The Inference Problem in Multilevel Secure Databases

Xue Ying Chen
Department of Computer Science
Lakehead University

Supervisor: Dr. Ruizhong Wei

April 2005

Your file Votre référence
ISBN: 0-494-10651-4
Our file Notre référence
ISBN: 0-494-10651-4

Abstract

Conventional access control models, such as role-based access control, protect sensitive data from unauthorized disclosure via direct accesses, however, they fail to prevent unauthorized disclosure happening through indirect accesses. Indirect data disclosure via inference channels occurs when sensitive information can be inferred from non-sensitive data and metadata, which is also known as "the inference problem". This problem has drawn much attention from researcher in the database community due to its great compromise of data security. It has been studied under four settings according to where it occurs. They are statistical databases, multilevel secure databases, data mining, and web-based applications.

This thesis investigates previous efforts dedicated to inference problems in multilevel secure databases, and presents the latest findings of our research on this problem. Our contribution includes two methods. One is a dynamic control over this problem, which designs a set of accessing key distribution schemes to remove inference after all inference channels in the database has been identified. The other combines rough sets and entropies to form a computational solution to detect and remove inferences, which for the first time provides an integrated solution to the inference problem. Comparison with previous work has also been done, and we have proved both of them are effective and easy to implement.

Since the inference problem is described as a problem of detecting and removing inference channels, this thesis contains two main parts: inference detecting techniques and inference removing techniques. In both two aspects, some techniques are selectively but extensively examined.

Acknowledgements

I would like to express appreciation to my thesis supervisor, Professor Ruizhong Wei, for leading me into this interesting research field and guiding me throughout my master's studies. I would also like to thank the external examiner of my thesis, Professor Michael J. Jacobson at University of Calgary, for his helpful comments. Finally, I would like to thank my families for their love and support.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Conventional access control models, such as role-based access control, protect sensitive data from unauthorized disclosure via direct accesses, however, they fail to prevent indirect accesses. In Figure 1.1, under a traditional access control model, the direct access to sensitive data is prohibited. However, the sensitive data can still be revealed via inference by the user. We say an inference problem occurs when sensitive information can be inferred from non-sensitive data and metadata.

Metadata refers to database constraints, or outside information, such as domain knowledge and query correlations. Depending on the level of accuracy by which the sensitive information is revealed, full disclosure or partial disclosure may occur. For example, suppose there is a database providing statistical data which contains a sensitive attribute *Salary*. A common practice to protect attribute *Salary* is to block any direct queries over it. Unfortunately, this measure is not good enough. Since the computation on it is available, users can still use mathematical techniques to infer a certain person's salary.

The inference problem has been studied by researchers in the database community for a long time, for example, [25][38] surveyed research efforts seeking to formalize and resolve the problem. In summary, the inference problem can be classified into four categories depending on the setting where it occurs: statistical databases, multilevel secure(MLS) databases, data mining, and web-based applications. Although thesis focuses on the inference problem in MLS databases, I would like to give a brief description for each of them for the sake of completeness.

1.1 Inference Problem in Statistical Databases

The inference problem was first considered in statistical databases, the above example is a typical one of this case. The security requirement in statistical databases is to provide access to statistics about groups of entities, such as mean, median, standard

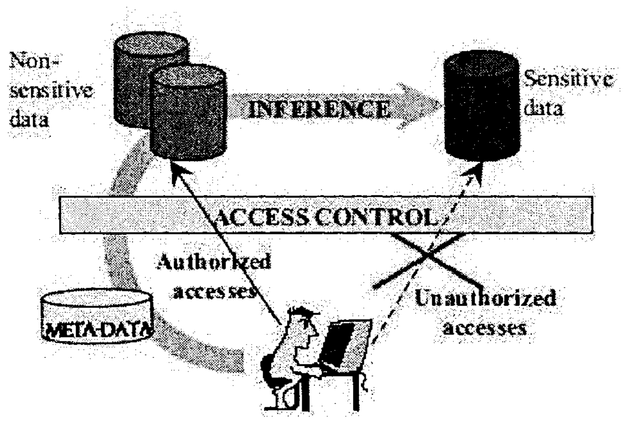


Figure 1.1: Indirect information access via inference

deviation, and so on, while protecting the confidentiality of the individual entities. The threat is that an attacker may pose queries on aggregate statistics over a period of time and perform arithmetic operations on the answers received, thus obtain confidential information about an individual entity. For example, suppose that an attacker wants to find out the salary of $A1$. He can do this by asking for the average salaries of $A1, A2$ and $A3$; of $A2, A3, A4$ and $A5$; and of $A4$ and $A5$. Alternatively, he can ask for the average salary of some set of individuals of which he knows that $A1$ is the only member.

Inference control in statistical databases has been extensively studied in [15][18][43][52]. A number of inference control mechanisms such as query size and query overlap, data swapping, and multidimensional transformation, were developed and their limitations established. The main problem is that simple inference control mechanisms are easily subverted. Mechanisms providing confidentiality with high assurance are often too complex and difficult to implement, or not applicable in general purpose databases and thus are limited to certain applications(e.g., U.S. Census Bureau database).

1.2 Inference Problem in Multilevel Secure Databases

Researchers have started considering the inference problem in multilevel secure(MLS) databases since early 1980's. Most of previous works focused on defining the problem and providing frameworks to address specific types of inferences. They provide an overview of different inference channels and present techniques to detect and remove them. They show that most of the inference channels in MLS databases are created by combining metadata (e.g., database constraints) with data in order to obtain information that has a higher security classification than the original data.

As we mentioned above, the inference problem in MLS databases has been described as a problem of detecting and removing inference channels. An inference channel is the minimum set of data needed to deduce the sensitive information. Correspondingly, previous papers on the detection of inference channels can be classified into two categories. One is to examine the functional dependencies among the attributes in the database schema (see [3][20][29][30][32][34][35][46][72]). These papers propose graph-based methods to detect illegal inferences. An inference channel is detected when there are two or more paths among the attributes, and the paths are labelled at different classification levels. The other category is related to partial or imprecise inferences (see [9][10][11][12][36][37][45][74]). Partial inferences occur when an unauthorized user is able to infer the value or a set of values for a data item with certain probability. Although the derived information is imprecise in nature, the granularity of the disclosed data item may be enough to create a security leak. The other feature of these methods is that all of them are trying to find a balance between the security and the functionality of databases.

The other aspect of the inference problem is the removal of inference channels. Previously proposed techniques can be organized into three categories. The first category is to remove inference channels during database design. This can be done by either modifying the database design or increasing the classification levels of some of the data items (see [4][27][29] [46][47][62][63][67][32]). However, these techniques often result in over-classification of data, and thus reduce the availability of data. For example, Hinke [29][30] presents a semantic-graph based inference detection tool to represent semantic relationships and to detect inference channels. Methods belonging to the second category seek to eliminate inference channels during query time. If an inference channel is detected, the query is either refused or modified to avoid security violations (see [16][42][48][48][68][69][70][79]). These methods allow inference detection at both the data and schema level. While data level inference detection allows increased data availability, it is computationally expensive. For example, Denning [16] recommends an authorized view equivalence schema to remove any unauthorized data from the answers to select-only, select-project, and select-project-join queries. However, she does not address the inference problem in the presence of database constraints. The work of Thuraisingham [69] presents a general and powerful logic-based framework. Dynamic models[13][65] are the most recently proposed techniques on removing the inference channels. In [13][65][78], accessing keys distribution schemes are designed to fulfill dynamic control over the inference problem. These methods consider both the query processing time and the availability of data.

1.3 Inference Problem and Data Mining

Data mining can be defined as the process of mining for implicit, previously unknown, and potentially useful information from very large databases by efficient knowledge discovery techniques. While data mining opens up an area for extracting valuable data, it also introduces significant security concerns. The inference problem is one of them. Since data mining is an attempt to answer the question " What does all this data mean?", and inference is basically an attempt to establish the relationships between data sets, we can say that data mining automates the inference problem. In addition, with the rapid increase of electronically available information, data mining represents an greater risk than in conventionally centralized databases for that information originating from different sources can be analyzed.

Papers considering the inference problem in data mining emerged during the mid 1990's (see [24][39][51][54][64][73]). Security threats based on data mining can be addressed either in preprocessing phase or during run-time. For preprocessing, a set of mining tools are applied on the database to check whether sensitive information can be disclosed from the learned patterns. For run-time, the inference controller evaluates the result to a user's request, and permits or rejects the release of the result based on this evaluation. In either mode, data mining abilities are reduced. Moreover, none of these approaches protect from inferences when the pattern discovered in one database is applied on a different database, since it is unrealistic to assume that all related database would enforce the same security policy.

Recently, several works[1][2][7][8] addressing privacy preserving data mining have surfaced. Their main motivation is to allow data accessible for mining purposes, while preserving the confidentiality of the data. Techniques such as data estimation, perturbation and sample size restrictions are used to remove any unwanted inferences. The main aim of this research is to apply minimal modification to the original data without disturbing the data mining results. In addition to secure data mining, efficient methods of data sharing is important in distributed settings because of the large size of the involved databases. Finally, a practical method should be independent from any specific data mining patterns due to the diversity of them.

1.4 Web-Based Inferences

With the further development of World Wide Web, new privacy problems surfaced. Almost at the same time, works to provide controlled accesses to documents in eXtensible Makeup Language (XML) format surfaced[5][23][40]. These models focus on defining access controls on XML documents, thus preventing privacy violations via direct data accesses. While these mechanisms are necessary, none of them provide technical solutions to enforce privacy requirements in the presence of possible inferences, or give assurance on the level of protection.

The development of technologies supporting the Semantic Web[6] , increases the risk of illegal data accesses via inference channels. Automated analysis allows software agents to integrate large amounts of possibly distributed data. Such integration is impossible for humans because of the size of the data sources. The main purpose of Semantic Web research is to provide interoperation and intelligent query processing[19][21][22][28][55] . While some inferences[66] are considered from the perspective of enabling machine processing of the Web, there is no comprehensive security analysis available.

Existing inference control technologies are insufficient to deal with this problem in the Web environment. Moreover, tracing user collaborations poses a challenge for security, where it might be desirable to support anonymous access to the web-based information resources.

1.5 Structure of the Thesis

As previously mentioned, the inference problem is a problem of detecting and removing inference channels. The purpose of this thesis is not only to provide a comprehensive view of the inference problem in MLS databases, but to present methods that assist DBAs in finding optimal solutions to this problem in practice. Therefore in later chapters, previous works are summarized, and the findings of our research are presented as well. To this end, the remainder of this thesis is structured as follows. Chapter 2 introduces the early efforts in formalizing the inference problem. Chapter 3 focuses on the detection of the inference problem, meanwhile corresponding removal techniques for some of them are given. Chapter 4 discusses inference removal techniques. In chapter 5, we give the conclusion of this thesis and put forward some future research topics.

Chapter 2

General Characterizations of Inferences

Although it is not easy, some efforts have been dedicated in formalizing the inference problem ever since its emergence. As a case in point, consider the following two stories from [75]. The stories concern the US government's attempt to keep the existence and the location of the Manhattan Project secret during World War II:

Finally I learned that we were going to New Mexico, to a place not far from Santa Fe. Never having heard about New Mexico, I went to the library and borrowed the "Federal Writers' Project Guide to New Mexico". At the back of the book, on the slip of paper on which borrowers signed their names, I read the names of Joan Hinton, David Frisch, Joseph McKibben, and all the other people who had been mysteriously disappearing to hush-hush war jobs without saying where. I had uncovered their destination in a simple and unexpected fashion. It is next to impossible to maintain absolute secrecy and security in war time.

This reminds me of another story. Since I knew Stebbins well, about a month after arriving at Los Alamos. I wrote to him. I did not say where I was but I mentioned that in January or February I had seen the star Canopus on the horizon. Later it occurred to me that as an astronomer he could easily have deduced my latitude since this star of the Southern skies is not visible above the 38th parallel.

There are several important things to note about these stories. First of all, in figuring out the destination of his mysteriously vanishing colleagues, Ulam did not use standard logical deduction. Instead, he looked for the best explanation for the fact that they had all checked out a guidebook for the same place. Second, although Ulam was able to deduce the destination of his friends from the guidebook, he did not know enough to look for the guidebook until he already knew some sensitive information: the location of the project he himself would be working on. If he had not known that information, he might have had to look at every guidebook in the library to find out the location of the Manhattan Project. Third, Ulam also needed some relatively nonsensitive information to make his deduction: namely, the fact that some of his colleagues had been leaving to work on what were apparently secret government projects. Finally, it is not always possible to predict what information an individual will know that can be used to deduce facts from other, apparently nonsensitive facts; this is the point of the story about Canopus and the 38th parallel.

2.1 Goguen and Meseguer's Theory

Probably the earliest formal characterization of the inference problem in databases is that of Goguen and Meseguer[27]. Consider a database in which each data item is given an access class, and suppose that the set of access classes is partially ordered. Define the relation $\longrightarrow$ as follows.

Definition 2.1.1 *Given data items x and y, we say that $x \longrightarrow y$ if it is possible to infer y from x.*

The relation $\longrightarrow$ is reflexive and transitive.

Definition 2.1.2 *A set S is said to be inferentially closed if whenever x is in S and $x \longrightarrow y$ holds, then y belongs to S as well.*

Definition 2.1.3 *For an access class L, let $E(L)$ denote the set consisting of all possible responses that are classified at access class less than or equal to L. There is an inference channel if $E(L)$ is not inferentially closed.*

Goguen and Meseguen do not set forth any one candidate for the relation $\longrightarrow$. They merely require that it be reflexive and transitive, and say that it will probably be generated according to some set of rules of inference, for example, first-order logic, statistical inference, and so on. However, they do denote that for most inference systems of interest, determining that $A \longrightarrow b$ (where A is a set of facts and b is a fact) is at best semidecidable, that is, there is an algorithm that will give the answer in a finite amount of time if $A \longrightarrow b$, otherwise may never halt.Complexity-theoretic properties of inference relations in multilevel databases are considered further by Thuraisingham in [71].

Goguen and Meseguer, besides not fixing on any inference system or set of inference systems, leave several other questions unexplored. For instance, they do not consider the effect on the inference problem of information that may exist outside the database. Nor do they attempt to include any measure of the difficulty of computing an inference.

2.2 Denning, Dorothy, and Morgenstern's Theory

As a refinement of Goguen and Meseguer's theory, Denning, Dorothy and Morgenstern in [17] derive the inference relation from classical information theory. Given two data items x and y, let $H(y)$ denote the uncertainty of y, and let $H_x(y)$ denote the uncertainty of y given x (where uncertainty is defined in the usual information-theoretic way). Then, the reduction in uncertainty of y given x is defined as follows:

$$INFER(x \rightarrow y) = \frac{H(y) - H_x(y)}{H(y)}$$

The value of $INFER(x \rightarrow y)$ is between 0 and 1. If the value is 0, then it is impossible to infer any information about y from x. If the value is between 0 and 1, y becomes somewhat more likely given x. If the value is 1, then y can be inferred given x.

The INFER relation can be used in the following way. Choose an $\epsilon > 0$ and define the sphere of influence of a set of data to be the set of all data items y such that there exists an x in the data set such that $INFER(x \rightarrow y) > \epsilon$. A system is safe from inference if, for each security class C, the sphere of influence of all data items with security level less than or equal to C does not contain any data elements of a higher or incomparable class.

Note that unlike the inference relations $\rightarrow$ defined in previous section, the sphere influence relation is nontransitive. For example, knowing the street on which someone lives may reduce our uncertainty about the person, knowing that an individual works on a project may reduce out uncertainty about the nature of the project, but knowing the name of a street inhabited by someone working on a project reduces our uncertainty about the project somewhat less.

This formulation is especially nice since it shows that inference is not an absolute problem, but a relative one. It gives us a way to quantify the bandwidth of the illegal information flow. On the other hand, it has serious drawbacks.

1. In most cases it is difficult, if not impossible, to determine the values of $H_x(y)$.
2. It does not take into account the computational complexity that is required to draw the inference.

To illustrate the second point, the following example from cryptography is presented in [17]. With few exceptions, the original text can be inferred from the encrypted text by trying all possible keys (so $H_x(y) = 1$ in this case). However, it is hardly practical to do so.

2.3 Categories of Inference Channels

A discussion of the kinds of inference functions that would be appropriate for determining the security of a multilevel database is given by Garvey, Lunt and Stickel [26]. Three kinds of inference channels are summarized:

1. Deductive channels, in which high data may be formally derived from low data.
2. Abductive channels, in which a deductive proof could be completed if certain low-level axioms were assumed.
3. Probabilistic channels, which occur when there is low data that can be used to decrease the uncertainty about the nature of high data, and it is likely that the low data will be known by a low user.

Ulam's second story, about the sighting of Canopus, is an example of abductive reasoning. Ulam wrote his friend that he had seen Canopus on the horizon and then realized that, not only could the low-level fact, together with another low-level fact, be used to deduce the high-level fact, but that the second low-level fact was likely to be known to the low-level recipient of his letter.

Techniques and tools that make use of abductive reasoning and probabilistic reasoning provide a good way of characterizing cases in which high-level information can be derived using low-level information from both inside and outside a database. Garvey and his colleagues discuss how these techniques might be used to make sure that a database is free from inference channels. For each high-level fact in the database, one attempts to discover what low-level facts could be used either to derive the high-level fact or to make it more likely to be derived. One also measures the likelihood that these low-level facts are known. They also point out that automated tools used for abductive proofs have features that measure the difficulty of a proof, and that such a feature can provide a guide to the difficulty a low-level user would have in performing the inference.

Chapter 3

Detection of Inferences

In this chapter, we will introduce some representative techniques in detecting the inference problem, meanwhile, some techniques to remove the problem are also briefly described. Section 3.1 and 3.2 discuss methods detecting inferences based on the functional dependencies among attributes. Section 3.3 introduces an inference detecting method that runs during query processing time. The last two sections are about partial inferences.

3.1 Database Inference Engine Based on Semantic Relationship Graph

In this section we are going to introduce an inference detection approach based on a graph based description of data, both database resident data and data commonly known in the application area. It shows how inference is detected basing on finding paths in what is termed a semantic relationship graph. It also suggests two measures to conquer the major challenge of this approach, the elimination of false inference.

3.1.1 Graph Based Inference Detection

In [30], Hinke presents an approach for a graph based method of inference detection. This approach uses a graphical means to describe the relationship of data. The graph is called the semantic relationship graph. It can be viewed as a variant of the entity relationship (E-R) graph, modified such that no distinction is made between entity sets and attributes of entity sets. The graph consists only of entity sets and links between entity sets which indicate binary relationships between entity sets.

An entity is some distinguishable thing. In this view, an entity could be something physical, such as a car, or some attribute of a car, such as its color. A set of such entities is called an entity set. A relationship between two entity sets exists if an

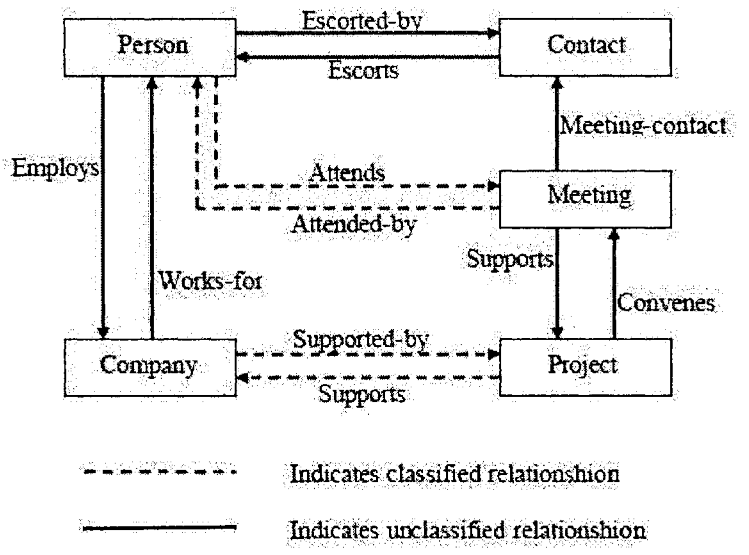


Figure 3.1: Example Semantic Relationship Graph

entity in one entity set is related to an entity in another entity set. For example, consider the entity sets "Project" and "Company". Each project is supported by one or more companies that is performing the work on the project. This means that there is a relationship between elements of the project entity set and the company entity set, and hence a relationship between the "Project" and "Company". In this case, since multiple companies can support a single project, the project-company relation is said to have a fan-out of n. A relationship between each project entity set and the manager entity set could have a fan-out of just 1, if each project has only a single manager.

The semantic relationship graph is a directed graph, hence the relationship between entity set X and entity set Y consists of two links - one from X to Y and the other from Y to X. Each link has a distinct fan-out n, $n \geq 1$. If both links have a fan-out that is greater than one, this is what is considered an $n : n$ relationship in database theory.

Figure 3.1 shows portions of a semantic relationship graph for an R&D project application area. Consider the situation in which the companies that support a sensitive project are themselves sensitive. Note that in Figure 3.1 the Project-Company relationship is shown with a dotted line, indicating that this is sensitive information and not generally available. To discover whether the available data permits this relationship to be inferred, we search the semantic relationship graph for a second path. One such path is Project-Meeting-Person-Company. This path connotes the case in which a person from company XYZ attends a meeting for project X-1. The fact that this person works for company XYZ can be used to make the inference that company XYZ is supporting project X-1.

Fortunately, the path Meeting-Person is also sensitive, thus the path Project-Meeting-Person-Company is not generally available. However, there exists another path Meeting-Contact-Person. A contact is the name of a person who schedules a room for a particular meeting. The contact also represents the escort for a visitor to an R&D facility. Using contact, we now have a second path (an inference path) from Project to Company, Project-Meeting-Contact-Person-Company.

This second path involves two inference rules. The first inference rule is that if a meeting and a person share a contact, then the person is attending the meeting. The second inference rule is that if a person working for a company attends a meeting for a project, then the company is supporting the project. Together, both of these rules permit us to infer that a particular company is supporting a particular project.

Note that while inference rules can be stated from the graph, they were not used to discover the inference path. The inference path was discovered by attempting to find a second path between project and company, since the direct path was effectively closed due to the fact that it is sensitive and thus protected.

While many second paths may exist, it suffices to find just one to perform an inference. If this path is closed, then the technique can search for any other second paths. Ultimately, all such second paths must be eliminated to eliminate the inference possibilities from the database.

The graph based approach has all the rules that it needs to find all inferences. The price for the completeness and yet simplicity is that the inference channels it finds also includes numerous false inference. Examples of false inference channels are shown in Figure 3.2. Consider the path that runs from Project to Company, through the banks-with link. Under the path finding rule, the objective is to find an alternative path between Project and Company. That path consisting of Project-Meeting-Contact-Person-Company(for which the person works) would be viewed as a legitimate inference path. However the path consisting of Project-Meeting-Contact-Person-Company(at which the person Banks), would not be considered a legitimate inference path. It is what we are calling a false inference. The figure also illustrate the false inference arising from the case where a Person(whose contact is Contact 1) is actually attending meeting A (with meeting Contact 2). But the inference rules we have been using would infer that the person is attending Meeting B. This would also be a false inference.

While it may not be possible to eliminate all false inferences, the strategy under the graph based approach must sift out as many false inferences as possible.

This approach permits one to identify what constitutes false inferences. Moreover,

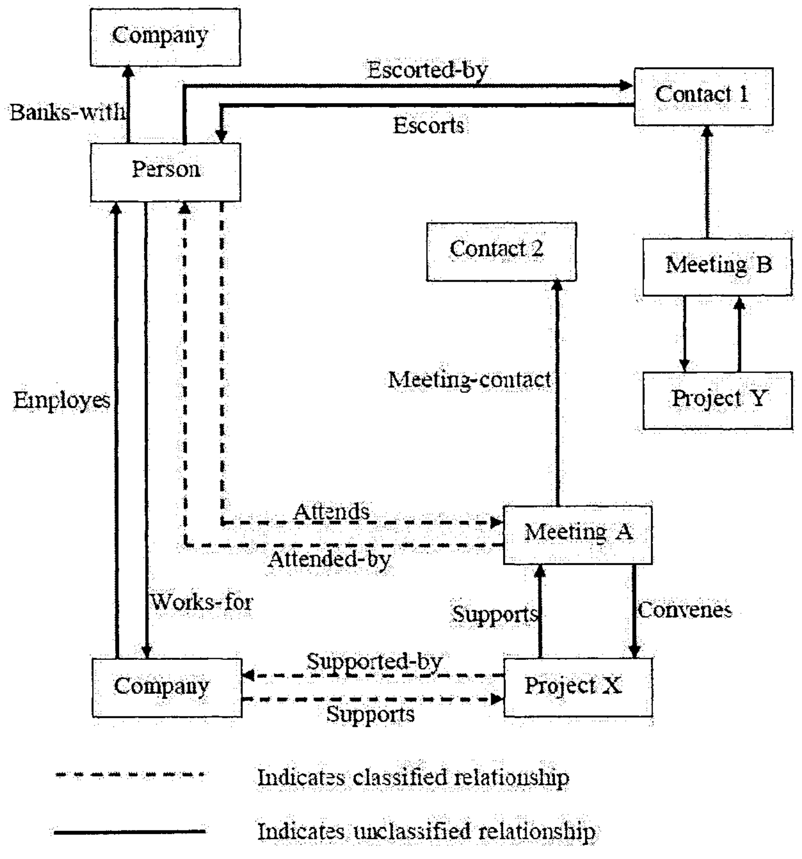


Figure 3.2: False Inferences

an inference engine constructed above this approach permits the inference engine to assist in knowledge engineering. One can, for example, begin with an inference engine consisting of just the semantic relationship graph for an application area. As inference paths are detected, they can be reviewed by the users and application domain experts to assess whether they constitute legitimate inference paths. If so, then there is no problem. If, however, they are judged to constitute false inference paths, then additional false inference detection rules can be added to the rule base to permit the inference engine to sift these false inference paths out of any future inference detection runs.

With this approach, inference paths are never missed, but as the inference engine is used, the intelligence of the inference controller can be enhanced to permit it to be more expert at differentiating between false inference paths and legitimate inference paths.

The problem then is to reduce the number of false inference paths. There are two approaches can be used: semantic specificity (SS) and rule based analysis (RBA), which will be described in turn.

3.1.2 Semantic Specificity (SS)

An example of how semantic specificity can reduce the number of false inference paths can be seen in the false inference example involving Project-Meeting-Contact-Person-Company(with which one banks) as an inference path to infer the Company associated with a particular classified project. This false inference arises because the semantic relationship graph has been stated in terms of the semantic entity Company, which includes a Defense Contractor type company supporting a project and a Banking type company providing banking services for a Person. Since the semantic relationship graph describes both of them as Companies, the relationship Project-Company, which is sensitive, can be inferred by finding the two paths from Project to Company(for which a Person works) and Company(with which a Person banks). One of them is a legitimate inference path and the other is a false inference path.

If Project-Company were presented as Project-Defense_Company, and the Person-Company(for which a person works) as Person-Defense_Company and Person-Banking _Company for the Person-Bank relationship, then the path Project-Meeting-Contact-Person-Banking_Company would not be found as an alternate path for Project-Defense_Company.

Of course, the price paid is that the Person-Company(for which the person works) relation is now expanded from a single node relationship between two nodes, to multiple Person-Company relationships, one for each Person-Company combination. Thus one would have Person-Insurance_Company, Person-Retail_Company, Person-Factory, etc. This can clearly get out of hand very rapidly. The solution to this is to recognize that the semantic relationship graph should exist within the context of a semantic

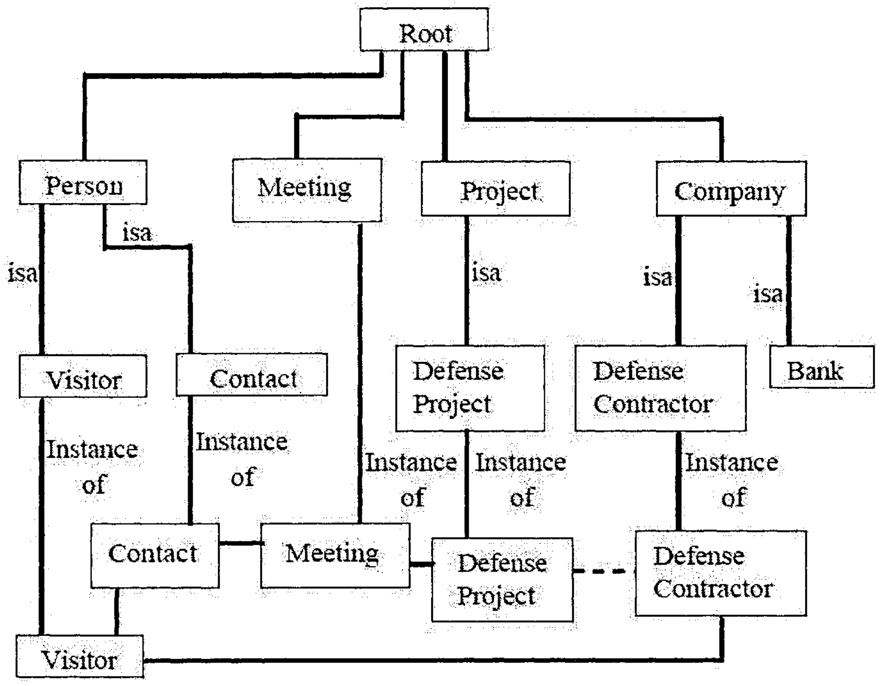


Figure 3.3: Semantic Relationship Graph as Instances of Semantic Net

net consisting of ISA, part-of, and instance hierarchies. The semantic net describes semantics of each entity that is part of the semantic relationship graph. Note that the semantic relationship graph represents how one set of entities is related to another, while a semantic net permits each semantic relationship graph node to be strongly typed by relating it to a node in the semantic net through the instance-of relationship. Figure 3.3 shows the semantic relationship graph for the project-company example as instances of a semantic net.

A node within the semantic net is called a Semantic-Designator node, since it designates the semantics of lower level nodes - those nodes which inherit its properties. Since a Semantic-Designator-node may be a superior type of some lower level Semantic-Designator-nodes, we talk of a Semantic-Designator-node designating a type set consisting of the Semantic-Designator-nodes and all instance nodes which are subtypes of the particular Semantic-Designator-node. Thus, in figure 3.3, Company designates the type set Company consisting of Defense_Companies and Banking_Companies, and all the instances of them.

Paths within the semantic relationship graph are created by piecing together binary relationships, such as Project-Meeting with other joinable binary relationships such as Meeting-Contact. Joinable binary relationships are those binary relationships which have nodes that are semantically equivalent and hence joinable.

Two semantic relationship graph nodes are joinable if their type sets intersect. The composite node takes on the type set of the intersection of the type sets of the components. Hence, this is similar to multiple inheritance, but only the common elements are inherited. If, for example, the joining of three nodes yields an empty set, then pairwise joins will be performed, with the result that three composite nodes could result (e.g. 1 intersect 3, 1 intersect 3, 2 intersect 3).

The strategy is to initially type a semantic relationship node as high in the semantic net as possible, so that its type set is as large as possible, yielding the maximum number of possible paths. As experience is gained with a particular semantic relationship graph, false inference paths may arise. One method is to constrain the type of certain semantic relationship graph nodes to eliminate these false paths. For example, the Company in the Company-Project binary relation could initially be specified as of type Company, which includes Defense_Contractor and Bank. When the false inference from the join of Company of Project with Bank of Person arises, one can re-specify the Company of Project to be of type Defense_Contractor. Now the type set of Defense_Contractor in Figure 3.3 includes only Defense_Contractor and its instances. One also needs to re-specify the Company for which a person works as of type Defense_Contractor and the Company with which a Person banks as of type Bank, to eliminate the false inference.

3.1.3 Rule Based Analysis (RBA)

Under the rule based analysis (RBA) method, the domain expert writes rules that are designed to detect paths that constitute false inferences and eliminate them from those returned to the user.

Figure 3.4 illustrates an example of a semantic relationship graph that is amenable to RBA. The senario is that Missions have Countries that they target. For classified missions, the targets are assumed to be classified. However, Missions are staffed by Person, and each Person speaks one or more Languages. These Languages are used in one or more Countries. The path to infer the Country which a Mission targets is thus Mission-Person-Language-Country.

While this semantic relationship graph will catch all of the legitimate inferences that can be made from this information, it will also generate numerous false inferences. For example, if the missions are staffed by personnel from the United States, then presumably they all speak English. Since English is spoken in the U.S.A., Canada, The United Kingdom, Australia, as well as various other countries, all missions will show each of these countries as potential destinations as well as the countries associated with the other languages spoken by mission personnel. This example shows that those false inferences can not be removed by the SS discussed previously, however,

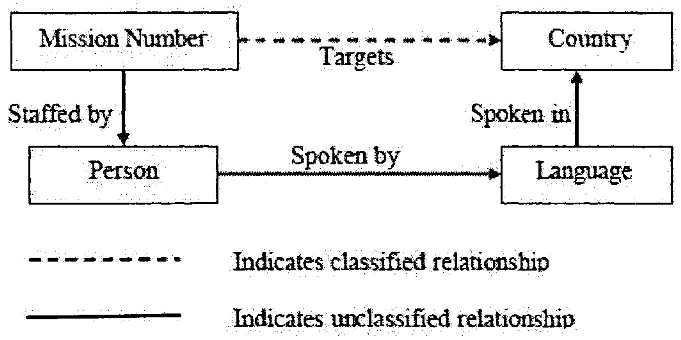


Figure 3.4: Mission Target Example Semantic Relationship Graph

many can be eliminated through RBA.

Consider some of the rules can be applied to this example to eliminate false inference paths. The first rule states that this inference path only applies to cases involving languages which are very good country designators. For example, English and Spanish are poor country designators since they are the native language in too many countries of the world. But Persian is a good country indicator since it is spoken primarily in Iran. So the rule base would be enhanced with rules indicating which countries are good country indicators. One may also want to add a rule that says 30% of the personnel assigned to a mission must speak the same language before this inference path is considered. The addition of this rule should be quite effective in trimming false inferences from the paths. The paths remaining should have a high potential for indicating targets for a mission.

3.1.4 Performance and Remarks

Path finding in a graph is $O(N^2)$, where N is the number of entity sets. Since for N entity sets, there are at most N^2 possible binary relationships among entities, that says checking all binary relations for second paths connecting them is $O(N^4)$. When a path is found, appropriate action must be taken to remove some piece of data in the path so that the path is broken and this particular second path is removed. The algorithm then needs to be run again to see if any second paths still remain. Assuming one node is eliminated each time that the algorithm is run for a given pair of entities, the algorithm will have to be run at most $N - 2$ times, making the total algorithm for a single pair of entities $O(N^3)$, and for all entities $O(N^5)$.

It is easy to see that when a legitimate inference path is identified (in other words, an inference channel is identified), at least one object in it should be unavailable to the generic users in order to protect the classified information. One common practice is to raise the objects' classification level so that the generic users cannot access them without extra authorization.

3.2 The Use of Conceptual Structures to Detect Inference

In last section, Hinke's work, which is based on the use of graphs for representing the applications and detects inferences by traversing alternate paths between two nodes in the graph, is discussed. While this technique enables simple inference to be detected via transitive property, it does not enable the detection of more complex inferences.

A method using conceptual structures to handle the inference problem is presented in [72] by Thuraisingham. Conceptual structures have many applications in artificial intelligence systems and natural-language processing. They have been used extensively for representing and reasoning about real-world knowledge. Multilevel conceptual structures are developed to represent multilevel applications and the reasoning strategies for them are used to detect security violations via inference during database design.

From Thuraisingham's point of view, to successfully handle inferences, it is not only important to be able to present the application semantics, but it is also essential that appropriate reasoning strategies be applied. Therefore conceptual structures, which are not only powerful representation schemes, but have a complete set of reasoning strategies, are investigated. The main focus is on the use of semantic nets. This is because semantic nets have been used extensively for a variety of data modelling, artificial intelligence, and natural language processing applications. However, to deal with a multilevel environment, the concept of multilevel semantic nets is developed and it is showed how multilevel information can be captured by such a representation. Then the complete set of reasoning strategies are used to detect security violations via inference.

While semantic nets are powerful for representing and reasoning about a variety of applications, it has been shown that they do not have the capabilities of first-order logic. As a result, several extensions to semantic nets have been proposed. One particular extension which is theoretically complete is the conceptual graph of Sowa [61], which will be briefly investigated later in this section. In particular, issues on developing multilevel conceptual graphs, and how reasoning techniques such as restriction, joining and simplifying could be applied to handle the inference problem are discussed .

3.2.1 Semantic Nets and the Inference Problem

1. Multilevel semantic nets

   A multilevel semantic net is a semantic net with nodes and links classified at different security level. Figure 3.5 shows some example multilevel semantic nets. We assume that there are only two security levels, Unclassified and Security. Note that the darkened shapes and lines are assumed to be Secret.

   Consider figure 3.5*a*. It states that all ships carry some weapons. In figure 3.5*a*, both the node and the link are Unclassified. That is, the fact that ships carry weapons can be seen by all. In figure 3.5*b*, the concepts SHIPS and WEAPONS can be seen by all, while the fact that ships carry weapons cannot be seen by Unclassified users since the link is classified Secret. In figure 3.5*c*, the Unclassified users know that something carries weapons, but they do not know that ships carry weapons. In figure 3.5*d*, the Unclassified users know that ships carry something, but they do not know what is carried by the ships. In figure 3.5*e*, the Unclassified users know that something is carried by something else, but they do not know anything about ships and weapons. In figure 3.5*f*, Unclassified users know only about ships. In figure 3.5*g*, Unclassified users know only about weapons. In figure 3.5*h*, nothing is visible to the Unclassified users.

   It needs to be determined whether all the links described in figure 3.5 should be permitted. For example, it may make sense to classify a link at a level which dominates the levels of the nodes associated with the link. That is, the level of the CARRY relationship must dominate the levels of SHIPS and WEAPONS.

   Figure 3.6 shows a more elaborate multilevel semantic net. The Unclassified interpretation of it is as follows: CHAMPION carries passengers. Its captain is Smith, who has 20 years' experience. The ship is located in the Mediterranean Sea on 16 June 1990. Its destination is Greece. The Secret interpretation is: CHAMPION carries SPARK which is an explosive. Its captain is Smith, who has battle management experience. The ship is located in the Mediterranean Sea on 16 June 1990. Its destination is Libya.

   We can see that certain information is polyinstantiated. Note that polyinstantiation occurs when users at different security levels have different views of the same "thing" in the real world. By "thing" we mean concept, entity, event, or any relationship. Cover stories result in polyinstantiation. Figure 3.6 illustrates how cover stories may be represented.

   In addition, a semantic net also has two standard links: ISA links and AKO links (See Figure 3.7). An ISA link is used to specify that a particular individual belongs to a particular group. Figure 3.7*a* shows an ISA link where CHAMPION is defined to be a particular shio. An AKO link defines a subset of a collection

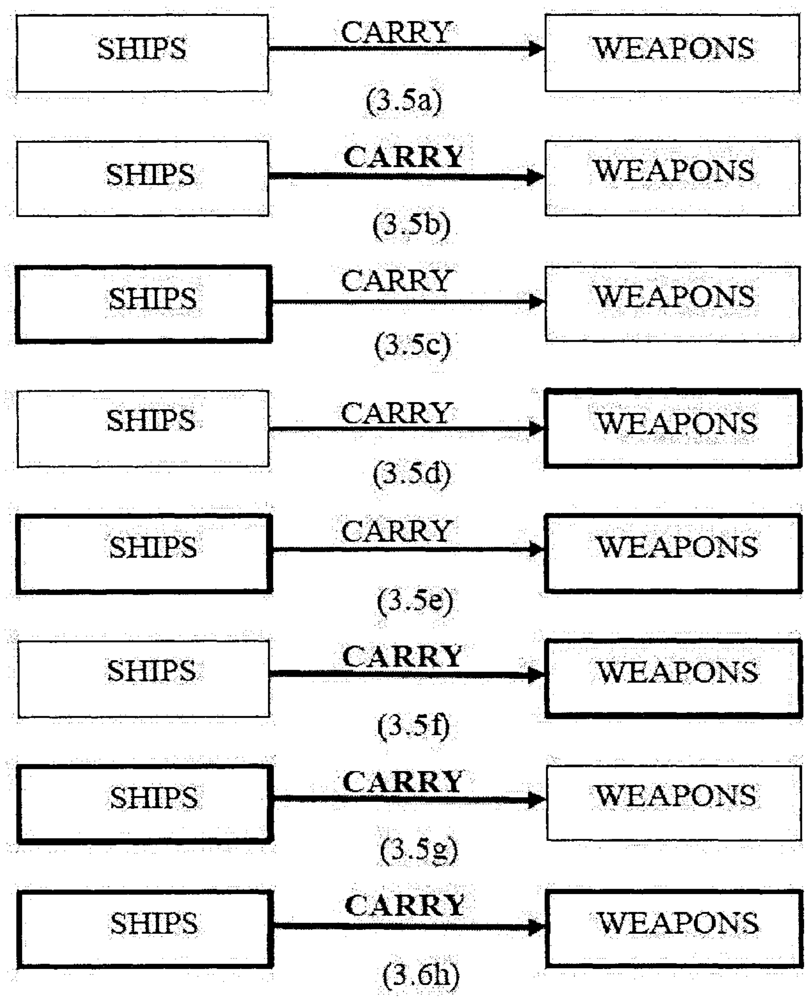


Figure 3.5: Multilevel Semantic Nets

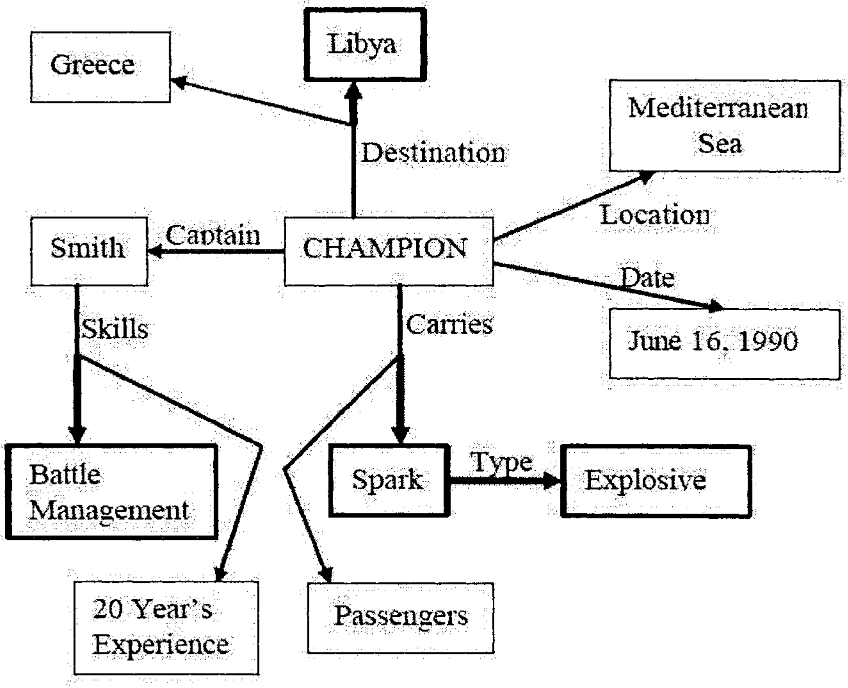


Figure 3.6: Complex Multilevel Semantic Net

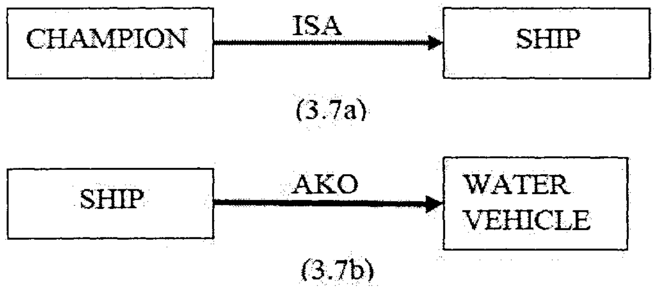


Figure 3.7: ISA, AKO Links

of individuals. Figure 3.7*b* defines the collection of ships to be a subset of a collection of water vehicles. Note that the AKO relationship is assigned to be Secret.

It does not make sense to classify CHAMPION at the Secret level and SHIP at the Unclassified level. This is because CHAMPION is an instantiation of SHIP. By classifying SHIP at the Secret level we are implicitly assuming that any ship must be classified at least at the Secret level. It should also be noted that it does not make sense to classify SHIP at the Unclassified level and WATER VEHICLE at the Secret level. This is because classifying SHIP at the Unclassified level implicitly assumes that a ship should be classified at least at the Unclassified level. Classifying WATER VEHICLE at the Secret level implicitly assumes that any water vehicle (including a ship) should be classified at least at the Secret level. This results in a conflict. Therefore, we enforce the following rules for consistency:

- Rule $A1$: If X ISA Y, then Level(X)$\geq$Level(Y).
- Rule $A2$: If X AKO Y, then Levle(X)$\geq$Level(Y).

2. Reasoning with multilevel semantic nets

   Most real-world applications deal with very large quantities of information. Therefore, capturing all of the information in a semantic net would make the net extremely complex. What we need is a minimal semantic net with a powerful set of reasoning strategies so that other information can be deduced. The information that is deduced is implicit information. For a multilevel application it should be ensured that the level of implicit information that can be deduced by a user at level L should be dominated by L.

   Some rules for deducing implicit information are the following:

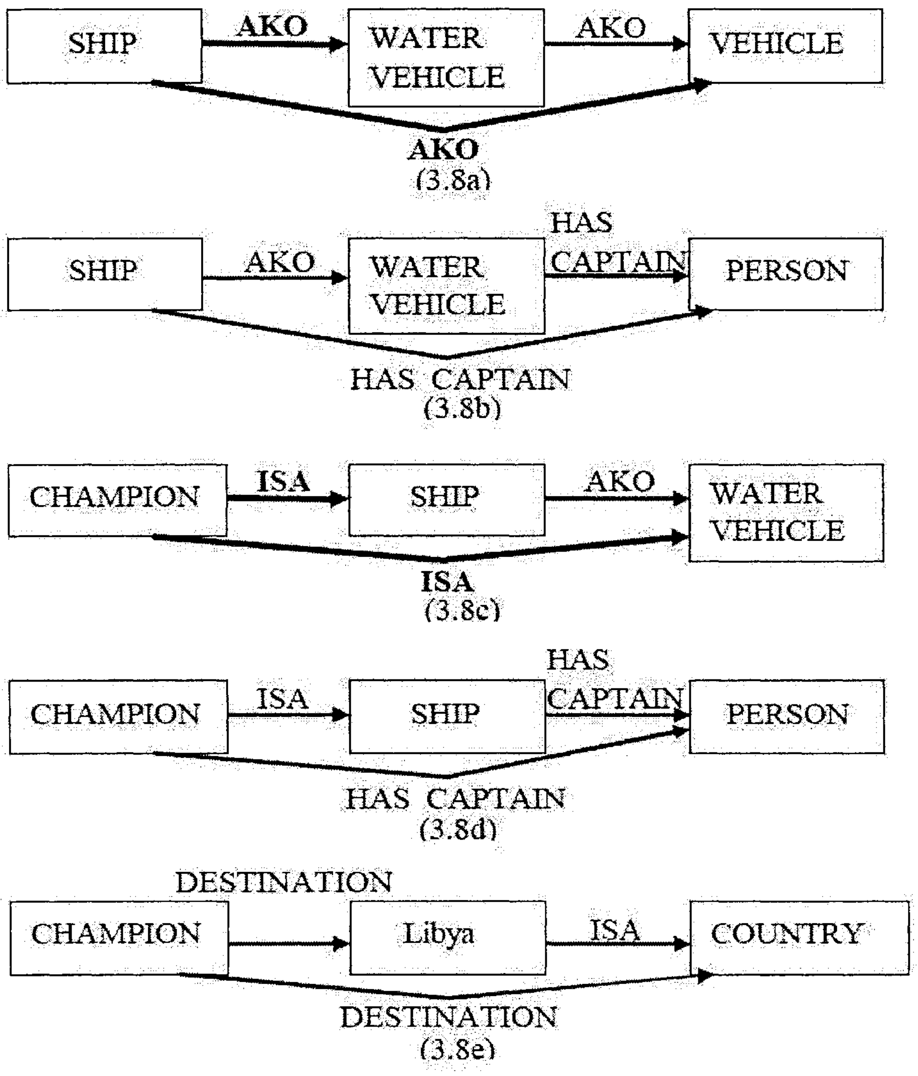


Figure 3.8: Sample Rules

- Rule *A*3: If X AKO Y and Y AKO Z then X AKO Z. The level of the AKO link from X to Z is the least upper bound of the levels of AKO links from X to Y and Y to Z.

  Figure 3.8*a* illustrates an example. The semantic net has SHIP AKO WATER-VEHICLE and WATER-VEHICLE AKO VEHICLE, and the AKO link from SHIP to WATER-VEHICLE is Secret. Then at the Secret level, one can conclude that SHIP AKO VEHICLE.

- Rule *A*4: If X AKO Y and Y has relationship R with Z, then X has relationship R with Z. The level of the relationship R that X has with Z is the least bound of the level of the AKO link from X to Y and the relationship R from Y to Z.

  Figure 3.8*b* illustrates an example. The semantic net has SHIP AKO WATER-VEHICLE and WATER-VEHICLE has captain PERSON. Then SHIP has captain PERSON.

- Rule *A*5: If X ISA Y and Y AKO Z, then X ISA Z. The level of the ISA link from X to Z is the least upper bound of the levels of the AKO link from Y to Z and the ISA link from X to Y.

  Figure 3.8*c* illustrates an example. CHAMPION ISA SHIP. This link is Secret. SHIP AKO WATER-VEHICLE. Therefore, there is a Secret ISA link from CHAMPION to WATER-VEHICLE.

- Rule *A*6: If X ISA Y and Y has a relationship R with Z, then X has relationship R with Z. The level of the relationship R that X has with Z is the least upper bound of the levels of the ISA link from X to Y and the relationship R from Y to Z.

  Figure 3.8*d* illustrates an example. The semantic net has CHAMPION ISA SHIP. SHIP has captain PERSON. Therefore, CHAMPION has captain PERSON.

- Rule *A*7: If X ISA Y and Z has relationship R with X, then Z has relationship R with Y. The level of the relationship R that Z has with Y is the least upper bound of the levels of the ISA link from X to Y and the relationship R from Z to X.

  Figure 3.8*e* illustrates an example. The semantic net has Libya ISA COUNTRY. The ship CHAMPION's destination is Libya. Therefore, the destination of CHAMPION is a country.

Conditional statements are of the form: A if B1 and B2 and ...Bn where B1, B2, ..., Bn are the antecedents and A is the consequent.

Consider the following conditional statement: CHAMPION's destination is

Libya, if it is located in the Mediterranean, and it carries SPARK, which is an explosive.

The conditional statement is represented by the auxiliary net shown in figure 3.9*a*. That is, the conditions are represented by dotted lines, and the conclusion is represented by solid lines. The transfer rule is applied in order to process conditional statements. The following is the transfer rule:

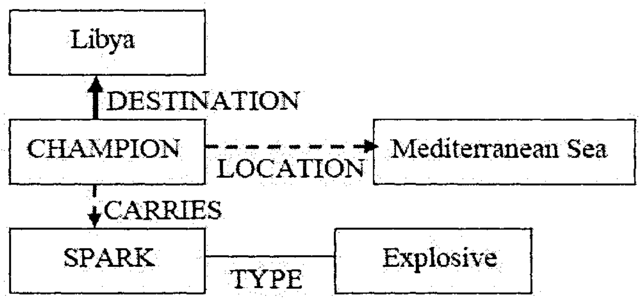
- If all the dotted lines in the auxiliary net are shown as solid lines in a main multilevel semantic net, and the level of each solid line of the main net dominated the level of the corresponding dotted line in the auxiliary net, then the solid line in the auxiliary net is drawn as a solid line in the main net. The security level of the line drawn is the least upper bound of the levels of all the lines in the auxiliary net and the levels of all the corresponding solid lines already in the main net.

Figure 3.9*b* shows that the dotted lines in the auxiliary net occur as solid lines in the multilevel semantic net. Figure 3.9*c* shows that the solid line in the auxiliary net is added to the multilevel semantic net at the appropriate security level.

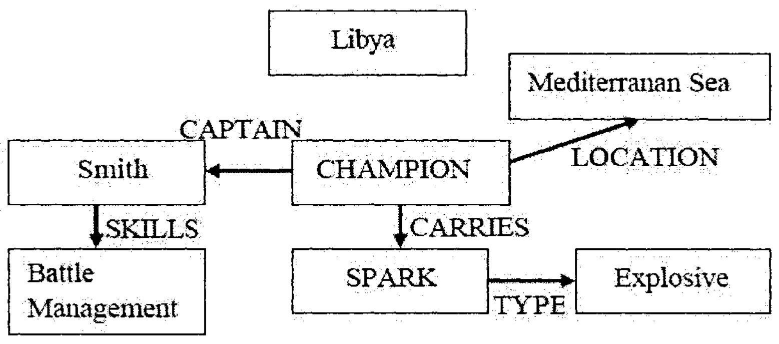
3. Enforcing security constraints

Security constraints are rules which assign security levels to the data. We represent security constraints by "constraint nets" (See Figure 3.10). A constraint net is a semantic net or an auxiliary semantic net which specifies only constraints. However, while semantic nets are used in general to represent application information, constraint semantic nets are used to represent the security constraints so that any security violations in the application can be detected. Similarly, while auxiliary semantic nets are used to derive implicit information, security constraints which are represented as auxiliary semantic nets are used to detect security violations. Therefore, we differentiate between ordinary auxiliary nets and constraint auxiliary nets. Figure 3.10*a* classifies the fact that CHAMPION carries something at the Secret level. Figure 3.10*b* shows a constraint which classifies the destination country of CHAMPION at the Secret level, if CHAMPION is located in the Mediterranean and it carries SPARK.

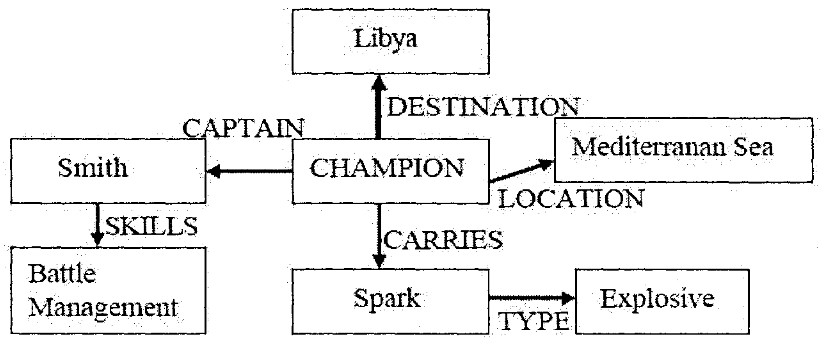
Security violations occur (either directly or indirectly) if the constraint net contradicts the multilevel semantic net which represents the application (either directly or indirectly). For example, the semantic net of Figure 3.11*a* violates both constraints of Figure 3.10 directly. Since in Figure 3.11*a*, the fact that CHAMPION carries something is not Secret, which violates the constraint of Figure 3.10*a*. Also in Figure 3.11*a*, it is unclassified that CHAMPION is located



(3.9a)

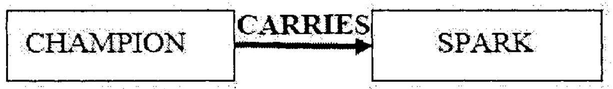


(3.9b)

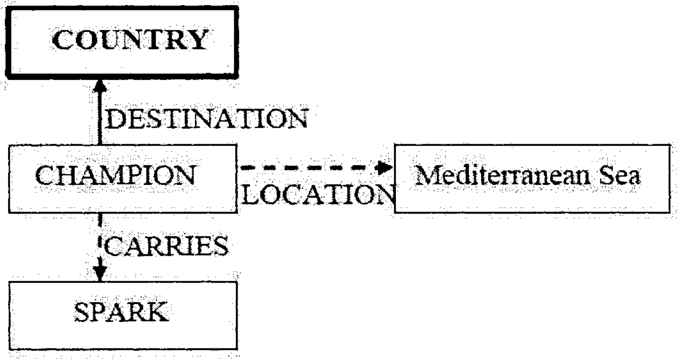


(3.9c)

Figure 3.9: Application of Transfer Rule



(3.10a)



(3.10b)

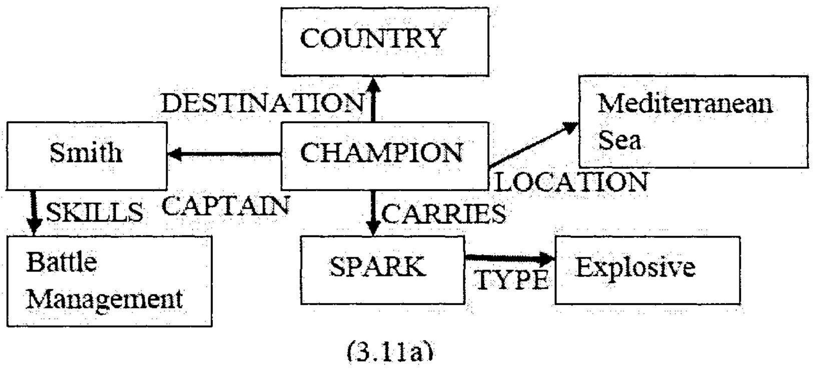
Figure 3.10: Representing Security Constraints

in the Mediterranean and carries SPARK. This directly violates the constraint of Figure 3.10*b*. Constraints can be violated indirectly when the implicit information that can be inferred contradicts the security constraints. Figure 3.11*b* shows how the security constraint of Figure 3.10*b* is violated indirectly. Here, CHAMPION carries SPARK and it is located in the Mediterranean. Its destination country Libya is Unclassified. Since Libya is a country, by rule *A*7, the destination Libya of CHAMPION must be Secret. Therefore, the constraint of Figure 3.10*b* is violated indirectly.

4. Universal and existential conditionals

   Constraint nets can also be used to represent universal and existential conditionals. There are two rules for constraint nets for universal and existential conditionals.

   - Rule *A*9: Let A and B be subnets of two multilevel semantic nets. Note that a subnet is any subnet (nodes and links) of a semantic net. Subnets are also called vectors. A matches B if the following are satisfied:
     (a) The links are labelled the same and have the same security levels.
     (b) If a node in A is labelled with a constant, then the corresponding node in B is labelled with the same term, and the two nodes have the same



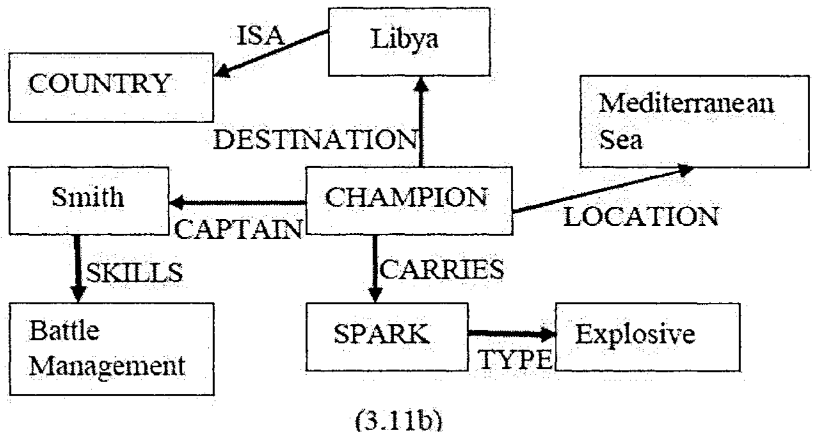


Figure 3.11: Security Constraint Violation

security levels.

(c) If two nodes in A are labelled with the same variable, then the corresponding two nodes in B are labelled with the same constant. Further, the security levels of the corresponding nodes in the two nets are the same.

- Rule *A*10: If a vector A matches with vector B, then for any variable x in A, the corresponding node in B which matches with x is called the binding of x.

3.2.2 Conceptual Graphs and the Inference Problem

It can be seen that semantic nets are very useful for representing a multilevel application and applying reasoning rules. However, they do not have the full power of first-order logic. Therefore, the conceptual graph, which has the full power of first-order logic, is introduced to handle the inference problem.

A conceptual graph is a graph-based notation developed by Sowa[61] for representing and reasoning about knowledge. It evolved from other graph-based representation schemes, such as semantic nets. The basic constructs of conceptual graphs are the notions of concepts and relationships between concepts. A concept could be any entity, action, or event. Concepts are represented as nodes and relationships as links. Each concept is an instantiation of a concept type. The concept types are ordered by the relation "<". For example, the concept types PERSON and MAN are ordered by MAN<PERSON. This means that every man is also a person. An instantiation of the type MAN is "John". The relationships are also instantiations of RELATIONSHIP types. For example, if AGENT is a relationship type, it can be instantiated as John is the agent of drinking. This means that John is drinking.

1. Representational issues

   A multilevel conceptual graph is a conceptual graph with security levels assigned to concept types, concepts, relationship types and relationships. We enforce the following rules for consistency.

   - Rule *B*1: If X is an instantiation of concept type C, then Level(X)$\geq$ Level(C).
   - Rule *B*2: If Y is an instantiation of a relationship type R, then Level(Y)$\geq$ Level(R).
   - Rule *B*3: If C1 and C2 are concept types and if C1<C2, the Level(C1)$\geq$ Level(C2).

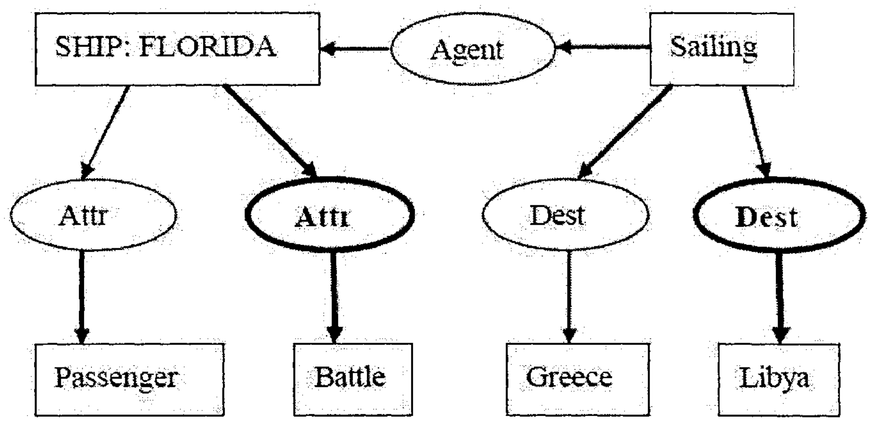


Figure 3.12: Multilevel Conceptual Graph

Figure 3.12, 3.13, 3.14, 3.15 show how the conceptual graphs represent applications. Figure 3.12 shows how to present different information at the Secret level and the Unclassified level. Figure 3.13 shows the negative information "It is not the case that there is a passenger ship FLORIDA which is going to Greece". Figure 3.14 shows the use of PROPOSITION type, representing the assertion "Enemy is going to attack on 6/16/90". Figure 3.15 expresses a universal statement "Every ship is managed by a captain".

2. Reasoning strategies

   Several reasoning strategies have been proposed for conceptual graphs. Three deduction rules are:

   - Restriction: This rule takes a graph and replaces any of its concept nodes by a subtype concept node or by a special instance of the concept. For example, the concept type PERSON may be restricted to MAN or an instance such as John.
   - Joining: This rule takes two graphs with a common concept (or concept type) and joins them over this concept.
   - Simplifying: This rule removes duplicate relationships between two concepts (or concept types) which could possibly arise after a join operation.

   The following security properties should be enforced whenever a deduction rule is applied:

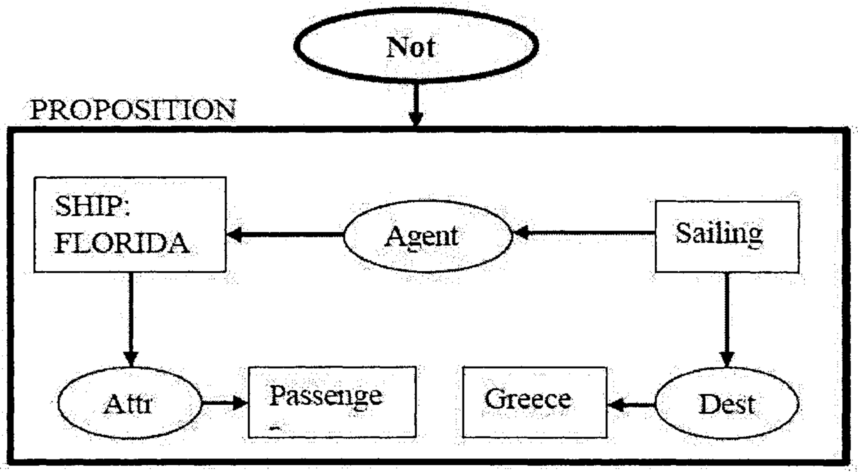


Figure 3.13: Negative Statement

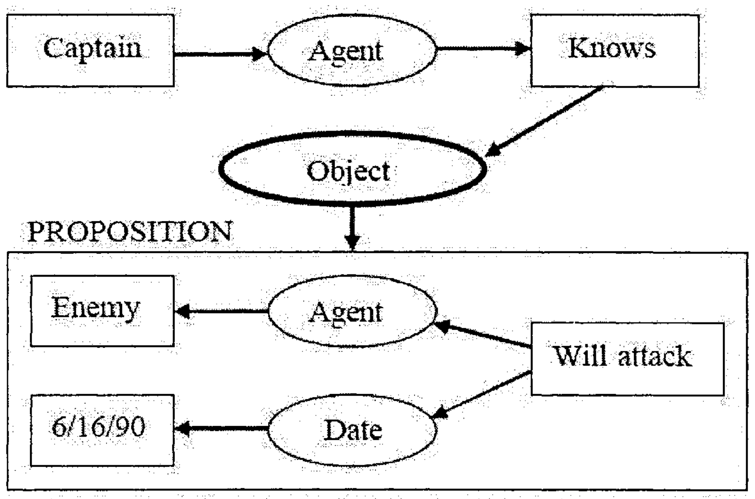


Figure 3.14: Complex Assertions

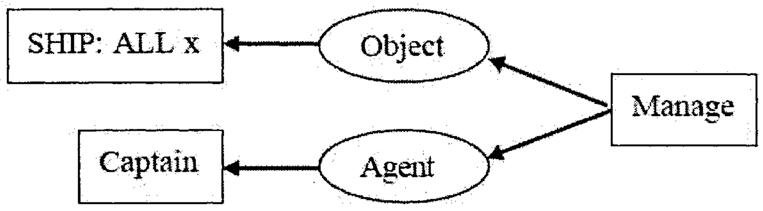


Figure 3.15: Universal Statement

- Let C1 be a conceptual graph and C2 be the result of applying the restriction rule to C1. Any security level of the restricted concept (or concept type) in C2 must dominate the level of the original concept in C1.
- Let C1 and C2 be conceptual graphs which are joined to form C3. Any concept, concept type, or relationship in C3 must have the security level of the corresponding concept, concept type, or relationship in C1 or C2.
- Let C1 be a conceptual graph simplified to C2. Let R1 and R2 be identical relationships between the same two concepts in C1. The security level of this relationship in the simplified graph C2 is the greater lower bound of the levels of R1 and R2.

3.2.3 Remarks

Comparing with the method discussed in last section, this method is nicer because it develops multilevel conceptual structures to capture the semantics of multilevel applications. In addition, it has a set of reasoning strategies applied to detect security violation. The SSO(System Security Officer) could use these techniques to manually analyze the multilevel application and detect possible security violations.

3.3 Data Level Inference Detection in Database Systems

In last two sections (Section 3.1 and 3.2), we introduced techniques employing functional dependencies in database schema to detect inferences. It has been noticed that analyzing the data stored in the database also help to detect more inferences. In [79], an inference detection technique based on data level is presented.

In this method, five inference rules used to infer data are identified: ‘subsume’, ‘unique characteristic’, ‘overlapping’, ‘complementary’ and ‘functional dependency’. The goal is to detect if a user can indirectly access data using two or more queries. In particular, the system determines if the user can infer the return tuples from different queries corresponding to the same tuple in the database.

3.3.1 Overview of the Inference Rules

The intuition behind the five inference rules are as follows. The result of each user query is a set of return tuples. The user cannot identify each return tuple unless the primary key of the tuple is also returned. However, a certain group of attribute values of a tuple may uniquely identify the tuple. The unique identification rule handles this situation. Another way to identify a return tuple is to compare it with other return tuples that have already been identified.

There are two possible relationships between two sets of return tuples. One possibility is that for each return tuple t_1 of a query, there is a return tuple t_2 of the other query, such that t_1 and t_2 correspond to the same tuple in the database. The subsume inference rule handles this case. Another possibility is that only some return tuples of a query correspond to some return tuples of another query. The overlapping inference rule identifies the corresponding return tuple that are common to both queries. The complementary inference rule identifies the corresponding return tuples by taking the "difference" between two sets of return tuples. The functional dependency inference rule is introduced to simulate the schema level inference detection scheme.

Once the corresponding return tuples between two queries are identified, the user can generate inferred queries without directly issuing them to the database. For example, the user can infer a new query with returning tuple "common" to both queries, or a new query that returns tuples from one query but not from another query. The user can also combine several queries into a single query. The effect of applying inference rules to unions of queries will be discussed later. Essentially, the five inference rules cover the set intersection, difference and union relationships between two sets of return tuples.

When the user issues a query, the inference detection system compare it with previously issued queries and inferred queries, and applies inference rules when appropriate. An occurrence of inference will result in either the modification of the existing queries, or the generation of new inferred queries. These may trigger further applications of the inference rules. Hence, the inference rules are applied repeatedly until no new inference occurs. This is a terminating process as the number of inferences that can occur is bounded by the size of the database. When two users are suspected of cooperating in performing inference, the inference detection system can be run against their combined set of queries.

3.3.2 Preliminaries and Notation

Consider inference detection in a relation database with a single table (A database with multiple tables can be transformed into a universal relation [33]). It is assumed that the only way the user can learn about the data in the database is by issuing queries to it.

Let A_i denote an attribute in the table, and a_i denote an attribute value from the domain of A_i. $t[A_i]$ denotes the attribute value of a single tuple t over the attribute A_i. A query is represented by a 2-tuple: *(attribute-set, selection-criterion)*, where *attribute-set* is the set of attributes projected by the query, and *selection-criterion* is the logical expression that is satisfied by each return tuple of the query. No aggregation function (for example, maximum and average) is allowed in the attribute-set. In general, Q_i refers to the query $\{AS_i; SC_i\}$. $|Q_i|$ denotes the set of return tuples of Q_i. For each query Q_i, AS_i is expanded with an attribute A_i when "$A_i = a_i$" appears in SC_i as a conjunct. A query Q is a "partial query" if the user can determine $|Q|$, but not all return tuples of Q. "$\cap$", "$\cup$", and "$\backslash$" stand for the set intersection, union and difference operation respectively.

Definition 3.3.1 *A tuple t over a set of attributes AS "satisfies" a logical expression E if E is evaluated to true when each occurrence of A_i in E is instantiated with $t[A_i]$, for all A_i in AS. t "contradicts" with E if E is evaluated to false.*

For example, the tuple $(35, 60K)$ that is projected over $(Age, Salary)$ satisfies $E = (Age > 30 \land Salary < 70K)$;while the tuple (25,50K) projected over the same set of attributes contradicts with E. The tuple $(45K, Manager)$ projected over $(Salary, Job)$ neither satisfies nor contradicts E. This is because after the instantiation, E becomes $(Salary < 70K)$ whose truth is undermined.

Definition 3.3.2 *Given two queries, Q_1 and Q_2, we say that Q_1 is "subsumed" by Q_2, denoted as $Q_1 \sqsubset Q_2$, iff*

1. *$SC_1 \rightarrow SC_2$; or*

2. *for each tuple t_1 in Q_1, t_1 satisfies SC_2.*

where $\rightarrow$ is the logical implication. $\sqsubset$ is a reflexive, anti-symmetric, and transitive relation. A return tuple t_1 "relates" to another return tuple t_2 if then two tuples are selected from the same tuple in the database. Hence, $Q_1 \sqsubset Q_2$ implies that for each return tuple t_1 of Q_1, there is a return tuple t_2 of Q_2, such that t_1 relates to t_2.

When evaluating a logical implication, we need to consider the integrity constraints that hold in the database. Consider the following implication,
$(age > 18 \land age < 35) \rightarrow (age > 20 \land age < 50)$,

which is false. Suppose the youngest person in the database is 22 years old. By adding this constraint to both sides of the implication, it becomes,

$((age > 18 \wedge age < 35) \wedge (age \geq 22)) \rightarrow ((age > 20 \wedge age < 50) \wedge (age \geq 22)) \equiv (age \geq 22 \wedge age < 35) \rightarrow (age \geq 22 \wedge age < 50)$,

which is true.

Definition 3.3.3 *A return tuple t_1 of Q_1 is "indistinguishable" from a return tuple t_2 of Q_2 iff*

1. *for all A_i in $(AS_1 \cap AS_2)$, $t_1[A_i] = t_2[A_i]$;*
2. *t_1 does not contradict with SC_2; and*
3. *t_2 does not contradict with SC_1.*

t_1 is "distinguishable" from t_2 if t_1 is not distinguishable from t_2.

Intuitively, t_1 is indistinguishable from t_2 if it is not possible to conclude that t_1 and t_2 are selected from two different tuples in the database. Two tuples that relate to each other are indistinguishable from each other, while two tuples that are indistinguishable from each other does not imply that they relate to each other.

3.3.3 Inference Rules

As mentioned above, five inference rules are summarized. They are illustrated by using the sample database as shown in Table 3.1. The user is allowed to access all data in the database. However, it is suspicious if the user can infer the salaries of employees. We assume that the security policy is to determine if the user can infer the associations between *Name* and *Salary*. In general, the policy can specify detecting inferences of any association among the attributes. Unless otherwise stated, all queries appear in the inference rules are not partial queries.

Name	Job	Age	Salary	Department	Office
Alice	Manager	35	60K	Marketing	2nd Floor
Bob	Secretary	35	45K	Marketing	2nd Floor
Charles	Secretary	40	40K	Production	1st Floor
Denise	Manager	45	65K	Sales	2nd Floor

Table 3.1: Sample Database

1. **Subsume inference** Given two queries Q_1 and Q_2, such that $Q_1 \sqsubset Q_2$.

- **SI1** If there is an attribute A in $(AS_2 \backslash AS_1)$, such that all return tuples of Q_2 take the same attribute value a over A, then for each return tuple t_1 of Q_1. $t_1[A] = a$. Q_1 may be a partial query.
- **SI2** If there is a return tuple t_1 of Q_1 that is indistinguishable from one and only one return tuple t_2 of Q_2, then t_1 relates to t_2. Q_1 may be a partial query.
- **SI3** Let S be the set of return tuple of Q_2 that are distinguishable from the return tuples of Q_1. If $|S| = (|Q_2| - |Q_1|)$, then two inferred queries are generated: $(AS_2; SC_2 \wedge \urcorner SC_1)$ with S as the set of return tuples, and $(AS_2; SC_2 \wedge SC_1)$ with $(Q_2 \setminus S)$ as the set of return tuples. If $|S| < (|Q_2| - |Q_1|)$, then an inferred partial query is generated: $(AS_2; SC_2 \wedge \urcorner SC_1)$ with S as the partial set of return tuples.

$Q_1 \sqsubset Q_2$ implies that for each return tuple t_1 of Q_1, there is a return tuple t_2 of Q_2, such that t_1 relates to t_2. $SI1$ says that when all return tuples of Q_2 share a common attribute value, say a, over an attribute A, the user can infer that each return tuple of Q_1 also takes the attribute value a over the attribute A. For example, consider the following two queries:

$Q_1 = (Age; Name = \text{"}Alice\text{"})$
$Q_2 = (Department; Age < 40)$

Q_1 returns a single tuple (35) which says that Alice is 35 years old. Q_2 returns two tuples ($Marketing$) and ($Marketing$) which shows that all employees at the age less than 40 work in the Marketing department. By $SI1$, Alice works in the Marketing department.

$SI2$ says that if t_2 is the only return tuple of Q_2 that is indistinguishable from a return t_1 of Q_1, then t_1 relates to t_2. Consider the following two queries:

$Q_3 = (Age; Name = \text{"}Charles\text{"})$
$Q_4 = (Age, Salary; Age \geq 40)$

Q_3 returns a single tuple $t_3 = (40)$ which says that Charles is 40 years old. Q_4 returns two tuples $(40, 40K)$ and $(45, 65K)$ which says that there are only two employees who are at the age greater than or equal to 40. As $Q_3 \sqsubset Q_4$ and $(40, 40K)$ is the only return tuple of Q_4 that is indistinguishable from t_3, by $SI2$, Charles earns $40K$.

$SI3$ says that if a user identifies all the return tuples of Q_2 that relate to the return tuples of Q_1, then the user can infer these two queries: $(AS_2; SC_1 \wedge SC_2)$ which includes return tuples of Q_2 that relate to the return tuples of Q_1, and $(AS_2; SC_2 \wedge \urcorner SC_1)$ which includes return tuples of Q_2 that do not relate to the return tuples of Q_1. Continue from the above example on Q_3 and Q_4, after the application of $SI2$, the following two inferred queries are generated:

$Q_{21} = (Age, Salary; Name = \text{“}Charles\text{”} \wedge Age \geq 40)$
$Q_{22} = (Age, Salary; Name \neq \text{“}Charles\text{”} \wedge Age \geq 40)$

Q_{21} returns a single tuple $(40, 40K)$, and Q_{22} returns a single tuple $(45, 65K)$. The two inferred queries together contains more information than Q_2. For example, Q_{22} says that the employee who is at the age of 45 and earns $65K$ must be someone other than Charles.

2. **Unique characteristic inference**

   Definition 3.3.4 *A logical expression E is a unique characteristic of a tuple t iff t is the only tuple in the database that satisfies E.*

   For example, if Alice is the only manager at the age of 35, the $(Job = \text{“}Manager\text{”} \wedge Age = 35)$ is the unique characteristic of Alice in the database.

   Given a tuple t_1 with unique characteristic C_1 in a database D, and another tuple t_2 with unique characteristic C_2 in D. If $C_1 \rightarrow C_2$, $C_2 \rightarrow C_1$, or $C_1 \leftrightarrow C_2$, then t_1 relates to t_2 in D.

   For example, the query

   $(Salary; Job = \text{“}Manager\text{”} \wedge Age \leq 40)$

   returns a single tuple $(60K)$. This query together with the above unique characteristic of Alice implies Alice earns $60K$. Unique characteristic inference is a special case of the subsume inference. Suppose $(AS_1; UC_1)$ returns a single tuple t_1, and $(AS_2; UC_2)$ returns a single tuple t_2. Then, UC_1 is the unique characteristic of t_1, and UC_2 is the unique characteristic of t_2. If $UC_1 \rightarrow UC_2$, $UC_2 \rightarrow UC_1$, or $UC_1 \leftrightarrow UC_2$ holds, then by $SI2$, t_1 relates to t_2.

   If all inferred queries are identified, unique characteristics are determined as follows,

   - if Q_i returns all but one tuple t in the database, then the unique characteristic of t is $(\neg SC_i)$
   - if Q_i and Q_j have only one overlapping return tuple t, then t has the unique characteristic $(SC_i \wedge SC_j)$.
   - if Q_i returns one more tuple t than Q_j, then the unique characteristic of t is $(SC_i \wedge \neg SC_j)$.

   where both Q_i and Q_j are not partial queries.

3. **Overlapping inference** Given n queries $Q_1, \ldots, Q_n$, where $n >= 3$.

- **OI1** Let S be the set of return tuples of Q_2 that are indistinguishable from the return tuples of Q_3. If $Q_1 \sqsubset Q_2$, $Q_1 \sqsubset Q_3$, $|S| = |Q_1|$, and t_2 is the only return tuple of Q_2 that is indistinguishable from *a* return tuple t_3 of Q_3, then t_2 relates to t_3. Q_1 may be a partial query.
- **OI2** Let $QS = Q_2, \ldots, Q_n$. Suppose for each Q_i in QS, $Q_i \sqsubset Q_1$, and the return tuples of Q_i are indistinguishable from the return tuples of at most one other query in QS. Also, the total number of indistinguishable tuples in all queries in QS is equal to $(2 \times (|Q_2| + \ldots + |Q_n| - |Q_1|))$. For any two queries Q_j and Q_k in QS, if t_j is the only return tuple of Q_j that is indistinguishable from a return tuple t_k of Q_k. Q_1 may be a partial query.
- When all relating tuples between Q_i and Q_j are identified, three inferred queries are generated (possibly partial): $(AS_i; SC_i \wedge \neg SC_j)$, $(AS_j; SC_j \wedge \neg SC_i)$, and $(AS_i \cap AS_j; SC_i \wedge SC_j)$.

4. **Complementary inference** Suppose there are four queries, Q_1, Q_2, Q_3 and Q_4, where $Q_1 \sqsubset Q_2$, and $Q_3 \sqsubset Q_4$. Also, suppose that the return tuples of Q_1 that relate to the return tuples of Q_3 are identified (for example, using the overlapping inference rule), and similarly for those between Q_2 and Q_4. If one of the following three conditions holds:

   - for each return tuple t_1 of Q_1 that does not relate to any return tuple of Q_3, t_1 is distinguishable from all return tuples of Q_4
   - $Q_4 \sqsubset Q_3$
   - $|Q_3| = |Q_4|$

   then $Q_1' \sqsubset Q_2'$, where $Q_1' = (AS_1; SC_1 \wedge \neg SC_3)$, and $Q_2' = (AS_2; SC_2 \wedge \neg SC_4)$.

5. **Functional dependency inference** Suppose that attribute A_1 functional determines attribute A_2, and there exists a tuple t, such that $t[A_1] = a_1$ and $t[A_2] = a_2$. If there is a tuple t_i, such that $t_i[A_1] = a_1$, then $t_i[A_2] = a_2$. The same applies when A_1 or A_2 is a composite attribute (i.e, a group of attributes).

3.3.4 Inference with Union Queries

Union queries is a special case when handling the inference problem. Consider the following three queries,

$Q_1 = (Job; Age < 50 \wedge Age > 40)$
$Q_2 = (Job; Age > 45 \wedge Age < 60)$
$Q_3 = (Job; Age > 30 \wedge Age \leq 45)$

since the following implication holds.

$(Age < 50 \wedge Age > 40) \rightarrow ((Age > 45 \wedge Age < 60) \vee (Age > 30 \wedge Age \leq 45))$, $Q_1 \sqsubset (Q_2 \cup Q_3)$ holds.

The inference rules can still be applied by treating $(Q_2 \cup Q_3)$ as a single user query. We call such a union of queries a "union query". In contrast, a user query is called a "simple query". If Q_u is a union query that consists $Q_i, \ldots, Q_j$, then $AS_u = (AS_i \cap \ldots \cap AS_j)$, and $SC_u = (SC_i \vee \ldots \vee SC_j)$. The applications of the unique characteristic and functional dependency inference rules on a union query are the same as their applications on the simple queries of the union query. Hence, we only consider the applications of the subsume, overlapping and complementary inference rules on union queries.

Consider the applications of the subsume inference rule on union queries. Suppose $(Q_2 \cup Q_3) \sqsubset Q_1$. This implies that $Q_2 \sqsubset Q_1$ and $Q_3 \sqsubset Q_1$. If the subsume inference rule is applicable due to $(Q_2 \cup Q_3) \sqsubset Q_1$, then it is also applicable due to $Q_2 \sqsubset Q_1$ and $Q_3 \sqsubset Q_1$. Hence, we do not need to consider the application of the subsume inference rule when the union query occurs on the left hand side of a "$\sqsubset$" relation. Now suppose $Q_1 \sqsubset Q_u$, where Q_u is a union query. The application of $SI1$ on union queries is the same as when only simply queries are involved. To apply $SI2$, the user must have identified all the overlapping tuples among the simple query of Q_u that correspond to the return tuples of Q_1. The subsume inference rule can still be applied when the simple queries of Q_u have no common projected attribute.

Consider the applications of the overlapping inference rule on union queries. Firstly, consider the application of $OI1$. If $Q_u \sqsubset Q_1$ is involved, $|Q_u|$ must be known to the user. If $Q_1 \sqsubset Q_u$ is involved, the user must has identified all the overlapping tuples among the simple queries of Q_u that relate to the return tuples of Q_1. Now consider the application of $OI2$. If $Q_u \sqsubset Q_1$ is involved, the user must has identified all the overlapping tuples among the simple queries of Q_u that relate to the return tuples of Q_1. If $Q_1 \sqsubset Q_u$ is involved, $|Q_u|$ must be known to the user. In either case, the attribute set of the union query cannot be empty.

To apply the complementary inference rule on union queries, the overlapping tuples of the simple queries in the union query must have been identified. Also the attribute set of the union query cannot be empty.

3.3.5 Performance and Remarks

This method identifies five inference rules: subsume, unique characteristic, overlapping, complementary, and functional dependency. These rules are sound but they are not necessarily complete. A prototype of the inference detection system using Perl on a Sun SPARC 20 workstation. The preliminary results show that the system on average takes seconds to process a query for a database of thousands of records.

Although in theory detecting inferences at data level is an NP-hard problem, in practice, there are cases where the use of such approach is practical. In particular, the case when there is a limited amount of overlapping among the return tuples of the queries can be solved efficiently.

3.4 The Application of Decision Trees to the Inference Problem

In methods mentioned above, when an attribute is removed from the dataset, all the values of that attribute are hidden from the generic users. If only some values of the sensitive attribute are classified, and consequently should be hidden for the sake of security, while the others should be released for the reason of performance and functionality, then how can the classified values be protected from unauthorized disclosure? The inference problem exists in this case because the relation between attributes is exposed to generic users due to the exposure of some values in the sensitive attribute. Generic user can conclude rules from the available dataset, and then infer the missing data by applying those rules. Therefore, in most cases not only the sensitive values, but some non-sensitive values should be trimmed off from the released dataset in order to prevent the generic users from forming rules. It can be easily seen that there is a trade-off between functionality and security, which means more data should be released as long as the classified data is well protected. In [10], a paradigm for dealing with the inference problem in this situation is presented.

3.4.1 Trade-off between Functionality and Security

In this section, an example is quoted from [10] to explained more detail the trade-off between functionality and security of a database. High represents users of higher security classifications and the data set available for them, and Low, the users of lower security classification and the modified data set available for them (see Table 3.2, Table 3.3). When High wishes to downgrade a set of data to Low, it may be necessary to trim the set, where a question mark represents a missing value.

It is necessary for High to downgrade to Low the first eight rows in their entirety, and the ninth row with the result missing. Low uses only the first eight rows of its database to form its rules because that they are the only complete rows (tuples). Rows that are downgraded in their entirety are referred as the *base set*, see Table 3.4.

Can High assume that the information it tried to keep hidden from Low is still hidden? If Low if stupid, then this is true. However, if Low analyzes the base set, Low will find out that every blonde who did not use lotion got burned. Since Tony is a blonde who did not use lotion, Low knows that Tony got burned. We formalize this rule as

Name	Hair	Height	Weight	Lotion	Result
Hillary	blonde	average	light	no	burned
Janet	blonde	tall	average	yes	no
Bill	brown	short	average	yes	no
Tipper	blonde	short	average	no	burned
Newt	red	average	heavy	no	burned
Ken	brown	tall	heavy	no	no
Al	brown	average	heavy	no	no
Paula	blonde	short	light	yes	no
Tony	blonde	average	heavy	no	burned

Table 3.2: High Database

Name	Hair	Height	Weight	Lotion	Result
Hillary	blonde	average	light	no	burned
Janet	blonde	tall	average	yes	no
Bill	brown	short	average	yes	no
Tipper	blonde	short	average	no	burned
Newt	red	average	heavy	no	burned
Ken	brown	tall	heavy	no	no
Al	brown	average	heavy	no	no
Paula	blonde	short	light	yes	no
Tony	blonde	average	heavy	no	?

Table 3.3: Low Database = Downgrade

Name	Hair	Height	Weight	Lotion	Result
Hillary	blonde	average	light	no	burned
Janet	blonde	tall	average	yes	no
Bill	brown	short	average	yes	no
Tipper	blonde	short	average	no	burned
Newt	red	average	heavy	no	burned
Ken	brown	tall	heavy	no	no
Al	brown	average	heavy	no	no
Paula	blonde	short	light	yes	no

Table 3.4: Base Set

Name	Hair	Height	Weight	Lotion	Result
Hillary	blonde	average	light	no	burned
Janet	blonde	tall	average	yes	no
Bill	brown	short	average	yes	no
Tipper	blonde	short	average	?	burned
Newt	red	average	heavy	no	burned
Ken	brown	tall	heavy	no	no
Al	brown	average	heavy	no	no
Paula	blonde	short	light	yes	no
Tony	blonde	average	heavy	no	?

Table 3.5: Low Database = Reduced Downgrade

$$(hair = blonde) \wedge (lotion = no) \Longrightarrow (result = burned)$$

Now how can High prevent this? One option is not to downgrade any information, but this is a little bit overkill. Instead, High should decide what not to downgrade based on the rules that it thinks Low can infer, and on the importance of the information that Low should receive. If the information is of trivial value, it might also send incorrect data to Low (only for some attribute values) to impinge upon Low's ability to infer rules and therefore infer High information. High could decide not to downgrade both *Hillary_Lotion* = *no* and *Tipper_Lotion* = *no*. Then Low could not determine the above rule and the result concerning Tony would not be apparent to Low. What is the impact of not downgrading the information about Hillary's and Tipper's lotion? If for functionality and performance reasons, Low must have this information, then there is a problem. If the importance of the information about Hillary's and Tipper's lotion is so great, perhaps it is worth compromising the information about Tony's lotion use. This is worth thinking about. Security, as has been noted in [31], need not be a yes/no world. Perhaps it is extremely important for Low to know that Hillary did not use lotion but it is not really important for Low to know about Tripper's lotion use. Then High could downgrade everything as in Table 3.3 with the exception of *Tipper_Lotion*, as in Table 3.5, for the Low database. How does this impact Low's rule making process?

Now we form the *reduced base set*, see Table 3.6. Unlike the original base set given in Table 3.4, we still include a row even though there is an unknown attribute value. This is because the result is still visible to Low. It is possible, though, that High decides to keep the result unknown to Low, then that row should not be included in the reduced base set because it would not assist Low in forming a rule. Note that using the reduced base set in Table 3.6 and deleting the Tipper row, the same rules as before would still be generated. However, the confidence in the rules concerning blondes has

Name	Hair	Height	Weight	Lotion	Result
Hillary	blonde	average	light	no	burned
Janet	blonde	tall	average	yes	no
Bill	brown	short	average	yes	no
Tipper	blonde	short	average	?	burned
Newt	red	average	heavy	no	burned
Ken	brown	tall	heavy	no	no
Al	brown	average	heavy	no	no
Paula	blonde	short	light	yes	no

Table 3.6: Reduced Base Set

decreased, because the data backing our rule has decreased. The data, both in quality and quantity, should affect which rules are generated and the confidence in these rules.

3.4.2 Decision Tree Analysis

From the above example, we can see that the trade-off between functionality and security of a database plays a critical role when deciding the data set to be downgraded to Low. Therefore, measurements against both the functionality and the security are critical. In [10], Chang and Moskowitz suggest decision tree analysis as a measurement against the security of a database.

Consider the base set as given in Table 3.3. An information theoretical approach [56] is used to generate the decision trees. Shannon [60][59] first put information theory on a firm foundation. His concept of entropy and mutual information are important to this method. The columns *Hair*, *Height*, *Weight*, and *Lotion* make up the attributes. We wish to see which has the greatest influence upon the result. To determine this we use the condition entropy. Let A be the random variable representing an attribute (we have four choices for this random variable) which takes on the values a_i and let R be the random variable representing the result which takes on the values r_1 = *burned*, and r_2 = *not burned*. We need to determine the mutual information $I(R, A)$ between the result and the attribute (use base two for the logs):

$$I(R, A) = H(R) - H(R|A)$$

where

$$H(R) = -\sum_j p(r_j) \log p(r_j)$$

and

$$H(R|A) = -\sum_j p(a_i) \sum_j p(r_j|a_i) \log p(r_j|a_i)$$

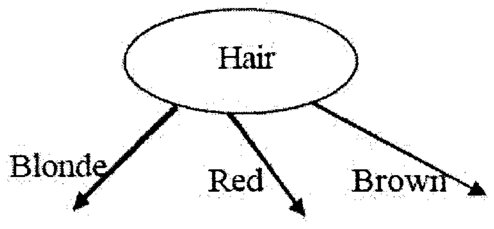


Figure 3.16: The First Branching

The probabilities are determined by a frequency count based on the data. The attribute that has the most effect upon the result is the attribute that has the greatest mutual information. Since $H(R)$ is constant and $H(R) \geq H(R|A)$, the optimization condition is equivalent to finding the attribute that minimizes the conditional entropy $H(R|A)$. Thus we have the following:

Gain Condition[56]: Find A such that $H(R|A)$ is minimized.

Let us take the first attribute $A = Hair, a_1 = blonde, a_2 = brown, a_3 = red$. This gives us

$H(R|A) = -\frac{4}{8}[\frac{2}{4}\log\frac{2}{4} + \frac{2}{4}\log\frac{2}{4}] - \frac{3}{8}[\frac{0}{3}\log\frac{0}{3} + \frac{3}{3}\log\frac{3}{3}] - \frac{1}{8}[\frac{1}{1}\log\frac{1}{1} + \frac{0}{1}\log\frac{0}{1}] = .5$

Similarly, we see that $H(R|Height) = .69$, $H(R|Weight) = .94$, and $H(R|Lotion) = .61$. Thus we see that the attribute that has the most influence upon *Result* is *Hair*. See Figure 3.16.

Now we must repeat the process for each node, until there are no more decisions to be made. Since every *Red* is *Burned*, and every *Brown* is *not Burned*, those decisions are done. However, blonde is still not decided upon so we must find another attribute that "maximally" influences result. See Figure 3.17.

Now we must repeat the gain condition but we restrict ourselves to the blondes. So we must minimize $H(R|A)$, where $A = Height$, *Weight* or *Lotion*. Let us try *Lotion*, $a_1 = no$, and $a_2 = yes$. So,

$H(R|Lotion, Hair = Blonde) = -\frac{2}{4}[\frac{2}{2}\log\frac{2}{2} + \frac{0}{2}\log\frac{0}{2}] - \frac{2}{4}[\frac{0}{2}\log\frac{0}{2} + \frac{2}{2}\log\frac{2}{2}] = 0$

We need not calculate any other conditional entropies. All they can do is tie (but they do not). So the next attribute we put down as a node is *Lotion*. See Figure 3.18.

Now we can read off the following four rules:

$(hair = blonde) \wedge (lotion = no) \Longrightarrow (result = burned)$
$(hair = blonde) \wedge (lotion = yes) \Longrightarrow (result = not\ burned)$
$(hair = brown) \Longrightarrow (result = not\ burned)$
$(hair = red) \Longrightarrow (result = burned)$

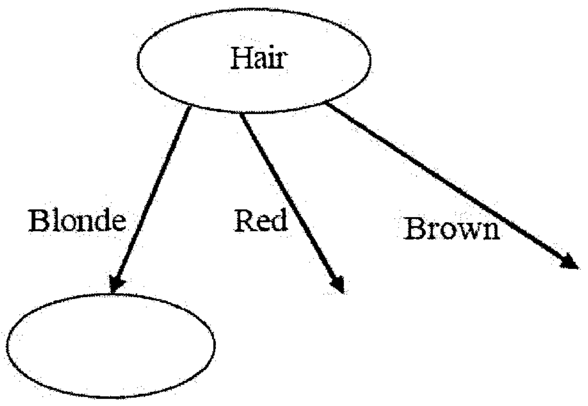


Figure 3.17: The First Branching with Partial Decisions

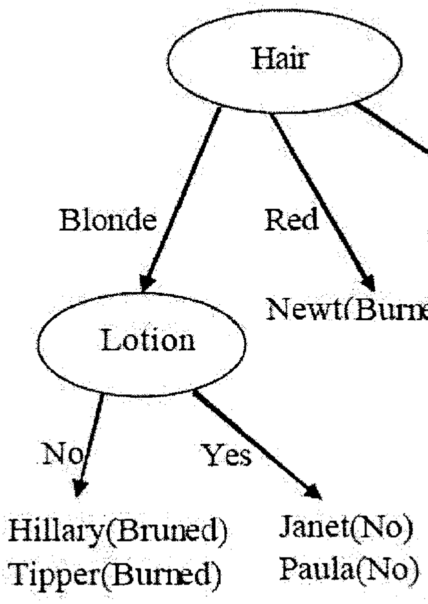


Figure 3.18: Decision Tree

We see that the first rule is the obvious one that we discussed before.

3.4.3 Remarks

Decision rules have been proven to be fruitful and accurate predictors in the AI world [57][77]. They are computationally feasible and they have a firm information theoretical foundation. As a measurement against the security of a database, decision analysis reflects the change of confidence upon decision rules between data set before and after modification. Although Chang and Moskowitz fail to define a practical measurement against the functionality of a database, their idea of evaluating these two aspects of a database indicates a new direction to resolve the inference problem.

3.5 A Scheme Using Rough Sets and Entropy to Handle the Inference Problem

In [14], we propose an integrated scheme to handle the inference problem based on rough sets and entropy functions. Previous efforts[9][10][45] gave us inspiration for developing our scheme. In [9], Bayesian networks are used to form decision rules from dataset, and in [10], decision trees analysis is used for the same purpose. But none of them presents an integrated solution like ours. Our scheme provides explicit computational methods to evaluate both security and functionality of the dataset.

3.5.1 Outline of the Scheme

From the discussion in previous section, it can be seen that the optimal solution for the inference problem should maximize both the security and the functionality. Our scheme is proposed for this purpose. We will use rough sets and entropies for explicit computations of the security and the functionality. Based on these computations, we can obtain an optimal solution. Using our scheme, the procedure of handling the inference problem can be described as follows.

1. *Identify and trim off sensitive information from the dataset for generic users.*
   This step depends on the types of database and the users. We will not discuss this in detail in this paper.

2. *Evaluate the security of the dataset.*
   Use rough sets theory to form a set of decision rules R from the dataset (See section 3.5.2). If the sensitive information cannot be inferred by employing the decision rules, in other words, there is no inference problem existing in the dataset, end the procedure. Otherwise, go to step 3.

3. *Find out all the possible values which will cause inference problems.*
   From decision rules in step 2 and the sensitive values identified in step 1, find out all the possible candidates which may cause the inference problem. Note that usually, there will be different choices of these values. We will discuss that in section 3.5.2.

4. *Choose the values to be hidden.*
   Determine the values need to be hidden in order to reduce the inference problem. We will use entropy functions to compute the functionality of the modified dataset and find out the right values to be hidden. The details are discussed in section 3.5.3.

5. *Reevaluate the security of the modified dataset.*
   Use rough sets to get a new set of decision rules for the modified dataset. This would end in two possible cases. One is that the new decision rules are different from the old rules, such that the sensitive information can no longer be inferred, which means the sensitive information is well protected. The other case is that the generic users may still infer the sensitive information by applying the decision rules, but the probability of conducting a right inference is reduced. In case that the decreasing is not satisfied, go back to step 3.

6. *Evaluate the modified dataset.*
   Consider the change in security and in functionality according to the requirements of real application. If it is viewed as balanced, end the procedure. Otherwise, repeat the procedure from step 3.

Since we want our scheme to be computationally explicit, the main point is to quantify the security and the functionality of a dataset. For this purpose, we introduce rough sets and information theory to estimate these two aspects respectively. By analyzing the changes in them, the optimal solution can be computed.

In the next two sections, we show some details of our scheme.

3.5.2 Making Decision Rules

Consider the following example, which is modified from [53]. In Table 3.7, attribute a represents "age" with values of "1: young, 2: pre-presbyopic, 3: presbyopic"; attribute b means "spectacle" with values of "1: myope, 2: hypermetrope"; attribute c is "astigmatic" with values of "1: no, 2: yes"; attribute d is "tear production rate" with values of "1: reduced, 2: normal"; attribute e is "the optician's decision" with values of "1: hard contact lenses, 2: soft contact lenses, 3: no contact lenses".

Suppose for the reason of security, the value of e of patient 13 is classified as sensitive for the generic users, and the other values are not (See Table 3.8, in which the classified value is replaced by a question mark).

U	a	b	c	d	e
1	1	1	2	2	1
2	1	2	2	2	1
3	1	2	2	2	1
4	1	2	2	2	1
5	2	1	2	2	1
6	3	1	2	2	1
7	3	1	2	2	1
8	1	1	1	2	2
9	1	2	1	2	2
10	1	2	1	2	2
11	2	1	1	2	2
12	2	1	1	2	2
13	2	1	1	2	2
14	2	2	1	2	2
15	3	2	1	2	2
16	3	2	1	2	2
17	1	1	1	1	3
18	1	1	2	1	3
19	1	1	2	1	3
20	1	1	2	1	3
21	1	2	1	1	3
22	1	2	1	1	3
23	1	2	2	1	3
24	2	1	1	1	3
25	2	1	1	1	3
26	2	1	2	1	3
27	2	2	1	1	3
28	2	2	1	1	3
29	2	2	1	1	3
30	2	2	1	1	3
31	2	2	2	1	3
32	2	2	2	2	3
33	3	1	1	1	3
34	3	1	1	1	3
35	3	1	1	2	3
36	3	1	2	1	3
37	3	2	1	1	3
38	3	2	1	1	3
39	3	2	2	1	3
40	3	2	2	2	3

Table 3.7: Original Database

U	a	b	c	d	e
1	1	1	2	2	1
.	.	.	.	.	.
9	1	2	1	2	2
10	1	2	1	2	2
11	2	1	1	2	2
12	2	1	1	2	2
13	2	1	1	2	?
14	2	2	1	2	2
15	3	2	1	2	2
.	.	.	.	.	.
40	3	2	2	2	3

Table 3.8: Modified Database

For the definition of rough set and its application to data reasoning, readers are referred to [53].

We use the 39 complete records in Table 3.8 to form decision rules. Table 3.8 can be viewed as a decision table in which {a, b, c, d} are condition attributes, whereas {e} is the decision attribute. The aim is to compute the minimal set of decision rules associated with this decision table. To do that, we first simplify the decision table by eliminating duplicate rows. Since all condition attributes in the table are different, according to rough sets theory in reasoning about data, we say that there is a total dependency between the condition attributes and the decision attribute, i.e. the dependency $\{a, b, c, d\} \Rightarrow \{e\}$ is valid.

Then we need to compute the reduction of the set of condition attributes necessary to define the decision attribute. We can do that by removing one condition attribute at a time from the table, and check if the reduced table becomes inconsistent. It is readily checked that in this example none of condition attributes can be removed. Hence the set of condition attributes is e-independent.

Now we need to check whether we can eliminate some superfluous values of condition attributes. To this end, we have to find out which values of condition attributes are indispensable in order to discern the values of the decision attribute. We know the value core (see [53]) is the set of all indispensable values with respect to e. To check whether a specific value is dispensable or not, we have to remove the value from the table and see if the remaining values in the same row uniquely determine the decision attribute value in this row. If not, this value belongs to the core. All core values of each decision rule are given in Table 3.9.

Knowing all the core values, we can easily compute reducts of each decision rule by adding to the core values of each decision rule values of condition attributes of the rule such that the predecessor of the rule is independent and the rule is true. Reducts

U	a	b	c	d	e	U	a	b	c	d	e
1	-	-	2	2	1	13	-	-	-	1	3
2	1	-	2	2	1	14	-	-	-	1	3
3	-	1	2	2	1	15	-	-	-	1	3
4	-	1	2	2	1	16	-	-	-	1	3
5	-	-	1	2	2	17	-	-	-	-	3
6	-	-	1	2	2	18	2	2	2	-	3
7	2	-	1	2	2	19	-	-	-	-	3
8	-	-	1	2	2	20	3	1	1	-	3
9	-	2	1	2	2	21	-	-	-	1	3
10	-	-	-	1	3	22	-	-	-	1	3
11	-	-	-	1	3	23	-	-	-	-	3
12	-	-	-	1	3	24	3	2	2	-	3

Table 3.9: Core Values of Decision Rules

of each decision rule are listed in Table 3.10.

U	a	b	c	d	e	U	a	b	c	d	e
1	x	1	2	2	1	13	x	x	x	1	3
1′	1	x	2	2	1	14	x	x	x	1	3
2	1	x	2	2	1	15	x	x	x	1	3
3	x	1	2	2	1	16	x	x	x	1	3
4	x	1	2	2	1	17	x	x	x	1	3
5	1	x	1	2	2	17′	2	2	2	x	3
6	1	x	1	2	2	18	2	2	2	x	3
6′	x	2	1	2	2	19	3	1	1	x	3
7	2	x	1	2	2	19′	x	x	x	1	3
8	2	x	1	2	2	20	3	1	1	x	3
8′	x	2	1	2	2	21	x	x	x	1	3
9	x	2	1	2	2	22	x	x	x	1	3
10	x	x	x	1	3	23	x	x	x	1	3
11	x	x	x	1	3	23′	3	2	2	x	3
12	x	x	x	1	3	24	3	2	2	x	3

Table 3.10: Reducts of Decision Rules

It is easy to find the minimal set of decision rules now. We can get it by removing superfluous rules from Table 3.10. Thus the final result can be written as:

$a_1c_2d_2 \rightarrow e_1$; $b_1c_2d_2 \rightarrow e_1$

$a_1c_1d_2 \rightarrow e_2$; $a_2c_1d_2 \rightarrow e_2$; $b_2c_1d_2 \rightarrow e_2$

$d_1 \rightarrow e_3$; $a_2b_2c_2 \rightarrow e_3$; $a_3b_1c_1 \rightarrow e_3$; $a_3b_2c_2 \rightarrow e_3$

Combining all decision rules for one decision class, we get the following decision rules:

Rule 1 : $(a_1 \vee b_1)c_2d_2 \rightarrow e_1$
Rule 2 : $(a_1 \vee a_2 \vee b_2)c_1d_2 \rightarrow e_2$
Rule 3 : $d_1 \vee (a_3b_1c_1) \vee ((a_2 \vee a_3)b_2c_2) \rightarrow e_3$

- **Find Out Inference Values**

  Now we come back to the inference problem. Our purpose is to hide the rule *Rule h* : $a_2b_1c_1d_2 \rightarrow e_2$. To do that, we use the decision rules obtained to "decompose" the hiding rule as follows. Basically, we consider the "and" for *Rule h* and the decision rules of the dataset.

  From *Rule* 1 and *Rule h* we obtain a rule: *Rule* h_1 : $b_1d_2 \rightarrow e_1 \vee e_2$.

  From *Rule* 2 and *Rule h* we obtain a rule: *Rule* h_2 : $a_2c_1d_2 \rightarrow e_2$.

  From *Rule* 3 and *Rule h* we obtain a rule: *Rule* h_3 : $a_2 \vee b_1c_1 \rightarrow e_2 \vee e_3$.

  The decomposition gives us strong information that the rule $a_2c_1d_2 \rightarrow e_2$ causes inference problems. This rule involves records 11,12, and 14. For each of these three records, at least one value of the attributes a, c, d or e should be hidden. At this point, we still cannot decide which value to be hidden. We will discuss how to choose the values to be hidden in the next section.

  After the chosen values are hidden, we repeat the above process to make the decision rules for the modified dataset. If there is some conflict in making decision rules or the new rules cannot decompose *Rule h* into a meaningful solution, then we solve the inference problem.

  In Table 3.11, 3 more values are hidden from Table 3.8. Using the same method to compute the decision rules, we will find that the *Rule* h_1 and *Rule* h_3 are unchanged, but *Rule* h_2 is changed to: $c_1d_2 \rightarrow e_2$. Therefore it is difficult to find *Rule h* from the modified dataset.

3.5.3 Quantifying Information in Databases

In this section, we show how to use "entropy" to evaluate the amount of information contained in a database. In [59][60], entropy is defined as a measure of the amount of information contained in an information source. The entropy function is defined as:

$$H(p_1, \ldots, p_n) = -\sum_{i=1}^{n} p_i \log p_i$$

where $(p_1, \ldots, p_i)$ is a probability distribution (using base 2 logarithms).

Intuitively, a table containing some missing values provides less information than a table without any missing values. We can use the probability to measure the availability of data. Suppose we select a random entry in a table. What is the probability that the value is not missing? If the table has no missing value, then the probability is 1. Otherwise, we can compute the probability by counting the

U	a	b	c	d	e
1	1	1	2	2	1
.	.	.	.	.	.
9	1	2	1	2	2
10	1	2	1	2	2
11	?	1	1	2	2
12	2	1	?	2	2
13	2	1	1	2	?
14	2	2	1	?	2
15	3	2	1	2	2
.	.	.	.	.	.
40	3	2	2	2	3

Table 3.11: Further Modified Database

missing values. However, since we are under the circumstance of databases, it is more complicated than tables or regular information source. We have to consider the distribution of missing values in rows and in columns. Therefore, we need to evaluate the amount of information with respect to both rows and columns. We use H_r and H_c to represent them respectively. If there are r rows in the dataset, then define $H_r = -\sum_{i=1}^{r} p_i \log p_i$, where p_i is the probability of availability of the ith row. If no value is missing, then $p_i = \frac{1}{r}$ for each i. If some value is missing in ith row, then $p_i < \frac{1}{r}$. H_c is defined in a similar way.

For example, consider the database in Table 3.8 and the trimmed version presented in Table 3.11.

The entropies of the Table 3.8 are:

$H_r = 39 \times \frac{1}{40} \times \frac{5}{5} \times \log(\frac{40}{1} \times \frac{5}{5}) + \frac{1}{40} \times \frac{4}{5} \times \log(\frac{40}{1} \times \frac{5}{4}) = 5.302$

$H_c = 4 \times \frac{1}{5} \times \frac{40}{40} \times \log(\frac{5}{1} \times \frac{40}{40}) + \frac{1}{5} \times \frac{39}{40} \times \log(\frac{5}{1} \times \frac{40}{39}) = 2.318$

The entropies of Table 3.11 are:

$H'_r = 36 \times \frac{1}{40} \times \frac{5}{5} \times \log(\frac{40}{1} \times \frac{5}{5}) + 4 \times \frac{1}{40} \times \frac{4}{5} \times \log(\frac{40}{1} \times \frac{5}{4}) = 5.242$

$H'_c = \frac{1}{5} \times \frac{40}{40} \times \log(\frac{5}{1} \times \frac{40}{40}) + 4 \times \frac{1}{5} \times \frac{39}{40} \times \log(\frac{5}{1} \times \frac{40}{39}) = 2.304$

The entropy function has the property that the value of the function increases when the probability becomes evener. For instance, if we hide the value of attribute a of record 12 and 14 instead, then the H'_c of Table 3.11 would be 2.303.

Finally, we define the information rates of the database as $IR_r = H_r/\log r$ and $IR_c = H_c/\log c$, which reflects the availability of information in a database, i.e. the functionality of a database. Therefore, $IR_r = 0.996$ and $IR_c = 0.998$, while $IR'_r = 0.985$ and $IR'_c = 0.992$.

- **Hiding Values**

  In the previous subsection we find more values should be hidden from generic users in order to protect the classified information. It remains to explain how to choose those values?

We will use information rates to decide the missing values. For each of the possible choices, we compute the information rates of the modified dataset. We will choose one modification which keeps the information rate as large as possible, in other words, the modification that has the largest entropies. We already know that the entropy function has the property that the evener the distribution, the larger the entropy. Let's take a look at the Table 3.8, from section 3.5.2, we decide to hide one value of attribute a, c, d or e in record $11, 12$ and 14. There are 64 ($P_4^1 P_4^1 P_4^1$) ways to do that. Among those choices, there are 24 ($P_4^1 P_3^1 P_2^1$) even distributions, Table 3.11 shows one of them. We already computed that $IR_r = 0.985$ and $IR_c = 0.992$. If we hide uneven distributed values in this example, we will find that the value of IR_c slightly smaller.

In real applications the importance of different attributes usually varies. So in practice, we can also consider that factor in determining which vales to hide. By employing our method and considering the importance of attributes, an optimal solution can be determined.

3.5.4 Remarks

This section presents a new scheme for handling inference problems, which considers both security and functionality of a dataset. The scheme uses two main tools. One is the application of rough sets to form a minimal set of decision rules from the dataset. The other is the use of entropy, an important concept from information theory, to evaluate the amount of information contained in the dataset. By analyzing the changes of confidence in decision rules and in the amount of information, an optimal solution can be decided. The scheme is explicit and also easy to be implemented.

Chapter 4

Removal of Inferences

In this chapter, we introduce two dynamic techniques to remove the inference problem after all inference channels have been identified. They can be combined with techniques in detecting the inference channels, such as the techniques introduced in last chapter, to form an integrated solution for this problem.

4.1 Dynamic Inference Control

In [65], dynamic control is proposed to handle the inference problem. This method considers the trade-off between the granularity of the access control and the query processing time.

This method allows for fast query processing while enabling fine-grained access control, and thus, flexible information access. To do this, access-enabling tokens are used to associate with objects in the database, and users are allocated keys that they use to generate the necessary tokens. Once a token is used to query an object, the key it was derived from cannot be used to query any other object in the inference channel. This is implemented by deleting the tokens generated with this key from other objects in the channel. Hence, query processing depends on the length of the channel rather than the ever-growing user query histories. In addition, because initially the same tokens are associated with each object, it allows for flexible information access. A user can access any objects in the inference channel provided doing so will not enable the user to make the undesired inference, even through collusion with other users.

4.1.1 Preliminaries

Let n denote the number of users of the database, and $U_1, \ldots, U_n$ denote the users themselves. Let m denote the number of objects in an inference channel, and $O_1, \ldots, O_m$ denote the objects in the channel. An inference channel of length m is sometimes referred as an $m-$ *channel.*

In the initialization phase, users receive a set of encryption keys that they use to prove authorization to access the objects in an inference channel. Users prove this by encrypting some information specific to their object of interest. Let K_i denote the set of keys allocated to U_i, $i = 1, \ldots, n$, each encryption key may be known to several users. Before any queries have been made, each encryption key can potentially be used to access any object in the channel. However, users only have enough keys to generate tokens for a proper subset of the objects in the channel. A user is said to have *maximally queried the channel* if they have used all possible keys to query the channel. By limiting the number of keys per user, collusion resistance (Collusion resistance is that a coalition of users cannot together query all the objects in the inference channel) is guaranteed.

Definition 4.1.1 *Let c be an integer,* $0 < c \leq n$. *We say that an inference protection scheme is c-collusion resistant if c users acting in collusion are unable to query all the objects in any inference channel.*

Once an encryption key is used to gain access to object Q_i, the same key cannot be used to gain access to any object O_j, $j \neq i$, in the same inference channel. This is accomplished by deleting the token generated by that key from the rest of the inference channel. In other words, part of the automated access control mechanism is to update the list of acceptable tokens for objects in the inference channel each time an object in the channel is accessed. Hence, because a key may be used by many users, the queries of each user potentially affect the access capabilities of many users. For example, if two of the keys used to query an object in the channel belong to the same user, then this user will be unable to maximally query the channel. These properties allow us to achieve a property that we call crowd control: if a lot of users have queried a maximal portion of an inference channel (e.g. $m-1$ out of m objects) then no user should be able to complete the inference channel by making the remaining query. The reasoning here is that if an object has been queried a lot, it is more likely to have been leaked and so it may be prudent to consider the query results to be in the public domain. So, if this is the case for most of the channel, access to the remaining object should be explicitly prohibited.

Definition 4.1.2 *Let* $0 < \epsilon < 1$, *and let* U *denote a randomly selected user. Consider a c-collusion resistant inference protection scheme. If when more than* x *sets of* c *users have together queried some set of* $m-1$ *objects,* $O_{i_1}, \ldots, O_{i_{m-1}}$, *in inference channel* $\{O_1, \ldots, O_m\}$, *then with probability at least* $1-\epsilon$, U *cannot access the remaining object,* $\{O_1, \ldots, O_m\} - \{O_{i_1}, \ldots, O_{i_{m-1}}\}$, *we say the scheme has* (x, c, ϵ)*-crowd control.*

Figure 4.1 is a high level example of how token sets, and consequently access control, changes as a result of user queries. It simplifies this approach in two ways. First,

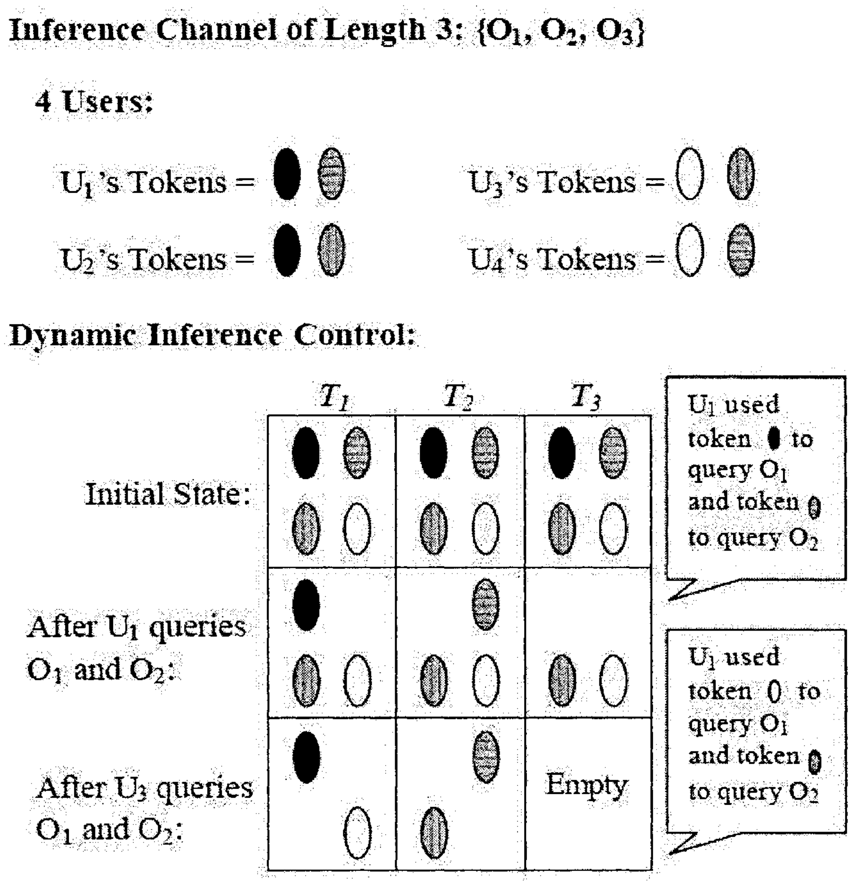

Inference Channel of Length 3: {O1, O2, O3}
4 Users:
U1's Tokens =
U2's Tokens =
U3's Tokens =
U4's Tokens =
Dynamic Inference Control:
T1
T2
T3
Initial State:
After U1 queries O1 and O2:
After U3 queries O1 and O2:
Empty
U1 used token to query O1 and token to query O2
U1 used token to query O1 and token to query O2

Figure 4.1: Dynamic Inference Control

it does not show the keys each user receives, but rather just the tokens they generate from them (depicted here as ovals). The second, and more important, simplification is that tokens corresponding to different objects appear identical. In reality, the tokens are particular to the object with which they are associated, while the encryption keys used to generate them may be the same. In this example there are four users each with two tokens (or, two keys that they use to generate tokens) and collusion resistance is $c = 1$. For $i = 1, 2, 3$, the ith column indicates which tokens can be used to access object O_i after the queries listed on the left hand side have been executed. After both U_1 and U_2 have queried objects O_1 and O_2, no one can query object O_3 (it has no more acceptable tokens) but everyone can still access both O_1 and O_2.

An important part of the analysis is assessing how information access changes as more users query the channel. To do this we need to understand how one user's set of keys relates to another user's set of keys. This is important because with each query the set of acceptable tokens for every other query can change. More precisely, we often study how much the key sets of one group of users, say $U_1, \ldots, U_r$, cover the key set of user, U_{r+1}. The size of the cover is the number of keys U_{r+1} has in common with at least one of $U_1, \ldots, U_r$, that is, the value: $|K_{r+1} \cap (\cup_{i=1}^{r} K_i)|$. Finally, for simplicity of exposition, it is assumed that a fractional quantity (i.e. $\frac{a}{b}$) is an integer.

4.1.2 Dynamic Inference Control

We assume that inference channels have been identified (for example, using a tool such as [58]) prior to the deployment of this inference control scheme. The protocol consists of three phases:

1. Key allocation
   Users are allocated $(m_{max} - 1)/c$ keys, where m_{max} is the maximum length of an inference channel in the database, and c is the desired degree of collusion resistance.

2. Initialization of the database
   For each inference channel $Q = \{O_1, \ldots, O_m\}$, a set of tokens, T_i, is associated with each object such that each user is capable of generating exactly $(m-1) = c$ tokens in T_i, for $i = 1, \ldots, m$. Initially, that is prior to any queries, the token sets are identical: $T_1 = T_2 = \ldots = T_m$.

3. Query processing
   If token $t \in T_i$ is used to gain access to O_i, then for every $s \neq i$, any token in T_s that was generated by the same key is deleted. Hence, the token sets change as queries are made.

Two schemes, which differ in how the first stage is completed, are presented. Initialization of the database is essentially the same for both and query processing is as described above. The first is a simple randomized scheme that achieves probabilistic guarantees on crowd control (that is, $\epsilon > 0$). The other is an algebraic scheme that offers deterministic guarantees on information access.

1. **A probabilistic key allocation scheme**
   To allocate keys to the users a bucket-based approach, which is often used in secure group communication (see, for example, [41]), is adopted. Let there be $(m_{max} - 1)/c$ buckets, $B_1, \ldots, B_{m_{max-1}}/c$, each containing q encryption keys (that is, there are $q(m_{max} - 1)/c$ keys in total). The keys themselves are randomly generated and are of one bit length suitable for the symmetric encryption scheme being used. For $i = 1, \ldots, n$, U_i receives a randomly selected key from each bucket for a total of $(m_{max} - 1)/c$ keys per user.

   Token sets for an inference channel of length $m \leq m_{max}$ are formed by choosing a subset of $(m-1)/c$ buckets, $B_{i_1}, \ldots, B_{i_{\frac{m-1}{c}}}$, and using the keys in each bucket to generate the tokens for each object in the *m − channel*. Hence, before any queries are made, each user has the ability to query any $\frac{m-1}{c}$ objects, and so the scheme is c-collusion resistant. The following theorem shows how, for a given value of ϵ, the degree of crowd control afforded by the scheme depends on m and q. The idea behind the theorem is that a large set of users are likely to cover the key set of another user, so if these users have maximally queried the channel, the other user will be blocked.

Theorem 4.1.1 *Let $0 < \epsilon < 1$, and let c denote the collusion resistance of an instance of the probabilistic inference control scheme. Let $x = \frac{\ln(1-(1-\epsilon)^{c/(m-1)})}{\ln 1 - c/q}$ then the scheme has (x, c, ϵ)-crowd control.*

PROOF. It suffices to show that if more than $x = \frac{\ln(1-(1-\epsilon)^{c/(m-1)})}{\ln 1 - c/q}$ sets of c users have queried $m - 1$ objects, $Q_{i_1}, \ldots, Q_{i_{m-1}}$, in an inference channel of length m then the probability that a user U can query Q_{i_m} is less than ϵ. U is unable to query Q_{i_m} if the key sets of the users who have queried any of the other objects in the channel cover the relevant part of $U's$ key set (i.e. those keys of $U's$ that are useful for querying this channel). Of course, this happens with probability 1 if U has made $\frac{(m-1)}{c}$ other queries, and otherwise, the probability of this covering is at least $(1 - (1 - c/q)^x)^{(m-1)/c})$. Setting the latter quantity to be at least $1 - \epsilon$ and solving for x gives the quantity in the theorem statement.

This theorem shows that if a particular $(m-1)$-subset of the objects in a channel has been queried a lot, then users will be unable to query the remaining object in the channel whether or not they have already made some queries. Its likely that most users will still be able to access some $((m-1)/c)$-subset of the queried objects, however, as the following lemma shows.

Lemma 4.1.2 *Let $0 < \epsilon < 1$, and let c denote the collusion resistance of an instance of the probabilistic inference control scheme. If $x > \frac{\ln(1-(1-\epsilon)^{c^2/m^2})}{\ln(1-1/q^2)}$ users have each maximally queried an m-channel, then with probability greater than $1-\epsilon$, a user U, who is not amongst the x users, can maximally query the same channel.*

PROOF. A user U cannot maximally query the channel if two of $U's$ keys have been used to query a single object. This is impossible if for every pair of $U's$ keys, one of the users who has maximally queried the channel has both such keys. The probability of this is at least (note this is a coarse bound): $(1-(1-1/q^2)^x)^{(m/c)^2}$. The bound in the lemma can be obtained by setting this quantity to be greater than $1-\epsilon$.

Theorem 4.1.3 *Let $0 < \alpha < 1$. Consider a c-collusion resistant instance of the probabilistic inference control scheme. If the number of users who have maximally queried the channel is $x < \frac{q(1-\alpha)\epsilon^c/((m-1)(1-\alpha))}{e}$, then with probability at least $1-\epsilon$, another user can access any subset of objects in the channel of size $\frac{\alpha(m-1)}{c}$.*

PROOF. Consider a set of $x+1$ users, $U \notin \{U_1, \ldots, U_x\}$, the expected number of $U's$ keys that $U_1, \ldots, U_x$ cover, is $\frac{(m-1)c}{cq}$. We show that the probability that $U_1, \ldots, U_x$ cover more than $1-\alpha$ of the keys U has for querying the channel is less than ϵ. From [49], it follows that this is true when the following inequality holds: $\left(\frac{e^{(q(1-\alpha)/x)-1)}}{(\frac{q(1-\alpha)}{x}^{\frac{q(1-\alpha)}{x}})}\right)^{\frac{(m-1)x}{cq}} < \epsilon$. This inequality is satisfied by the bound on x given in the statement of the theorem.

2. **A variant with deterministic guarantees**
   The above key allocation scheme guarantees crowd control and information access probabilistically as a function of the number of users querying an inference channel. In some settings deterministic guarantees are necessary. We can achieve some deterministic guarantees by using error correcting codes to allocate keys to users. Specifically, we use Reed-Solomon codes (see, for example, [44]) to allocate keys to users. Reed-Solomon codes use a polynomial that is unique to each user to determine each users codeword, or in our case, key set. Because two polynomials of the same degree intersect on at most as many points as their degree, two users in our inference control scheme will share at most as many keys as the degree of their polynomials. Using this fact, we construct a scheme with deterministic (i.e. $\epsilon = 0$) information access guarantees. The following makes the key allocation part of such a scheme more precise, the initialization of the database and the query processing are both just as before.

We consider all polynomials of degree t, $0 < t < (m_{min}-1)/c$, over the finite field F_q of q elements, where m_{min} is the minimum length of an inference channel. To each user, U, we associate a unique such polynomial, $pU(x) \in Fq[x]$. For each element $(\gamma, \beta) \in F_q \times F_q$ we generate a random key, $k_{\gamma,\beta}$ of the desired bit length. User U receives the set of keys $K_U = \{k_{\gamma,pU(\gamma)} \in A \subseteq F_q\}$, where A is a set of size $(m_{max} - 1)/c$. Note that this is very similar to the bucket-based construction except that using polynomials to allocate keys from the "buckets" gives more control over the overlap between users' key sets. The following lemma demonstrates one of the deterministic guarantees provided.

Lemma 4.1.4 *Consider a c-collusion resistant inference protection scheme that uses the key allocation method of this section. If $x < \frac{(m-1)(1-\alpha)}{ct}$ users have maximally accessed an m-channel then another user can access any subset of objects in the channel of size $\frac{\alpha(m-1)}{c}$ with probability 1.*

PROOF. Consider a user U who is not among the x users who have maximally accessed the channel. U shares at most t keys with each of the x users, and so at most $tx < t(\frac{(m-1)(1-\alpha)}{ct}) = \frac{(m-1)(1-\alpha)}{c}$ keys total. Hence, U has more than $\frac{\alpha(m-1)}{c}$ keys that none of the x users have and can access a different object in the channel with each key.

An analysis very similar (but a bit more involved due to the fact that keys are not assigned independently) can be performed to prove crowd control and lower bounds on information access. The scheme performs comparably to the earlier one.

4.1.3 Remarks

Though proposing a creative idea to handle the inference problem, this method has some serious drawbacks. For example, it concentrates on a single inference channel for simplicity of exposition. When using our methods to prevent inferences across multiple channels a potential problem arises when a single object appears in channels of different lengths. To ensure that no inferences can be made by exploiting the varying channel lengths it may be necessary to reduce the number of acceptable keys associated with objects in overlapping channels, thus reducing information access. Also, it assumes a fixed user base. Over time, however, it is likely that users will lose access to the database (i.e. be revoked) and new users will be added. To allow for the addition of new users we can choose q to be larger than is currently required. Of course, this will have consequences for crowd control: increasing q means users' key sets are more disjoint and so the queries of individual users tend to have less of an impact on the access capabilities of others. Thirdly, when a user is revoked from accessing the database their keys must no longer be valid. Simply deleting the tokens generated from their keys might unfairly restrict the access of others, so rekeying of the valid users may be needed.

The performance of this method will be compared with another dynamic control presented in next section in a more detailed manner.

4.2 Accessing Key Distribution Schemes

In [13], we design several accessing key distribution schemes to solve the inference problem in MLS databases. They are collusion resistant, efficient and easy to implement. Unlike previous proposed schemes, they guarantee the maximum accessibility for users, at the same time prevent inference problems and retain fast query processing.

4.2.1 Overview

We assume that all inference channels are identified in an initial pre-query processing stage. Therefore, we can focus on inference control during the query processing. For the sake of simplicity, we consider only users at the same security classification level in MLS database. The processing for users of other security levels is similar.

In our scheme, the foremost task is to generate an accessing key set, which is usually done by running a key generation algorithm at the system end. Each key contains information about the association to objects. The number of keys in a key set depends on the length of the inference channels. We use K to denote a key set. Two kinds of key schemes are proposed in this paper. In one of them, the key set is only used by the database system so that users do not need to keep any keys. In the other, each user has one secret key. In the initialization phase, all objects in an inference channel are associated with all or part of the keys in the key set. When a key is used to access an object by a query algorithm, other queries have to use the same key to access the object. This is done by deleting the association between the object and other keys. By doing so, our scheme ensures the most flexible access control (users are able to decide which object they would like to access provided that they can access all but one object in an inference channel), as well as fast query processing since the processing time depends on the length of inference channels instead of users' query histories. Besides, all keys are different and different keys cannot be used to access the same object. We will see how this feature provides the property of collusion resistance in later discussion.

We separate the schemes into two phases. The first is key initialization. In this phase, associations between keys and objects are established. The other is query processing which details the algorithm of a query. The initial algorithm runs one time (unless the whole system is going to refresh). Then the query processing algorithm runs whenever a user wants to access an object.

4.2.2 Single Key Set Schemes

In this section, we present three key distribution schemes to handle the inference control problem under three different settings.

1. **A Simple Scheme for Single Inference Channels**

   First we consider the case that there is only one inference channel in the database. Let m denote the length of the inference channel, and $O_1, \cdots, O_m$ the objects. Let U denote a user of database.

   In our schemes, an object O is associated with a set of keys denoted by $K(O)$. In this simple scheme, the total number of keys is $m-1$: the set of keys $K = \{k_1, \cdots, k_{m-1}\}$, and every object in the inference channel is associated with all the $m-1$ keys initially, i.e., $K(O_i) = K, i = 1, 2, \cdots, m$.

   The query processing is as follows. When a user tries to access one object, a key is selected randomly by the algorithm. The association between the object and other keys will be deleted. At the same time, the association between the selected key and other objects will be deleted as well. When all $m-1$ keys have been used, $m-1$ objects of m objects in the channel are associated with keys. However, there is one object left, which is associated with no key at all. We call that object "the reserved object", which means no user can access it. Once a "reserved object" has been determined, the system puts an indicator on it. In this way, no user can access the reserved object. In other words, all users are able to access all objects but the reserved object in the inference channel. Therefore, it is pointless for a group of users to try to do inference in collusion. Figure 4.2 is an small example. In this example, there are 4 objects O_1, O_2, O_3, O_4 in the channel. Initially, each object is associated with 3 different keys (see the upper part of the picture). The lower part of the picture shows the key sets after one user requested O_2. In this case, O_2 is associate only with one key while other objects are associated with other 2 keys.

   We now give the formal description of the scheme in the following algorithm. This algorithm consists of a key initialization algorithm and a query processing algorithm. Let K denote a key set of $m-1$ keys.

   Algorithm 4.2.1 *Single channel scheme for a user requesting object O_i.*

   ```
   Key initialization:
   ```

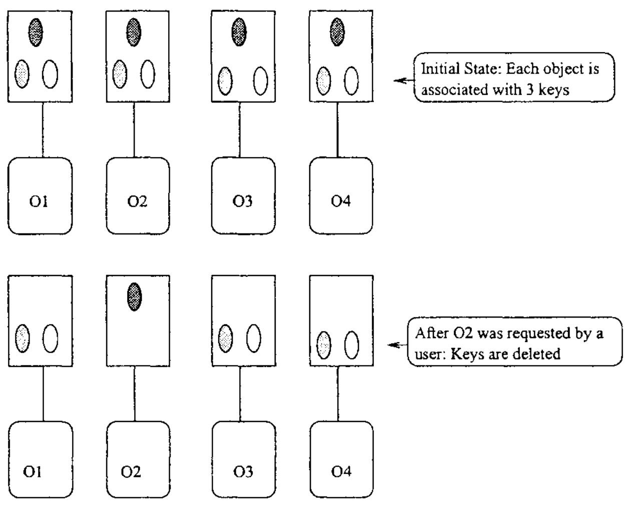

Initial State: Each object is associated with 3 keys
O1
O2
O3
O4
After O2 was requested by a user: Keys are deleted
O1
O2
O3
O4

Figure 4.2: Example of inference control

```
$K(O_i) = K, i = 1, \cdots, m$.
Query processing:
Input: $i$;
if $K(O_i) = \emptyset$ then
    output "access denied";
else
    Select randomly a $k_j \in K(O_i)$;
    $K(O_i) = \{k_j\}$;
    $K(O_s) = K(O_s) \backslash \{k_j\}$ for all $s \neq i$;
    Deliver $O_i$ to the user.
```

2. **Multiple Inference Channels Without "Repeated Object"**

   Now, let us consider the situation of more than one inference channel in the database. The solution is to allocate one set of keys to each inference channel. Let C denote an inference channel, and l denote the number of inference channel in a database: $C_1, \cdots, C_l$, the length of one inference channel C_j is denoted as $m_j, j = 1, \cdots, l$, m_{max} denote the maximum length of all inference channels. In this subsection, we only consider the situation that all channels are disjoint each other. In this case, the key set K contains $m_{max} - 1$ keys.

   Query processing is similar to the algorithm for a single inference channel. The main difference is the initial state of key sets. Initially, for channel C_j, let K_j be a set of $m_j - 1$ keys selected randomly from $K, j = 1, 2, \cdots, l$. Let $K(O_i) = K_j$ if $O_i \in C_j$.

Algorithm 4.2.2 *Disjoint multiple channels scheme for a user requesting object O_i.*

```
Key initialization:
$K_j = \{m_j - 1$ random keys from $K\}, j = 1, 2, \cdots l$;
if $O_s \in C_j$ then  $K(O_s) = K_j$, for all possible $s$.
Query processing:
Input: $i$;
if Find $j$ such that $O_i \in C_j$ then {
    if $K(O_i) = \emptyset$ then
        output "access denied" and quit;
    else
        Select randomly a $k_p \in K(O_i)$;
        $K(O_i) = \{k_p\}$;
        $K(O_s) = K(O_s) \setminus \{k_p\}$ for all $O_s \in C_j, s \neq i$;
        Deliver $O_i$ to the user;
}
else
    output "information not found".
```

3. **Multiple Inference Channels With "Repeated Object"**

   Finally, let us consider the situation that some objects appear in more than one inference channel. For illustration, we give another example in the same database as the example mentioned in section 1. The relationship between the project and its supporting company is classified. While users can access "training course" on the project, "person(engineer)" who attends the training course, and for which company the person(engineer) works. Thus, users can infer the company which supports the project. In this case, the inference channel has four objects: "project", "training course", "person(engineer)" and "company". We name an object appearing in more than one channel after "repeated object". Referring to our examples, "project" and "company" are repeated objects.

   There may exist two different cases during the query processing. One is that the repeated object is not a reserved object of any channel. In this case, users can access the repeated object in a way the same as what they do to access the other objects. The other case is that the repeated object, which users are trying to access, is a reserved object, then the request should be denied.

   We use the following algorithm. The initialization of the key sets is the same as that described in Algorithm 4.2.2. So the number of total keys we used is $m_{max} - 1$. For a repeated object, the number of key sets associated with it is equal to the frequency of occurrence in all inference channels. We use $K_j(O_i)$ to denote the key set which is associate with the object O_i in channel C_j.

Algorithm 4.2.3 *Multiple channels scheme for a user requesting object O_i.*

```
Key initialization:
$K_j = \{m_j - 1$ random keys from $K\}, j = 1, \cdots, l$;
for All possible $s$ do {
    if $O_s \in C_j$ then $K_j(O_s) = K_j, j = 1, \cdots, l$.
}

Query processing:
Input: $i$;
if $O_i \in C_j$ then {
    if $K_j(O_i) = \emptyset$ then
        output "access denied" and quit;
    else {
        for every $C_j$ such that $O_i \in C_j$ do {
            Select randomly a key $k_p \in K_j(O_i)$;
            $K_j(O_i) = \{k_p\}$;
            while $O_s \in C_j$ and $s \neq i$ do {
                $K_j(O_s) = K_j(O_s) \backslash \{k_p\}$
                if $K_j(O_s) = \emptyset$ then
                    $K_s(O_s) = \emptyset$ for all $r$ such that $O_s \in C_r$;
                }
            }
        Deliver $O_i$ to the user;
    }
}
else
    output " information not found".
```

We would like to mention the processing procedure of the repeated object in a more detailed way because it is special case. Let us assume that there exists an object which is a repeated object, and at the same time a reserved object of one inference channel. In order to prevent users from receiving this object from other inference channels, we should synchronize the status of this object in all inference channels that it belongs to. The idea is that when a repeated object is accessed, the system should perform a function on the same object in other channels such that the status of the object is accessed. Similarly, once a repeated object is indicated as a reserved object, the system should make it the reserved object of all other channels in which it appears. In practice, we suggest that a number of system tables, which contain statistical data about the inference channels, should be set up once all inference channels are identified.

4. **Performance**

   Now we will give summary of the performance of our method. To solve the inference problem, three key elements must be considered. They are listed as follows in the descending order of importance.

   - Security, which requires that no user can do the undesirable inference, neither alone nor collaboratively.
   - Access flexibility, which means that maximum ability of access should be guaranteed provided that the possibility of performing an undesirable inference is zero.
   - Response time, which means that in practice the query processing time meets the requirement of applications.

   Under our key schemes, all users are able to access the same $m-1$ objects in the inference channel. Therefore it is impossible for database users to access objects enough to perform an inference, neither alone nor in collusion. Moreover, we can see that our schemes actually promise the maximum access flexibility with the fact that $m-1$ out of m objects are accessible. As to the response time, because the number of keys depends only on the maximum length of inference channels in the database, the key space is so small that the response time is almost the same as the response time in a system without considering the inference problem.

   As we mentioned in the previous section, [65] provides dynamic control over the inference problem. We now compare it with our scheme, meanwhile we will explain other benefits provided only by ours. For the sake of simplicity, we use " Staddon's Scheme " to denote the approach in [65] during the following discussion. Here we give a brief description of the scheme. In Staddon's Scheme, a Probabilistic Key Allocation Scheme is used to ensure dynamic control over the inference problem. Suppose there are m objects $O_1, O_2, \cdots, O_m$ in an inference channel and n users in the database. In the initialization phase, $(m-1)/c$ buckets of random keys are set up, where c is the desired degree of collusion resistance. Each bucket contains q random keys (so there are in total $q(m-1)/c$ keys used). Every user receives $(m-1)/c$ randomly selected keys such that no keys are from a same bucket. In this phase, an object O_i in the channel is associated with a set of keys T_i (In the original scheme, T_i is a set of tokens.

	Staddon's Scheme	Our Scheme
Inference channel modes	1. Single channel 2. Multiple channels no repeated objects	1. Single channel 2. Multiple channels no repeated objects 3. Multiple channels repeated objects
User base	fixed	variable

Table 4.1: Comparison of System Environment

	Staddon's Scheme	Our Scheme
Cost	1. A list of acceptable key for each object 2. Each user has $(m-1)/c$ keys 3. Mechanism to prevent against key leaks	1. System tables
Parameters	1. Processing time: depending on m and q 2. Access flexibility:$[1,(m-1)/c]$ 3. Key space: $q(m-1)/c$ 4. Key size: large 5. Collusion resistance: $\leq c$	1. Processing time: depending on m 2. Access flexibility: $m-1$ 3. Key space: $m-1$ 4. Key size: small 5. Collusion resistance: Any size

Table 4.2: Comparison of Cost and Benefit

However, essentially, we can view the set as a key set). Initially, T_i contains all $q(m-1)/c$ keys so that any key can be used to access the object. During query processing, a key in T_i must be used to access an object O_i. If a key is used to access an object O_i, then for every $s \neq i$, that key is deleted from T_s.

Firstly, we consider both two schemes' performance in terms of system environment (see Table 4.1). In our scheme, the user base is variable since no user needs to keep any keys. Table 4.2 shows the performance on system cost.

We can see that Staddon's scheme has some drawbacks.

- In the Staddon's Scheme, each user's key set is different from the others, so the scheme has to keep and maintain a key set for every user, as well as provide prevention against key leaks.
- Since different keys are allowed to be used to access the same object, the scheme needs to set up and maintain a list of acceptable keys for each object. This feature also makes the number of accessible objects ($(m-1)/c$) unwarranted. In fact, the number of accessible object is between 1 and $(m-1)/c$. For example, if there is a popular object in the channel that many users visit it (unfortunately, this is a common phenomenon), then

many keys were deleted from the key sets of other objects. In this way, many other objects are unavailable for users.

- When the inference channel is short (m is small), the system only can let $c = 1$. That means the scheme cannot protect against a collusion of users to attack in this case.

We should point out that in our schemes, none of these problems exists. Our schemes can protect against any size of collusion while Staddon's scheme only can bear c users' collusion. Note that in their scheme, there is a trade-off between the value of c and the accessibility of a user so that c cannot be large. Moreover, since the keys are used only by the database system, we can use any set of size $m_{max} - 1$ as the key set. Therefore the size of a key can be very small. For example, we may use the set $\{1, 2, \cdots, m-1\}$ as the key set. However, the size of a key in Staddon's scheme should be large enough to prevent from key leaking.

Another useful feature of our schemes is the scalability. Suppose an inference channel was identified and a key scheme is set. However, the system finds later that some objects should be added or deleted from the channel. In this case, we just need add or reduce some keys from the scheme.

Our scheme is also easy to refresh because users do not possess keys. A system can run the key initialization algorithm in any time.

In practice, we can use system tables to store statistical data on inference channels, such as the number of inference channel within the database, what objects are contained in an inference channel, etc. The data is important when system functions are performed, for instance, when a function is run to change the status of a repeated object. However, they are statistical data, so they would not occupy much system resources.

4.2.3 Multiple Key Set Schemes

Schemes in Section 4.2.2 are very simple and efficient. The main drawback of them is that a user can perform "block an object" attack as follows. The user just visits all the $m-1$ other objects in the inference channel so that the last object is blocked. This problem will be more serious if the channel is "short" (the number of the objects is small). In practice, the system may establish a time frame. Within this time frame, any coalition of users cannot get the all information in the inference channel. However, after the time frame, the key set is refreshed and the channel returns to its initial stage. Another method to prevent the attack is the requirement of extra authorization. When a user is requesting the reserved object, the system reassesses and decides whether to grant extra authorization to the user.

In this section, we propose another scheme which prevents the attack with the price that more keys are used. For simplicity, we only describe a scheme for a single inference channel. However, it is straightforward to modify this scheme to the other cases discussed in Section 4.2.2.

Recall that in a single inference channel, there are m objects. In the previous scheme, the system associates $m-1$ keys to each object. In this scheme we associate qt keys to each object in the channel (we will discuss later that $t = (m-1)/c$, where $c \geq 1$ related to the size of collusion). The keys are divided into q disjoint subsets each containing t keys. The main idea is that when a user queries an object, only one of the subsets is involved in the processing so that other users can still access any object with high probability even if a user performed "block an object" attack. In order for the system to recognize users, each user needs a key in this scheme. For users with the same secret key, the algorithm is just like Algorithm 4.2.1. However, there are q "copies" of the key set and users with different keys will involve different copies. We describe the scheme in the following subsection.

1. **The scheme**

   To initialize the scheme, the system chooses two random key sets Q and T, where $|Q| = q$ and $|T| = t$. In the initialization phase, each object is associated with the key set $K = Q \times T$. In other words, each object is associated with q copies of set T. For a user U, the system assigns a random key $q_U \in Q$ to him/her. When the user U wants to access an object O_i, the key q_U must be submitted. We use the following algorithm to describe the scheme.

   Algorithm 4.2.4 *Multiple key set scheme for the user U requesting object O_i.*

```
Key initialization:
$K_i^j = \{q_i\} \times T$, for all $q_i \in Q, j = 1, \cdots, m$;
$K(O_j) = \cup_{q_i \in Q} K_i^j, j = 1, \cdots, m$.
Query processing:
Input: $i, q_U$;
if $K_{q_U}^i = \emptyset$ then
    output "access denied";
else
    Select randomly a $k_j \in K_{q_U}^i$;
    $K_{q_U}^i = \{k_j\}$;
    $K_{q_U}^s = K_{q_U}^s \backslash \{k_j\}$ for all $s \neq i$;
    Deliver $O_i$ to the user.
```

	Scheme [65]	Our Scheme
Key space	$q(m-1)/c$	$q(m-1)/c$
User key	$(m-1)/c$	1
Query time	$q(m-1)/c$	$q+(m-1)/c$
Access flexibility	$[1,(m-1)/c]$	$(m-1)/c$
Suitability	1 and 2	1, 2 and 3
Collusion resistence	c	At least c

Table 4.3: Comparison with multiple key set scheme

In this scheme, each user can access at most t objects of the channel. A coalition of c users, where $ct \leq m-1$, cannot obtain the information of all the objects in the channel. Since each q_U is randomly selected from the key set Q, the probability that a user get a specific key is $\frac{1}{q}$. In order to block an object for a user U, there should be a coalition of c users who hold the same key q_U. Therefore the probability that a coalition of c users can perform the "block an object" attack to a user is $(\frac{1}{q})^c$. There is obviously a trade-off between c and t. Here c is the size of a collusion the scheme can tolerate and t is a measure of the accessibility for a user.

2. **Performance**

Now we compare our multiple key set scheme to the scheme of [65]. For convenience, we let both schemes use a key space of size $q(m-1)/c$. Table 4.3 lists general parameters of these two schemes.

In our scheme, each user only keeps one secret key while a user needs to keep $(m-1)/c$ keys in the other scheme. During the query processing, our scheme first finds a t-subset using q_U and then treats the t-set. So the time requirement is $q+t = q+(m-1)/c$ in this case. Staddon's scheme needs to treat all the $q(m-1)/c$ keys. Therefore our scheme is more efficient in both space and time complexity.

Moreover, we can see that our scheme does not have the main weaknesses indicated in last section. In our scheme, each user only has to keep one key and each user can always visit $(m-1)/c$ objects even if one object was visited by all of the users. In our scheme, a coalition of $c+1$ users may not be able to do the inference, because some users may have the same key. Suppose there are b users. Then the probability that $c+1$ users can get all the information in the inference channel is

$$\frac{\binom{q}{c+1}}{q^{c+1}} < \frac{1}{(c+1)!}.$$

This shows that even in the case of $c = 1$ (when inference channel is short), our scheme still provides a little collusion resistance while Staddon's scheme cannot.

We should indicate that although we put the key spaces as $q(m-1)/c$ in both schemes, the key space of our scheme is much smaller. Note that in Staddon's scheme, all the $q(m-1)/c$ keys should be randomly selected from a large set (much larger than $q(m-1)/c$). Otherwise, a user can perform a simple attack by randomly selecting several keys and using these keys to query the objects in the inference channel. In our scheme, only q secret keys are used. So we just randomly select q keys from a large set (much larger than q). The set of T can be any t-set, e.g., $T = \{1, 2, \cdots, t\}$, because set T is used and maintained by the system. Therefore the size of key in T can be very small. In practice, we can use $(q_U; 1, 2, \cdots, t)$ to record a initial key subset and delete integers accordingly during the query processing.

4.2.4 Remarks

In this section, we proposed several key schemes to handle inference problems in a multilevel database. All these schemes are simple and very efficient while providing collusion resistance to information inference. Different situations are considered in this paper including that several inference channels were identified in a database, which are not disjoint.

Our scheme is very easy to implement if all the inference channels are identified. We note that so far there is no efficient method to identify all the inference channels in a database [36][29][70], which limits the application of our schemes. In fact, many inference problem is a probabilistic problem in nature. It means that the probability of a correct inference depends on the information obtained. In this case, no ideal inference channel can be identified. One open question for our schemes is whether these schemes can be modified by adding some probabilistic weighting to handle this situation.

Chapter 5

Conclusion and Future Research

This thesis provides a comprehensive view of the inference problem, and some representative techniques on detecting and removing the problem in MLS databases are discussed as well. Furthermore, the findings of our research are shown. There are two paper from this thesis: "A Scheme for Inference Problems Using Rough Sets and Entropy" and "A Dynamic Method for Handling the Inference Problem in Multilevel Secure Databases", which are represented in Section 3.5 and 4.2 respectively.

In section 3.5, we proposed an integrated, computational solution for the inference problem. The solution contains two main parts. First we use rough sets to deduce the minimal set of decision rules from dataset. Then we use information rates to measure the amount of information contained in the dataset. By quantifying both security and functionality, we are able to analyze the possible changes in them. Depending on these analysis, we suggest a practical way to decide the set of values to be hidden which solves the inference problem.

Our scheme to handle the inference problem is the first one to propose computationally feasible methods for both security and functionality of a dataset. Although some other techniques, such as decision trees and Bayesian networks, were used to form decision rules, we feel that rough sets have some advantages in reasoning about data. For example, rough sets use decision tables to represent data when forming decision rules. Tabular form of representation allows us for easy computation of truth of decision rules, and consequently provides an easy way to simplify the rules. A disadvantage is that inconsistent records are abandoned when deducing decision rules in the beginning. This may cause some useful data to be excluded because of a small number of noisy data. One possible solution for that is to give some weight to the rules according to the number of same records.

In section 4.2, we proposed a set of key distribution schemes to apply dynamic control over the inference problem. Some of these schemes use one small key set used by the database system, thus the database provides the maximum flexible accessibility. Some of them use multiple key sets, thus blocking one object attacks (blocking one object attack means that one object in an inference channel cannot be accessed by any user after other objects have been queried by users), can be prevented. Although this is decided by the nature of the inference control problem, we have put much effort into seeking solutions for it.

One important factor for a inference control scheme is the efficiency of the scheme. For the efficiency, we mainly considered two things. One is the availability of the information and the other is query processing time. It does not make much sense to us that a system extremely limits the users' accessibility to protect against inference attacks. Our scheme allows a user to access as much information as possible. Since we use keys to control the system, the number of keys and the size of a key determine the query processing time. Therefore we tried to reduce the key space used in the schemes. One of the ideas is to keep as many as possible the keys in database system. In this way, keys with small size can be used because there is no key leaking problems.

For future research, as the technology of database management system develops, and the security of real applications increases, techniques handling the inference problem should be improved to adopt to the new feathers. In addition, we also feel that the inference problem occuring in data mining and web-based inference would be interesting topics.

Bibliography

[1] D. Agrawal, C. C. Aggarwal. *On the design and quantification of privacy preserving data mining algorithms.* Symposium on Principles of Database Systems, 2001.

[2] R. Agrawal, R. Srikant. *Privacy-preserving data mining.* Proc. of the ACM SIGMOD Conference on Management of Data, pp. 439-450, 2000.

[3] L. J. Binns. *Inference through secondary path analysis.* Database Security VI: Status and Prospects, pp. 195-209, 1993.

[4] L. Buczkowski. *Database inference controller.* Database Security III: Status and Prospects, pp. 311-322, 1990.

[5] E. Bertino, S. Castano, E. Ferrari, M. Mesiti. *Controlled acess and dissemination of XML documents.* Workshop on Web Information and Data Management, pp. 22-27, 1999.

[6] T. Berners-Lee, J. Hendler, O. Lassila. *The semantic web.* Scientific American, May 2001.

[7] C. Clifton. *Using sample size to limit exposure to data mining.* Journal of Computer Security, 8(4), 2000.

[8] C. Clifton, D. Marks. *Security and privacy implications of data mining.* Workshop on Data Mining and Knowledge Discovery, number 96-08, pp.15-19, 1996.

[9] L. Chang, I. S. Moskowitz. *Bayesian methods applied to database inference problem.* Database Security XII, pp. 237-251, 1998.

[10] L. Chang, I. S. Moskowitz. *Parsimonious downgrading and decision trees applied to the inference problem.* Proc. New Security Paradigms Workshop, 1998.

[11] L. Chang, I. S. Moskowitz. *An integrated framework for database inference and privacy protection.* Data and Application Security (Proc. IFIP WG 11.3), pp. 161-172, 2000.

[12] L. Chang, I. S. Moskowitz. *A study of inference problem in distributed database systems.* Data and Application Security (Proc. IFIP WG 11.3), 2002.

[13] X. Chen, R. Wei. *Key schemes for the inference control problem.* Accepted by the IEEE International Conference on Information Technology: Coding and Computing 2005.

[14] X.Chen, R. Wei. *A Scheme for Inference Problems Using Rough Sets and Entropy.* Submitted.

[15] D. Denning. *Cryptography and Data Security.* Addison-Wesley, Mass, 1982.

[16] D. Denning. *Commutative filters for reducing inference threats in multilevel database systems.* Proc. IEEE Symposium on Security and Privacy, pp. 134-146, 1985.

[17] D. Denning, E. Dorothy, M. Morgenstern. *Military database technology study: AI techniques for security and reliability.* SRI tech. report, Aug. 1986.

[18] G. Duncan, S. Fienberg. *Obtaining information while preserving privacy: a markov perturbation method for tabular data.* Statistical Data Protection, pp. 351-362, 1998.

[19] A. Deutch, M. Fernandez, D. Florescu, A. Lecy, D. Suciu. *A query language for XML.* Computer Networks, 31(11-16):1155-1169, 1999.

[20] H. Delugach, T. Hinke. *Wizard: A database inference analysis and detection system.* IEEE Trans. on Knowledge and Data Engineering, 8(1):56-66, 1996.

[21] G. Denker, J.R. Hobbs, D. Martin, S. Narayanan, R. J. Waldinger. *Accessing information and services on the DAML-enabled web.* SemWeb, 2001.

[22] S. Decker, S. Melnik, F. van Harmelen, D. Fensel, M. C. A. Klein, J. Broekstra, M. Erdmann, I. Horrocks. *The semantic web: The roles of XML and RDF.* IEEE Internet Computing, 4(5):63-74, 2000.

[23] E. Damiani, S. D. C. di Vimercati, S. Paraboschi, P. Samaıati. *Design and implementation of an access control provessor for XML documents.* WWW9/Computer Networks, 33(1-6):59-75, 2000.

[24] C. Farkas, S. Fenner, M. Valtorta. *Medical privacy versus data mining.* Proc. Fifth Multiconference on Systemics, Cyberbetics and Informatics, pp. 194-200, 2001.

[25] C. Farkas, S. Jajodia. *The inference problem: a survey.* ACM KIGKDD Explorations Newletter, Volume 4 Issue 2, December 2002.

[26] T. D. Garvey, T. F. Lunt, M. E. Stickel. *Abductive and approximate reasoning models for characterizing inference channels.* Proc. Computer Security Workshop IV, pp. 118-126, 1991.

[27] J. Goguen, J. Meseguer. *Unwinding and inference control.* Proc. IEEE Symposium on Security and Privacy, pp. 75-86, 1984.

[28] G. Gardarin, F. Sha. *Using conceptual modeling and intelligent agents to integrate semi-structured documents in federated databases.* Lecture Notes in Computer Science, 1565:87-99, 1999.

[29] T. H. Hinke. *Inference aggregation detection in database management systems.* Proceedings of the 1988 IEEE Symposium on Security and Privacy, pp. 96-106, 1988.

[30] T. Hinke. *Database inference engine design approach.* Database Security II: Status and Prospects, 1990.

[31] H. Hosmer. *Security is fuzzy!.* Proceedings of New Security Paradigms Workshop, pp. 175-184, 1993.

[32] T. H. Hinke, H. S. Delugach, A. Chandrasekhar. *A fast algorithm for detecting second paths in database infeence analysis.* Computer Security,3(2,3):147-168, 1995.

[33] T. H. Hinke, H. S. Delugach, and R. P. Wolf. *Iliad: An integrated laboratory for inference analysis and detection.* In Database Security IX: Status and Prospects, pp. 333-348, 1995.

[34] T. Hinke, H. Delugach, R. Wolf. *A framework for inference directed data mining.* Proc. 10th IFIP WG11.3 Workshop on Database Security, pp. 229-239, 1996.

[35] T. Hinke, H. Delugach, R. Wolf. *Proteciting datbases from inference attacks.* Computers and Security, 16(8):687-708, 1997.

[36] J. Hale, S. Shenoi. *Catalytic inference analysis: Detecting inference threat due to knowledge discovery.* Proc. of the 1997 IEEE Symposium on Security and Privacy, pp. 188-199, 1997.

[37] J. Hale, J. Threet, S. Shenoi. *A practical formalism for imprecise inference control.* IFIP Trans. Computer Science and Technology, 60:139-156, 1994.

[38] S. Jajodia, C. Meadows. *Inference problems in multilevel secure database management systems.* Information Security: An Integrated Collection of Essays, M. Abrams et al., eds., IEEE Computer Society Press, pp. 570-584, 1995.

[39] W. Klosgen. *Knowledge discovery in databases and data privacy.* IEEE Expert, April 1995.

[40] M. Kudo, S. Hada. *XML document security based on provisional authorization.* Proc. of the 7th ACM Conference on Computer and Communication Security, November 2000.

[41] R. Kumar, S. Rajagopalan, A. Sahai. *Coding constructions for blocklisting problems without computational constructions.* In Advances in Cryptology - Crupto'99, pp. 609-623.

[42] T. Keefe, M. Thuraisingham, W. Tsai. *Secure query-peocessing strategies.* IEEE Computer, Vol 22, No. 3, pp. 63-70, 1989.

[43] T. Lunt. *Current issues in statistical database security.* Database Security V: Status and Prospects, IFIP WG 11.3, pp.381-385, 1991.

[44] J. Van Lint. *Introduction to Coding Theory.* Springer-Verlag, 1999.

[45] T. Y. Lin, T. H. Hinke, D. G. Marks, B. Thuraisingham. *Security and data mining.* Database Security Vol.9: Status and Prospects, pp. 391-399, 1996.

[46] D. G. Marks. *Inference in mls database systems.* IEEE Transactions on Knowledge and Data Engineering, 8(1):46-55, 1996.

[47] M. Morgenstern. *Controlling logical inference in multileve database systems.* Proc. IEEE Symposium on Security and Privacy, pp. 245-255, 1988.

[48] D. Marks, A. Motro, S. Jajodia. *Enhancing the controlled disclosure of sensitive information.* Proc. European Symposium on Research in Computer Security, 1996.

[49] R. Motwani, P. Raghavan. *Randomized algorithms.* Cambridge University Press, 2000.

[50] S. Mazumdar, D. Stemple, T. Sheard. *Resolving the tension between integrity and security using a theorem prover.* Proc. ACM Int'l Conf. Management of Data, pp. 233-242, 1988.

[51] D. O'Leary. *Some privacy issures in knowledge discovery: OECD personal privacy guidelines.* IEEE Experts, April 1995.

[52] G. Ozsoyoglu, T. Su. *On inference control in semantic data models for statistical databases.* Journal of Computer and System Sciences, 40(3): 405-443, 1990.

[53] Z. Pawlak. *Rough sets: theoretical aspects of reasoning about data.* Kluwer Academic Publisher, 1992.

[54] G. Piatetsky-Shapiro. *Knowledge discovery in databases vs. personal privacy.* IEEE Expert, April 1995.

[55] Y. Papakonstantinou, V. Vianu. *DTD inference for views of XML data.* Proceedings of the 19th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp. 35-46, 2000.

[56] J. R. Quinlan. *Discovering rules by induction from large numbers of examples: a case study.* Expert Systems in the Micro-Electronic Age. D. Michie, editor. Edinburg Univ. Press, 1979.

[57] J. R. Quinlan. *Programs for machine learning.* Morgan-Kaufmann. 1993.

[58] X. Qian, M. Stickel, P. Karp, T. Lunt and T. Garvey. *Detection and elimination of inference channels in multilevel relational database systems.* In IEEE Symposium on Security and Privacy, 1993.

[59] S. Roman. *Coding and information theory.* Springer-Verlag. 1992.

[60] C. Shannon. *A mathematic theory of communication.* Bell System Technical Journal, Vol. 27, pp. 379-423, 623-656, 1948.

[61] J. Sowa. *Conceptual structures: information processing in minds and machines.* Addison Wesley, 1984.

[62] G. Smith. *Modeling security-relecant data semantics.* Proc. IEEE Symposium on Research in Security and Privacy, pp. 384-391, 1990.

[63] M. Stickel. *Elimination of inference channels by optimal upgrading.* Proc. IEEE Symposium on Research in Security and Privacy, pp. 168-174, 1994.

[64] P. Selfridge. *Privacy and knowledge discovery in databases.* IEEE Expert, April 1995.

[65] J. Staddon. *Dynamic inference control.* DMKD03: 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, 2003.

[66] A. Stoica, C. Farkas. *Secure XML views.* IFIP WG11.3, Working Conference on Database and Application Security, 2002.

[67] T. Su, G. Ozsoyoglu. *Inference in MLS database systems.* IEEE Trans. Knowledge and Data Eng., 3(4):474-485, 1991.

[68] P. Stachour, B. Thuraisingham. *Design of LDV: A multilevel secure relational database management system.* IEEE Trans. Knowledge and Data Engineering, 2(2):190-209, 1990.

[69] B. Thuraisingham. *Security checking in relational database management systems augmented with inference engines.* Computers and Security, 6:479-492, 1987.

[70] B. Thuraisingham. *Towards the design of a secure data/knowledge base management system.* Data Knowledge and Engineering, 1990.

[71] B. Thuraisingham. *Recursion theoretic properties of the inference problem in database security.* MTP 291, MITRE Corp., Bedford, Mass, May 1990.

[72] B. Thuraisingham. *The use of conceptual structures for handling the inference problem.* Database Security V, pp. 333-362, 1992.

[73] B. Thuraisingham. *Security issues for data warehousing and data mining.* DBSec, 1996.

[74] J. Tracy, L. Chang, I. S. Moskowitz. *An-Agent-Based approach to inference prevention in distributed database systems.* Proc. ICTAI, Washington DC, pp 413-422, 2002.

[75] S. M. Ulam. *Adventures of a mathematician.* Charles Scribner's Sons, New York, 1976.

[76] S. Urban, L. Delcambre. *Constraint analysis for specifying perspectives of class objects.* Proceeding of the 5th IEEE International Conference on Data Engineering. 1989.

[77] H. Winston. *Artificial Intelligence.* 3rd edition. Addison-Wesly. 1991.

[78] D. Woodruff, J. Staddon. *Private inference control.* Proceedings of 11th ACM Conference on Computer and Communications Security. 2004.

[79] R. Yip, K. Levitt. *Data level infernce detection in database systems.* Proc. of the 11th IEEE Computer Security Foundation Workshop, pp. 179-189, 1998.