Enhancing the Controlled Disclosure of Sensitive Information

Donald G. Marks, Amihai Motro, and Sushil Jajodia

Department of Information and Software Systems Engineering
George Mason University
Fairfax, VA 22030-4444

Abstract. The so-called "aggregation problem" is addressed, where the issue is how to release only a *limited* part of an information resource, and foil any attacks by users trying to aggregate information beyond the preset limits. The framework is that of relational databases, where sensitive information can be defined flexibly using view definitions. For each such view, the tuples that have already been disclosed are recorded *intensionally* rather than *extensionally*; that is, at each point, sub-view definitions are maintained that describe all the sensitive tuples that have been released to each individual. While our previous work foiled sequences of single-query attacks attempted by individual users, it did not consider multi-query attacks, where a *combination* of queries is used to invade the sensitive information. In this study we enhance our previous solutions to guard the sensitive information against two kinds of multi-query attacks: join attacks, and complement attacks. We then argue that the enhanced algorithm renders the sensitive information immune to attacks.

1 Introduction

In has been observed that often the release of a limited part of an information resource poses no security risk, but the release of a sufficient part of that resource might pose such risks. This problem of controlled disclosure of sensitive information is known as the *aggregation* problem. In [10] we argued that it is possible to articulate the specific sensitive concepts within a database that should be protected against over-disclosure, and we provided an accounting system to enforce such controlled disclosure. Our methods foil any attempt to attack these predefined secrets either by disguising queries or by surreptitiously accumulating tuples. The accounting methods that we developed to thwart such attempts were shown to be both accurate and economical.

Our previous work tracked continuously the queries attempted by each individual user, but it assumed that each "attack" consisted of a single query at a time; that is, it did not consider attacks through combinations of several

The work of Motro was supported in part by ARPA grant, administered by the Office of Naval Research under Grant No. N0014-92-J-4038. The work of Jajodia was supported in part by NSF Grant No. IRI-9303416.

queries. In this study we enhance our previous solutions to guard the sensitive concepts against two kinds of multi-query attacks: (1) *join attacks*, in which two queries targeting narrower (and therefore unprotected) associations of attributes are joined, to create larger associations that are supposed to be protected; and (2) *complement attacks*, in which a query targeting a larger set of tuples (which embeds the protected tuples, but is itself unprotected) is combined with a query on the unprotected tuples within the larger set (which is unprotected), to derive tuples in the protected set. In addition, we show that the enhanced algorithm prevents any kind of attack on sensitive concepts.

Section 2 summarizes these concepts and methods as developed in [10]. Section 3 describes the multi-query attack strategies, Section 4 shows how these attacks can be foiled, and Section 5 argues that the enhanced algorithm provides concepts with immunity to attacks. Section 6 considers several related considerations, and Section 7 discusses further research problems.

1.1 Related Work

Related work was discussed in [10] and has not changed significantly in the interim. Major studies of interest include [11, 4, 7, 6, 5]. Many of the previous studies suggest terminology and reason from examples to derive specific solutions for specific problems. Recent work on the problem has been meager, with an occasional reference in the more general context of inference problems. For example, Campbell [2] notes that aggregation is a "big security problem" but offers no references to detailed studies, only to approaches that work in some cases. Of course, many of the techniques used in this study have been developed previously for use on other problems. These efforts are noted as appropriate.

Similar problems occur for statistical databases (SDB). In statistical databases users can retrieve various characterizations (such as salaries), but not identities (such as people or institutions). The work of [1] and [9] provide excellent overviews of statistical database security problems. None of the SDB techniques assume that the data is kept in a relational databases, rather the problems are couched in terms of simple tables. Common techniques for controlling disclosure include linear programming and answer size restrictions. Little research has been done on auditing or query sequence control as is addressed here. If the holders of statistical data, such as census bureaus, wish to make data more widely available through on-line access to relational databases, variations of the techniques presented here may have applicability.

Our approach allows a flexible (view-based) definition of the sensitive information, but develops an accurate method for accounting access to such views. The main contribution of [10] was a method to articulate specific sensitive concepts and to account for individual user access to these sensitive concepts. The main contributions of this study is to control alternative ways of determining tuples in these sensitive concepts.

We choose views as our mechanism for articulating more specific sensitive concepts. This was previously proposed by Denning et al. [3] as an access control mechanism, but this is the first application of the technique to the aggregation

problem. As queries are also views, we can then check each query to see if it intersects a sensitive view. This intersection is also a view; in fact, it is a sub-view of both the sensitive concept and the query. By maintaining a "history" of these sub-views of the sensitive views and their sizes, we know how many, and which, tuples of each sensitive concept have been accessed. This eliminates the need to maintain a history of the sensitive tuples themselves that have been accessed. The technique integrates the accounting and naming mechanisms of secrets, resulting in an efficient and complete system for tracking access.

1.2 Phonebook Example

The Secret Government Agency (SGA) Phonebook is a common example of the aggregation problem. In this example, the entire phonebook is a classified document and is not available without the appropriate clearance; yet, individual phonebook entries are available to inquiring callers. A simple example of such a phonebook, with scheme *Emp = (Name, Tel, Div, Mail, Bldg, Room)* is shown in Figure 1. The reason that the entire phonebook is classified is because a phonebook provides a way of grouping the *individuals* into *concepts*. In relational database systems tuples are associated together by means of *views*, so each view may be regarded as a concept.

Name	Tel	Div	Mail	Bldg	Room
A. Long	x3333	A	m505	2	307
P. Smith	x1111	B	m303	2	610
E. Brown	x2345	B	m101	1	455
C. Jones	x1234	A	m202	1	307
M. Johnson	x1234	A	m101	3	103
B. Stevenson	x2222	A	m202	1	305
S. Quinn	x2222	C	m606	3	101
R. Helmick	x1234	A	m404	1	307
A. Facey	x1122	C	m505	2	400
S. Sheets	x2345	B	m101	1	455

Fig. 1. The Phonebook example

2 The Model

2.1 Basic Assumptions

The model is mostly unchanged from [10], and we repeat here the assumptions and features that are relevant to this paper. We assume that the sensitive information is a relatively small portion of a relational database. We adopt the usual

definition of relational databases, but restrict our attention to databases that are *single* relations, each with a *simple key*, and to *projection-selection* views, where all selections are conjunctions of simple clauses of the form *attribute* = *value*. We denote the database scheme $R = (A_1, \ldots, A_n)$. The domain D_i of an attribute A_i is the projection of the given instance of R on this attribute. This so-called *active* domain is the finite set of values used for A_i in the database instance. All tuples $t = (t_1, t_2, \ldots, t_n)$ are therefore elements of the set $D_1 \times D_2, \cdots, \times D_n$.

We will define both sensitive information and queries in terms of such selection-projection views. More comprehensive views may be formed by taking complements and unions of views. Databases consisting of several relations may be treated view the Universal Relation formalism [12]. The limitation of selection clauses to the form *attribute* = *value* is fairly serious, as it prohibits clauses of the form *attribute* < *value*. While such views may be handled by decomposing them into a set of *attribute* = *value* views, handling such views satisfactorily is a topic for future research.

2.2 Queries and Concepts

A *query* is a view. Its extension in the present database instance is the *answer* to the query. Queries are defined by users and describe the information they are seeking. A *concept* is also a view. Concepts are defined in the system and describe the information that needs to be protected. Views (queries or concepts) may be syntactically different, but yet describe the same information. Consider the example database scheme *Emp = (Name, Tel, Div, Mail, Bldg, Room)* and the views **select** *Name, Room* **where** *Room*=103 and **select** *Name* **where** *Room*=103. Both view definitions are identical, except that the latter view does not project a selection attribute which is projected by the former (*Room*). Nevertheless, because the values of selection attributes are known (in this case, the constant value 103), there is no difference in the information these views describe. Consequently, regardless of their syntax, we shall treat all views as is their projection attributes include all their selection attributes.

2.3 Concept Disclosure

Let U and V be views of database scheme R. U *overlaps* V, if their selection conditions are not contradictory, [1] and U's projection attributes contain V's projection attributes. When U overlaps V, then the extension of U could be processed by another view that will remove the extra attributes. Some of the resulting tuples may be in the extension of V.

Assume that U overlaps V. The *restriction* of V to U, denoted $V \mid U$, is the view obtained from V by appending to its selection condition the selection condition of U. The *exclusion* of U from V, denoted $V \mid \neg U$, is the view obtained

[1] The selection conditions of U and V are *contradictory*, if U's selection condition includes the clause $A_i = a$ and V's selection condition includes the clause $A_i = b$, for some attribute A_i and two different constants a and b.

from V by appending to its selection condition the *negation* of the selection condition of U.[2] Obviously, $V = (V \mid U) \cup (V \mid \neg U)$.

Let C be a concept view and let Q be a query view. Q *discloses* C, if Q overlaps C. Intuitively, a query discloses a concept, if its result could be processed by another query, to possibly derive tuples from the protected concept. The disclosure relationship between a query and a concept is illustrated schematically in Figure 2.

As an example, with the previous database scheme, consider this concept

$$C = \pi_{Name,Div,Room}\sigma_{(Room=103)\wedge(Div=\mathrm{B})}$$
(names of those in division B and in room 103)

and these three queries

1. $Q_1 = \pi_{Name,Tel,Div,Room}\sigma_{(Room=103)\wedge(Div=\mathrm{B})\wedge(Tel=\mathrm{x2345})}$
   (names of those in room 103, in division B, and with telephone x2345)
2. $Q_2 = \pi_{Name,Div,Room}\sigma_{Div=\mathrm{B}}$
   (names and rooms of those in division B)
3. $Q_3 = \pi_{Name,Div,Room}\sigma_{Room=102}$
   (names and divisions of those in room 102)

Q_1 discloses C, because applying the query $\pi_{Name,Div,Room}$ to the result of Q_1 may yield some tuples in C. Q_2 discloses C in its entirety, because applying the query $\sigma_{Room=103}$ to the result of Q_2 yields all the tuples of C. Q_3 does not disclose any tuples of C because their selection conditions are contradictory.

Notice that a concept protects its tuples, but not its sub-tuples; i.e., a query on a *subset* of the concept's projection attributes does not disclose the concept. On the other hand, a query on a *superset* of the attributes would disclose the concept (unless their selection conditions are contradictory).

As mentioned earlier, disclosure control requires that the number of tuples disclosed from a given concept does not exceed a certain predetermined number. For each concept C we define three integer values called *concept total*, *concept threshold* and *concept counter*, and denoted respectively, N, T and D. N denotes the total number of tuples in the extension of this concept, T denotes the maximal number of tuples that may be disclosed from this concept, and D denotes the number of tuples from this concept that have already been disclosed. If $T \geq N$, then the concept is *unrestricted*; we shall assume that none of the concepts are unrestricted. As queries are processed, the database system must keep track of D to ensure that $D \leq T$. The number of tuples in the extension of a view V will be denoted $\|V\|$; e.g., $\|C\| = N$.

[10] described a quick method that determines whether Q discloses C, and then defines the precise sub-view of C that is disclosed by Q. This method was at the basis of several algorithms for controlling the disclosure of sensitive concepts. The main feature of the solution is that tuples that have already been disclosed are recorded *intensionally* rather than *extensionally*; that is, at each point, view definitions are maintained that describe all the concept tuples that have released to each individual.

[2] Note that the resulting selection condition is no longer a simple conjunction.

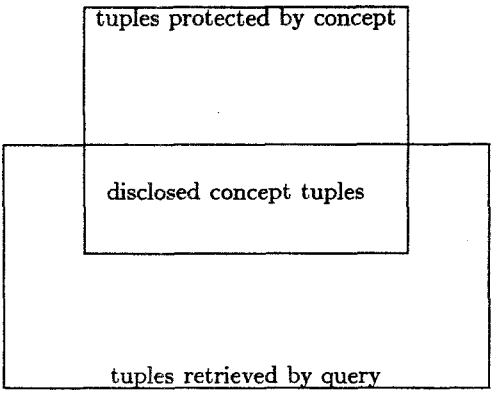

Fig. 2. Disclosure relationships between a query and a concept

3 Attack Strategies

As already noted, an essential principle behind these methods is that a concept protects its tuples, but not its sub-tuples; i.e., a query on a *subset* of the concept's projection attributes is always allowed. The idea is that concepts are designed to protect *minimal associations* of attributes; any lesser associations are assumed to be "harmless". This, however, leaves concepts vulnerable to attacks that attempt to construct additional concept tuples from information that is available freely.

Recall that a concept is a set of projection attributes α and a selection condition ϕ (and α includes the attributes used in ϕ). The obvious way to generate tuples over α that satisfy ϕ is to start with "larger" views, where either the set of attributes contains α and/or the condition does not contradict ϕ, and then use projection and/or selection to generate concept tuples. However, such views are tracked by the algorithms described in [10].

The only other possibility is to generate concept tuples from views in which the set of attributes is *strictly contained* in α and/or the condition *contradicts* ϕ, as such views are not controlled by these algorithms.

Given the sensitive concept

$$C = \textbf{select } \alpha \textbf{ where } \phi$$

two attacks are possible:

1. **Join.** In the join attack two queries are submitted:

$$Q_1 = \textbf{select } \alpha_1 \textbf{ where } \phi_1$$
$$Q_2 = \textbf{select } \alpha_2 \textbf{ where } \phi_2$$

where $\alpha_1 \cup \alpha_2 = \alpha$ and $\alpha_1 \cap \alpha_2$ contains a *key* to C, and ϕ_1 and ϕ_2 are conditions that do not contradict ϕ. Both queries are allowed, because

the attribute sets α_1 and α_2 are not protected. Clearly, their natural join $Q_1 \bowtie Q_2$ yields tuples in C.

2. **Complement.** In the complement attack two queries are submitted:

$$Q_1 = \textbf{select } \alpha' \textbf{ where } \theta$$
$$Q_2 = \textbf{select } \alpha \textbf{ where } \theta \wedge \neg\phi$$

where θ is a condition that is less restrictive than ϕ, and α' is obtained from α, by removing the selection attributes that are no longer necessary, because θ requires less attributes than ϕ. The former query is allowed because α' is not protected; the latter query is allowed because its condition contradicts ϕ. Clearly, their difference $Q_1 - Q_2$ (the complement of Q_2 within Q_1) yields tuples in C.

In both attacks, some additional information was used. In the first attack, it was knowledge of the database scheme and the key attribute. In the second attack, the condition $\neg\phi$ would have to be expressed via specific values that "complement" the values used in ϕ. In both cases, however, the system must assume that such knowledge might be available to the attacker.

As an example, assume the sensitive concept

select *Name* **where** *Bldg*=1 **and** *Room*=307

The key to this concept is *Name*.

1. **Join.** Consider the queries

$$Q_1 = \textbf{select } \textit{Name}, \textit{Tel} \textbf{ where } \textit{Bldg}=1$$
$$Q_2 = \textbf{select } \textit{Name}, \textit{Tel} \textbf{ where } \textit{Room}=307$$

Both would be allowed as neither contains the complete set of the concept's attributes (*Name, Bldg, Room*). Yet, their natural join "contains" the concept (appropriate selecting and projecting from this join will yield the concept in its entirety).

2. **Complement.** Consider the queries

$$Q_1 = \textbf{select } \textit{Name} \textbf{ where } \textit{Bldg}=1$$
$$Q_2 = \textbf{select } \textit{Name} \textbf{ where } \textit{Bldg}=1 \textbf{ and } \textit{Room}=305$$
$$Q_3 = \textbf{select } \textit{Name} \textbf{ where } \textit{Bldg}=1 \textbf{ and } \textit{Room}=455$$

The first would be allowed because it does not contain the entire set of the concept's attributes, and the other two because their selection conditions are contradictory with that of the concept. Yet, the difference of the first and the union of the other two corresponds to the concept.

4 Guarding against Attacks

The common element in both attacks was the lack of control over views that ask for a subset of the concept's attributes that contains the key attribute of the concept (Q_1 and Q_2 in the first attack, and Q_1 in the second attack). By extending our control to such views, both kinds of attacks would be foiled.

This extension implies a significant change to the semantics of a sensitive concept: *a concept now protects also all its key projections.* To implement the new semantics, we define a new view relationship.

Let U and V be two views of database scheme R. U *critically overlaps* V, if their selection conditions are not contradictory, and the intersection of their projection attributes contains a *key* of V. When U critically overlaps V, then the extension of U could be processed by a projection that removes the attributes in U but not in V, and possibly generate sub-tuples of tuples in the extension of V, that include its key attribute. The definitions of the restriction $V \mid U$ and the exclusion $V \mid \neg U$ remain unchanged.

We now update the *disclosure* relationship between a query and a concept. Let C be a concept and let Q be a query. Q *discloses* C if Q critically overlaps C. Intuitively, a query discloses a concept if its result could be processed by another query to possibly derive sub-tuples from the protected concept that include its key attribute. The new disclosure relationship between a query and a concept is illustrated schematically in Figure 3.

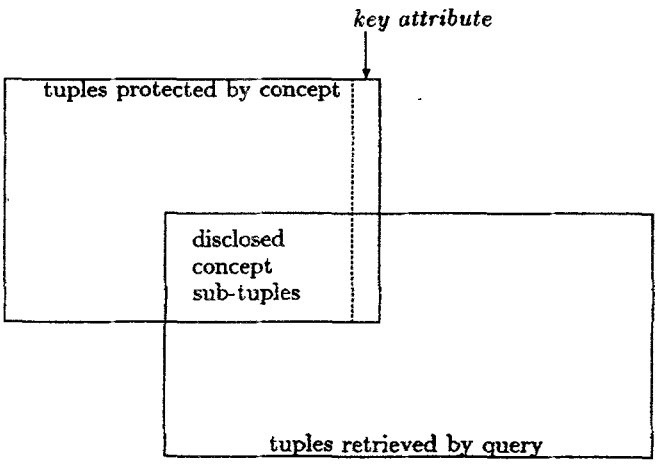

Fig. 3. Disclosure relationships based on critical overlap

By considering queries that target the key of a concept (i.e., critically overlap the concept), as if they "fully" overlap the concept, attacks of the kind described above would be foiled. With this extended definition of disclosure, the

earlier disclosure control algorithms are still valid. Figure 4 reproduces such an algorithm.

This algorithm associates with each concept C a *predicate* P that describes the concept tuples that have already been disclosed. P is initialized to *true*. Assume that $Q_1, \ldots, Q_p$ have already been processed when Q_{p+1} is received, and let $\alpha_1, \ldots, \alpha_p$ denote their respective selection conditions. The present value of P would be $\alpha_1 \vee \cdots \vee \alpha_p$. After computing the restriction of C to Q_{p+1} (the tuples in this concept that are disclosed by this query), we exclude from it the view σ_P (the tuples of this concept that have already been disclosed by the previous queries). The tuples in this new query are those that have *not* been delivered already.

The input to this algorithm is a set $C_1, \ldots, C_m$ of protected concepts, each with its associated predicate P_i and counters N_i, T_i and D_i, and the query Q whose selection predicate is α. When it terminates, the value of *permit* indicates whether the answer to Q should be presented to the user or not.

```
Algorithm (disclosure)
permit := true
materialize Q
i := 0
while permit and i < m
do
    i := i + 1
    M_i := 0
    if Q critically overlaps C_i
    then
        M_i := ||(C_i | Q) | ¬σ_{P_i}||
        if D_i + M_i > T_i
        then
            permit := false
            break
        endif
    endif
done
if permit
then
    for i = 1, ... m
    do
        P_i := P_i ∨ α
        D_i := D_i + M_i
    done
endif
```

Fig. 4. Disclosure control algorithm that defeats join and complement attacks

It should be noted that key-containing sub-tuples are counted as if they were full tuples. That is, a query that overlaps a concept and a query that critically overlaps a concept incur the same "cost" to the user, against that concept. However, a user who, in two separate queries, extracts two sub-tuples of the *same* tuple, is only "charged" once!

Note that a query that intersects only with the non-key attributes of a concept, is answered freely, as concepts protect only their key projections. The reason is that such queries cannot be used in any of the attacks described earlier. It should be noted, though, that it might be necessary to consider *near keys* (i.e., concept attributes whose active domains are nearly the size of the concept) as if they were keys.

As an example, consider the previous concept **select** *Name* **where** *Bldg*=1 **and** *Room*=307, and the join attack and complement attacks specified earlier. In the join attack, both Q_1 and Q_2 critically overlap the concept, and their tuples will be accounted for. Similarly, in the complement attack, Q_2 and Q_3 will be delivered freely, as they do not critically overlap the concept (their selection conditions are contradictory to the concept's), but Q_1, which critically overlaps the concept, will be accounted for. Altogether, these attempts no longer provide any additional opportunities.

5 Immunity to Attacks

In this section we argue that the algorithm presented in Section 4 provides sensitive concepts with immunity to attacks.

Consider a set X of elements and a binary property p, where each $x \in X$ either has or does not have this property p, and assume that we are tasked with finding the subset Y of X of elements that have the property p. It is obvious that Y could be built in only two ways:

1. **Positively**: by starting with $Y = \emptyset$, and then examining every element of X and adding it to Y iff it has p.
2. **Negatively**: by starting with $Y = X$, and then examining every element of X and removing it from Y iff it does not have p.

Transferring this problem to relational databases, we assume a set of unique values, each such value is associated with a non-unique set of values, and an extra value that denotes whether the element has the property. Altogether, an element is now $x = (x_1, \ldots, x_n)$, where x_1 provides the identity of the element (the key), x_n denotes (e.g., using the values 1 and 0) whether the element has the property or not (the condition), and the other values constitute the description of the element. The task is now to isolate elements $x = (x_1, \ldots, x_n)$ such that $x_n = 1$.

In accordance with the previous observation, these elements could be isolated in one of two ways:

1. **Positively**: $Y = \sigma_{x_n=1}(X)$
2. **Negatively**: $Y = X - \sigma_{x_n \neq 1}(X)$

Note that the first X in the second formula does not use its x_n values, and the method will still work even when this value is unknown.

Assume now that we are tasked to *prevent* the retrieval of tuples of X that satisfy the condition. Then defeating these two methods of construction is guaranteed to accomplish this task.

Clearly, barring all tuples $(x_1, \ldots, x_n)$, unless they are certain not to satisfy the condition, will defeat both methods of construction, because when X contains only tuples that do not satisfy the condition, both formulas evaluate to the empty set. Thus, to populate the set Y, one needs to construct tuples $(x_1, \ldots, x_n)$ that *might* satisfy the condition.

Intuitively, to populate the set X in the first formula (the positive method) with tuples $(x_1, \ldots, x_n)$ that might satisfy the condition, one may use (1) queries that specify all these attributes (and possibly others) and might satisfy the condition; or (2) queries that specify fewer attributes and might satisfy the condition. Only the former kind of queries was controlled by the earlier model. The latter kind is the source for the join attack. The set X in the first term of the second formula (the negative method) can be populated in similar ways (though x_n need not be retrieved). Once this term is populated, it is combined with the second term to form a complement attack. Hence the two new attack methods, the join and the complement.

Yet, regardless of the specific method, by barring access to any tuple that contains the key x_1, unless it is certain not to satisfy the condition x_n, it is clear that tuples $(x_1, \ldots, x_n)$ that might satisfy the condition would never become available (and Y will remain empty). Queries that contain tuples with the key and might satisfy the condition, were said to critically overlap the concept. This discussion is summarized in the following theorem.

Theorem. Monitoring queries that critically overlap a concept provides complete protection to the concept.

Of course, inference based on other knowledge may still be possible [8], but users will not be able to attack the concept by queries alone.

6 Additional Considerations

6.1 Partial Answers

All our disclosure algorithms behaved similarly, when the size of an answer to a disclosing query exceeds the allotment remaining on a particular concept: such a query is denied in its entirety. This approach maintains the *completeness* of the answers issued; that is, queries are either answered completely, or not answered at all.

At times, it would seem preferable in such situations to deliver the remaining allotment, even if it does not answer the query completely. It should be empha-

sized that in abandoning completeness, we are violating a basic premise of query answering mechanism, by which all answers must be sound and complete.[3]

Modifying our disclosure algorithms to deliver partial answers is straightforward, though a question that still remains is which tuples to deliver, and whether users should be notified when answers are incomplete.

6.2 Disclosing Key Projections

The enhancements of the controlled disclosure algorithm against attacks offered in Section 4 required new semantics for concepts: concepts protect all their key projections. At times, however, these semantics may be at odds with reality. In our example, consider the sensitive concept **select** *Name* **where** *Bldg*=1 **and** *Room*=307. To protect this concept, in every query that includes *Name*, the number of employees in room 307 of building 1 is noted, and the cumulative number is not allowed to exceed a predetermined threshold. However, this might prove impossible, if, for example, the institution needs to make public its entire list of employees in building 1; i.e., the concept **select** *Name* **where** *Bldg*=1. In such a case, the concept becomes vulnerable to complement attacks, via a query on the names of employees in building 1, and queries on the names of employees in building 1 but in rooms other than 307.

Hence, when a key projection of a concept cannot be protected, the concept remains vulnerable to complement attacks. Formally, assume that

$$C = \textbf{select } \alpha \textbf{ where } \phi$$

is declared as sensitive, but

$$D = \textbf{select } \alpha' \textbf{ where } \theta$$

is disclosed, where θ is a condition that is less restrictive than ϕ, and α' is obtained from α, by removing the selection attributes that are no longer used, because θ requires less attributes than ϕ. In this case D may be combined with the query

$$Q = \textbf{select } \alpha \textbf{ where } \theta \wedge \neg\phi$$

to attack C, using the complement strategy $D - Q$.

Our solution in this case is to enlarge the condition of C from ϕ to θ; that is, to replace C with

$$C' = \textbf{select } \alpha \textbf{ where } \phi$$

The query Q would then return no tuples, its condition being $\theta \wedge \neg\theta$, and $D - Q$ will return D. In the above example, this would mean changing the sensitive concept to **select** *Name* **where** *Bldg*=1.

[3] It may be interesting to note, that whereas here we maintain soundness while abandoning completeness, in statistical databases the dual approach has been suggested, in which completeness is maintained while soundness is abandoned. Specifically, complete answers are augmented with fictitious information.

7 Conclusion

In this paper we extended our previous work on the controlled disclosure of sensitive information to foil multi-query attacks, and we argued that the enhanced algorithm prevents any kind of attack on sensitive information. Much work remains to be done and we mention three research problems.

First, we are interested in extending this work to remove the simplifying assumptions that have been made. Chiefly among them are the assumptions of a single relation database, and the limitations on the kind of views that may be used for both concepts and queries.

Second, we have assumed that the databases are "static"; i.e., when considering a sequence of queries by the same user, we assumed that the extensions of concepts do not change via insertions or deletions of tuples. While this may be the nature of statistical databases, a general disclosure control algorithm must account for updates as well (for example, when the tuples previously released are deleted from the database).

This study extended the analysis from sequences of single-query attacks to sequences of multi-query attacks. However, each user continues to maintain an individual "account" with the system. The methods are thus still vulnerable to groups of several users who in collusion aggregate an amount of information that is considered to pose a security risk.

References

1. M. R. Adam and J. C. Wortmann. Security-control methods for statistical databases: a comparative study. *ACM Computing Surveys*, 21(4):515–556, December 1989.
2. J. R. Campbell. A brief database security tutorial. In *Proceedings of 18th National Information System Security Conference* (Baltimore, Maryland, October 10–13), pages 740–757, 1995.
3. D. E. Denning, S. G. Akl, M. Morgenstern, P. G. Neumann, R. R. Schell, and M. Heckman. Views for multilevel database security. In *Proceedings of IEEE Symposium on Security and Privacy, (*Oakland, California*)*, 1986.
4. S. Jajodia. Aggregation and inference problems in multilevel secure systems. In *Proceedings of the 5th Rome Laboratory Data Security Workshop*, 1992.
5. T.Y. Lin. Database, aggregation and security algebra. In *Proceedings of the 4th IFIP Working Conference on Database Security*, September 1990.
6. T.F. Lunt. Aggregation and inference: Facts and fallacies. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 102–109, May 1989.
7. T.F. Lunt and R.A. Whitehurst. The Sea View formal top level specifications. Technical report, Computer Science Laboratory, SRI International, February 1988.
8. D. G. Marks. Inference in MLS databases. *IEEE Transactions on Knowledge and Data Engineering*, 8(1):46–55, February 1996.
9. Z. Michalewicz. Security of a statistical database. In Z. Michalewicz, editor, *Statistical and Scientific Databases*. Ellis Horwood, Chichester, England, 1991.

10. A. Motro, D. G. Marks, and S. Jajodia. Aggregation in relational databases: Controlled disclosure of sensitive information. In *Proceedings of ESORICS-94, Third European Symposium on Research in Computer Security,* (Brighton, UK, November 7–9), Lecture Notes in Computer Science No. 875, pages 431–445. Springer-Verlag, Berlin, Germany, 1994.
11. B. Thuraisingham, editor. *Proceeding of the 3rd RADC Database Security Workshop*, Report MTP 385. Mitre Corp., Bedford, Massachusetts, 1991.
12. J. D. Ullman. *Database and Knowledge-Base Systems, Volume II.* Computer Science Press, Rockville, Maryland, 1989.