NCSC TECHNICAL REPORT - 005

Volume 1/5

Library No. S-243,039

FOREWARD

This report is the first of five companion documents to the *Trusted Database Management System Interpretation of the Trusted Computer System Evaluation Criteria*. The companion documents address topics that are important to the design and development of secure database management systems, and are written for database vendors, system designers, evaluators, and researchers. This report addresses inference and aggregation issues in secure database management systems.

May 1996

Keith F. Brewster
Acting Chief, Partnerships and Processes

ACKNOWLEDGMENTS

The National Computer Security Center extends special recognition to the authors of this document. The initial version was written by Victoria Ashby and Sushil Jajodia of the MITRE Corporation. The final version was written by Gary Smith, Stan Wisseman, and David Wichers of Arca Systems, Inc.

The documents in this series were produced under the guidance of Shawn P. O’Brien of the National Security Agency, LouAnna Notargiacomo and Barbara Blaustein of the MITRE Corporation, and David Wichers of Arca Systems, Inc.

We wish to thank the members of the information security community who enthusiastically gave of their time and technical expertise in reviewing these documents and in providing valuable comments and suggestions.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE **PAGE**

LIST OF TABLES

TABLE **PAGE**

SECTION 1

INTRODUCTION

This document is the first volume in the series of companion documents to the *Trusted Database Management System Interpretation of the Trusted Computer System Evaluation Criteria* [TDI 91; DoD 85]. This document examines inference and aggregation issues in secure database management systems and summarizes the research to date in these areas.

1.1 BACKGROUND AND PURPOSE

In 1991 the National Computer Security Center published the *Trusted Database Management System Interpretation* (TDI) *of the Trusted Computer System Evaluation Criteria* (TCSEC). The TDI, however, does not address many topics that are important to the design and development of secure database management systems (DBMSs). These topics (such as inference, aggregation, and database integrity) are being addressed by ongoing research and development. Since specific techniques in these topic areas had not yet gained broad acceptance, the topics were considered inappropriate for inclusion in the TDI.

The TDI is being supplemented by a series of companion documents to address these issues specific to secure DBMSs. Each companion document focuses on one topic by describing the problem, discussing the issues, and summarizing the research that has been done to date. The intent of the series is to make it clear to DBMS vendors, system designers, evaluators, and researchers what the issues are, the current approaches, their pros and cons, how they relate to a TCSEC/TDI evaluation, and what specific areas require additional research. Although some guidance may be presented, nothing contained within these documents should be interpreted as criteria.

These documents assume the reader understands basic DBMS concepts and relational database terminology. A security background sufficient to use the TDI and TCSEC is also assumed; however, fundamentals are discussed whenever a common understanding is important to the discussion.

1.2 SCOPE

This document addresses inference and aggregation issues in secure DBMSs. It is the first of five volumes in the series of TDI companion documents, which includes the following documents:

- *Inference and Aggregation Issues in Secure Database Management Systems*
- *Entity and Referential Integrity Issues in Multilevel Secure Database Management Systems* [Entity 96]
- *Polyinstantiation Issues in Multilevel Secure Database Management Systems [Poly 96]*
- *Auditing Issues in Secure Database Management Systems [Audit 96]*
- *Discretionary Access Control Issues in High Assurance Secure Database Management Systems* [DAC 96]

This series of documents uses terminology from the relational model to provide a common basis for understanding the concepts presented. This does not mean that these concepts do not apply to other database models and modeling paradigms.

1.3 INTRODUCTION TO INFERENCE AND AGGREGATION

Many definitions of inference and aggregation have been suggested in the literature. In fact, one of the challenges to understanding inference and aggregation is that there are different (but sometimes similar) notions of what they are. We will discuss the differences and similarities at length in later sections. At this point, the following general definitions are sufficient:

> *Inference or the inference problem* is that of users deducing (or inferring) higher level information[1] based upon lower, visible data [Morgenstern 88].
>
> *Aggregation* or the *aggregation problem* occurs when classifying and protecting collections of data that have a higher security level than any of the elements that comprise the aggregate [Denning 87].

Inference and aggregation problems are not new, nor are they only applicable to DBMSs. However, the problems are exacerbated in multilevel secure (MLS) DBMSs that label and enforce a mandatory access control (MAC) policy on DBMS objects [Denning 86a].

Inference and aggregation differ from other security problems since the leakage of information is not a result of the penetration of a security mechanism but rather it is based on the very nature of the information and the semantics of the application being automated [Morgenstern 88].

1.4 AUDIENCES OF THIS DOCUMENT

This document is targeted at four primary audiences: the security research community, database application developers/system integrators, trusted product vendors, and product evaluators. Inference and aggregation are problems based on the value and semantics of data for the part of the enterprise being automated. Understanding inference and aggregation is important for all of the audience categories, but the audience for whom this document is most relevant includes those involved in designing and engineering automated information systems (AIS)--in particular, the database application developers/system integrators. In general, this document is intended to present a basis for understanding the issues surrounding inference and aggregation especially in the context of MLS DBMSs. Implemented approaches and ongoing research are examined. Members of the specific audiences should expect to get the following from this document:

<u>Researcher</u>

This document describes the basic issues associated with inference and aggregation. Important

1. We recognize that information, data, and knowledge may be considered by many to be distinctly different things. On the other hand, different terms may be legitimately used to describe the same collection of characters. For example, a "rule" for a knowledge-based system would constitute data for the knowledge engine, but would be considered information or knowledge when accessed by a human. For the purposes of this document, there is no need to distinguish between data and information; thus, the terms will be used interchangeably.

research contributions are discussed as various topics are examined. By presenting current theory and debate, this discussion will help the research community understand the scope of the issue and highlight approaches and alternative solutions to inference and aggregation problems. For additional relevant work, the researcher should consult two associated TDI companion documents: *Polyinstantiation Issues in Multilevel Secure Database Management Systems* [Poly 96] and *Entity and Referential Integrity Issues in Multilevel Secure Database Management Systems* [Entity 96].

Database Application Developer/System Integrator

This document highlights the need for analysis of the application-dependent semantics of an application domain in order to understand the impact of inference and aggregation on MLS applications. It describes the basic issues and current research into providing effective solutions.

Trusted Product Vendor

This document describes the types of mechanisms from the research literature that can be incorporated into products to assist application developers with enforcing aggregation-based security policies. Research on tools to support inference analysis provides the basis for additional tools that might be provided with a DBMS product.

Evaluator

This document presents an understanding of inference and aggregation issues and how they relate to the evaluation of a candidate MLS DBMS implementation.

1.5 ORGANIZATION OF THIS DOCUMENT

The organization of the remainder of this document is as follows:

- Section 2 discusses several *first principles* that form the foundation for reasoning about inference and aggregation. Included in Section 2 are subsections that define and explain many aspects of inference and aggregation, present relevant security terminology (e.g., comparing inference to direct disclosure), and describe two fundamental processes: the traditional security analysis paradigm and the database security engineering process.

- Section 3 presents a framework for reasoning about inference and aggregation based on the two processes described in Section 2. Section 3 also introduces a detailed example that is used throughout the document to illustrate the material. Section 3 concludes with a matrix that shows how each research effort discussed in Sections 4 and 5 relate to the framework.

- Section 4 summarizes research efforts on inference.

- Section 5 summarizes research on aggregation.

- Section 6 summarizes the contents of this document.

SECTION 2

BACKGROUND

Inference and aggregation are different but related problems. In order to understand what they are, what problems they embody, and their relationship, it is essential to revisit several *first principles*. In Section 2.1, we give further descriptions and discuss inference and aggregation in some detail, including examples. Some important terminology and security concepts are covered in Section 2.2. The next two sections describe fundamental processes that form the basis for reasoning about inference and aggregation: the traditional security analysis paradigm (Section 2.3) and the database security engineering process (Section 2.4). The final section summarizes the operational and TCSEC requirements for inference and aggregation.

Although much of what will be discussed in this section will be considered by some as intuitively obvious, it is these principles that are often forgotten and lead to misunderstandings. Even experienced researchers and practitioners should review this section.

2.1 INFERENCE AND AGGREGATION - DEFINED AND EXPLAINED

This section provides further definitions and explanations of inference and aggregation problems. Many examples are given to illustrate the relationship and complexity of these problems.

2.1.1 Inference

Inference, the inferring of new information from known information, is an everyday activity of most humans. In fact, inference is a topic of interest for several disciplines, each from its own perspective. In the context of this document:

> Inference addresses the problem of preventing authorized[2] users of an AIS from inferring unauthorized information.

Preventing unauthorized inferences is complicated both by the variety of ways in which humans infer new information and by the wide variety of information they use to make inferences.

Some researchers characterize the inference problems in terms of *inference channels* [Morgenstern 88; Meadows 88a]. SRI's research identifies three types of inference channels based on the degree to which **HIGH** data may be inferred from **LOW** data [Garvey 94]:

- Deductive Inference Channel. The most restrictive channel, requiring a formal deductive proof (described by propositional or first-order logic) showing that **HIGH** data can be derived from **LOW** data.
- Abductive Inference Channel. A less restrictive channel where a deductive proof could be completed <u>assuming</u> certain **LOW**-level axioms.

---

2. Explicit meanings of "authorized" user and "unauthorized" information are described in Section 2.2.

- Probabilistic Inference Channel. This channel occurs when it is possible to determine that the likelihood of an unauthorized inference is greater than an acceptable limit.

These descriptions of inference channels combine the method by which the inference is made and the information used to make the inference. Both of these areas of complexity are discussed further in the following paragraphs. Section 2.1.1.3 identifies related disciplines and how they address inference.

2.1.1.1 Inference Methods

One of the factors that makes the inference problem so difficult is that there are many methods by which a human can infer information. The following list [Ford 90] illustrates the variety of inference strategies that can be used:

- <u>Inference by Deductive Reasoning</u> New information is inferred using well-formed rules, either through classical or non-classical logic deduction. Classical deductions are based on basic rules of logic (e.g., assertion that "A is true" and if "A then B" deduces "B is true"). Non-classical logic-based deduction includes probabilistic reasoning, fuzzy reasoning, non-monotonic reasoning, default reasoning, temporal logic, dynamic logic, and modal logic-based reasoning.

- <u>Inference by Inductive Reasoning</u> Well formed rules are utilized to infer hypotheses from observed examples.

- <u>Inference by Analogical Reasoning</u> Statements such as "X is like Y" are used to infer properties of X given the properties of Y.

- <u>Inference by Heuristic Reasoning</u> New information is deduced using various heuristics. Heuristics are not well defined and may be a "rule of thumb."

- <u>Inference by Semantic Association</u> The association between entities is inferred from the knowledge of the entities themselves.

- *Inferred Existence* One can infer the existence of entity Y from certain information (e.g., from the information, "John lives in Boston," one can infer that "there is some entity called John").

- <u>Statistical Inference</u> Information about an individual entity is inferred from various statistics compared on a set of entities.

2.1.1.2 Inference Information

The other factor that makes preventing unauthorized inferences within a DBMS a difficult problem is the wide variety of information that can be used to make an inference. The following are some types of information that can be used to make inferences:

- Schema metadata including relation and attribute names

- Other metadata such as constraints (e.g., value constraints or other constraints that may be implemented through triggers or procedures)

- Data stored in the relations (i.e., the value of the data)
- Statistical data derived from the database
- Other derived data that can be determined from data in the database
- The existence or absence of data (as opposed to the value of the data)
- Data disappearing (e.g., upgraded) or changing value over time
- Semantics of the data that are not stored in the database but are known within the application domain
- Specialized information about the application domain (both process and data) not stored in the database
- Common knowledge, and
- Common sense

Denning [Denning 86a] and many other researchers adopt a so-called “closed-world assumption” that limits inference to the database. With this limitation, both the information used to make the inference and the inferred information must be represented in the database. The first seven types of information described above encompass information that would be included using this closed-world assumption. Unfortunately, in the real world in which a DBMS must operate, significant amounts of the remaining types of information will be used by an adversary to attempt to infer unauthorized information.

2.1.1.3 Related Disciplines

Inference in the context of this document focuses on the information contained in a DBMS and preventing inference of unauthorized information by authorized users of the DBMS. Inference is also a key issue in several other disciplines including:

- <u>Operations Security (OPSEC)</u> OPSEC is concerned with denying external adversaries the ability to infer classified information by observing unclassified indicators. The key difference is that the adversary is external to the organization, while this document addresses individuals that are internal to the organization (i.e., DBMS users).
- <u>Artificial Intelligence (AI)</u> The ability to infer new facts from existing knowledge is a positive goal of several of the sub-disciplines of AI. The AI research community is looking for better ways to represent knowledge and approaches for inferring new information.
- <u>Uncertainty Management</u> The inverse problem of preventing inferences is the need for individuals, such as intelligence analysts, to infer information about an adversary based upon evidence that can be collected. There are several approaches to evaluating evidence in an effort to quantify one’s confidence in the evidence and the resulting inferred information [Schum 87].

2.1.2 Aggregation

Like the word *inference*, the word *aggregation* has multiple meanings in the information technology community. An *aggregation*, or aggregate, is a collection of separate data items. In the database context, data aggregation normally has a positive connotation. Aggregation is an important and desirable side effect of organizing information into a database. Information has a greater value when it is gathered together, made easily accessible, and structured in a way that shows relationships between pieces of data [National Academy Press 83]. In some sense, every database is an aggregation of data.

The term aggregation has a different, more common use in the commercial database management community, where it usually refers to the use of aggregate operators. The five standard operations are sum, average, count, maximum, and minimum. Others such as standard deviation or variance may also be found in query languages [Ullman 88]. These operators are applied to columns of a relation to obtain a single quantity. They allow conclusions to be drawn from the mass of data present in a database.

The aggregate operators in commercial DBMSs relate closely to the operations performed on statistical databases. Statistical databases are used to answer queries dealing with sums, averages, and medians, and use aggregation to hide the separate data items found in individual records. The concern in a statistical database is for security in the sense of privacy, not classification. Security in statistical databases has a large body of literature and will only be discussed briefly in this document.

In the context of this document:

> Aggregation involves the situation where a collection of data items is explicitly considered to be classified at a higher level than the classification assigned to the individual data elements [Lunt 89].

Denning has identified two classes of the aggregation problem [Denning 87]. We describe them here in terms of entities and their attributes.

- <u>Attribute Associations</u> Associations between instances of *different* data items of which there are two sub-classes.

  <u>Associations between Entities</u> Associations between instances of attributes of different entities.

  <u>Associations between Attributes of an Entity</u> Associations between instances of different attributes of an entity.

- <u>Size-based Record Associations</u> Associations among multiple instances of the *same* entity.

We will use the current terminology to represent these classes of aggregation: *Data Association* [Jajodia 92] for Attribute Associations (both Inter- and Intra-Entity) and *Cardinal Aggregation* [Hinke 88] for Size-based Record Associations.

The difference between data association and cardinal aggregation is important. Data association refers to aggregates of instances of different data objects; cardinal aggregation refers to aggregates of instances of the same type of entity. Distinguishing between data association and cardinal aggregation as two types of aggregation is also important since data association can be effectively controlled using database design techniques with labeled data [Denning 87]. On the other hand, effective mechanisms to enforce cardinal aggregation policies are still a research issue.

2.1.2.1 Data Association Examples

A traditional example of intra-entity data association involves salary: the identity of an employee and the actual number that represents the employee's salary may both be unclassified, but the association between a particular employee and the employee's salary is considered sensitive. This is an instance of intra-entity data association as defined above: employee name (or other identifying attributes such as employee number) and salary are both attributes of the employee entity.

An example of inter-entity data association is the following: the identity of commercial companies and government agencies are unclassified, but the association of a particular company as the contractor for a particular government agency may be considered classified. In this case, the data association is between two entities, commercial company and government agency.

2.1.2.2 Cardinal Aggregation Examples

The traditional example of cardinal aggregation is the intelligence agency telephone book: individual entries are unclassified but the total book is classified [Lunt 89]. The challenge with this example is understanding what information the agency is trying to protect with the policy. Is it the name and number of staff personnel (and therefore capability) of each (or some) organizational element? The total staffing size of the agency? Identification of special employees? Special skills? A clear understanding of what information needs to be protected must precede a cardinal aggregation policy.

Another example of a cardinal aggregation policy relates to an Army troop list, that is, a list of Army organizations [Smith 91]. Information about each Army organization is unclassified (even if the organization's mission is classified), but the list as a whole is classified. Getting someone to determine how many organizations can be aggregated and still remain unclassified information is difficult. If the intent is to protect the Army's war fighting capability, then aggregating the combat arms units (as opposed to the support and civilian-based organizations) will directly disclose[3] that information. Of course, one has to know about the war fighting capabilities of each type of combat arms unit, which is unclassified information and can be obtained externally to the troop list. Unfortunately, there may be other ways to infer the unauthorized information. For example, if one can determine the tactical communications capability (i.e., aggregation of communications units), can one then infer the war fighting capability? And what about any number of other aggregates of units where there is a reasonable understanding of their relationship to war fighting capability? It quickly becomes a difficult problem to describe, let alone to provide mechanisms to solve this type of aggregation problem. Yet keeping information about individual units unclassified is essential

3. Explicit meanings for disclosing information directly or through an inference are discussed in Section 2.2.

for operational efficiency.

A final example is network management data: information about outages of individual circuits is not sensitive, but the aggregation of information about all circuit outages for a network is sensitive. In this case, one can postulate that this cardinal aggregation policy is attempting to protect the identification of vulnerabilities of the network from an adversary that might want to disrupt or render the network inoperative. With information about only one or two circuit outages, an adversary cannot mount an effective attack to bring the network down. On the other hand, if the adversary knows all the circuit outages, the network may have a temporary single point of failure, a vulnerability that the adversary can exploit. It is operationally impractical to treat information about each circuit as sensitive, yet in the face of a credible threat, it is important to protect the aggregate information.

All of these example cardinal aggregation policies were established as a trade-off between the need to protect sensitive information versus the need for operational efficiency. To some researchers, cardinal aggregation policies are considered fundamentally unsound [Meadows 91). Thus, it should not be surprising that effective mechanisms to implement what some consider an unsound policy elude the community.

2.2 SECURITY TERMINOLOGY

This section discusses several security-relevant terms and concepts. We distinguish between two ways in which information can be revealed:

- Direct Disclosure. Classified[4] information is directly revealed to unauthorized individuals.
- Inference. Classified information may be deduced by unauthorized individuals with some level of confidence often less than 100%.

The primary distinction, then, between direct disclosure and inference is whether. the information is revealed (direct disclosure) or must be deduced (inference). For example, if a user at **LOW**[5] can execute a query to a DBMS[6] and receive a response where the data displayed to the **LOW** user is actually classified **HIGH** (i.e., based on an explicit classification rule), then this is a direct disclosure problem not an inference problem. In most instances of inference, the adversary[7] has some uncertainty with respect to the accuracy of the inference of the <u>unauthorized</u> information.

For the purposes of this document, we do not distinguish between where the **HIGH** data that is

4. We use the term *classified* even though the concepts discussed in this document are applicable to information that is sensitive for other than national security reasons, such as, proprietary or “bet your company” information for private sector companies.
5. For the purposes of this report, we define a simple classification scheme with three hierarchical levels: UNCLASSIFIED, LOW, HIGH.
6. This example assumes the DBMS correctly enforces a mandatory access control policy over the classification scheme from footnote 5, using labels on DBMS objects (i.e., all storage objects which include named objects).
7. We use “adversary” as a generic term to refer to the threat agent that tries to obtain unauthorized information. In general, the adversary can be an authorized user (e.g., employee) or external individual (e.g., hacker, intelligence agent).

directly disclosed or inferred is located (i.e., in the database or not). The important consideration is that **HIGH** data was directly disclosure or inferred. This consideration is independent of whether the **HIGH** data is actually stored in the database or it exists somewhere else in the application domain of discourse.

Data association policies are stated to prevent direct disclosure of classified information related to an association between data elements. In most cases, cardinal aggregation policies are stated to prevent inferences that could be made from the aggregate data.

The discussion above refers generically to “unauthorized” individuals. In fact, we must distinguish between two types of unauthorized access to information:

- based on authorized access to classified information (i.e., a label-based policy); and
- based on informal “need-to-know” policies.

“Need-to-know” is a well-known concept and is defined in the DoD Information Security Regulation [5100.1-R 86] as:

- “A determination made by a possessor of classified information that a prospective recipient, in the interest of national security, has a requirement for access to, or knowledge, or possession of the classified information in order to accomplish lawful and authorized government purposes.”

There are two distinct types of need-to-know: “informal” (sometimes called discretionary), and “formal” need-to-know. With informal need-to-know, individuals may have the clearance (i.e., are considered trustworthy to properly protect the information) but not the job-related need to actually access the information. Informal need-to-know is enforced in the people and paper world by the individual who physically controls access to a classified document. In an AIS, informal need-to-know is normally enforced by discretionary access control (DAC) mechanisms as described in the TCSEC. Formal need-to-know is enforced through compartments (or other release markings) to which individuals are formally granted access. Mandatory access control (MAC) categories are used in an AIS to enforce access based on formal need-to-know.

Direct disclosure is normally only thought of as being a problem when classified information is revealed to an individual without the appropriate clearance, including formal access to compartments (i.e., formal need-to-know). Given that U.S. Government information is classified based on laws or executive orders, direct disclosure of classified information is punishable by imprisonment. On the other hand, disclosing information in violation of an informal need-to-know policy is normally not considered a compromise of classified information, since the individual has the appropriate clearance to see the information and is trusted to properly protect it. Thus, a violation of an informal need-to-know policy may result in an employee being fired or sued, but not imprisoned.

Although an authorized user can attempt to infer information for which the user does not have an informal need-to-know, research efforts to date in inference and aggregation have dealt exclusively with direct disclosure or inference based on compromise of classified information, not on violations of informal need-to-know policies. As such, this document focuses almost exclusively on protecting classified information against inference and aggregation threats from individuals who are not cleared to receive this information.

2.3 THE TRADITIONAL SECURITY ANALYSIS PARADIGM

To better understand inference and aggregation and how they relate to each other, they need to be discussed in the context of the traditional security analysis paradigm shown in Figure 2.1. A brief description of the activities in the paradigm follows:

- <u>Identify information protection requirements</u> The protection requirements traditionally addressed by the security community relate to confidentiality. However, information protection requirements for integrity and availability should also be specified. Although, the inference and aggregation problems strictly address confidentiality protection requirements, solutions to these problems typically adversely impact the ability to meet the integrity and availability requirements.

- <u>Identify vulnerabilities</u> Given a set of protection requirements, one must identify the vulnerabilities that exist which could be exploited to prevent those requirements from being satisfied by the system. As the next section shows, vulnerabilities need to be assessed continuously throughout the design and engineering process.

- <u>Identify threats</u> Given identified vulnerabilities, one must identify credible threat agents that can exploit those vulnerabilities. For inference and aggregation, the primary threat is the insider; an authorized user of the system who is authorized access to some, but not all, information in the system. Malicious code is a secondary threat.

- <u>Conduct a risk analysis</u> The risk analysis must take into account the likelihood of a threat agent actually exploiting a vulnerability. The risk analysis considers tradeoffs between effectiveness, cost, and the ability to meet other requirements (e.g., integrity and availability) to determine appropriate controls.[8]

- <u>Select controls</u> Based on the risk analysis, controls are selected to eliminate or reduce a threat agent's ability to exploit the vulnerability. Note that some controls are selected to <u>prevent</u> while others are selected to <u>reduce</u> or <u>detect</u> a threat agent's ability to exploit the vulnerability. This distinction is important in that most cases dealing with inference involve determining during the risk analysis the likelihood of an individual inferring unauthorized information given a particular set of controls. The goal is to establish controls that provide an acceptable level of risk at an affordable cost, where cost is cast in terms of both money and adverse impact on functionality, ease of use, and performance.

---

8. We use the term "control" although other terms, such as countermeasures and safeguards, are also commonly used.

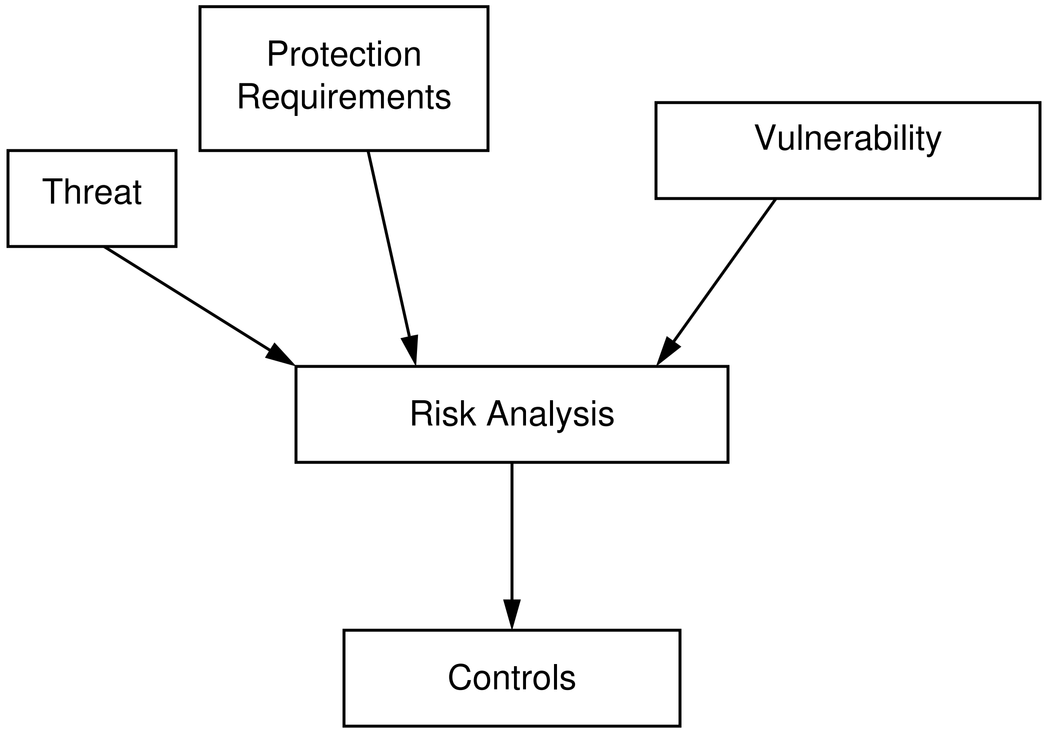

Figure 2.1: Traditional Security Analysis Paradigm

Details of how this process is related to inference and aggregation are discussed in the next section.

2.4 THE DATABASE SECURITY ENGINEERING PROCESS

The process of implementing an AIS with a database that meets the specified security requirements is shown in Figure 2.2. The process should begin with a clear understanding of what information needs to be protected, in this case from unauthorized disclosure. If one does not have a clear understanding of what has to be protected, then it makes secure database design difficult. Unfortunately, many inference and aggregation problems are stated at the database design level (e.g., with sensitivity labels on tables) without a description of what information is to be protected from disclosure.

Given a precise statement of the information to protect, there are two distinct aspects of establishing rules for implementing those confidentiality protection requirements:

- **classification rules** must be established that specify the confidentiality properties of the information, and

- **access control rules** must be established that specify which authorized users can access information based on the classification rules.

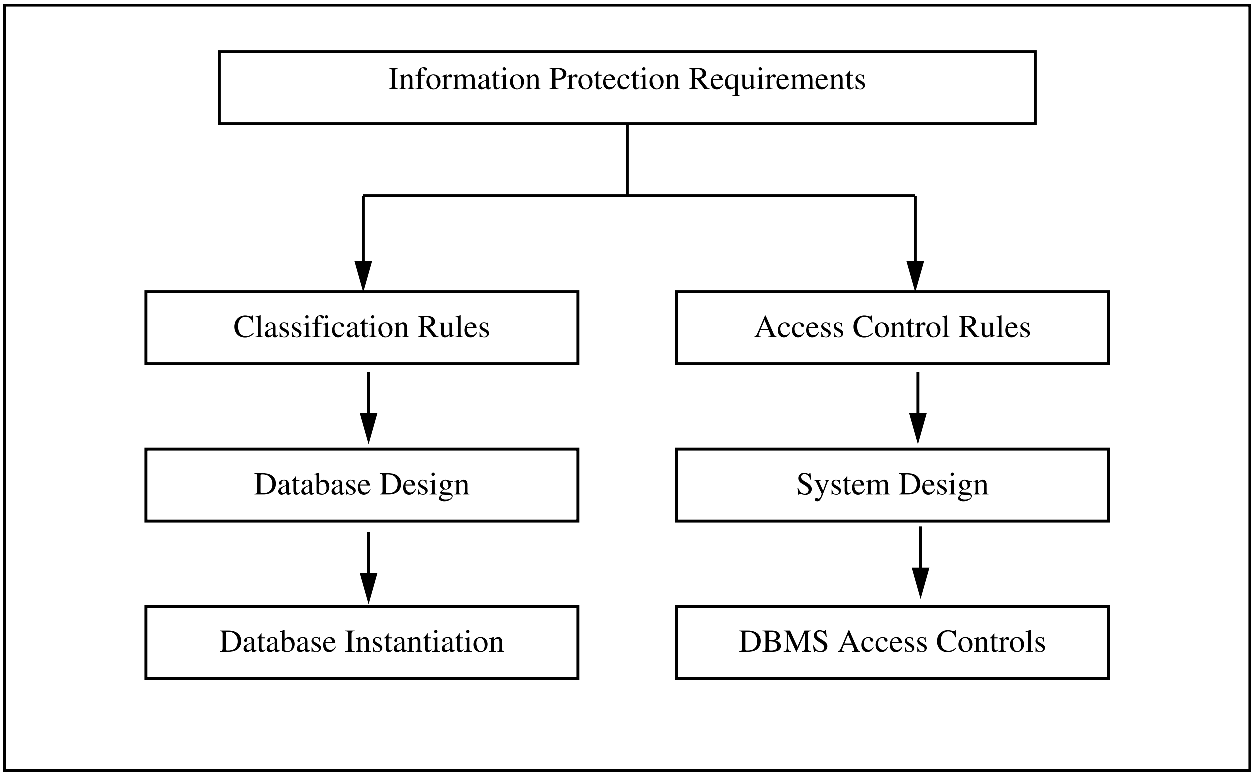

Figure 2.2: Database Security Engineering Process

Classification rules should address all information that needs to be protected, not just information that must be protected for national security reasons (i.e., Confidential, Secret, and Top Secret) based on Executive Order 12356. For example, the Computer Security Act of 1987 requires unclassified but sensitive information have confidentiality protection. Categories of unclassified but sensitive information include, but are not limited to, personnel (i.e., subject to the Privacy Act of 1984), financial, acquisition contracting, and investigations. Certain other information is classified under the Atomic Energy Act of 1954 as Restricted Data or Formerly Restricted Data.

Confidentiality protection requirements are also applicable to non-government organizations. For example, proprietary information of many types may be considered “bet your company” data and needs to be protected to ensure the viability of the company.

Given a set of classification rules, the database designer must provide a database design which labels data objects and correctly implements the classification rules. The database instantiation represents the database as it is populated with data and used.

Access control rules are generically described as MAC for access based on the military classification system (including formal need-to-know) and DAC for access based on informal need-to-know. However, to effectively implement the protection requirements, one must explicitly identify the application-dependent information which identifies which authorized users, or classes of users (e.g., roles), are authorized access to each type of information identified in the classification rules.

The access control rules must be incorporated into the application system design. Although the access control rules can be implemented through mechanisms in the operating system or application software, inference and aggregation concerns are primarily addressed by DBMS access controls.

Classification rules are normally found in a Classification Guideline as required by the DoD Information Security Program Regulation [DoD 5100.1-R]. Generic access control rules are documented in the system security policy as required by the TCSEC. The instantiation of MAC and DAC with application-dependent access control rules that identify specific users and data is usually documented in a Security Concept of Operations.

The classification rules for an application domain need to be complete. That is, all information to be contained in the support information system should be explicitly addressed.

2.5 THE REQUIREMENTS FOR PROTECTING AGAINST INFERENCE AND AGGREGATION

In this section, we identify both operational and TCSEC requirements related to protecting against inference and aggregation. Operational requirements, identified in Section 2.5.1, include more than just the need to protect against inference and aggregation but also operational considerations, or tradeoffs, that impact decisions on how, or to what extent, inference and aggregation protections are implemented. Section 2.5.2 discusses the lack of direct TCSEC requirements for protecting against inference and aggregation.

2.5.1 The Operational Requirement

There are at least four operational requirements related to inference and aggregation protection.

1. The enterprise's information must be properly protected from disclosure either directly or through inference.

2. Employees (i.e., users) must have access to the information they need to do their jobs.

3. The enterprise must conduct its operations in an efficient and cost-effective manner.

4. The enterprise must interact efficiently with external organizations, which may involve mostly unclassified information.

Unfortunately, these operational requirements often are in conflict. Over-classifying data to meet the first requirement may reduce availability of information to certain employees (conflicts with second requirement). Cost-effective operations (the third requirement) may demand granting the lowest clearances possible to employees, which may conflict with the established classification rules to protect information (first requirement). Finally, the need to protect information (first requirement) may conflict with the efficient interface with other organizations (fourth requirement). This last conflict is the genesis of most cardinal aggregation policies.

2.5.2 The TCSEC Requirement

There are no requirements in the TCSEC, TDI, or *Trusted Network Interpretation* (TNI) [TNI 87] that directly address inference and aggregation issues. Inference and aggregation are properties intrinsic to data values and to the structure of a database (schema, metadata). Inference and aggregation are based on an interpretation of the enterprise's security policy and on specific requirements of the application domain being automated. Thus, inference and aggregation should best be addressed by the system developer, acquisition agency, and certification and accreditation authorities for each system and should not be looked at during a TCSEC /TDI evaluation.

Since inference is often characterized as an inference channel, there is often some confusion as to the relationship between inference and covert channels. Inference and aggregation are possible because of the existence of a set of specific known data values and a set of logical implication rules. Knowledgeable database administrators or information security officers may be able to apply inferential engines to the analysis of a populated database to uncover a set of potential unauthorized inferences which they can then protect.

However, new inferences may be introduced through error on the part of users authorized to update the database. It is important to note that the introduction of inference or aggregation is a data value-based problem (including metadata that represents the database design), and not one of DBMS (i.e., the product) design or implementation. Unlike Trojan horses or covert channels, values needed to support an inferential attack are introduced consciously by benign actions of users, and not clandestinely by malicious implanted code or by flaws in a DBMS trusted computing base (TCB) design or implementation. In this sense, inference problems are different from covert channel issues since all the data used to make the inference are at **LOW**, and there is no need for an active agent at **HIGH** to signal information from **HIGH** to **LOW** [Jajodia 92]. Note also that inference will still produce the same result, independent of whether the compromised (inferred) data remains in (or ever existed in) the database at **HIGH**, as opposed to the need for **HIGH** data to be present in order for it to be signaled to **LOW** by a covert channel.

SECTION 3

AN INFERENCE AND AGGREGATION FRAMEWORK

The purpose of this section is to present a framework to reason about inference and aggregation. Section 3.1 begins with a description of a detailed example application domain that will be used to illustrate parts of the framework. Section 3.2 describes the framework using instances from the detailed example for illustration. Section 3.3 presents a matrix which summarizes how the research efforts described in Sections 4 and 5 relate to the framework.

3.1 DETAILED EXAMPLE

This section introduces a detailed example used to illustrate the concepts presented in this document. We use a mythical company, the Acme Technology Company or ATC. ATC contracts with many government agencies and commercial companies to provide technology-oriented products and services. ATC also conducts extensive applied research for its clients in several areas of technology. ATC needs extensive support for its business operations by AISs that must represent information about:

- entities of interest in the application domain, and
- functional or business processes used in the application domain.

Although there are other security considerations that should be addressed in designing functional processes, for the purpose of this document, we will only address the process view as it relates to access control.

We are most interested in understanding and modeling the information about entities. Tn particular, we are concerned with identifying the entities, associations, and attributes of interest to ATC.

Entities (real objects or concepts of interest):

- <u>Employees</u> ATC employs a wide variety of personnel from clerks to very specialized engineers.
- <u>Education</u> Data about the education of each ATC employee is retained.
- <u>Research Projects</u> Although ATC has thousands of projects that cover a wide variety of purposes, research projects represent a special category of projects, especially from a security standpoint.
- <u>Critical Technologies</u> Certain technologies are particularly sensitive.
- <u>Clients</u> Information about the organization that sponsors a project.

Associations (relationships between instances of the entities). Figure 3.1 shows the following associations (in italics) between entities:

- Employees *work-on* Projects.
- Employees *have* Education.
- A Client *sponsors* Projects.
- Some Projects *use* Critical Technologies.

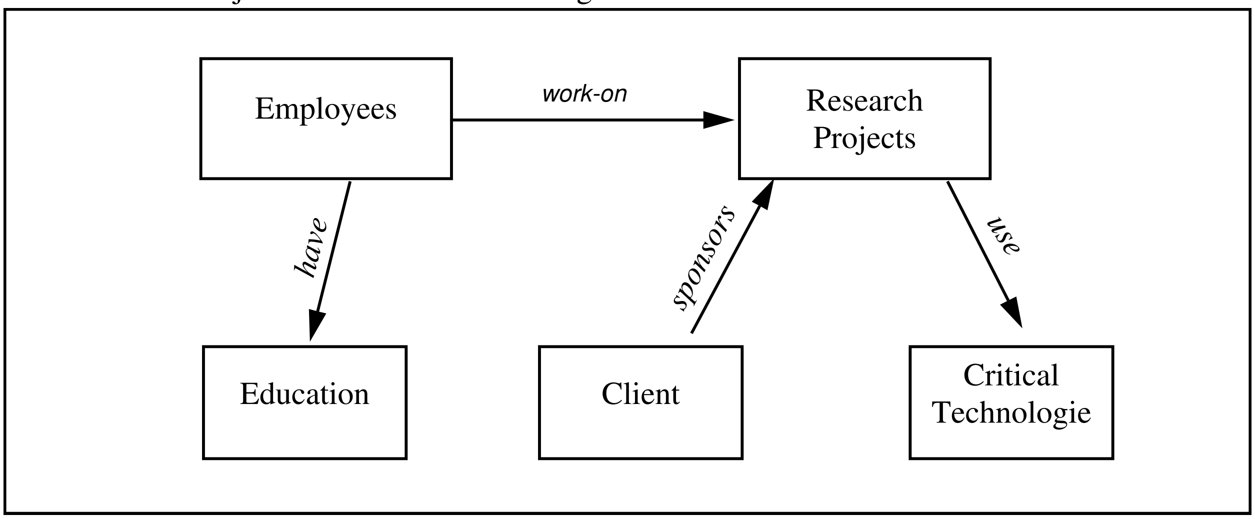

Figure 3.1: Entities and Associations

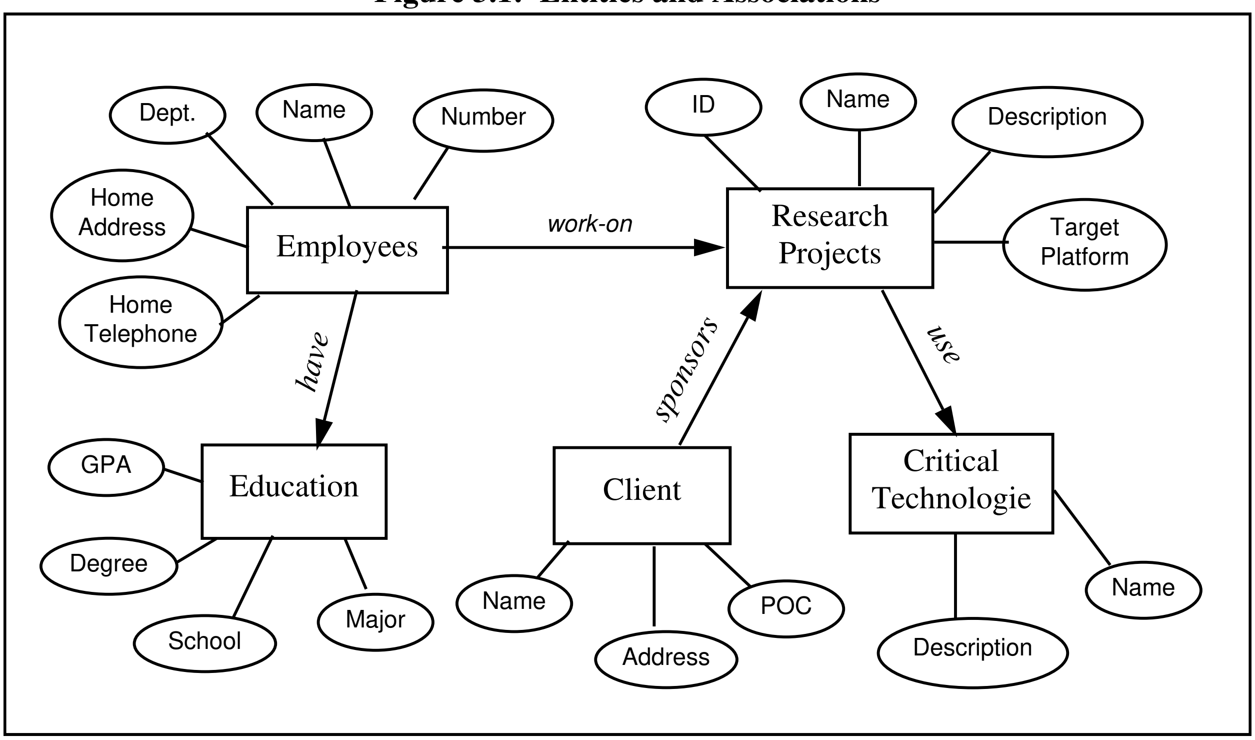

Figure 3.2.: ATC Attributes

Attributes (characteristics or properties of an entity). The addition of attributes adds more detail and, therefore, realistic complexity to the example. Figure 3.2 shows selected attributes for each ATC entity.

3.2 THE FRAMEWORK

The framework, shown in Figure 3.3, is an extension to the database security engineering process described in Section 2.4. The framework provides the context to discuss how inference and aggregation are applicable at each step in the development process. Given that inference and aggregation are inherently application-dependent, examples, based on the ATC example in the previous section, are given to illustrate the concepts. This framework with examples provides a basis to illustrate and compare the proposed approaches described in Sections 4 and 5.

3.2.1 Information Protection Requirements and Resulting Vulnerabilities

There are at least three types of information protection requirements: confidentiality, integrity, and availability. Inference and aggregation relate exclusively to the confidentiality objective of security. Therefore, although integrity and availability protection requirements are also important, this document will only address confidentiality requirements.

Given a set of information confidentiality protection requirements, there are two vulnerabilities that must be addressed by appropriate controls: direct disclosure and inference. Both of these vulnerabilities were discussed in Section 2.2. Aggregation is not yet a problem, because the protection requirements address specific facts or information that must be protected as opposed to a protection requirement for an aggregate of data.

3.2.2 Classification Rules

Classification rules for information are stated such that direct disclosure will be prevented, and the ability to infer unauthorized information is reduced to an acceptable level. Preventing direct disclosure is fairly straightforward: state classification rules for entities, attributes, and/or associations and then restrict access based on user clearance and data classification. Unfortunately, in practice, protection requirements can be so complex that ensuring that a set of classification rules are correct (i.e., consistent and complete) can be difficult.

Each of the following types of classification rules from Figure 3.3 is described in the following paragraphs:

- entity;
- attribute;
- data association; and
- cardinal aggregation.

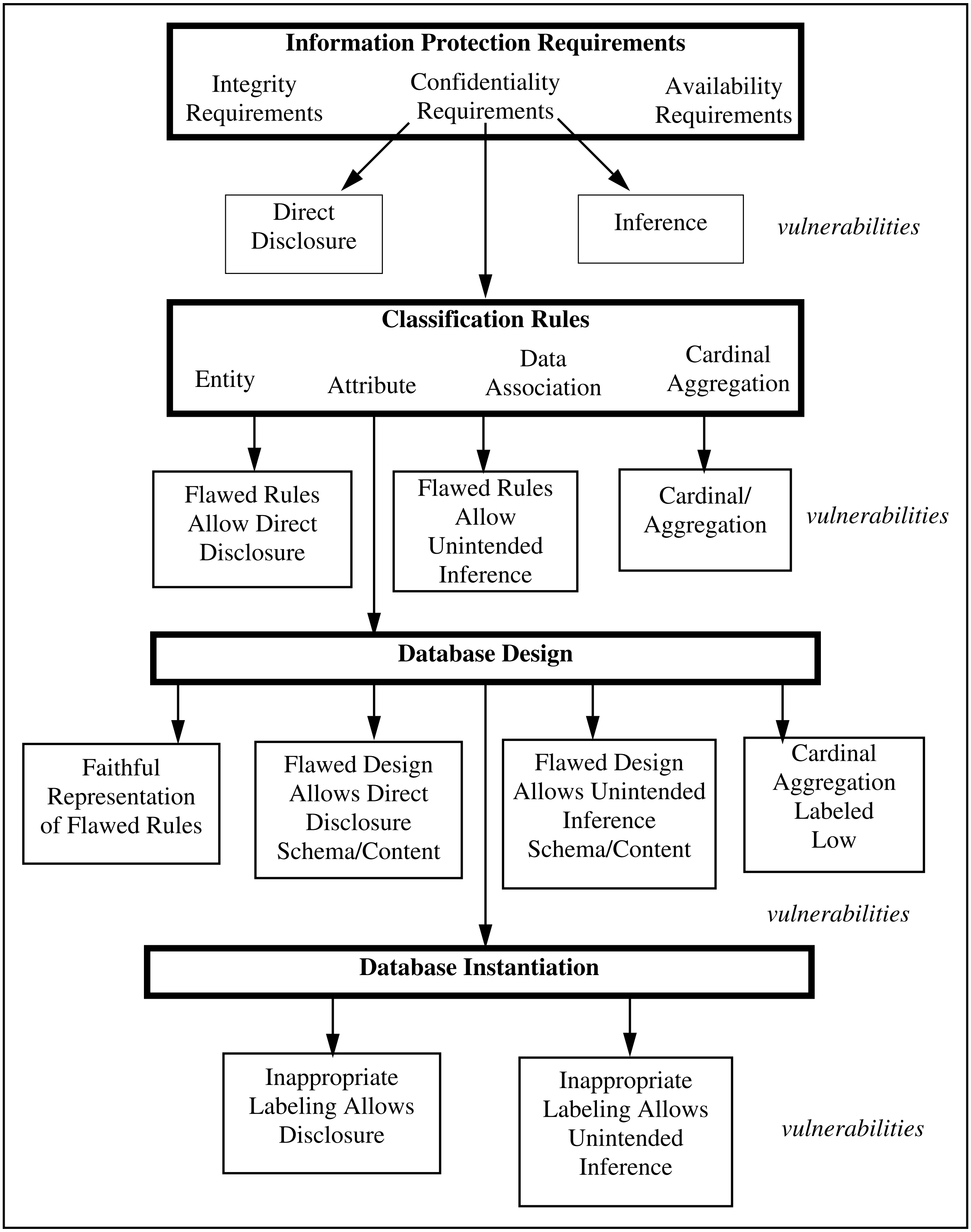

Figure 3.3: Inference and Aggregation Framework

Table 3.1 summarizes this section by showing a taxonomy of types of confidentiality protection requirements along with examples of protection requirements and supporting classification rules for the ATC example. The ATC classification rules from Table 3.1 are used to illustrate the types of classification rules described in this section. The types of protection requirements are a synthesis of those found in papers by Smith, Pernul, and Burns, [Smith 90a; Pernul 93; Burns 92] and are meant to be illustrative of the primary types of protection requirements but not necessarily an exhaustive taxonomy.

Type Protection		Example Protection Requirement	Example Classification Rules
Entity Classification			
1a	existence of all instances of an entity	• the fact that ATC is involved with Critical Technologies must be protected at LOW	• the existence of all instances of Critical Technologies entity are classified LOW
1b	existence of selected instances of an entity	• the fact that a Research Project involves Critical Technologies must be protected at HIGH	• the existence of Research Projects that use Critical Technologies are classified LOW • the existence of instances of all other entities is UNCLASSIFIED
2a	identity of all instances of an entity	• the identity of Critical Technologies must be protected at HIGH • the identity of Clients of Research Projects must be protected at LOW (this is a requirement of the client organization)	• the value of all instances (i.e., all attributes) of the Critical Technologies entity is classified HIGH • the value of all instances (i.e., all attributes) of the Client entity is classified LOW
2b	identity of selected instances of an entity	• selected Research Projects will be protected at LOW based on the contents of Project Description as determined by the classification authority	• all attributes of the Research Project entity are classified LOW when the Description attribute contains LOW information • the identity of instances of all other entities is UNCLASSIFED
Attribute Classification			
3	existence of an attribute (could also be of selected instances of an attribute)	• the fact that specific Target Platforms are identified for Research Projects is to be protected at HIGH	• the existence of the Target Platform attribute of Research Project is classified HIGH • all instances of the Target Platform attribute are classified HIGH • the existence of instances of all other attributes is UNCLASSIFIED except privacy information
4	value of an attribute	• privacy information for employees (i.e., home address and phone number) is to be protected at LOW	• the existence of the privacy information attributes of employees is classified LOW
Data Association			
5	the association between two entities or two attributes of different entities	• the association between a specific project and the sponsoring client must be protected at HIGH	• the association between the Project entity and the Client entity is HIGH • the association between all other entities is UNCLASSIFIED
6	the association between two attributes of the same entry	• the association between name and salary is to be protected at LOW	• the association between the Salary attribute of the Employee entity and Employee identifying attributes (name, number) is LOW
Cardinal Aggregation			
7	a collection of instances of the same entry	• the identity of Critical Technologies must protected at HIGH	• the education entity and all its attributes for all employees are UNCLASSIFIED • the collection of the Degree and Major attributes for all employees is LOW

Table 3.1: Taxonomy of Protection Requirements and Classification Rules

Entity classification rules can address the need to protect both the *identity* as well as the *existence* of instances of entities of interest. In some operational environments, the fact that an entity (or some selected instances of an entity) exists would disclose, directly or through an inference, classified information. In many other operational environments, knowing that something classified exists is not a problem; but you must not be able to know the classified information (i.e., its identity or value). A good example is a compendium of documents including some that are classified. The reader can know that the documents exist without having the clearance to read them. Entity classification rules from the ATC example include:

- The existence of all instances of the Critical Technologies entity are classified **LOW**.
- The values of all instances (i.e., all attributes) of the Critical Technologies entity are classified **HIGH**.
- All attributes of the Research Project entity are classified **LOW** when the Description attribute contains **LOW** information.

Attribute classification rules are similar to entity classification, except that they address specific attributes of an entity. Attribute classification rules from the ATC example include:

- The existence of the Target Platform attribute of Research Project is classified **HIGH**.
- Instances of the Employee entity are **UNCLASSIFIED**, but the value (not existence) of two attributes, Home Address and Home Phone#, are classified **LOW**.

Data Association classification rules are stated when a relationship <u>between</u> entities or attributes is considered more sensitive than information about the entities or attributes. This type of classification rule implements inter- and intra-entity aggregation as discussed in Section 2. The ATC example includes these data association rules:

- The association between a specific project (classified **LOW**) and the sponsoring client (classified **LOW**) must be protected at **HIGH**.
- The association between an employee and the employee's salary is to be protected at **LOW**.

These three types of classification rules (entity, attribute, and data association), along with appropriate access control rules (e.g., the military classification system) will define the basis for preventing direct disclosure. Direct disclosure can be prevented if the classification rules correctly implement the protection requirements (i.e., are consistent and complete) and if the AIS correctly implements the rules.

On the other hand, eliminating inferences is difficult, if not impossible, in that an individual's capability to infer information is based on their already acquired knowledge and the innovative ways in which information can be combined [Morgenstern 88]. Much of the knowledge that may be used to infer unauthorized information is normally outside of the system in the form of domain-specific knowledge or even common sense. At one end of the spectrum, one can ensure that no unauthorized inferences are possible by classifying all information at **HIGH** and then giving all users a **HIGH** clearance. Although this approach solves the inference problem (since every user is authorized to access all information in the system), we now have introduced new vulnerabilities (e.g., users have clearances they should not have, and therefore access to information they should not see) and potentially decreased the operational efficiency of the organization (e.g., having to manually downgrade a large percentage of the information produced by the system)[9].

Establishing classification rules that will reduce unwanted inferences involves a constant trade-off between the desire to raise the classification of data to reduce inferences and the desire to maintain or even lower the classification of data to improve operational efficiency without directly disclosing classified data to those not cleared to see it. This process of evaluating the relative advantages of over-classifying information to prevent inferences is part of the risk analysis activity identified in Section 2.3.

At this point, a fourth type of classification rule, cardinal aggregation, is often added. Cardinal aggregation rules address the classification of collections of instances of the same entity. Often, these rules are established to reduce inference. At this point in the framework we can conclude the high-level relationships between inference and aggregation are:

- inference is a vulnerability, e.g., authorized users can deduce information not authorized by the protection requirements, and
- aggregation is a class of classification policies that are stated as one form of control, in many cases, to reduce the inference vulnerability.

The ATC example includes a cardinal aggregation classification rule:

- the education entity and each of its attributes for all employees are individually **UNCLASSIFIED**, but
- the collection of the Degree and Major attributes of all instances of the education entity is **HIGH**.

The ATC cardinal aggregation classification rule illustrates the difficulty of enforcing the intent of the rule, even though we have specified the information to be protected (i.e., the identity of Critical Technologies must be protected at **HIGH**). The difficulty is to determine how many employees' educational information has to be aggregated to allow an inference. Is it all employees on a particular project? Ten employees? Ten percent of the employees? All employees with blue eyes? To have an effective implementation of a cardinal aggregation classification rule, someone has to state explicit criteria for which collections must be considered HIGH. Unfortunately, cardinal aggregation classification rules introduce a new vulnerability, as discussed in the next section.

At this point it is appropriate to distinguish between an aggregation policy, such as those discussed above, and a *Chinese Wall* policy. Brewer and Nash introduced an important commercial security policy called the Chinese Wall policy [Brewer 89]. They described the policy as follows:

---

9. Indeed, these vulnerabilities were among the major motivations for providing multilevel secure technology.

It can most easily be visualized as the code of practice that must be followed by a market analyst working for a financial institution providing corporate business services. Such an analyst must uphold the confidentiality of information provided to him by his firm's clients; this means that he cannot advise corporations where he has *insider knowledge* of the plans, status, or standing of a competitor. However, the analyst is free to advise corporations which are not in competition with each other, and also draw on general market information.

This type of policy seems to involve aggregates of information and therefore is sometimes considered an "aggregation problem." In fact, the Chinese Wall policy does not meet the definition of an aggregation problem; there is no notion of some information being sensitive with the aggregate being more sensitive. The Chinese Wall policy is an access control policy where the access control rule is not based just on the sensitivity of the information, but is based on the information already accessed. The situation can be considered similar to having compartmented data and subjects which can be cleared for exactly one compartment. The choice of which compartment the subject is cleared for is effectively made when the first compartmented data is viewed.

3.2.3 Vulnerabilities from Classification Rules

Figure 3.3 identifies three residual vulnerabilities given a set of classification rules designed to prevent direct disclosure and reduce inference (i.e., the vulnerabilities of information confidentiality protection requirements):

- <u>Flawed Rules that allow Direct Disclosure</u> With complex operational environments, the classification rules can be flawed with respect to how faithfully they implement the information confidentiality protection requirements. They can be inconsistent, incomplete, or just incorrect such that they allow direct disclosure.

- <u>Flawed Rules that allow an Unintended Inference</u> Although the entire set of classification rules are intended to prevent unwanted inferences, when faced with a complex set of classification rules there may be opportunities for unintended inferences. Again, this may be caused by rules that are inconsistent, incomplete, or incorrect.

- <u>Cardinal Aggregation</u> If a cardinal aggregation rule has been stated that classifies an aggregate as **LOW** but individual elements are **UNCLASSIFIED**, then a vulnerability has been added since **UNCLASSIFIED** users may be able to assemble aggregates in an unintended or undetectable way.

3.2.4 Database Design

Database designers and application developers attempt to implement classification rules by accurately representing and attaching security labels to database objects. The Relational Model [Codd 70], one of several data models, is the model of choice for existing commercial MLS products. Figure 3.4 shows a relational model representation of the ATC example using tuple-level labeling.

The following are examples of how each type of classification rule can be implemented in a database design:

- The rule, "the existence of all instances of the Critical Technologies entity is classified **LOW**," is implemented by classifying the Critical Technologies table metadata at **LOW**. This is done by the database designer.
- The rule, "the value of all instances of the Critical Technologies entity is classified **HIGH**," is implemented by labeling each tuple in the Critical Technologies at **HIGH**. This is enforced by the user entering the data.
- The rule, "all attributes of the Research Project entity are classified **LOW** when the Description attribute contains **LOW** information," is implemented by having tuples labeled **UNCLASSIFIED** or **LOW** based on the content of the Description attribute. This is enforced by the user entering the data.
- The rule, "the existence of Research Projects that use Critical Technologies are classified **HIGH**," is implemented by classifying each Research Project tuple at **HIGH** when the project uses a Critical Technology. This is enforced by the user entering the data.
- The rule, "the existence of the Target Platform attribute of Research Project is classified **HIGH**," is implemented by decomposing the Target Platform attribute (which would normally be an attribute in the Research Project table) into a separate table, Project-Target. All tuples in Project-Target are classified **HIGH**. Note that the metadata can also be classified **HIGH** to prevent obtaining the knowledge that projects have a target platform that could contribute to unauthorized inferences.
- The rule, "all instances of the Target Platform attribute are classified **HIGH**," from an implementation standpoint, is redundant with the previous rule. Thus, this rule requires no changes to the database design.
- The rules for data association between the Project and Client entities are, "the value of all instances (i.e., all attributes) of the Client entity is classified **LOW**," "the value of all instances of the Project entity (except those that use Critical Technologies) are classified **LOW**," and "the association between the Project entity and the Client entity is **HIGH**." These rules are implemented by labeling tuples (and metadata) in the Client table as **LOW**, labeling tuples in the Project table as **LOW** (except those that are labeled **HIGH** because they use Critical Technologies), and constructing an additional table, Client-Project, to represent the data association between Clients and Projects. The Client-Project table (metadata and tuples) is classified **HIGH**. Note that in the absence of a data association rule, the association between Project and Client tables would be represented by a foreign key from the Project table to the Client table.

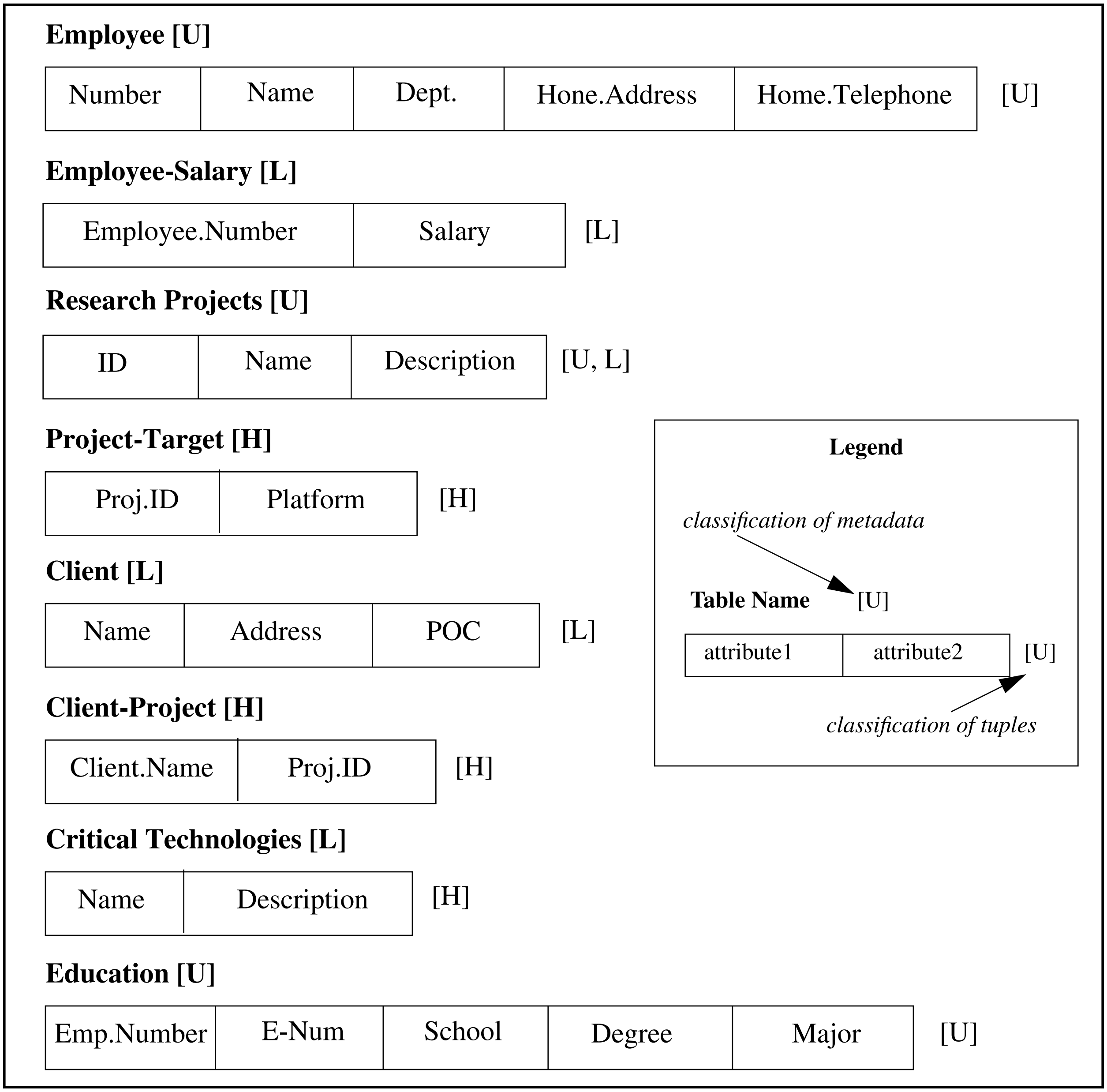

Figure 3.4: ATC Example Database Design

- The data association rule, "the association between Salary attribute of the Employee entity and Employee identifying attributes (name and number) is **LOW**," is implemented by decomposing the Salary attribute into a separate table, Employee-Salary. All tuples in the Employee-Salary table are labeled **LOW**.
- The cardinal aggregation rule established by the rules, "the education entity and all its attributes for all employees are **UNCLASSIFIED**" and "the collection of the Degree and Major attributes of the entity for all employees is LOW," is implemented by labeling all tuples in the Employee table as **UNCLASSIFIED** and requiring some (unspecified) mechanism to control access to multiple records.

3.2.5 Vulnerabilities from Database Design

Given a database design that attempts to faithfully represent the established classification rules, four vulnerabilities are identified in Table 3.3:

- <u>Faithful Representation of Flawed Rules</u> A database design that faithfully implements a flawed set of classification rules will not fix the flawed rules.
- <u>Flawed Design Allowing Direct Disclosure</u> A flawed design may allow a direct disclosure at the schema level for uniformly classified data objects or at the content level for data objects where instances may be classified at one of several levels.
- <u>Flawed Design Allowing Unintended Inference</u> The flawed design may allow unintended inferences at the schema level for uniformly classified data objects, or at the content level for data objects where instances may be classified at one of several levels.
- <u>Cardinal Aggregation Labeled **LOW**</u> If the data is labeled **LOW**, then a vulnerability exists, since **LOW** users may be able to access aggregates in an unintended way.

These vulnerabilities are related to the values, application semantics, and the security labels assigned to data objects, not to the functionality (beyond enforcing a MAC policy) or assurance provided by the TCB. The TCSEC class of a TCB that enforces MAC has no impact on countering these vulnerabilities. An Al TCB is just as effective (or ineffective) as a B1 TCB in countering these vulnerabilities.

3.2.6 Database Instantiation and Vulnerabilities

For completeness, Table 3.3 shows the database instantiation as the final activity. Databases are populated with data, and users use the database to conduct their job tasks. The database administrator or security officer is tasked with ensuring the system is operating in a secure manner. Assuming clearances are properly assigned to users, two vulnerabilities are identified:

- <u>Inappropriate Labeling Allowing Disclosure</u> Current MLS systems, including DBMSs, depend on the user being logged in at the access class that correctly represents the classification of the data being added to the database. If a user with a **HIGH** clearance enters **HIGH** data when the user's current access class is **LOW**, then **HIGH** data has been directly disclosed, since users with only a **LOW** clearance can access the **HIGH** data that is incorrectly marked **LOW**.
- <u>Inappropriate Labeling Allowing Unintended Inference</u> In a similar manner, a user with a **HIGH** clearance may enter data at **LOW** that will allow other users with only a **LOW** clearance to infer **HIGH** data.

The TCSEC class of a TCB that enforces MAC also has no impact on countering these vulnerabilities.

3.3 OVERVIEW OF RESEARCH

This section provides a brief overview of the research that will be discussed in Sections 4 and 5.

Each of the papers and research efforts on inference and aggregation address one or more parts of the framework. Table 3.2 lists formal approaches to defining the problems. Table 3.3 maps inference and aggregation research efforts against the framework in three categories (techniques, mechanisms, and analysis tools). As Table 3.3 shows, much of the research has been in developing techniques and designing mechanisms. There has also been limited work in developing tools to help address inference and aggregation. The remainder of the table shows which vulnerabilities each of the analysis tools and mechanisms attempt to address.

Inference	Aggregation
Database Partitioning (Denning)	Security Algebra (Lin)
Set Theory (Goguen and Meseguer)	Modal Logic Framework (Cuppens)
INFER Function (Denning and Morgenstern)	
FD and MVD (Su and Ozsoyoglu)	
Sphere of Influence (Morgenstern)	
Fact Model (Cordingley and Sowerbutts)	
Inference Secure (Lin)	
Inference Secure (Marks)	
Imprecise Inference Control (Hale)	

Table 3.2: Formal Definitions

	Techniques	Mechanisms	Analysis Tools
Classification Rules			
Entity and Attribute			
Direct Association	Database Design (Denning, Lunt)		
Cardinal Aggregation			
Classification Rule Vulnerabilities			
Direct Disclosure			
Unintended inference			
Cardinal Aggregation Rules			
Faithfully implements flawed classification rules			

Table 3.3: Research Summary (Part 1)

	Techniques	Mechanisms	Analysis Tools
Data Design Vulnerabilities			
Direct Disclosure at schema level	Secondary Path Analysis (Hinke, Binns)		
Direct disclosure at content level			
Unintended inference at schema level	Basic Security Principle (Meadows) Semantic Data Models (Buczkowski, Hinke, Smith) Classification Constraints (Akl) Constraint Satisfaction (Morgenstern) Integrity Constraints (Denning) Secondary Path Analysis (Hinke, Binns)		DISSECT (Garvey et al.) AERIE (Hinke and Delugach) Constraint Processor (Thuraisingham et al.) Hyper Semantic Model (Marks et al.)
Unintended inference at content level			
Cardinal Aggregation-LOW Data	Separae Structures (Lunt) Use of Views (Wilson) Restrict Ad hoc Queries Use of Audit Some or All Data HIGH	Aggregation Constraints (Haigh and Stachour) Aggregation Detection (Motro et al)	
Database Instantiation Vulnerabilities			
Labeling allows disclosure		Constraint Processor (Thuraisingham et al.) DB Inference Controller (Buczkowski) Secondary Path Monitoring (Binns) Polyinstantiation (Denning) Auditing Facility (Haigh and Stachour) Snapshot Facility (Jajodia)	
Labeling allows unintended inference			

Table 3.3: Research Summary (Part 2)

SECTION 4

APPROACHES FOR INFERENCE CONTROL

Providing a solution to the inference problem is beyond the capability of currently available MLS database management systems. It is also recognized that a general solution to the inference problem is not possible given the inability to account for external knowledge and human reasoning [Binns 94a]. To provide partial solutions, the inference problem must be bounded.

Section 4.1 identifies several efforts by researchers to formally define the inference problem. These approaches are summarized to provide different perspectives on the problem. Implementing theoretical inference solutions has proven to be difficult. Section 4.2 describes database design techniques that could be used to prevent inference problems from occurring. Mechanisms for inference prevention and detection during run-time are described in Section 4.3. Finally, Section 4.4 describes several tools that assist the database designer to detect and avoid inference channels.

4.1 FORMALIZATIONS

Researchers have proposed numerous ways to characterize or define the inference problem. This section presents several formal definitions from the ongoing research into the discovery of fundamental laws that determine whether the potential for undesirable inferences exist.

4.1.1 Database Partitioning

Database partitioning is used by Denning to define the inference problem [Denning 86a]. For each user *, the data in the database can be partitioned into two sets: a VISIBLE set and an INVISIBLE set, as in Figure 4.1. The user * is allowed to access only elements from the VISIBLE set, and in is not allowed to know what elements are in the INVISIBLE set. (Note that if VISIBLE is null, the user has no access to database information and, therefore, the problem is outside the domain of the DBMS. In what follows, we assume that VISIBLE is not null.) Let KNOWN denote the set of data elements that is known to *. The set KNOWN is constructed by the user either as a result of the previous queries or as a result of some external knowledge that in possesses. There is no inference problem if the intersection of the two sets INVISIBLE and KNOWN is empty, as in Figure 4.2. There is an inference problem if the two sets INVISIBLE and KNOWN intersect, as in Figure 4.3.

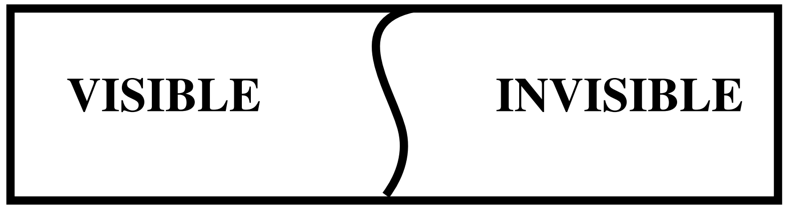

Figure 4.1: Partitioning of the Database for a User U

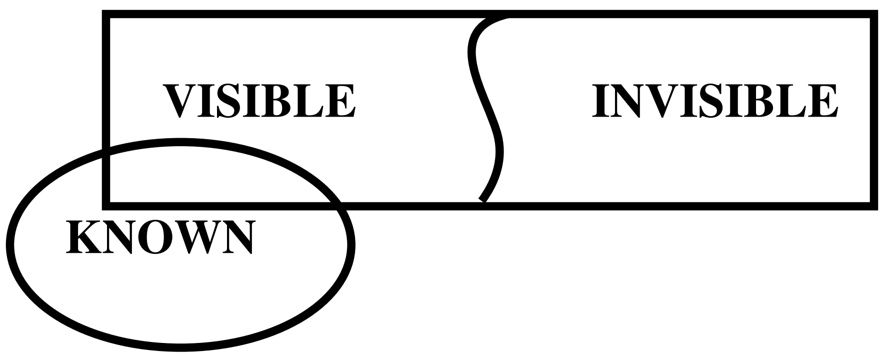

Figure 4.2: Database without an Inference Problem

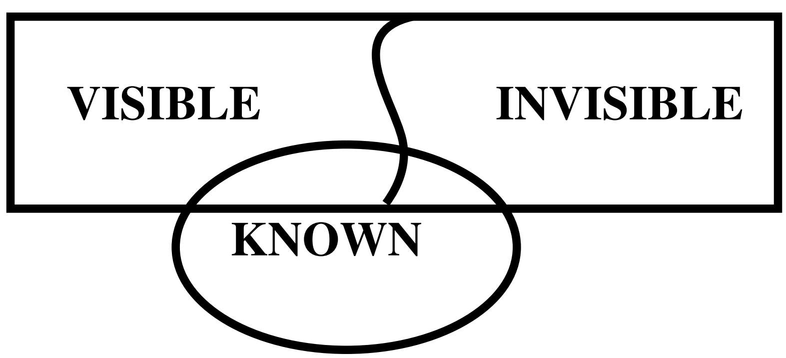

Figure 4.3: Database with an Inference Problem

4.1.2 Set Theory

Another formulation of the inference problem has been given by Goguen and Meseguer [Goguen 84]. 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 $\rightarrow$ as follows: Given data items x and y, relation x $\rightarrow$ *y* is said to hold if it is possible to infer y from x. The relation $\rightarrow$ is reflexive and transitive. A set *S* is said to be *inferentially closed* if whenever *x* is in *S* and $x \rightarrow y$ holds, then *y* belongs to *S* as well. Now, for an access class *L*, let *E(L)* denote the set consisting of all possible responses that are classified at access classes dominated by *L*. There is an inference problem if *E(L)* is not inferentially closed.

Goguen and Meseguer do not set forth any one candidate for the relation $\rightarrow$; 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, monotonic logic, and knowledge-based inference). They do note, however, that for most inference systems of interest, determining that $A \rightarrow b$ (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 $A \rightarrow b$, but otherwise may never halt.

4.1.3 Classical Information Theory

Yet another definition that uses classical information theory has been given by Denning and Morgenstern [Denning 86a]. Given two data items *x* and y, let *H(y)* denote the uncertainty of *y*, and let *Hx (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:

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

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*, then *y* becomes somewhat more likely given *x*. If the value is *1*, then *y* can be inferred given *x*.

This formulation is especially useful since it shows that inference is not an absolute problem, but a relative one. It provides a way to quantify the bandwidth of the illegal information flow. On the other hand, Denning and Morgenstern point out its serious drawbacks [Denning 86a]:

1. It is difficult, if not impossible, to determine the value of *Hx (y)*.
2. It does not take into account the computational complexity that is required to draw the inference.

To illustrate the second point, they give the following example from cryptography: With few exceptions, the original text can be inferred from the encrypted text by trying all possible keys (so *Hx (y) = 1* in this case); however, it is hardly practical to do so.

4.1.4 Functional and Multivalued Dependencies

Su and Ozsoyoglu have studied inference problems that arise because of the functional and multivalued dependencies that are constraints over the attributes of a relation [Su 86,87,90]. Functional dependencies are defined below; the reader is referred to Date for a definition of multivalued dependencies [Date 86].

Let *R* be a relation scheme defined over attributes *, and let *X* and *Y* be subsets of in. A functional dependency $X \rightarrow Y$ is said to hold in *R* if for any relation *r*, the current instance for *R*, *r* does not contain two tuples with the same values for *X*, but different values for *Y*; that is, given any pair of tuples *t* and s of *r*, t[X] = *t[X]* implies *t[Y] = s[Y]*.

Su and Ozsoyoglu illustrate how inference problems can arise if certain functional dependencies are known to the low-level users. If attributes are assigned security labels in a manner that is consistent with the functional dependencies, then these inference threats can be eliminated. This process is formalized by Su and Ozsoyoglu.

4.1.5 Sphere of Influence

Morgenstern expands upon the INFER function (described in Section 4.1.3) to propose a theoretical foundation for inference [Morgenstern 87,88]. He proposes a framework for the analysis of logical inferences and for the determination of logical inference channels. Morgenstern introduces the concept of a *sphere of influence* (SOI) and *inference channel*.

The SOI delimits the scope of possible inferences given some base data, called the core that may consist of entities, attributes, relationships, and constraints. Specifically, the sphere of influence relative to some core, i.e., SOI (core), is the set of all information that can be inferred from the core. The SOI is defined in terms of the INFER function. The SOI models the process by which a user's knowledge of an application can give rise to inferences about additional information. The SOI utilizes a *forward chaining* inference process from the given core to determine the scope of such inferable data.

Morgenstern's concept of *inference channel* serves to isolate the lower level data which could give rise to inferences about higher level data H. The computation of an inference channel is a *backward-chaining* inference process from some resultant data H to determine all information which contributes to upward inferences about H. An inference channel exists if information about some set of data H may be inferred from data in another set C which is at a lower level than H, or is incomparable relative to H in the classification lattice.

4.1.6 The Fact Model

Sowerbutts and Cordingley provide a formal definition of inference based on an abstract data model called the *Fact Model* [Cordingley 89, 90; Sowerbutts 90]. They identify two aspects of the inference problem: static inference and dynamic inference. Static inference is knowledge of the database together with authorized facts that can be used to determine unauthorized facts. Dynamic inference is where database operations can be used together with knowledge of the structure of the database to determine unauthorized facts. The Fact Model addresses static inference.

The Fact Model is a model of knowledge described as a set of facts and a set of constraints. The authors provide a language for specifying application-dependent facts and constraints. The Fact Model provides a formal statement of the standard constraints, such as functional and multivalued dependencies, and defines explicitly the inference threat associated with each [Cordingley 89]. The relationship between facts and subfacts is defined by construction rules. An **INFER** function is defined on facts, constraints, and *construction rules*. Given an arbitrary subset of facts and constraints, the **INFER** function identifies all the facts that are implicit in the given subset, and hence can be inferred by a user. At this point the classification attribute can be added for each fact.

This approach forms the basis for assigning classifications such that any facts inferable from a given subset of facts whose classifications are dominated by a classification level k also have a classification level dominated by k. The authors call this a *Static Inferentially Secure Database*.

The Fact Model also is extended to represent operations such that a *Dynamic Inferentially Secure Database is defined*. The Fact Model was developed as an abstraction of the relational model but can be used to describe any conventional database structure such as network or hierarchical.

4.1.7 Inference Secure

Lin introduces the concept of *navigational inference* as the process of accessing **HIGH** data via navigating through legally accessible **LOW** data [Lin 92]. Lin states that inference problems exist if the security classifications of data are inconsistent with some database structures. He identifies three such inference problems:

- <u>Logical Inference Problems</u> The security classification of a theorem (a derivable formula) in a formal theory should be consistent with its proof. If not, then logical inference problems arise.

- <u>Algebraic Inference Problems</u> The security classification of “algebraically derivable data” should be consistent with its relational algebraic structure. If not, algebraic inference problems arise.

- <u>Navigational Inference Problems</u> The security classification of data should be consistent with its navigational operators (which generate navigational paths). If not, navigational inference problems arise.

Lin defines a multilevel data model as *inference secure* if:

(1) it is a Bell-LaPadula Model (secure under MAC), and

(2) it is navigational inference free.

Lin provides further descriptions and theorems to support the definitions [Lin 92].

4.1.8 An Inference Paradigm

Marks defines *database* inference as [Marks 94b]:

> Inference in a database is said to occur if, by retrieving a set of tuples {T) having attributes {A} from the database, it is possible to specify a set of tuples {{T'), having attributes {A'), where {T') is not a subset of {T) or {A') is not equal to {A).

The definition may be stated as an inference rule: IF ({T), {A}) THEN ({T'}), {A'}), which may be denoted as ({T}, {A}) → ({T'}, {A'}). However, it may be possible to retrieve a set of tuples {T} and the associated attributes {A} from a database and to reason, using data and attributes outside the database, to arrive at a set of tuples {T'} and attributes {A'} that are again within the database. That is, it is possible to form a *chain of reasoning*, ({T}, {A}) → ({T1}, {A1}) → ({T2}, {A2}) →... → ({T'}, {A'}) where some of the tuples and/or attributes are outside the database. When some of the data or attributes are outside of the database, the chain of reasoning cannot be followed by the database system. Fortunately it is not necessary to actually follow such a chain of reasoning in order to control the inference threat. If the database contains the endpoints of the chain (({T}, {A}) and ({T'}, {A'})) there will exist what is referred to in logic systems as a *material implication* relating the two sets.

If a material implication exists such that ({T}, {A}) → ({T'}, {A'}) where ({T}, {A}) is classified **LOW** and ({T'}, {A'}) is classified **HIGH**, then the database can offer no assurance that there does not exist some chain of inference, using outside knowledge, that can connect the two, enabling a **LOW** user to infer **HIGH** information. If, however, ({T}, {A}) not → ({T'}, {A'}) within the database, then it can be guaranteed that no chain of inference, using outside knowledge or not, exists which connects the two sets. That is, the absence of a material implication between two sets of data is *sufficient* to guarantee the absence of any chain of reasoning between the sets of data. It is not *necessary* for inference control, however, since material implications may be coincidental, and not related to any reasoning process. These arguments may be reduced to:

> *Limitations on Database Inference*: ({T}, {A}) → ({T'}, {A'}) is an inference rule capable of being controlled by the database if and only if all the tuples {T} and {T'} are in the database and all the properties {A} and {A'} are attributes in the database.

4.1.9 Imprecise Inference Control

Hale, Threet, and Shenoi introduce a powerful, yet practical, formalism for modeling and controlling imprecise functional dependency (FD) based inference in relational database systems [Hale 94]. The existence of an imprecise FD implies that if some tuple components satisfy certain equivalencies, then other tuple components must exist and their values must be equivalent. Imprecise FDs can specify constraints on precise and imprecise data. Examples of imprecise FDs are: "engineers have starting salaries about 40K" and "approximately equal qualifications and more or less equal experience demand similar salary."

An imprecise FD is formally defined from which the authors formally define an *imprecise inference channel*. The non-formal definition of an imprecise inference channel is a chain of imprecise FDs. To control and ultimately eliminate compromising imprecise inferences, it is necessary to specify a set of imprecise inference channels considered to be secure. This *compromise specification set* is defined by the database administrator. Suspect channels in a database are compared with these secure channels. A potential security compromise exists when a suspect channel allows the inference of information which is <u>not</u> coarser than information inferred by any secure channel. Potentially compromising imprecise inference channels are eliminated by hiding relations attributes or by "clouding" data manifested by imprecise FDs in the compromising channels.

4.2 DATABASE DESIGN TECHNIQUES

The problem of deciding how to label multilevel database objects - data, schemata, and constraints - should not be a problem for cleared database designers and users [Millen 91]. They ought to know the classification of whatever they are entering into the database. Unfortunately, many inferences are not simple ones, and the number and complexity of potential inferences can be quite large. Thus, raising the question of how well a person can anticipate such inferences in order to classify the data [Morgenstern 88]. The following subsections examine techniques for avoiding inference problems in the first place, and how to detect inference channels during database design.

4.2.1 Inference Prevention Methods

Ideally, if all unauthorized disclosures are to be prevented, the Basic Security Principle [Meadows 88a, 88b] should be followed:

> *Basic Security Principle for Multilevel Databases* - The security class of a data item should dominate the security classes of all data affecting it.

The reason for the Basic Security Principle is clear: if the value of a data item can be affected by data at levels not dominated by its own level, information can flow into the data item from data at other levels.

The task of predicting or detecting all inference problems appears to be very difficult. However, many of these problems can easily be prevented by careful consideration of the data items on which a data item is predicated. In practice, unfortunately, two problems with the Basic Security Principle have been identified:

1. The number of potential inferences can be quite large.
2. All possible inferences cannot be anticipated.

The combination of these problems makes the task of appropriately classifying data complex. However, several techniques that have been developed for dealing with inferences are discussed below. If information *x* permits disclosure of information *y*, one way to prevent this disclosure is to reclassify all or part of information *x* such that it is no longer possible to derive *y* from the disclosed subset of *x*. There are two approaches to dealing with violations of this type: reclassify either the data or the constraints appropriately. These approaches are discussed next.

4.2.1.1 Appropriate Labeling of Data

One of the approaches suggested to handle the inference problem is to design the multilevel database in such a way that the Basic Security Principle is maintained [Binns 92a; Burns 92; Hinke 92; Garvey 93; Smith 90b; Thuraisingham 90c]. Security constraints are processed during multilevel database design and subsequently the schemas are assigned appropriate security levels.

Several researchers have proposed that semantic database models be used to detect (and then prevent) some inference problems [Buczkowski 90; Hinke 88; Smith 90b, 91]. Conventional data models (such as hierarchical, network, and relational data models) use overly simple data structures (such as trees, graphs, or tables) to model an application environment. Semantic database models, on the other hand, attempt to capture more of the meaning of the data by providing a richer set of modeling constructs. Since integrity and secrecy constraints can be expressed naturally in semantic database models [Smith 90b], they can be used to detect inference problems during the database design phase.

Classification (or secrecy) constraints can also be used to eliminate inference problems [Akl 87]. Classification constraints are rules that are used to assign security levels to data as they are entered into the database. Classification constraints are required to be consistent (meaning that each value is assigned a unique security level) and complete (meaning that rules assign each value a security level). Inconsistent classification constraints indicate potential inference problems, and incomplete classification constraints point to incomplete labeling rules. Two different methods have been proposed for determining the consistency and completeness of the classification constraints. One is based on logic programming [Denning 86b], and the other is based on computational geometry [Akl 87].

Morgenstern views the overall process of classifying a database as a *constraint satisfaction problem* of the following form [Morgenstern 88]. Each potential inference which involves one or more relations or objects (data) from the database serves as a constraint. It limits the classifications which can be assigned to an object given the classification(s) of the other object(s). In some cases, the constraint may uniquely determine the classification of the remaining data object. Morgenstern defines safety for inference: a data object O, and its assigned classification label, are said to be *safe* for inference if there is no upward inference possible from O. That is, object O cannot be used to infer information about other data objects at higher or incomparable levels of the classification lattice. A classification level is *safe* for inference if all the data objects having that classification are safe.

The techniques described in this section rely on increasing the classification of one or more elements in an offending path following its detection. Note that in practice, however, it is not always possible to solve inference problems by raising the classification level of data that cause inference violations. Some data may be public knowledge, and therefore, cannot be reclassified. In some cases, certain data must be made public due to legal requirements or national policy. Binns also points out that an upgraded element may open apparently new inference channels [Binns 94b].

4.2.1.2 Appropriate Labeling Integrity Constraints

There are two ways of handling an explicit constraint defined over several data fields. One is to leave the constraint **UNCLASSIFIED**, but to identify the **LOW** data that, together with the constraint, will allow users to infer information about the **HIGH** data, and to reclassify that **LOW** data upwards. This approach satisfies the Basic Security Principle, but at the cost of reducing data availability to low-level users. The other approach is to classify the constraint itself at the least upper bound of the security classes of the data over which it is defined. This approach satisfies the Basic Security Principle and allows low-level users access to data, but at the possible cost of data correctness; data entered by a low-level user may not satisfy the constraint. This problem can be partially solved by allowing polyinstantiation to take place [Denning 87] and is briefly discussed in Section 4.3.2. When data at a **LOW** security level is entered, it is not checked against the constraint. However, the **LOW** data does not replace the **HIGH** data; both are stored in the database. Such an approach will guarantee correctness from the point of view of the high user. Inference controls mean that complete data correctness is not always possible for **LOW** users.

Since implicit constraints are always available to users whether or not they are allowed access to them in the database, it is useless to attempt to reduce inferences by classifying implicit constraints. Therefore, data over which the constraint are defined must be uniformly classified. Since the data over which the constraints are defined may be overlapping, it is difficult to assign security levels in a consistent manner while limiting the damage to data availability. For example, Su has shown that the problem of classifying data so that data availability is maximized, and the security class of each key in an implicit functional dependency dominates the classes of all data depending on that key, is NP-complete [Su 86] (see also [Su 87,90]).

4.2.2 Inference Detection Methods

Inference through secondary path analysis can be characterized as multiple paths between two attributes such that the paths' security levels are inconsistent [Binns 92a]. Given a large enough database, it is not likely that a database designer can be aware of all possible interactions between tables. Alternative, hence, secondary paths may exist which ultimately relate two attributes in a manner unforeseen by the designer. If the classification of the secondary path is different than the primary path, then a classification inconsistency exists. Classification inconsistency is not acceptable regardless of which path is conceived to be "primary". Binns describes an approach that aids the database designer by identifying potential aggregates and inference paths automatically.

The use of conceptual structures to handle the inference problem was first proposed by Hinke [Hinke 88]. A semantic network can be regarded as a collection of nodes connected via links. The nodes represent entities, events, and concepts. The links represent relationships between the nodes. Numerous variations of semantic networks have been proposed, including the conceptual graphs of Sowa, and are called conceptual structures [Sowa 84]. Conceptual structures are useful in that they can represent and reason about the real-world like humans. The database security research community became interested in conceptual structures because they can be used to represent a multilevel application [Thuraisingham 91].

Hinke showed how inference may be detected by traversing alternate paths between two nodes in the graph. Graph traversal in semantic nets corresponds to limited inference. The classification of implied links found by graph traversal is determined by the classifications of the traversed links. The technique enables detection of simple inferences via the transitivity property, but does not enable the detection of more complex inferences. Others since have also shown how to represent multilevel database applications using conceptual structures developed for knowledge-based system applications and subsequently reasoning about the application using deduction techniques [Thuraisingham 90f, 91; Delugach 92, 93; Hinke 92; Garvey 92].

Auxiliary semantic nets can be used to express security constraints. Smith suggests extensions to

Urban's semantic data model to represent multilevel applications [Smith 90b; Urban 90]. Smith states that eventually the representation could be translated into statements of a logic programming system that could then be used to detect inferences.

4.3 RUN-TIME MECHANISMS

The previous section contained many different methods of controlling inferences through careful database design. The methods surveyed in Section 4.2 are, for the most part, application dependent. For example, it was seen that using proper database design, using the designer's knowledge about the data, and classifying data at an appropriate level all help with inference control. These activities occur outside a DBMS and, therefore, are not within its sphere of control.

However, certain run-time mechanisms can be provided to help with other approaches. These mechanisms are listed below and discussed in the remainder of the section.

1. <u>Query response modification</u>. Users can build reservoirs of knowledge from responses that they receive by querying the database and it is from this reservoir of knowledge that they infer unauthorized information. Two methods to address this problem are 1) to restrict queries by modifying the query appropriately during the query evaluation phase, and 2) to modify query responses (see Section 4.3.1).

2. <u>Polyinstantiation</u>. Some inference threats arise due to key integrity, which is a basic integrity requirement of the relational model, and these violations can be eliminated if the DBMS allows the use of polyinstantiation (see Section 4.3.2).

3. <u>Audit Facility</u>. Since auditing can be used to detect inferences, a DBMS should provide an adequate audit facility so user activities can be monitored and analyzed for possible inference violations (see Section 4.3.3).

4. <u>Snapshot Facility</u>. There are situations in which it was desirable to release some **HIGH** information to **LOW** users for reasons of convenience or necessity (as in the case of U.S. Bureau of the Census described in Section 4.3.4). In these cases, snapshots can be used effectively to eliminate or minimize the harm done by release of **HIGH** information. Thus, it is desirable that multilevel secure DBMSs provide a snapshot facility as well.

4.3.1 Query Response Modification

To block an inference channel during a session, user queries can be restricted or the query results modified. Diminishing the potential of query restriction control is the finding that through certain sequences of seemingly innocuous unrestricted queries called trackers, the answers to restricted queries can always be deduced [Denning 78]. Therefore, response modification, which introduces uncertainty so as to reduce the risk of disclosure, has been the focus of much run-time inference control research.

4.3.1.1 Constraint Processors

Some MITRE research focuses on handling inferences during query processing [Keefe 89; Thuraisingham 87, 88, 90a]. The query modification technique is illustrated in Figure 4.4 below

[Ford 90].

MITRE has augmented Hewlett Packard's Allbase with a logic-based Constraint Processor and knowledge base in their current prototyping effort [Thuraisingham 94]. Security constraints contained in the knowledge base are utilized by the Constraint Processor to modify queries to prevent security violations through inference. Factors used by the Constraint Processor for query modification and response sanitization include 1) the security constraints, 2) the previously fetched tuples, and 3) external knowledge.

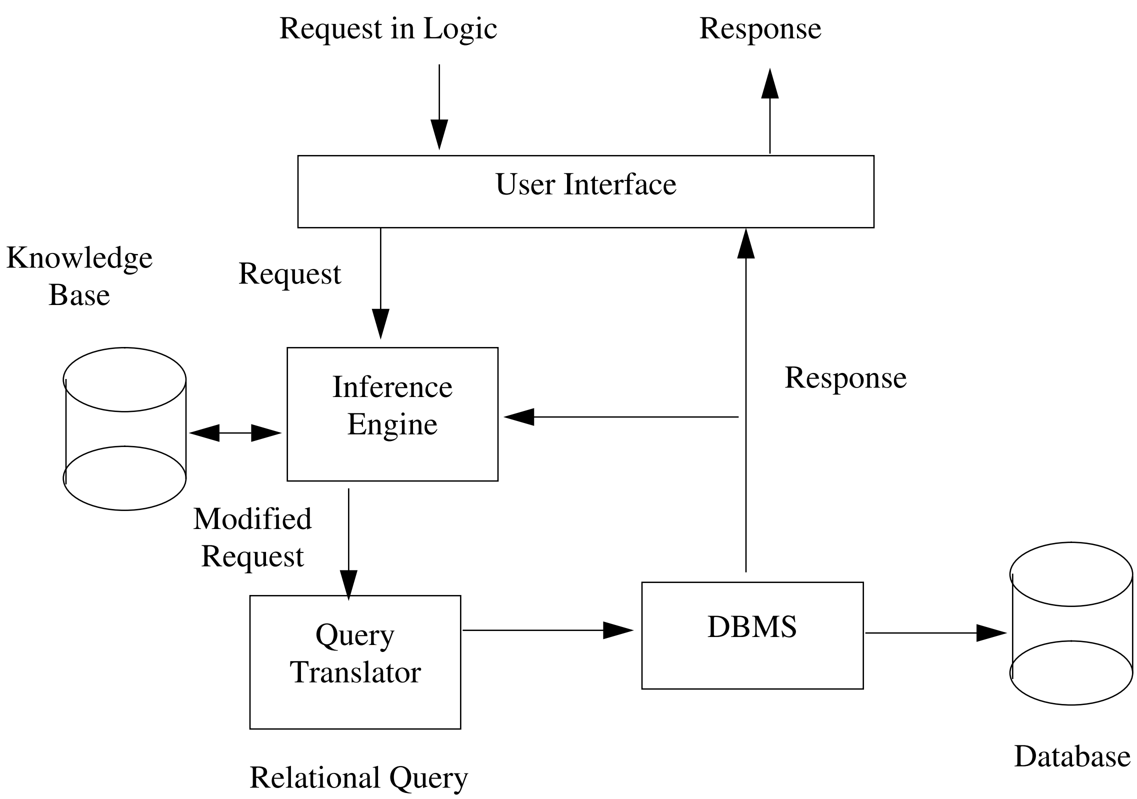

Figure 4.4: Query Processor

An Update Constraint Processor was developed to accurately label data as it is being loaded into the database during database update or design. This alleviated the need for the Constraint Processor to process all constraint types which could impact query response time. The Update Constraint Processor is designed to operate with the Constraint Processor to accurately label data associated with the simple and context-dependent constraints defined for an application.

The prototyping effort has a goal of extending from a centralized to a distributed environment with trusted DBMSs interconnected via a trusted network. MITRE implemented modules to augment the Constraint Processors to process security constraints in a distributed environment during the query and update operations.

In addition to the citations above, results of the MITRE-based research on the inference problem1 including previous prototyping efforts, have been presented at various security conferences and documented in MITRE technical reports [Thuraisingham 89a, 89b, 90b-f; Collins 91].

4.3.1.2 Database Inference Controller

The Ford Aerospace proposed Database Inference Controller (DBIC) was a knowledge- based tool or set of tools used by the System Security Officer (SSO) to detect and correct logical inferences [Buczkowski 89a, 90]. The DBIC system configuration is given in Figure 4.5 [Buczkowski 89b]:

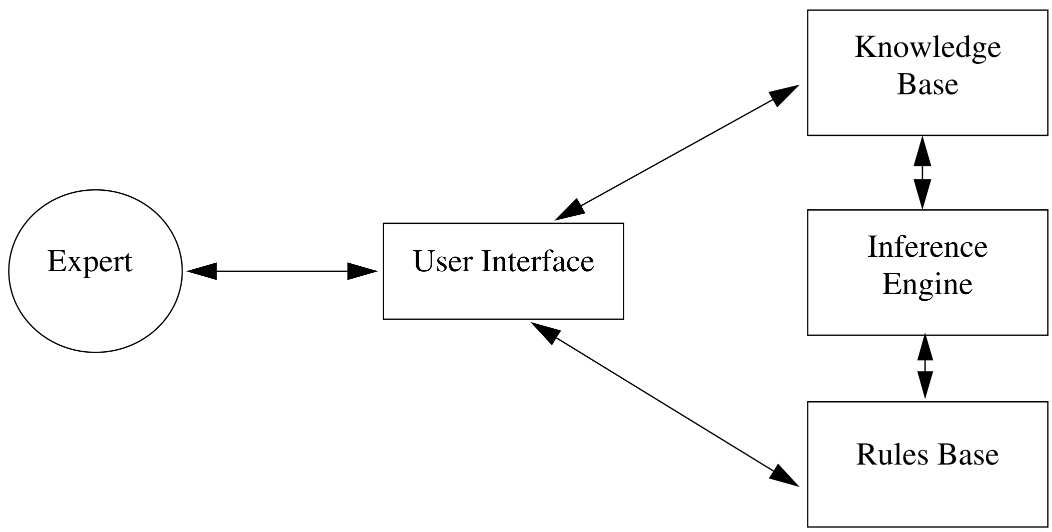

Figure 4.5: DBIC System Configuration

Rules would cause the Inference Engine to take the following actions:

1. Determine which queries (along with relevant data) are presented to the user during knowledge elicitation.

2. Check consistency and completeness of data provided by the user.

3. Initiate procedures to calculate the probability of inference and identify violations.

4. Determine how the Knowledge Base subsystem is updated with the data received from the user or procedure output.

5. Recommend classification changes to parameters to correct inference problems.

The DBIC was based on Probabilistic Knowledge Modeling to identify inference, creating and using a Probabilistic Inference Network on and integrated with a semantic model of the target database. The Probabilistic Inference Network is a structure that identifies the logical dependencies of classified parameters on aggregates of objects at lower classifications.

The Ford Aerospace research culminated in a top-level DBIC design [Buczkowski 89b]. One of the conclusions of the effort was that aggregation is a special case of the inference problem.

4.3.1.3 Secondary Path Monitoring

Section 4.2.3 describes secondary path analysis as an approach for the database designer to detect potential inference paths. Binns also notes that removing all potential inference paths is not always feasible and that run-time analysis based on the potential inference channels detected but not removed during design may be necessary [Binns 92a].

Path definition yields the attributes, relations, and order of traversal used to substantiate the path. Thus, the definition indicates what the path is. Given the potentially offending inference path, Binns proposes that run-time analysis could monitor its existence. Each time a relation in the primary or secondary path is modified, the run-time analysis would fire a trigger to detect if a classification inconsistency exists. Triggers that monitor the existence of an inference path must dominate both the primary and secondary paths, or run as a trusted process. The run-time mechanism can flag an active inference channel to the SSO.

4.3.1.4 MLS System Restrictions

Inference problems can arise if queries are conditioned on data that are supposed to be invisible to the user. The inference problems of this type can be eliminated by ensuring that all data used in evaluating a query are dominated by the level of the user. In general, it is difficult to determine when a given operation can be replaced by an operation defined over data classified at lower security levels. Thus, it is necessary to develop algorithms for determining when an operation at a **HIGH** security level is equivalent to an operation at a **LOW** security level. Some work has been done on this problem by Denning [Denning 85].

There is a simpler way to deal with this type of inference problem. In certain implementations, the system will be unable to access the higher level data, and the query will not pose an inference problem. For example, a multilevel system rated at class B1 or higher implements a mandatory security policy that includes the simple security condition. Since only. those relations which are dominated by the user's clearance level (**UNCLASSIFIED** in this case) will be available to a subject processing this query, such queries will not result in an inference violation based on access to higher level data.

4.3.2 Polyinstantiation

A solution to inference violations that occur because of key integrity enforcement has been proposed by Denning et al. [Denning 87]. Suppose a user operating at the **LOW** security level wants to enter a tuple in a relation in which the data are classified at either the tuple or element granularity. Whenever a tuple with the same key at a higher security level already exists in the relation, both tuples are allowed to exist, but the security level is treated as part of the key. Thus, the tuple entered at the **LOW** security level and the tuple entered at the higher security level will always have different keys, since the keys will have different security levels. A multilevel relation is said to be *polyinstantiated* when it contains two or more tuples with the same apparent primary key values. The TDI companion document on Polyinstantiation [Poly 96] reviews this topic in greater depth.

Garvey et al., suggest development of tools for automatic inference channel detection and cover story generation using polyinstantiation be a long-term goal to provide automated assistance for System Security Officers [Garvey 91c].

4.3.3 Auditing Facility

In some systems, auditing is used to control inferences [Haigh 90]. For instance, a history can be kept of all queries made by a user. Whenever the user makes a query, the history is analyzed to determine whether the response to this query, when correlated with the responses to earlier queries, could result in an inference violation. If a violation arises, the systems can take appropriate action (for example, abort the query).

The LOCK Data Views (LDV) project represents an early attempt in data-level monitoring of inference [Haigh 90,91]. The project has as its goal to design and build an MLS DBMS hosted on the Logical Coprocessor Kernel (LOCK). The LOCK system is being developed to satisfy the TCSEC Al criteria. It includes a type manager that allows assured pipelines to be built to enforce application-specific extensions to the system security policy enforced by LOCK. The word “pipeline” is used because the type manager supports both encapsulation and the security policy by funneling an operation through subjects in different domains, each of which has access only to objects of designated types.

The LDV project has used the type manager and assured pipeline feature to address DBMS-specific issues not covered by the LOCK security policy. This basic security policy is extended by incorporation of an explicit classification policy. The policy covers name-dependent, content-dependent, and context-dependent classification policies, as well as inference control [Stachour 90]. LDV maintains a set of history files that keeps track of which files have been accessed in answering queries from LOW users. As soon as enough files have been accessed by LOW users to compromise LOW data, LDV either upgrades the clearance of the querying process, or the classification of the query result, or the classification of the file that is being queried. The LDV project considers aggregation and inference as aspects of the same problem.

Unlike previous methods that seek to prevent inference problems from arising (by reclassifying data or metadata), auditing seeks to detect inference problems when they are about to occur and to take preventive action.

There is a side benefit of this approach: it may deter many inference attacks by threatening discovery of violations. There are three disadvantages of this approach:

1. It may be too cumbersome to be useful in practical situations.

2. It can detect a very limited type of inferences (since it is based on the hypothesis that a violation can always be detected by analyzing the audit records for abnormal behavior).

3. The length of time for which history files will be kept will impact the amount of audit data recorded

Collusion between users where each user retrieves some of the information remains a difficult threat to counter and is not addressed by this method.

4.3.4 Snapshot Facility

There are situations where it is possible to allow limited inferences [Jajodia 92]. These methods are useful in cases in which the inference bandwidth is so small that these violations do not pose any threat. A snapshot can be created based on a sanitized joined relation for **LOW** users. Although this snapshot cannot be updated automatically without opening a signaling channel, it can be kept updated by periodic recreation by a trusted subject. In off-line, static databases, the "snapshot" or "sanitized file" is an important technique for controlling inferences. In particular, this approach has been used quite effectively by the United States Bureau of the Census [Cox 86,87; Denning 82].

The Bureau of the Census has the responsibility of releasing collected data for public (or unrestricted) use; however, the Bureau must do so in such a way that it is not possible to identify any individual or organization by use of the released data. To achieve both goals, the Bureau uses a conceptual model that is different from the on-line, general purpose DBMS model [Cox 88].

Unlike the general-purpose DBMS model, which is shown in Figure 4.6, the Bureau constructs a separate database that is designed for public use. This public database is the sanitized version of the Bureau database (see Figure 4.7).

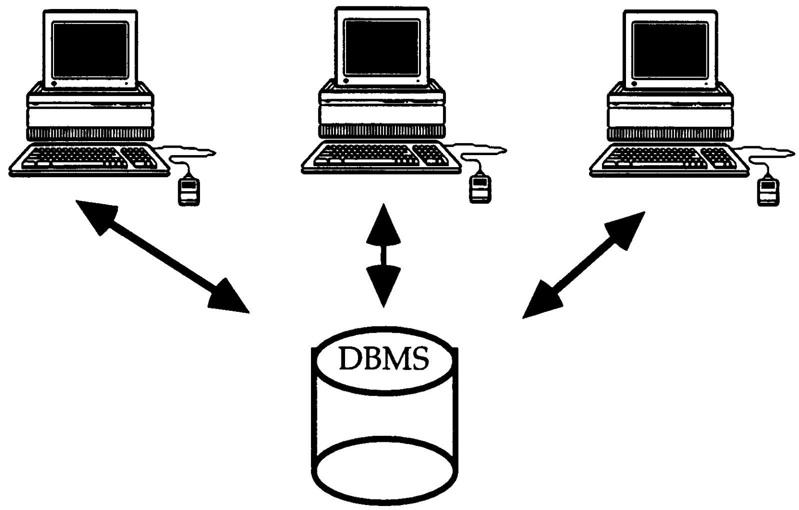

Figure 4.6: General-Purpose Database Management System Model

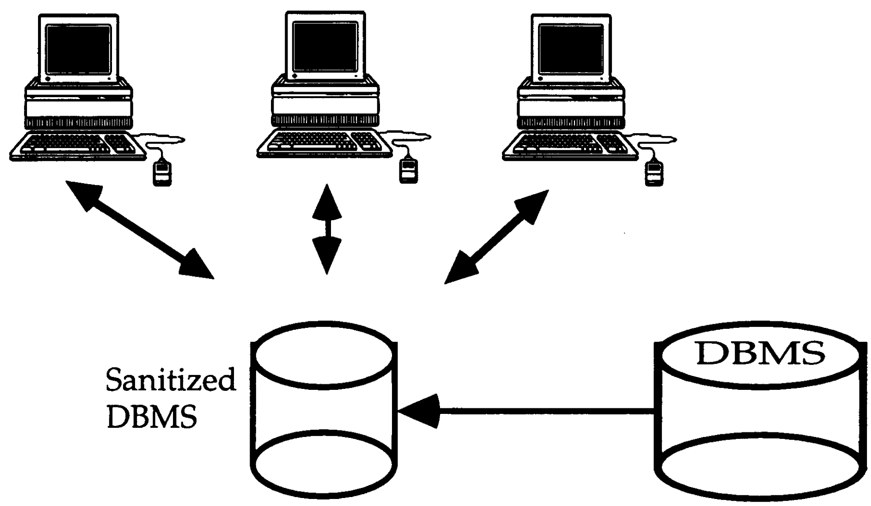

Figure 4.7: Database Management System Model at the U. S. Bureau of the Census

To sanitize the database for public use, the Bureau starts with a microdata file, which typically consists of a randomly selected sample from the database. All microrecords are first stripped of all identifiers (such as names, exact addresses, or social security numbers of individuals), and then different inference control methods are applied to reduce the risk of inference to an acceptable level. Some of these methods are listed below:

1. Subsampling–A second sample is chosen from the microdata to reduce further the probability of possible identification of individuals. Also, records having extreme values are eliminated.

2. Compression–The microrecords are merged together (e.g., income is reported in five-year intervals rather than every year).

3. Perturbation of data–Noise is added to data (by rounding off numerical values or by actually inserting bogus records).

4. Slicing–The microdata file is divided into several subfiles.

While these methods work well in static databases (especially for statistical accesses), these methods must be used with a great care in general-purpose DBMSs. Since these methods perturb the data by adding noise, they have an adverse effect on the integrity of data. Moreover, these methods may be too time-consuming to apply dynamically to on-line databases (in which both statistical and non-statistical accesses must be provided).

4.4 DATABASE DESIGN TOOLS

It is a difficult task for the trusted database designer to label data so that the labels accurately reflect the actual sensitivity of the data and adequately protect the information from inference. This section describes tools for detecting potential inference paths. Action can then be taken to minimize or eliminate these inference paths. The tools are based on the techniques described in Section 4.2.2.

4.4.1 DISSECT

SRI International (SRI) has produced a prototype tool for detecting specific types of inference channels in a multilevel database. DISSECT (Database Inference System Security Tool) primarily detects inferences where two relations are connected by multiple paths with different security classifications. Once these multiple paths are detected, DISSECT constructs the security constraint formulas that if satisfied would eliminate the problem. However, it is up to the user of DISSECT to ascertain whether the paths represent relations that are semantically similar enough that a lower-classification path comprises a higher path. Selected constraints can then be satisfied by manually editing the schema or by an automatic upgrading procedure. A limited set of external information and assumptions are used.

DISSECT accepts as input a multilevel database schema provided in the form of MSQL relation definitions (MSQL is the query language of the SeaView prototype MLS DBMS developed by SRI [Lunt 88]). DISSECT is interactive and allows the user to edit relevant features of the schema. DISSECT can discover a subclass of deductive inference channels (defined in Section 2.1.1) called compositional channels and, with the help of user-supplied type declarations, type-overlap channels:

- Compositional channels - A relationship can be inferred between any pair of entities that are connected by a sequence of foreign-key relationships. An inference channel may exist if two relations connected by a high foreign-key relationship are also connected by a path of one or more low foreign-key relationships. If the two paths between the relations represent essentially the same relationship, this is certainly an inference channel, since the relationship is explicitly classified high, but is accessible via an alternative path to a low user.

- Type-overlap channels - Attributes whose types overlap are joinable. A typeoverlap relationship occurs between two attributes when the two attributes have been declared to be of the same or overlapping types. This corresponds to the case in which a theorem proved using a high axiom is also provable from all low axioms. This may not be a problem, since consequences of high data are not necessarily high.

These channels can be eliminated (by changing attribute classifications) or ignored (e.g., because they are not really problems, are unavoidable, or should be deferred to datalevel handling) at the discretion of the user. Channels can be eliminated by editing the schema to raise or lower some attribute classifications. DISSECT also provides an automated means of upgrading attribute classifications to eliminate all inference channels chosen for elimination. User-specified costs of upgrading attributes express preferences of how channels should be eliminated.

The SRI DISSECT research effort has been completed and documented [Garvey 94]. The DISSECT prototype has demonstrated its ability to detect and to optimally upgrade some attributes to automatically eliminate compositional and type-overlap inference channels. Results of the DISSECT-based research has been presented at various security conferences and workshops [Stickel 94; Lunt 94; Qian 92, 93a, 93b, 94; Garvey 91a, 91b, 91c, 93; Binns 92a].

4.4.2 AERIE

The University of Alabama in Huntsville has developed an inference model called AERIE (Activities, Entities, and Relationships' Inference Effects). The project is examining approaches to inference modeling and detection using inference targets (sensitive data that are to be protected from unauthorized disclosure through inference). The approach requires that the specific information that is to be protected be named, and inference paths to this information will be sought. The approach describes the database schema, the data, and the sensitive target information using conceptual graphs. It utilizes a knowledge-base of domain-specific information that can be expressed as a conceptual graph to be used in inference analysis.

A precursor to AERIE was the TRW inference engine analysis and the effort to discover inference rules from fundamental relationships among data that pertain to a domain [Hinke 90]. In TRW's Advanced Secure DBMS project, a tool was proposed to identify potential schema-level inference paths, and queries could be run periodically to detect the existence of data-level inference channels. One of the results of the effort was that the specific rules required to find the classified relationship did not have to be explicitly stated. The AERIE research is based on these results and on conceptual graph structures. AERIE extends the inference techniques suggested by Thuraisingham [Thuraisingham 90f] by classifying inference targets and using a two-phase inference approach. Two supporting prototype tools were developed: an inference detection tool called Merlin to validate the AERIE algorithms; and a rule-based, automated inference database generator called Genie to facilitate the development of a large, coherent set of inference relevant instances.

Merlin uses functional dependencies to detect second paths with schema-level data. Merlin adapts a transitive association algorithm for testing a relational database decomposition for the lossless join property [Elmasri 94]. The Merlin adapted algorithm is used in the detection of second path inferences within schema-level data and includes subtype relationships between attributes within the schema. Merlin has demonstrated identification of a secondary path between two attributes and is able to categorize secondary paths based on schema classification. However, it is recognized that legitimate inferences are not being detected by Merlin. Future plans of the AERIE project are to 1) move beyond functional dependency-based inference detection, 2) integrate path detection with path finding, and 3) adapt Merlin to handle catalog and instance levels of data.

Genie was developed to facilitate the creation of an inference benchmark database for the testing of inference detection tools. The database created by Genie is to contain inferences that represent some coherent activity in some microworld with the potential to grow to a size representative of a real database. Genie is encoded with rules to simulate activities of a microworld, providing the basis for the generation of coherent data.

Results of the AERIE-based research have been presented at various conferences and workshops [Hinke 92, 93a, 93b, 94; Delugach 92, 93].

4.4.3 Security Constraint Design Tool

MITRE developed a tool for use during database design using an Informix DBMS [Thuraisingham 94]. The tool provides guidance in assigning minimum security classifications to the database schema as well as establishing minimum security classifications for the database schema of an existing non-secure database migrating to an MLS environment. The design tool examines simple, associative-based, logical and level-based security constraints and determines a minimum classification level for database attributes. This information can be utilized by the database designer in the design of an MLS database schema.

4.4.4 Hypersemantic Data Model and Language

Section 4 began with the recognition that a general solution to the inference problem is not possible. The techniques, mechanisms, and tools described in Sections 4.2,4.3, and 4.4 are partial solutions that complement each other. Unfortunately, the approaches are highly specialized. No single tool can handle all types of inference problems. Marks proposes that what is needed is a uniform representation scheme which can be transformed into specific representation schemes without much complexity (Marks 94a]. So while this approach is not a database design tool in and of itself, once done, various inference analysis approaches can be applied. For this objective, a hypersemantic data model called Multilevel Knowledge Data Model (MKDM), a graphical representation scheme called GRAPHICAL-MKDM, and a specification language called Multilevel Knowledge Data Language (MKDL) have all been developed.

Future research could include:

1. Developing inference analysis tools for MKDM.
2. Developing transformations to other representations so that current inference analysis tools can be applied.
3. Testing tools with examples.

SECTION 5

APPROACHES FOR AGGREGATION

There is no general approach or mechanism for dealing with general aggregation problems, only specific measures taken to protect specifically defined aggregates. This section does not describe how to identify sensitive aggregates. Rather, it presents approaches for protecting certain classes of aggregation problems. A discussion on identifying aggregates is not presented because aggregates can be defined in almost any arbitrary way in order to protect almost any fact. As such, identifying an aggregate is totally up to the owner of the data. Once an aggregate which requires protection has been identified, it hopefully can be classified as either one of the two sub-classes of aggregation problems defined below and an appropriate approach defined in Section 5.1 or 5.2 can be used to protect that aggregate. If an aggregate does not cleanly fit into one of the two aggregation sub-classes, it may be a combination of the two forms of aggregation or another problem altogether, such as access control or inference. If an aggregate exhibits both aspects of the aggregation problem, it needs to be broken down into its constituent parts and each part protected appropriately.

This section describes the various approaches that have been identified for dealing with specific classes of aggregation problems. Some of these approaches can be used now while others are still ongoing research. As stated in Section 1, the aggregation problem occurs when each individual data item is less sensitive than the collection (or aggregation) of all the data items. This definition distinguishes the aggregation problem from the inference problem which uses **LOW** data to infer **HIGH** data, rather than a collection of **LOW** data being considered **HIGH** because of the quantity of, or relationships between, the data. Aggregation has been divided into two sub-problems [Jajodia 92]:

- Data Association - associations between instances of different data items.
- Cardinal Aggregation - the associations among multiple instances of the same entity.

5.1 DATA ASSOCIATION

This section presents the research to date on the data association problem. It specifically describes the work being done to formalize the description of the data association problem and possible solutions, and then presents the primary technique that has been identified for solving specific instances of this problem. No tools for detecting data association problems or mechanisms which require changes to standard DBMSs are presented as none have been developed or proposed.

5.1.1 Formalizations

Researchers have attempted to apply mathematical approaches to the data aggregation problem, especially in the context of MLS DBMS implementations. This section discusses the approaches that have been proposed to date.

5.1.1.1 Lin's Work

Lin applies a security algebra defined over classifications and extended over relational database operations to mathematically express the aggregation problem [Lin 89a]. The security algebra works with classification labels under the constraints of Denning's classification lattice algebra [Denning 76]. Lin defines a homomorphism to relate the classification of a relational operation's output to the individual classifications of the operation's operands. The reader needs to pay close attention to the algebra's notation, as Lin's analysis requires comparisons between purely label-based operations on classification labels under the lattice algebra (called 'algebraic classification') and the real-world security class of data (the classification of operations on data). The algebra of label manipulation is fully intuitive to readers familiar with Bell-LaPadula [Bell 76]: the label of a process that can read a set of data needs to dominate the least upper bound of the labels on its individual inputs; the label on a process that creates a set of outputs needs to be dominated by the greatest lower bound of the labels on the individual outputs.

Lin observes that an aggregation problem exists when this dominance constraint fails to hold as an equality -- i.e., when the real-world classification of a set of inputs fails to exactly equal the least upper bound of their individual classifications. He allows for the traditional case (the classification of the set strictly dominates the classification of its members), but also for the case where the real-world classification of an operation is strictly dominated by the least upper bound of the inputs to the operation, as in the results of certain aggregate functions ('built-in' operators like SUM, AVERAGE, etc.). This algebra is then used to formally define the `aggregation problem for a database'.

Lin deals with the data-association aggregation problem, and not with the cardinality aggregation problem, and defines an aggregation to be 'simple' if it is a set of data whose real-world classification strictly dominates the algebraic classifications of each of its constituent elements with the further constraint that no proper subset of the set has this property (i.e., the classification of any proper subset equals the least upper bound of its constituents' classifications). Since under this definition, the removal of any element(s) from a simple aggregate suffices to 'solve' the aggregation problem, Lin presents an algorithm that requires the SSO to identify sets of elements that can be removed from a simple aggregate, and reclassified at a strictly dominating level in order to ensure that none of the resulting collections could result in disclosure of a simple aggregate to inadequately cleared users [Lin 89b]. Using this technique, it is up to the owner of the data to decide which sets of data are to be more highly protected and which are to be left at their original classification. Lin extends this work somewhat in [Lin 90a, 91]. An example problem that his technique would address would be if we had the following simple aggregates:

- A, B, C
- A, C, F
- D, F, Y, Z
- E, F, G, H

where a 'simple aggregation problem' is as defined above. Given these four simple aggregates, Lin's technique would identify that A or C, and F were important elements that must either be protected, or all other elements need to be protected in order to protect the aggregates. For example, if elements A and F were marked **HIGH**, and all other elements were marked **LOW** then no user can compile all the elements of these aggregates without accessing **HIGH** data. Lin's method would identify that C was equivalent to A such that it could be protected instead. Conversely, access to A and F or C and F could be granted at **LOW** while all other data was protected at HIGH and the aggregates would be equivalently protected. It is up to the owner of the data to decide whether it is more important to make available the largest number of elements, or certain critical or more interesting elements, while protecting enough other elements to ensure that an aggregate cannot be collected together solely at the **LOW** level.

Lin goes on to define an important homomorphism, H, to relate the operations of relational algebra and security classification algebra. It is essential that this mapping be a homomorphism, and that it satisfy commutative mapping diagrams (i.e., that the mapping under H into a classification of a relational operation on two operands equal the algebraic classification of the operands' mapped labels in the label-combining algebra -- e.g., the classification of a join equals the least upper bound of the classifications of the join's arguments). Aggregation problems exist when equality fails. Lin notes that there is a need to consider aggregations that could hold if the homomorphism were non-commutative aggregation, indicating that it will be addressed in a forthcoming paper [Lin 90b, 91]. Lin extends the work on simple aggregation problems to towers of aggregation problems [Lin 91]. The real-world classification of a tower strictly dominates the least upper bound of the classifications of the tower's components, but there exist proper subsets of the tower that are simple aggregation problems. That is, given

- Individual elements A and Band C may be **UNCLASSIFIED**
- A, B may be a **LOW** simple aggregation problem
- A, B, C may be a **HIGH** tower aggregation problem

Lin would solve <A, B> by classifying, say B as **LOW** and replacing it with an **UNCLASSIFIED** element E (e.g., a cover story). This would reduce the **HIGH** tower aggregation problem to <A, E, C>, which may not be a problem or which may now be a simple aggregation problem.

Lin also discusses the case of statistical aggregation problems where the application of an aggregate function (built-in operator) produces an output whose classification is strictly dominated by the least upper bound of its inputs [Lin 91].

5.1.1.2 Cuppens' Method

Cuppens has defined a formal method for addressing the aggregation problem which he refers to as a modal logic framework [Cuppens 91]. This logic is based on modeling what a user knows as compared to what a user has permission to know. This logic framework can be used to model a security policy with aggregation and then prove whether a particular specification is secure or not. It does not detect aggregation in a database management system but rather determines whether a particular specification properly protects data as specified by a security policy. An aggregate must be predefined as part of the security policy in order for the technique to determine whether a particular specification will properly protect the data as required by the security policy.

5.1.2 Techniques

Since the data association problem is concerned with the sensitive association between two or more distinct less sensitive data elements, Denning [Denning 86a] and Lunt [Lunt 89] have both proposed fairly straightforward solutions to this problem. They point out that since a quantifiable entity (the sensitive relationship) requires more protection than the individual elements, database design and access controls can be used to control the data association problem. The basic solution to this problem is for the individual data items to be protected as normal, but the ability to associate the items together is removed from each item's relation. A separate relation is then created which describes the relationship between the items and is protected appropriately.

Assuming you have an MLS DBMS, this new relation is classified at the proper level for the aggregate. Because the data items cannot be aggregated without reference to the higher level relation, only users cleared for the level of the aggregate will be able to view the aggregate as an entity. Less cleared users will be able to access the individual elements at their proper classification. This approach has the additional advantage that the mandatory access control mechanism will ensure that the aggregate is properly labeled at the least upper bound of the accessed data. In this case, the new relation's higher classification.

Using our fictional company ATC we see two examples of data association. In the ATC database, Project names are **UNCLASSIFIED** while Client names are **LOW** but the association between a Project and its sponsoring Client is classified **HIGH**. This example of inter-entity data association is properly protected by placing each entity, as well as the relationship between them, in separate tables that can be independently labeled with the appropriate classification. An example of intra-entity data association is the fact that all information about an employee is **UNCLASSIFIED**, except for the employee's salary, which must be protected at **LOW**. To properly protect the association between the employee and the employee's salary, the salary field is stored in a table separate from the rest of the employee data and protected at **LOW**. These two associations and their solutions are shown as follows:

Research Projects [U]

ID	Name	Description

[U, L]

Client [L]

Name	Address	POC

[L]

Client-Project [H]

Client.Name	Proj.ID

[H]

Inter-Entity Association
between Project and Client

Employee [U]

Number	Name	Dept.	Home.Address	Home.Telephone

[U]

Employee-Salary [L]

Employee.Number	Salary

[L]

Inter-Entity Association
between Project and Client

Figure 5.1: ATC Company Data Association Examples

If an MLS database is not available, then these more sensitive relationships will have to be removed from the database or protected using discretionary access controls (DAC). The SSO can define roles or individual users that are associated with each type of data and then each user or role can be restricted to access only the appropriate tables. The restrictions can be in the form of DAC on the underlying base tables, as well as providing predefined views that are restricted to the appropriate users or roles. For the example above, to use DAC, one would protect the Client-Project and Employee-Salary tables with DAC controls that restricted access to only those users who should be able to see this information. DAC may be a reasonable solution for mitigating the data association problem in some cases, but, given the inherent flaws of DAC, it may not provide sufficient assurance that the data is properly separated depending on the sensitivity of the aggregate. Such a solution can be used to separate categories of data (e.g., budget, sales, or cost data). However, if aggregating all the data within a specific category is an issue, this solution will not, for example, prevent a budget clerk from accessing all of the budget data. Accessing all the data within a particular category is a cardinal aggregation issue, and approaches to this problem are discussed in the next section.

The fundamental technique then to solving aggregation problems that fit into the sub-class called data association is to first identify whether a more sensitive aggregate is truly a data association problem, and if so, then properly construct the database to protect the more sensitive relationship. This protection implies separating the entity's relationship and defining a separate table which associates the entities. The table that relates the two entities together must then be appropriately protected at the level of the aggregate. This solution protects each entity at the appropriate level through careful database design without the need to overclassify any individual element of the data.

5.2 CARDINAL AGGREGATION

A fairly straightforward technique was presented for solving the data association problem. Unfortunately, similar straightforward techniques have not yet been identified which can address the cardinal aggregation problem in general. A number of techniques and mechanisms have been proposed to help address the cardinal aggregation problem. This section presents those that have been identified. No tools or formal methods have yet been developed for dealing with the cardinal aggregation problem.

5.2.1 Techniques

This section describes those techniques that rely solely on standard capabilities available in the typical commercial DBMS as well as those that would be expected to be present in Multilevel Secure (MLS) DBMSs.

The following techniques each address a certain aspect of the cardinal aggregation problem. They vary in complexity, effectiveness, and side effects or drawbacks. Remember that aggregation is a concern only when the security policy for a system specifically indicates that a collection of less sensitive items is considered more sensitive when aggregated together. If the policy does not indicate this is so, then no aggregates have been defined which must be dealt with. Once an aggregate has been identified, then the most appropriate approach for dealing with that particular aggregate will depend on the nature of the aggregate. The number and interrelationships between aggregates will also affect which techniques would be most appropriate. To date, most techniques do not consider interrelationships between aggregates and the complexity interrelationships can cause when proposing a solution to the cardinal aggregation problem. Such interrelationships can make it more difficult to provide acceptable solutions which balance the need to protect sensitive aggregates while still providing access to the less sensitive individual elements.

5.2.1.1 Single Level DBMS Techniques

Cardinal aggregation control is a problem in databases used in commercial applications as well as databases used in secure, system-high applications. In both these environments, techniques used for cardinal aggregation control have been developed in an ad hoc fashion and not uniformly applied. Most of these techniques are administrative and physical; examples are control and review of hard copy output and segregation of data across multiple machines. The following techniques rely on the use of DBMS security mechanisms and can be applied using most any commercial DBMS.

5.2.1.1.1 Database Design and Use of DAC

One of the simplest techniques to minimize cardinal aggregation in single level databases is to divide data that could aggregate into separate structures [Lunt 89]. Each of these separate structures could then be protected with discretionary controls that restrict individual users from aggregating enough information to construct an aggregate whose size was considered more sensitive than the individual elements. For example, if there were 1000 data elements which were divided into 10 blocks of 100, and an aggregate of over 300 was considered sensitive, then each user could be limited to accessing at most three of the 10 data blocks. Which three may or may not be arbitrarily selected. Such a structure would mitigate the cardinal aggregation problem to some degree, but it has the significant drawback that users with a legitimate need to access certain individual elements may be prevented from doing so, depending on whether the element resides in the part they are permitted to see, or in the part for which they are restricted. It is also difficult to administrate such a structure if the amount of data rapidly changes over time, or the number of users changes to any significant degree.

Now if the data in an aggregate can be distinguished in some way, such as budget, sales, and cost data, then it makes it easier to decide how to divide up access to an aggregate among sets of users. However, this is a data association problem, not a cardinal aggregation problem, since cardinal aggregation only deals with multiple instances of the same entity. A single aggregate can have both data association and cardinal aggregation aspects to it. As in the budget, sales, and cost example just discussed, the distinction between the elements is a data association problem, while the collection of a sufficient number of elements within a single category is a cardinal aggregation problem. Such an aggregate needs to be broken into its constituent components and then each aspect of the problem dealt with using its respective technique.

An inherent vulnerability in any solution relying on DAC is that any user with access to data can make that data available to others because of the limitations of DAC in controlling data propagation. As such, this type of solution may not be sufficiently strong to protect the data in addition to the inherent drawback of restricting access to data from users with a legitimate need to access it.

5.2.1.1.2 Use of Views

Views can be used to provide access to specific portions of the database. Use of a view can then be granted to individuals which might not have the right to access the underlying base relations. Views can then be used to more precisely control what a user sees when accessing the database. These controls can help protect aggregates in a number of ways. Summary and grouping data can be presented which shield the user from accessing individual elements. Depending on the nature of the aggregate, such data may not be considered more sensitive than the individual elements, while the individual elements when seen together may be more sensitive. For example, a view may allow a user to ask how many of a particular element exists, or the average element value, and this might be acceptable to reveal, as long as the values of all the individual elements are not revealed. Another view may show individual element values but restrict the user to only a certain number of elements in any given query. Both techniques may help protect the aggregate.

However, it is typically difficult to be sure that views provide sufficient protection, depending on the number of views available, their complexity, and the size and volatility of the database. Normal views do not keep track of what each user has accessed over time so the user could assemble an aggregate by issuing numerous queries which each access only a few elements at a time. Group functions which attempt to hide the underlying data using sums, averages, counts, etc. may be subject to statistical inference attacks described in Section 4. Views should be able to protect data from the casual viewer but a persistent attacker may be able to circumvent the restrictions placed by views.

5.2.1.1.3 Restricting Ad Hoc Queries

One desirable feature of a relational DBMS is that it gives users the ability to formulate ad hoc queries against any database object. However, if all users are allowed unrestricted ad hoc queries, the SSO may not be able to pre-identify the likely aggregation results and therefore may not be able to define controls appropriately. This decision to allow ad-hoc queries is a tradeoff between convenience and security.

If ad hoc queries are restricted, then the administrator can rely on protecting access to specific views as just described, thereby relying on the access controls they provide to limit each user's access to only that data they need to perform their assigned tasks. It may not be necessary to restrict ad-hoc queries from all users. If certain users are cleared for all the aggregates contained on the system anyway, it may be perfectly reasonable to allow them to execute ad-hoc queries. In this way, this capability is not arbitrarily restricted from all users. In fact, most systems would not really restrict this capability from every user, because at least one user (the database administrator) typically needs to be able to define new views and issue arbitrary queries.

5.2.1.1.4 Use of Audit

Another approach relies on the use of auditing tools to track the accesses of each user to elements of an aggregate. Assuming sufficient data has been audited, an administrator may be able to review the data and detect whether a user is trying to collect an abnormally large number of values of a particular data element (e.g., too many employee phone numbers). Hopefully this audit analysis can be done by automated means, but few automated audit analysis (i.e., intrusion detection) tools currently exist. More typically, these analyses are done manually, long after the event, by an SSO on a set schedule. The audit trail analyses look for outputs that contain aggregates more sensitive than their component data elements. Based on these results, the administrator can then either restructure the database or apply other controls that would prevent the same problem in the future.

This approach has a number of significant drawbacks, including 1) it attempts to detect, not prevent, an aggregate from being pulled together, 2) today's audit analysis tools are typically difficult and time consuming to use, 3) the likelihood that an administrator would have the time or ability to detect a non-obvious aggregation attempt is questionable, 4) a determined adversary can easily avoid detection through slow and careful gathering of data.

5.2.1.1.5 Store all Data System-High

One of the simplest solutions to the aggregation problem, but one with significant drawbacks, is to store all the data in a single level (system-high) database which is classified at the same level as the most sensitive aggregate in the database. By definition then, all users of the database must be cleared for all the data, thus eliminating the possibility of exposing an aggregate to an individual that is not cleared to see such an aggregate. In essence, this solution proposes that you remove all users from a system who you do not want to access all the data. This has the significant drawback that all the less sensitive data, which may be the bulk of the data, is overclassified, possibly highly so, and inaccessible to individuals who should otherwise be able to access the data. Thus, although this addresses the security concerns related to the aggregation problem, it is not normally a practical alternative depending on the nature of the data. Now for a particular dataset, if only a small portion of the data is normally classified less than this system-high level, then this may be a practical solution.

If this type of solution is used, the less sensitive data must be reviewed before it is regraded to a lower level before dissemination. This review can be automated for specifically formatted, tightly controlled data, but is typically reviewed manually. If done manually, then the output is typically hardcopy, rather than in electronic form. Depending on the nature and volume of data, this review task can be extremely burdensome to perform. Sufficiently so, that this solution is impractical in many situations.

5.2.1.2 Multilevel DBMS Techniques

There are some additional techniques for addressing the aggregation problem by specifically labeling the sensitivity of data (e.g., **UNCLASSIFIED**, **LOW**, **HIGH**) and using this labeling capability, and the ability to control access based on labels, to help control the cardinal aggregation problem. These techniques build upon the techniques described in the previous section and are discussed below.

5.2.1.2.1 Use of Labeled Views

Wilson proposed that views in a multilevel system be used to properly label aggregates with their sensitivity level [Wilson 88]. Views are predefined queries which are labeled. They allow a user to access certain portions of the database in a well known, controlled, manner. Views can be preconstructed to access portions of an aggregate. If a view allows access to a small portion of an aggregate then the result that is returned can be labeled at the level of the data accessed. However, if certain views access large portions of an aggregate, then the result returned should be labeled at the level of the aggregate, which may be higher than the sensitivity of the individual elements accessed by the view. This can be accomplished by labeling the sensitivity of the view definition at the level of the aggregate. This will cause the results returned to be marked at least at the level of the aggregate and prevent users with insufficient clearance from accessing this view. The idea here is to use views to define aggregates and have them properly label aggregates whenever they are accessed.

This technique relies on the fact that users cannot construct their own queries to get around the classification constraints provided by predefined views. If ad hoc queries are permitted, they should be labeled at system-high, independent of the user's session level, to prevent unauthorized access to aggregates. This means that ad hoc queries by lower cleared users will have to be reviewed and downgraded before being displayed to those users. Based on audit trail analyses, any ad hoc queries that show up frequently could be converted into standard queries with an appropriate label. Use of labeled views and query restriction appears to be a workable strategy in some applications, but has some significant drawbacks. If users rarely need to execute ad hoc queries, this solution may be reasonable. However, if ad hoc queries are normally necessary to accomplish the tasks at hand, then the review and downgrade process could quickly become onerous, depending on the nature and volume of the data being reviewed.

This technique will not prevent a user from accumulating sufficient information on their own to assemble an aggregate of sufficient size to be considered more sensitive than the individual elements. What it does do is automatically label aggregates at the appropriate level for those users that are cleared to see the aggregate. It also make it more difficult for an uncleared user to do so, and auditing may be able to detect the accumulation of data in an aggregate that a user is not cleared for.

This technique may also draw attention to the fact that a sensitive aggregate exists which may be inappropriate for some operational environments.

5.2.1.2.2 Start with All Data System-High

Another approach is to start with all data on the system stored at system-high and then downgrade individual elements as necessary to satisfy the access needs of less cleared users. This will cause the data that is needed to migrate down to its actual level into the hands of individual users while the rest of the data remains at system high. The security advantage here is that a review process is automatically invoked each time a user requires access to an element which is part of a sensitive aggregate and that the data is only given to that particular user, rather than all users at that level. This allows the reviewer to monitor the necessity for and volume of data granted to a particular user. This is similar to the paper word concept where you can call the operator for some organization and request an individual phone number, but you cannot gain direct access to the entire phone book. This allows the operator to verify that you are a legitimate caller (e.g., you at least know the employee's name and where they work).

This approach can work in certain situations, as in the phone book example above, but clearly could be unfeasible when the volume of requests to downgrade data is high.

5.2.1.2.3 Make Some Data High

Similar to the technique of using discretionary access controls to separate data into chunks to ensure that an aggregate of sufficient size cannot be pulled together, sensitivity labeling can also be used to separate data such that it cannot be aggregated beyond an allowable size. In this case, a sufficient portion of the aggregate is marked at a higher sensitivity level (e.g., the level of the aggregate) in order to prevent or make it more difficult to create an aggregate which exceeds the allowed size at the lower level. If such an aggregate were pulled together, it would be marked at the higher level since it would require the inclusion of some of the higher level data. This has the advantage that an aggregate of sufficient size would most likely be labeled at the appropriate level, but the disadvantage that some, but not all, of the data would be inaccessible to users at the appropriate level of the data.

5.2.2 Mechanisms

The previous section described techniques that could be used to reduce the cardinal aggregation problem. These techniques relied on database design techniques and mechanisms available in standard single and multilevel DBMSs. This section describes new mechanisms that have been proposed to address the cardinal aggregation problem which require some degree of modification to standard DBMS mechanisms.

In proposed or prototype database implementations using MLS DBMSs, data aggregation control problems that have arisen have been handled by application-specific ad hoc controls [Doncaster 90]. In addition, other control strategies, which are still ad hoc but are less application-specific, have been proposed. These are described below.

5.2.2.1 Aggregation Constraints

Section 4.3.3 described the Lock Data Views (LDV) approach for dealing with the inference aspects of classification consistency problems. The following describes the LDV approach for dealing with aggregation through the use of aggregation constraints. Aggregation constraints are defined using extended Structured Query Language (SQL) [ANSI 92, 94], and are enforced after the query results have been built. Aggregation constraints are similar to classification constraints, except the assigned security levels are applied only to the entire aggregate, and not to the individual elements [Denning 86a]. The module responsible for this examination may reclassify the query result based upon the constraints [Stachour 90]. The aggregation levels are then used to restrict how much data can be released to a given user. The constraints are enforced by building a rule-based classification scheme into the pipelines. In addition, a history-recording mechanism is consulted and updated whenever a query response is prepared by the DBMS [Haigh 90].

5.2.2.2 Aggregation Detection

The mechanisms just described attempt to control aggregates that are accessed by a single query. In order to completely protect an aggregate, a solution ultimately needs to track which data elements have been accessed by a particular user and then grant or deny access to additional data based on all previous data that user has seen. In this way, the system can ensure that the user cannot collect sufficient information to compile an aggregate which is more sensitive than the user's clearance. However, the difficulty of tracking all the data that any particular user has seen has seemed insurmountable.

Motro, Marks, and Jajodia have proposed an accounting based approach for tracking all information that a user has accessed previously [Motro 94]. This information is then used to make access decisions which ensure that the user cannot assemble an aggregate using multiple queries. This method uses views to define secrets that can be disclosed to a limited degree, and associates a threshold value with each such view, which they call *sensitive concepts*. If any view incorporates a sensitive concept, this broader view is also protected as well.

Access to sensitive concepts is then accounted for each user. In fact, to avoid overcounting due to multiple accesses to the same data, the scheme keeps track of the actual tuples that have been disclosed, rather than just the number of tuples disclosed from a particular sensitive concept. In this way, the method can accurately determine exactly what a particular user has seen before, in order to determine whether a subsequent query is allowed, or should be denied because a threshold would be exceeded. The method also determines whether other related data is equivalent to the data protected by a particular sensitive concept and controls access to this related data as well. For example, if in a particular database all the employees in Division A are located in Building 1 and the employees in Building 1 is a sensitive concept, then employees in Division A would also have to be protected as well.

The method described is limited to databases that are single relations and to queries and sensitive concepts that are projection-selection views, where all selections are conjunctions of simple clauses of the form attribute = value. The authors conjecture that this method can be extended to overcome these limitations.

SECTION 6

SUMMARY

Finding effective approaches for dealing with inference and aggregation continues to be a significant challenge for the security community. This report provides a survey of the state of the art of various aspects of the inference and aggregation problems. It defines and characterizes inference and aggregation as they relate to multilevel secure database systems and describes methods that have been developed for resolving some of these problems.

Inference addresses users deducing (or inferring) higher level information based upon lower, visible data [Morgenstern 88]. The aggregation problem occurs when classifying and protecting collections of data that have a higher security level than any of the elements that comprise the aggregate [Denning 87].

Inference and aggregation problems are not new, nor are they only applicable to database management systems (DBMSs). However, the problems are exacerbated in multilevel DBMSs that label and enforce a mandatory access control policy on DBMS objects [Denning 86a].

Inference and aggregation differ from other security problems since the leakage of information is not a result of the penetration of a security mechanism but rather it is based on the very nature of the information and the semantics of the application being automated [Morgenstern 88]. As a result, inference and aggregation are not amenable to complete solution through mechanisms in the DBMS. They inevitably have to be addressed in the context of the semantics of the data.

Inference and aggregation are different but related problems. Each of them can in turn take many different forms. Researchers have addressed different aspects of the problems making it difficult to relate different research efforts. In order to more precisely understand what inference and aggregation are and provide a framework for understanding partial solutions it is essential to revisit several *first principles* related to security concepts and processes. These concepts, described in Section 2, include:

- Identifying methods and types of information that can be used to make inferences.
- Characterizing two classes of aggregation problems: data association and cardinal aggregation.
- Distinguishing between direct disclosure and inference.
- Understanding the relevance of the traditional security analysis paradigm.
- Identifying the steps in a database security engineering process.

The framework presented for reasoning about inference and aggregation identified types of classification rules and vulnerabilities at each step of the database security engineering process. This approach allows one to:

- Establish the relationship among information protection requirements, implementing classification rules, database design, and database instantiation.
- Distinguish between the types of classification rules (entity, attribute, data association, and cardinal aggregation).
- Distinguish between direct disclosure and inference vulnerabilities.
- Identify specific vulnerabilities that are introduced at each step in the database security engineering process.
- Differentiate between inference as a vulnerability and cardinal aggregation as both a class of policies and as vulnerabilities.

Utilizing this framework, Sections 4 and 5 describe research efforts and other approaches to provide techniques, mechanisms, and analysis tools to address aspect of inference and aggregation.

Eliminating or reducing unauthorized inference remains a difficult problem because of diversity of methods and types of information that can be used to make inferences. The inference problem remains one that is dependent on understanding the value and semantics of data for the part of the enterprise being automated. Techniques, mechanisms, tools described in Section 4 provide support for identifying inference vulnerabilities during database design.

The data association class of aggregation policies can be solved today through database design techniques described in Section 5. Effective mechanisms to enforce cardinal aggregation policies remain a topic for research.

REFERENCES

[Akl 87] S. G. Akl and D. E. Denning. Checking Classification Constraints for Consistency and Completeness. In *Proceedings of the IEEE* Symposium on *Security and Privacy*, pp. 196-201, April 1987.

[ANSI 92] ANSI. Database Language SQL2. In *ANSI K3.135-1992*, American National Standards Institute, New York, December 1992.

[ANSI 94] ANSI. SQL3 - Working Draft. Draft Version, Available From X3 Secretariat, Computer and Business Equipment Manufacturers Association (CBEMA), 1250 Eye St. N.W., Suite 200 Washington, D.C. 20005-3992, September 1994.

[Audit 96] National Computer Security Center, Auditing Issues in *Secure Database Management Systems*, NCSC Technical Report-005, Volume 4/5, May 1996.

[Bell 76] D. E. Bell and L. J. LaPadula. *Secure Computer Systems: Unified Exposition and Multics Interpretation*, The MITRE Corporation, March 1976.

[Binns 91] L. J. Binns. Inference Through Polyinstantiation. In *Proceedings of the Fourth Rome Laboratory Database Security Workshop*, 1991.

[Binns 92a] L. J. Binns. Inference through Secondary Path Analysis. In *Proceedings of the Sixth IFIP 11.3 WG on Database Security*, August 1992.

[Binns 92b] L. J. Binns. Inference and Cover Stories. In *Proceedings of the Sixth IFIP 11.3 WG on Database Security*, August 1992.

[Binns 94a] L. J. Binns. Inference: A Research Agenda. In L. Notargiacomo, editor, *Research Directions in Database Security V*, June 1994.

[Binns 94b] L. J. Binns. Implementation Considerations for Inference Detection: Intended vs. Actual Classification. In *IFIP Transactions A (Computer Science and Technology)*, A-47:297-306, 1994.

[Brewer 89] D. F. Brewer and M. J. Nash, The Chinese Wall Security Policy, In *Proceedings of the 1989 IEEE Symposium on Security and Privacy*, May 1989.

[Buczkowski 89a] L. J. Buczkowski, and E. L. Perry. *Database Inference Controller*. Interim Technical Report, Ford Aerospace, February 1989.

[Buczkowski 89b] L. J. Buczkowski, and E. L. Perry. *Database Inference Controller Draft Top-Level Design*. Ford Aerospace, July 1989.

[Buczkowski 90] L. J. Buczkowski. Database Inference Controller. In *Database Security, III: Status and Prospects*, ed. D. L. Spooner and C. Landwehr, pp. 311-322, North-

Holland, Amsterdam, 1990.

[Bums 92] R. K. Burns. A Conceptual Model for Multilevel Database Design. In *Proceedings of the 5th Rome Laboratory Database Security Workshop*, 1992.

[Codd 70] E. F. Codd. A Relational Model of Data for Large Shared Data Banks. In *Communications of the ACM*, June, 1970, pp. 377-387.

[Collins 91] M. Collins, W. Ford, J. O'Keefe, and M. B. Thuraisingham. The Inference Problem in Multilevel Secure Database Management Systems. In *Proceedings of the 3rd Rome Laboratory Database Security Workshop*, 1991.

[Cordingley 89] S. Cordingley and B. J. Sowerbutts. *The Fact Database - A Formal Model of the Inference Problem for Multi-Level Secure Data bases*. Plessey Report No: 72/89/R420/U, November 1989.

[Cordingley 90] S. Cordingley and B. J. Sowerbutts. *A Formal Model of Dynamic Inference for Multi-Level Secure Databases*. Plessey Report No: 72/90/R078/U, March 1990.

[Cox 86] L. S. Cox, S. McDonald, and D. Nelson. Confidentiality Issues at the United States Bureau of the Census. In *Journal of Official Statistics*, Vol. 2, No. 2, pp. 135-160, 1986.

[Cox 87] L. S. Cox. Practices of the Bureau of the Census with the Disclosure of Anonymized Microdata. In *Forum der Bundesstatistik*, pp. 26-42,1987.

[Cox 88] L. S. Cox. Modeling and Controlling User Interface. In *Database Security: Status and Prospects*, ed. Carl E. Landwehr, North-Holland, Amsterdam, pp. 167-171,1988.

[Cuppens 91] F. Cuppens. A modal logic framework to solve aggregation problems. In *Proceedings of the 5th IFIP 11.3 WG on Database Security*, September 1991.

[DAC 96] National Computer Security Center, Discretionary *Access Control Issues in High Assurance Secure Database Management Systems*, NCSC Technical Report-005, Volume 5/5, May 1996.

[Date 86] C. J Date. An Introduction to *Database Systems*, Vol. I, 4th Edition, AddisonWesley, Reading, MA, 1986.

[Delugach 92] H. S. Delugach and T. Hinke. Aerie: Database inference modeling and detection using conceptual graphs. In *Proceedings of the 7th Annual Workshop on Conceptual Graphs*. New Mexico State University, July 1992.

[Delugach 93] H. S. Delugach, T. Hinke, and A. Chandrasekhar. Applying conceptual graphs for inference detection using second path analysis. In *Proceedings of the ICCS1993, Intl. Conf. on Conceptual Structures*, pages 188-7, Laval University, Quebec City, Canada, August 4-71993.

[Denning 76] D. E. Denning. A Lattice Model of Secure Information Flow. Communications of the ACM, Vol. 19, No. 5, May 1976, pp. 236 - 243.

[Denning 78] D. E. Denning, P. J. Denning, and M. D. Schwartz. The Tracker: A Threat to Statistical Database Security. In ACM *Trans. Database Systems*. 4,1,7-18, March 1978.

[Denning 82] D. E. Denning. *Cryptography and Data Security*, Addison-Wesley, Reading, MA, 1982.

[Denning 85] D. E. Denning. Commutative Filters for Reducing Inference Threats in Multilevel Database Systems. In *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 134-146, April 1985.

[Denning 86a] D. E. Denning. A Preliminary Note on the Inference Problem in Multilevel Database Management Systems. In *Proceedings of the National Computer Security Center Invitational Workshop on Database Security*, June 1986.

[Denning 86b] D. E. Denning and M. Morgenstern. *Military Database Technology Study: Al Techniques for Security and Reliability*, SRI Technical Report, August 1986.

[Denning 87] D. E. Denning, S. G. Akl, M. Heckman, T. F. Lunt, M. Morgenstern, P. G. Neumann, and R. R. Schell. Views for Multilevel Database Security. In *IEEE Transactions on Software Engineering*, Vol. 13, No. 2, pp. 129-140, February 1987.

[DoD 85] Department of Defense *Trusted Computer System Evaluation Criteria*, DoD 5200.28-STD, Washington, DC, December 1985.

[Doncaster 90] S. Doncaster, M. Endsley, and G. Factor. Rehosting Existing Command and Control Systems into an MLS Environment. In *Proceedings of the 6th Annual Computer Security Applications Conference*, 1990.

[Elmasri 94] R. Elmasri, and S. B. Navathe. *Fundamentals of Database Systems, Second Edition*. The Benjamin/Cummings Publishing Company, Inc., Redwood City, CA, 1994.

[Entity 96] National Computer Security Center, *Entity and Referential Integrity Issues in Multilevel Secure Database Management Systems*, NCSC Technical Report-005, Volume 2/5, May 1996.

[5100.1-R 86] Department of Defense, Information Security Program Regulation 5100.1- R.

June 1986.

[Ford 90] W. R. Ford, J. O'Keefe, and M. B. Thuraisingham. *Database Inference Controller: An Overview*. Technical Report MTR 10963 Vol. 1. The MITRE Corporation. August 1990.

[Garvey 91a] T. D. Garvey, T. F. Lunt, and M. E. Stickel. Abductive and approximate reasoning models for characterizing inference channels. In *Proceedings of the Fourth Workshop on the Foundations of Computer Security*, June 1991.

[Garvey 91b] T. D. Garvey, T. F. Lunt, and M. E. Stickel. Characterizing and Reasoning about Inference Channels. In *Proceedings of the Fourth Rome Laboratory Database Security Workshop*, 1991.

[Garvey 91c] T. D. Garvey and T. F. Lunt. Cover Stories for Database Security. In *Proceedings of the 5th IFIP 11.3 WG on Database Security*, November 1991.

[Garvey 93] T. D. Garvey, T. F. Lunt, X. Qian, and M. E. Stickel. Toward a tool to detect and eliminate inference problems in the design of multilevel databases. In M. B. Thuraisingham and C. E. Landwehr, editors, *Database Security, VI: Status and Prospects*, pages 149-167. North-Holland, 1993.

[Garvey 94] T. D. Garvey, T. F. Lunt, X. Quin, and M. E. Stickel. *Inference Channel Detection and Elimination in Knowledge-Based Systems*. Final Report ECU 2528, SRI International, October 1994.

[Goguen 84] J. A. Goguen and J. Meseguer. Unwinding and Inference Control. In *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 75-86, April 1984.

[Haigh 90] J. T. Haigh, R. C. O'Brien, P. D. Stachour, and D. L. Toups. The LDV Approach to Security. In *Database Security, III: Status and Prospects*, ed. D. L. Spooner and C. Landwehr, North-Holland, Amsterdam, pp. 323-339, 1990.

[Haigh 91] J. T. Haigh, R. C. O'Brien, and D. J. Thomsen. The LDV secure relational DBMS model. In S. Jajodia and C. Landwehr, editors, *Database Security IV: Status and Prospects*, pages 265-279. North- Holland, 1991.

[Hale 94] J. Hale, J. Threet, and S. Shenoi. A Practical Formalism for Imprecise Inference Control. In *Proceedings for the 8th IFIP 11.3 WG on Database Security*, August 1994.

[Hinke 88] T. H. Hinke. Inference Aggregation Detection in Database Management Systems. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pp. 96-106, April 1988.

[Hinke 90] T. H. Hinke. Database Inference Engine Design Approach. In *Database Security, II: Status and Prospects, Proceedings for the IFIP 11.3 WG on Database Security*, Kingston, Ontario, Canada, October 1988, C. E. Landwehr, ed., North-Holland, 1990.

[Hinke 92] T. H. Hinke and H. Delugach. AERIE: Database inference modeling and detection for databases. In *Proceedings 6th IFIP 11.3 WG on Database Security*, August 1992.

[Hinke 93a] T. H. Hinke, H. Delugach, and A. Chandrasekhar. Layered Knowledge Chunks for Database Inference. In *Proceedings for the 7th IFIP 11.3 WG on Database Security*, Lake Guntersville State Park Lodge, Alabama, September 1993.

[Hinke 93b] T. H. Hinke and H. Delugach. AERIE: Database inference modeling and detection for databases. In M. B. Thuraisingham and C. E. Landwehr, editors, *Database Security, VI: Status and Prospects: Proceedings for the Sixth IFIP 11.3 WG on Database Security, Vancouver, Canada, August 1992*. Elsevier Science Publishers, 1993.

[Hinke 94] T. H. Hinke and H. Delugach. Inference Detection in Databases. In *Proceedings of the Sixth Rome Laboratory Database Security Workshop*, June 1994.

[Jajodia 92] S. Jajodia. Aggregation and Inference Problems in Multilevel Secure Systems. In *Proceedings of the 5th Rome Laboratory Database Security Workshop*, 1992.

[Keefe 89] T. F. Keefe, M. B. Thuraisingham, and W. T. Tsai. Secure Query Processing Strategies. In *IEEE Computer*. Vol. 22, #3, pages 63-70, March 1989.

[Lin 89a] T. Y. Lin. Commutative Security Algebra and Aggregation. In *Proceedings of the Second RADC Database Security Workshop.*, 1989.

[Lin 89b] T. Y. Lin. L. Kerschberg, and R. Trueblood. Security Algebra and Formal Models, In *Proceedings of the 3rd IFIP 11.3 WG on Database Security*, September 1989.

[Lin 90a] T. Y. Lin. Database, Aggregation, and Security Algebra. In *Proceedings of the 4th IFIP 11.3 WG on Database Security*, September 1990.

[Lin 90b] T. Y. Lin. Messages and Noncommutative Aggregation. In *Proceedings of the 3rd RADC Database Security Workshop*, 1990.

[Lin 91] T. Y. Lin. Multilevel Database and Aggregated Security Algebra, In *Database Security, IV: Status and Prospects*, ed. S. Jajodia, and C. E. Landwehr, North-Holland, Amsterdam, pp. 325-350.1991.

[Lin 92] T. Y. Lin. Inference Secure Multilevel Databases. In *Proceedings of the Sixth*

IFIP !1.3 WG on Database Security, August 1992.

[Lunt 88] T. F. Lunt, R. R. Schell, W. R. Shockley, M. Heckman, and D. Warren. Toward a multilevel relational data language. In *Proceedings of the Fourth Aerospace Computer Security Applications Conference*, December 1988.

[Lunt 89] T. F. Lunt, Aggregation and Inference: Facts and Fallacies. In *Proceedings of the IEEE Symposium on Security and Privacy*, May 1989.

[Lunt 94] T. F. Lunt. The Inference Problem: A Practical Solution, In *Proceedings of the 17th NCSC Conference*, pages 507-509. Baltimore, MD, October 1994.

[Marks 94a] D. Marks, L. Binns, and M. B. Thuraisingham. Hypersemantic Data Modeling for Inference Analysis. In *Proceedings for the 8th IFIP 11.3 WG on Database Security*, August 1994.

[Marks 94b] D. Marks. An Inference Paradigm. In *Proceedings of the 17th NCSC Conference*, pages 497-506. Baltimore, MD, October 1994.

[Meadows 88a] C. Meadows and S. Jajodia. Integrity Versus Security in Multi-Level Secure Databases. In *Database Security: Status and Prospects*, ed. Carl E. Landwehr, North-Holland, Amsterdam, pp. 89-101,1988.

[Meadows 88b] C. Meadows and S. Jajodia. Maintaining Correctness, Availability, and Unambiguity in Trusted Database Management Systems. In *Proceedings of the 4th Aerospace Computer Security Applications Conference*, pp. 106-110, December 1988.

[Meadows 91] C. Meadows, Aggregation Problems: A Position Paper, In Proceedings of the Third RADC Database Security Workshop, May 1991.

[Millen 91] J. Millen. Honesty is the Best Policy. In *Proceedings of the 3rd Rome Laboratory Database Security Workshop*, 1991.

[Morgenstern 87] M. Morgenstern. Security and Inference in Multilevel Database and Knowledge-Base Systems. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 357-373, May 1987.

[Morgenstern 88] M. Morgenstern. Controlling Logical Inference in Multilevel Database Systems. In *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 245-255, April 1988.

[Motro 94] A. Motro, D. G. Marks, and S. Jajodia. Aggregation in Relational Databases: Controlled Disclosure of Sensitive Information. In *Proceedings of 3rd European Symposium on Research in Computer Security (ESORICS)*, pp. 431-445, Brighton, UK, November 1994.

[National Academy Press 83] *Multilevel Data Management Security*, National Academy Press, Washington, DC (FOR OFFICIAL USE ONLY). 1983.

[Pernul 93] G. Pernul, W. Winiwarter, and A. M. Tjoa, The Deductive Filter Approach to MLS Database Prototyping, *Proceedings of the Tenth Annual Computer Security Applications Conference*, December 1993.

[Poly 96] National Computer Security Center, *Polyinstantiation Issues in Multilevel Secure Database Management Systems*, NCSC Technical Report-005, Volume 3/5, May 1996.

[Qian 92] X. Qian. Integrity, secrecy, and inference channels. In *Proceedings of the Fifth Rome Laboratory Database Security Workshop*, October 1992.

[Qian 93a] X. Qian. A model-theoretic semantics of the multilevel secure relational model. Technical Report SRI-CSL-93-06, Computer Science Laboratory, SRI International, November 1993.

[Qian 93b] X. Qian, M. E. Stickel, P. D. Karp, T. F. Lunt, and T. D. Garvey. Detection and elimination of inference channels in multilevel relational database systems. In *Proceedings of the 1993 IEEE Symposium on Security and Privacy*, pages 6-205, Oakland, California, May 1993.

[Qian 94] X. Qian. Inference channel-free integrity constraints in multilevel relational databases. In *Proceedings of the 1994 IEEE Symposium on Security and Privacy*, pages 158-167, Oakland, California, May 1994.

[Schum 87] D. A. Schum, *Evidence and Inference for the Intelligence Analyst*, University Press of America, Lanham, MD, 1987.

[Smith 90a] G. Smith, A Taxonomy of Security-Relevant Knowledge, In *Proceedings of the 13th National Computer Security Conference*, Washington, D.C., October 1990.

[Smith 90b] G. Smith. Modeling Security-Relevant Data Semantics. In *Proceedings of the 1990 IEEE Symposium on Security and Privacy*, Oakland, California, May 1990.

[Smith 91] G. Smith. Thoughts on the Aggregation Problem. In *Proceedings of the 3rd RADC Database Security Workshop*, May 1991.

[Sowa 84] J. Sowa. *Conceptual Structures: Information Processing in Minds and Machines*, Reading, MA, Addison Wesley, 1984.

[Sowerbutts 90] B. Sowerbutts and S. Cordingley. Data Base Architectonics and Inferential Security. In *Proceedings 4th IFIP 11.3 WG on Database Security*, 1990.

[Stachour 90] P. Stachour, and B. Thuraisingham. Design of LDV: A Multilevel Secure

Relational Database Management System. In *IEEE Transactions on Knowledge and Data Engineering*, Vol. 2., No. 2, pp. 190-209, June 1990.

[Stickel 94] M. E. Stickel. Elimination of Inference Channels by Optimal Upgrading. In Proceedings of the 1994 *IEEE Symposium on Security and Privacy*, pages 168-174, Oakland, California, May 1994.

[Su 86] T. Su. Inferences in Database. Ph.D. Dissertation, Department of Computer Engineering and Science, Case Western Reserve University, August 1986.

[Su 87] T. Su and G. Ozsoyogiu. Data Dependencies and Inference Control in Multilevel Relational Database Systems. In *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 202-211, April 1987.

[Su 90] T. Su and G. Ozsoyoglu. Multivalued Dependency Inferences in Multilevel Relational Database Systems. In *Database Security III: Status and Prospects*, eds. D. L. Spooner and C. Landwehr, pp. 293-300, NorthHolland, Amsterdam, 1990.

[TDI 91] *Trusted Database Management System Interpretation of the Trusted Computer System Evaluation Criteria*, NCSC-TG-021, National Computer Security Center, April 1991.

[TNI 87] *Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria*, NCSC-TG-005, National Computer Security Center, July 1987.

[Thuraisingham 87] M. B. Thuraisingham. Security Checking in Relational Database Management Systems Augmented with Inference Engines. In *Computers & Security*, Vol. 6, pp. 479-492,1987.

[Thuraisingham 88] M. B. Thuraisingham, W. Tsai, and T. Keefe. Secure Query Processing Using Al Techniques. In *Proceedings of the Hawaii International Conference on Systems Sciences*. January 1988.

[Thuraisingham 89a] M. B. Thuraisingham. Security checking with Prolog Extensions. In *Proceedings of the 2nd RADC Database Security Workshop*, Franconia, NH, May 1989.

[Thuraisingham 89b] M. B. Thuraisingham. Secure Query Processing in Intelligent Database Management Systems. In *Proceedings of the 5th Computer Security Applications Conference*. Tucson, AZ, December 1989.

[Thuraisingham 90a] M. B. Thuraisingham. Towards the Design of a Secure Data/Knowledge Base Management System. In *Data and Knowledge Engineering Journal.* Vol. 5, #1, March 1990.

[Thuraisingham 90b] M. B. Thuraisingham. The Inference Problem in Database Security. In *IEEE CIPHER*. 1990.

[Thuraisingham 90c] M. B. Thuraisingham. Recursion Theoretic Properties of the Inference Problem in Database Security. MTP-291, The MITRE Corporation, Bedford, MA, May 1990.

[Thuraisingham 90d] M. B. Thuraisingham. Novel Approaches to Handling the Inference Problem in Database Security. In *Proceedings of the 3rd RADC Database Security Workshop*, Castile, NY, June 1990.

[Thuraisingham 90e] M. B. Thuraisingham. A *Nonmonotonic Typed Multilevel Logic for Multilevel Data/Knowledge Base Management Systems*. Technical Report MTR 10935, The MITRE Corporation, June 1990.

[Thuraisingham 90f] M. B. Thuraisingham. *The use of conceptual structures for handling the inference problem*. Technical Report M90-55, The MITRE Corporation, Bedford, MA, August 1990.

[Thuraisingham 91] M. B. Thuraisingham. The use of conceptual structures for handling the inference problem. In *Proceedings 5th IFIP 11.3 WG on Database Security*, 1991.

[Thuraisingham 92] M. B. Thuraisingham. Knowledge-Based Inference Control in a Multilevel Secure Database Management System. In *Proceedings of the 15th National Computer Security Conference*, October 1992.

[Thuraisingham 94] M. B. Thuraisingham, M. Collins, and H. Rubinovitz. Database Inference Control. In *Proceedings of the 6th Rome Laboratory Database Security Workshop*, 1994.

[Ullman 88] J. D. Ullman. *Database and Knowledge-Base Systems Volume I*, Rockville, MD, 1988.

[Urban 90] S. Urban and L. Delcambre. Constraint Analysis for Specifying Perspectives of Class Objects. In *Proceedings of the 5th IEEE International Conference on Data Engineering*, Los Angeles, CA, December 1990.

[Wilson 88] J. Wilson. Views as the Security Objects in a Multilevel Secure Database Management System. In *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA. April 1988.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	May 1996	Final

4. TITLE AND SUBTITLE
Inference and Aggregation Issues in Secure Database Management Systems

5. FUNDING NUMBERS

6. AUTHOR(S)

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
National Security Agency
Attn: V21, Partnerships and Processes
Fort George G. Meade, MD 20755-6000

8. PERFORMING ORGANIZATION REPORT NUMBER
NCSC Technical Report - 005
Volume 1/5
Library No. S-243,039

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for Public Release
Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This report is the first of five companion documents to the *Trusted Database Management System Interpretation of the Trusted Computer System Evaluation Criteria*. The companion documents address topics that are important to the design and development of secure database management systems, and are written for database vendors, system designers, evaluators, and researchers. This report addresses inference and aggregation issues in secure database management systems.

14. SUBJECT TERMS
Inference; Aggregation; Secure Database Management Systems

15. NUMBER OF PAGES
79

16. PRICE CODE

17. SECURITY CLASSIFCATION OF REPORT	18. SECURITY CLASSIIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	