Database Security

Course 2

Database Security Foundations (part 1)

Mandatory aspects of the security in database management systems

Plan

1. Introduction
2. Brief History of the Field (Previous developments)
    - 2.1 First initiatives
    - 2.2 Major research and developments
    - 2.3 Interpretation of the systems' security evaluation criteria for the secure databases
    - 2.4 Types of multi-level secure database systems
    - 2.5 Important problems
    - 2.6 Subsequent technologies
3. Design principles
    - 3.1 Mandatory access control
    - 3.2 Security architectures

1. Introduction

Security Policies (1)

- Before the design of a secure system, the first choice: the security policy which will be enforced by the system.
  - Policy = a set of **rules** which ensure the system's security.
- Security policies can be classified in the following two types:
  - **Mandatory** policies
  - **Discretionary** policies.

Security Policies (2)

- Mandatory policies – **mandatory** by their nature, independent of the application.
- Discretionary policies – specified by an administrator or by a user who is responsible for the environment in which the system will operate.
  - The most known discretionary security policy refers to the access control.

Course goal

- In this course, we will present the mandatory approaches of the security in the database management systems.
- The course contains:
  - An historic perspective of these approaches, especially of a specific type of mandatory security, the multi-level security (**MLS/DBMS –** *Multi Level Secure Database Management Systems*)
  - The design of this type of systems, from which it results security architectures for the DBMSs.
- **The goal of the course** is to present the **fundamentals** of the different types of MLS/DBMSs, including the ones of the relational, distributed and object-oriented database systems.

2. Brief History of the Field (Previous developments)

Brief History (1)

- A lot of the contributions in the ’80s and ’90s in the field of the database security were oriented towards **MLS/DBMS**.
- These systems are also named ***Trusted*** *Database Management Systems* (**TDBMS**).
- They are based on the definition of the security levels.

Brief History (2)

- In an MLS/DBMS, the users are classified on different security levels, such as *Unclassified*, *Confidential*, *Secret* and *TopSecret*.
- In their turn, data are assigned "sensitivity" levels, named like the levels applied to the users.
- Generally, it is assumed that these security levels form a **partially ordered lattice**.
  - For example, *Unclassified* < *Confidential* < *Secret* < *TopSecret*.
  - The partial order is because we may have different compartments, and these can be not comparable.

Brief History (3)

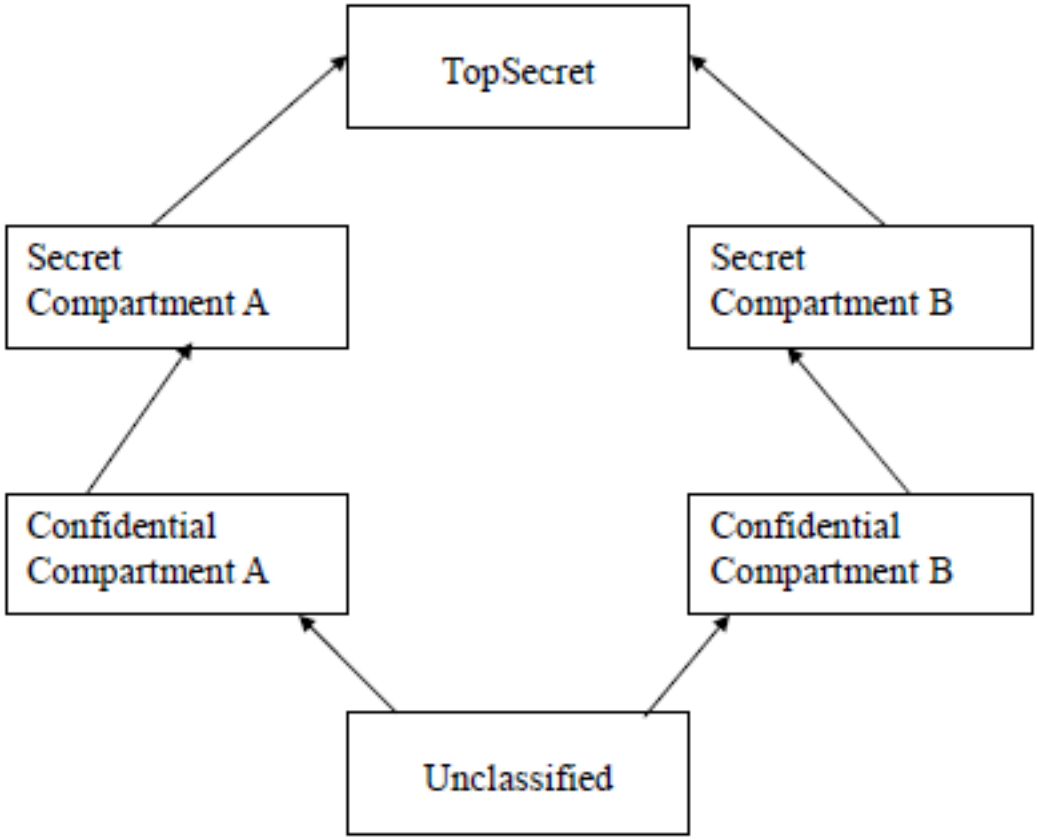


Figure 1. Security Levels

Brief History (4)

- MLS/DBMSs evolved from:
  - some of the multi level secured operating systems developments (ex: MULTICS, SCOMP) and
  - the database field developments.
- Many of the concerns focused on the relational model.
- At the end of the ’80s *National Computer Security Center* (NCSC)[1] proposed the **interpretation** of the secure systems evaluation criteria for the database systems.
  - This interpretation was called *Trusted Database Interpretation*.

[1] https://www.pcmag.com/encyclopedia/term/ncsc

Brief History (5)

- In the ’90s the research also considered the non-relational systems, including the MLS **object** databases.
- Also, efforts were focused on multi-level secured **distributed** databases.
- Challenges that appeared and were investigated included:
  - Multi-level data models
  - The inference problem
  - The secure transaction processing.
- The research was being done at the same time as the arrival of commercial products.
- Subsequently, the efforts were directed towards the investigation of the multi-level security for the new data management technologies.

Brief History (6)

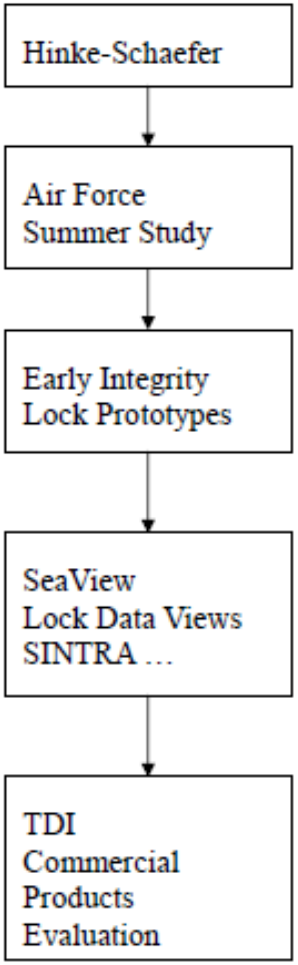


Figure 2. Moments of the MLS/DBMS's evolution

First initiatives

- An important moment in the development of the MLS/DBMSs was the ***Air Force Summer Study***.
- This event was, in turn, influenced by previous research and results, the most notable of these being the **Hinke-Schaefer approach** on providing the *mandatory* security by the **operating systems**.
- This approach developed a way to host the MLS/DBMSs on the operating system MULTICS MLS.
- The system was based on the relational one and the idea was:
  - the partitioning of the relation based on attributes and
  - their storage in different files and at different levels.
  - Next, the operating system will control the access to these files.

Air Force Summer Study (1)

- The goal of this study was to:
  - Analyze the problems in the field and
  - Propose viable approaches to the MLS/DBMS's design.
- The group of study brought together experts in the field and was divided in three.

Air Force Summer Study (2)

- The first group focused on the **short-term** approaches for the MLS/DBMS design. Among these, there are those referring to:
  - The integrity lock
  - The distributed architecture.
- The second group considered the military messages system's model for an MLS/DBMS.
  - The model was developed by that time
  - The goal of the group: to investigate its applicability for MLS/DBMS.
- The third group studied the long-term approaches and investigated problems :
  - Data classification based on content and context
  - Multi-level relational data models
  - The inference problem.

Air Force Summer Study (3)

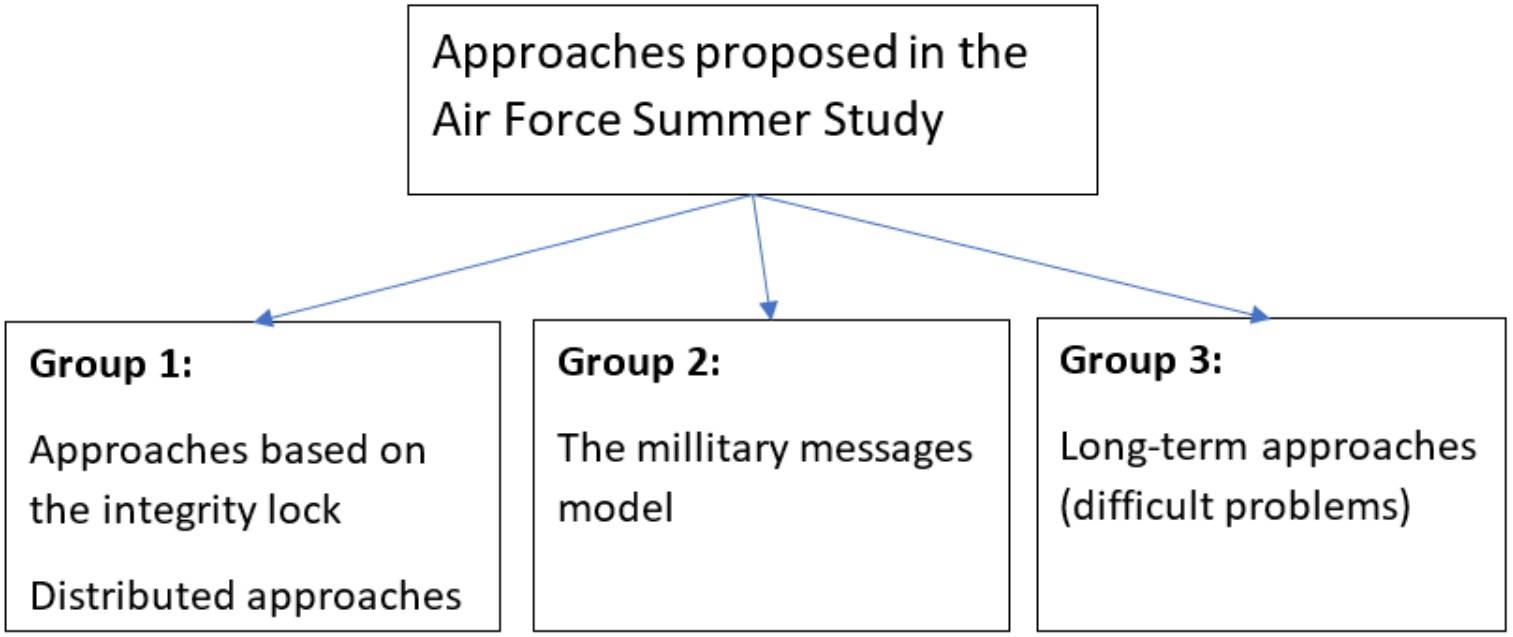


Figure 3. Approaches proposed by the Air Force Summer Study

Major research and developments (1)

- Many contributions on the design and development were reported subsequent to *Air Force Summer Study*.
- Initial developments were based on the integrity lock approach, which was developed by MITRE Corporation.
  - Two prototypes were designed and developed, one of which for the Ingres system.
- Two important systems were *SeaView System* and *Lock Data Views System*. These two initiatives were taken by *Rome Air Development Center* (RADC – now Air Force Rome Laboratory) and aimed the long-term approaches proposed by the Summer Study.
  - Both initiatives influenced the further commercial developments a lot.

Major research and developments (2)

- Other three important systems were:
  - The SINTRA system developed by the *Naval Research Laboratory*
  - The SWORD system developed by the *Defense Research Agency* (UK)
  - SDDBMS developed by Unisys.
- The SINTRA system based on the distributed architecture proposed by the *Air Force Summer Study*
- The SWORD system proposed alternatives to the *SeaView* and *Lock Data Views* data models
- SDDBMS investigated the approaches that used partitioning and replication for the design of MLS/DBMS.

The interpretation of the evaluation criteria for the secure databases (1)

- Following the progress made over the ’80s, *National Computer Security Center* (NCSC) anticipated that the commercial products (implementations) were going to appear.
- At that time, there were already commercial operating systems evaluated by the specific criteria stated for the secure systems.
- These criteria consist of rules that a system must satisfy in order to be certified at a certain level. The highest level was A, while the lowest was C2. There were several intermediate levels between them (C1, B1, B2, B3)[1].

[1] https://www.pcmag.com/encyclopedia/term/ncsc-security-levels

The interpretation of the evaluation criteria for the secure databases (2)

- In order to evaluate MLS/DBMSs, these criteria were no longer sufficient, so that their interpretation (adjustment) for the database systems was needed.
- This "adjustment" was named *Trusted Database Interpretation* (TDI).
- TDI focused on the TCB (*Trusted Computing Base*) approach. This is that part of the system which imposes the security policies.
- TDI was published in **1991**, moment at which there were already commercial products (*Oracle*, *Sybase*, *Informix*, *Ingres*) ready to be evaluated.

Types of multi-level secure database systems

- Many of the designed and developed MLS/DBMSs were based on the relational model.
- There were also concerns directed towards other models, like object and functional models.
- MLS/DBMS types:
  - MLS Relational Databases
  - MLS Object Databases
  - MLS Distributed Databases
  - MLS Parallel Databases
  - MLS Real-Time Databases
  - MLS Functional Databases
  - MLS Logic Databases

Relational database systems

- The first efforts regarding MLS were directed towards the relational databases.
- These systems were aimed at both:
  - designing the multi-level relational databases models and
  - providing access based on views.
- The notion of polyinstantiation was also stated – this allows users to view an element differently, based on their security level.
- Other research subjects that were investigated for the MLS/DBMSs based on the relational model included:
  - The secure transaction processing
  - The inference problem
  - The security constraints processing.

Entity-relationship systems

- In 1987, A*ir Force Research Laboratory in Rome* took the initiative to design an **MLS/DBMS** based on the entity-relationship model:
  - Developed initially by Peter Chen in 1976
  - Later used for modeling applications.
- This initiative aimed to explore:
  - The security properties for the E/R model
  - The use of the secure E/R models for the design of the DBMSs.
- The result: **the MLS E/R model**, which was later used for the secure application's modeling.
- Variants of this were used to explore the inference problem.
- This approach contributed to the design of the MLS applications, but there were no initiatives to design MLS/DBMSs based on the E/R model.

Object-oriented database systems

- The design of the MLS/DBMSs of this type can be based on different models.
- Generally, it is based on the design proposed for the MLS/DBMSs that are based on the relational model. Furthermore, it must secure the complex objects and manage the method's secure execution.
- While research on object-oriented MLS/DBMS design evolved, the use of object models was analyzed for secure applications design.
  - For example, following UML's development, there were initiatives to design secure applications with the help of this language.
- The object models were also used for the representation of the secure multimedia databases.

Distributed and heterogenous database systems

- The *Air Force Summer Study* initiative discussed about the design of MLS/DBMS based on distributed architectures, but partitioning data according to their security level.
- The initial goal was to develop secure centralized database systems.
- Subsequently, it was worked on the design and development of the distributed MLS/DBMS (MLS/DDBMS).
- Then, the focus went on the design and development of the heterogeneous distributed database systems.

Deductive database systems

- Deductive databases are based on logic.
- When the inference problem was investigated, multi-level secure deductive database systems were also designed.
- These systems base on the NTML logic (Nonmonotonic Typed Multilevel Logic)[1].
  - This type of logic has the security levels reasoning ability, which are non-monotonic, by their nature.
  - This logic incorporates constructions for the reasoning over the applications at different security levels.

[1] https://patents.google.com/patent/US5481700

Functional database systems

- Functional data models are based on functions, and the query evaluation leads to the execution of functions.
- MLS models were studied for this type of systems, the notion of multi-level function was introduced and the problem of the execution of these functions was also studied.

Real-time database systems

- These systems were investigated for many applications, including the command and control applications, as well as process control applications.
- The challenge is to embed security in the real-time processing. But, this two components are in **conflict**.
  - For example, if **all** access control checks need to be done, transactions may not be completed on time.
  - There is also potential for the emergence of covert channels when i**ntegrating security with real time processing**.
- Secure algorithms for real-time concurrency control were investigated, while issues related to security integration, real-time processing, and fault tolerance were analyzed.
- The problems concerning these systems become **critical** for many applications, including embedded systems.

Important problems (1)

- Previously, we made a short presentation of the different types of MLS/DBMSs.
- Some important and difficult issues in MLS / DBMS are:
  - Inference
  - Secure transactions processing
  - Development of a multi-level secure relational data model.
- The most notable of these problems is that of the inference. This consists in the process of formulating requests and deducing sensitive information from the legitimate answers received.
- Different approaches to solving this problem have been presented in the literature.
- It has been shown that the general problem of inference is unsolvable.
- The use of security constraints and conceptual structures to treat different types of inference has been explored.
- A particular case of inference is the aggregation problem.

Important problems (2)

1. Secure transaction processing

- Many actions have been directed towards reducing covert channels when processing transactions in MLS / DBMS.
- As mentioned earlier, the challenge is to design real-time MLS/DBMSs.
- This means not only that transactions must be secure, but also they must respect time constraints.

Important problems (3)

2. Development of a multi-level secure relational data model

- More proposals
- Different users - different views on the same item.
- If we use multiple values to represent the same entity, then we do not respect the databases integrity.
- If we do not impose polyinstantiation, there is a potential for signaling channels. The problem is still open.

Subsequent technologies (1)

- As new technologies have emerged, specific issues related to multi-level security can be examined.
- Data warehousing, e-commerce systems, multimedia systems, digital libraries, web applications etc.
- However, there was a limited number of initiatives to investigate multi-level security for new data management systems.

Subsequent technologies (2)

- This is partly because even in the case of the relational systems there are difficult problems to solve regarding multi-level security.
- As systems become more complex, the development of secure multi-level systems is becoming an increasing challenge.
  - How could secure multi-level systems be developed for different types of applications (digital libraries, e-commerce systems)?
  - How can an acceptable performance be achieved?
  - How can huge systems be verified?

Subsequent technologies (3)

- **What happened to pure MLS?**
  - its complexity, performance overhead, and the difficulty of problems like inference meant that "pure" MLS/TDBMS systems remained a niche, primarily for high-assurance government/military use.
- **How does this live on?**
  - the *concepts* are alive in modern databases, just implemented differently
  - **Oracle Label Security (OLS):** This is a direct commercial implementation of these MAC principles, built right into the Oracle kernel.
  - **Row-Level Security (RLS):** Features like RLS in SQL Server, PostgreSQL etc., are a *practical, flexible way* to achieve a similar outcome (controlling access to specific rows based on a user's context), even if it's not "pure" MAC.

3. Design Principles

Design Principles

- Next, we will describe the **design principles for MLS/DBMS**, presenting a taxonomy for the realization of different MLS/DBMS architectures.
- The type of access control used by MLS/DBMS is the *mandatory* access control.
- We will present this type of access control, we will introduce the Bell-La Padula security policies and their interpretation for MLS/DBMSs.
- Later on, we will present different MLS/DBMS security architectures.

Mandatory access control (1)

- Although DBMSs need to address most of the security issues that operating systems address (identification and authentication, access control, audit), there are specific features that generate additional issues.
- For example, objects in the DBMS tend to be of variable size and may have fine granularity (relationships, attributes, and records).
- This contrasts with operating systems, where granularity tends to be coarse (files, segments).
- Due to this fine granularity of MLS/DBMS (TDBMS), the objects on which the Mandatory Access Control (MAC) and the Discretionary Access Control (DAC) access is performed may differ.
- In Trusted Operating Systems (MLS), MAC and DAC are usually performed on the same object (for example, a file).

Mandatory access control (2)

- There are also some functional differences between operating systems and DBMSs.
  - Operating systems work with subjects who want to access / modify objects.
  - DBMSs are used to share data between users and provide users with means to link different objects.
  - DBMSs are dependent on operating systems, which provide them with resources (for example, communication between processes and memory management). Therefore, the design of secure DBMSs must consider the way in which operating systems handle security.

Mandatory access control (3)

- The differences discussed above lead to the fact that the traditional methods used in the design of the systems must be adapted for DBMSs.
- There is no standard architectural approach for MLS/DBMS.
- Different methods have been proposed for the design and construction of MLS / DBMS.
- Some of them will be presented in the following.
- In essence, the MLS/DBMSs were designed based on one of the architectures that will be presented.
- Figure 4 illustrates the difference between access control in operating systems and access control in DBMSs.

Mandatory access control (4)

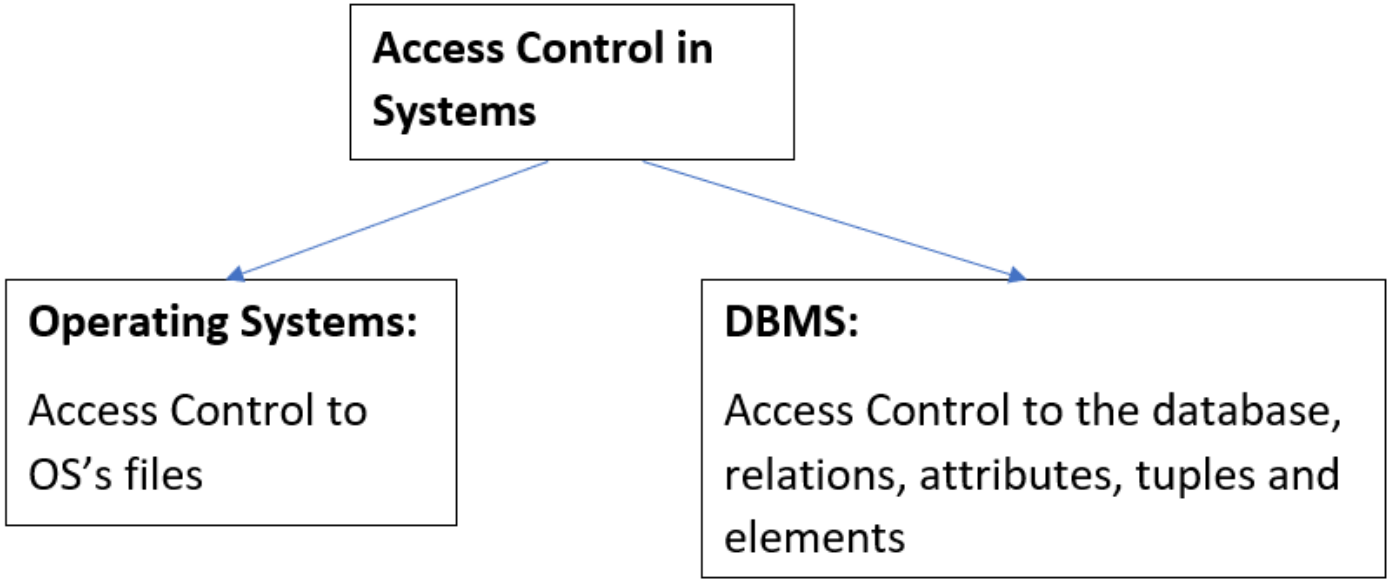


Figure 4. Access control in operating systems and in DBMSs

Mandatory access control policies (1)

- MAC policies refer to subjects' access to objects.
- Many of the commercial DBMSs are based on the Bell and La Padula policies specified for operating system design.
- We will present these policies, and then how they have been adapted for DBMS.
- Other mandatory policies include those referring to non-interference (Goguen, Messeguer).

Mandatory access control policies (2)

- In Bell - La Padula policies, subjects are associated with certain levels and can operate up to their level. Objects are also assigned levels of "sensitivity".
- The two rules of B-LP policy are:

- **Simple security property:** A subject has read access on an object if its security level dominates the level corresponding to the object.
- **Property *:** A subject has write access on an object if the security level of the subject is dominated by that of the object.

Mandatory access control policies (3)

- These properties also apply to DBMSs.
- In their case, the property * is modified as follows:

A subject has write access on an object if the subject's level is the same as the object’s level. Therefore, a subject can modify relations at his level.

Mandatory access control policies (4)

- An important aspect of security policies for database systems is the polyinstantiation.
  - This assumes that the same object can have different interpretations and values at different levels.
  - For example, an employee's salary can have the value 30,000 at the Unclassified level and 70,000 at the Secret level.
  - In multi-level relational models, we can have both entries but accompanied by security levels, as an additional attribute.
- One of the motivations for using polyinstantiation is to avoid covert (hidden) channels.
  - For example, if we have an entry at the Secret level that provides the value 70,000 for X's salary, and an Unclassified subject wants to enter the value 30,000 for the same salary and the update is not allowed, it could indicate a channel from a higher level to a lower one. Over time, this could become a hidden channel.

Mandatory access control policies (5)

- There have been many discussions and debates about polyinstantiation
- No consensus was reached
- Different systems have implemented multi-level relational data models in different ways.

Mandatory access control policies (6)

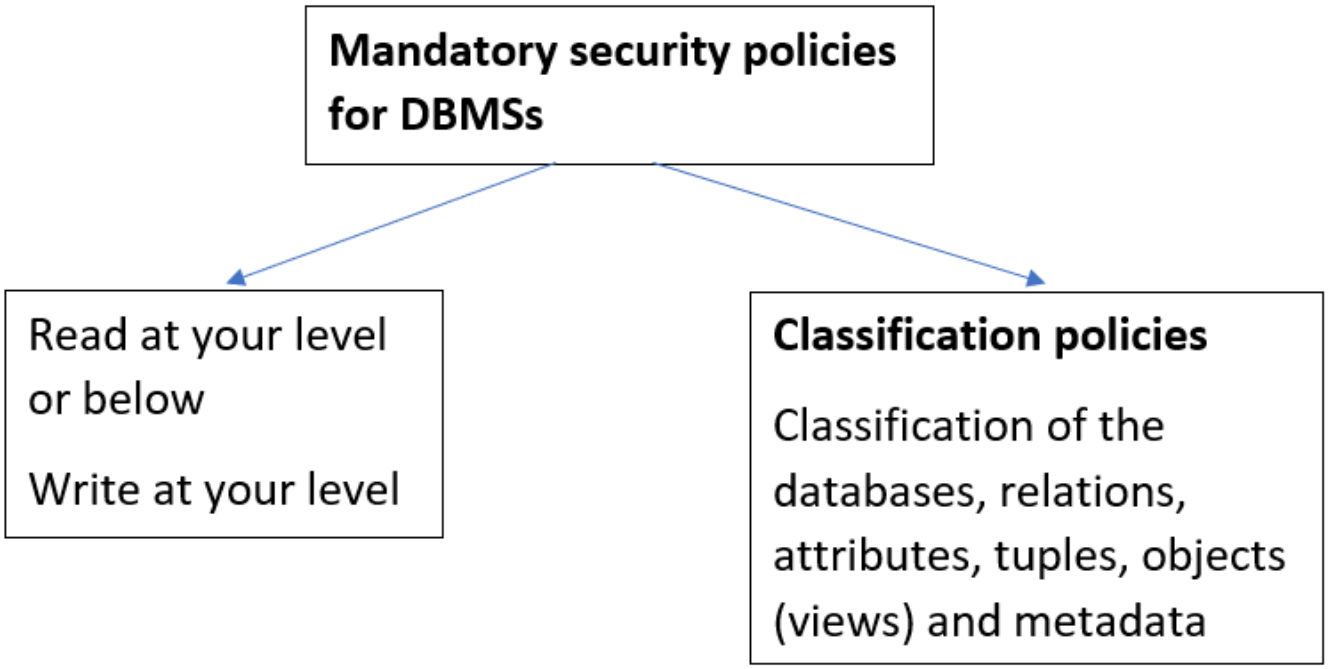


Figure 5. Mandatory policies for DBMSs

Security Architectures

- Security architectures are system architectures, which were designed to address certain aspects of security.
- We will present 5 architectures for MLS/DBMSs. As mentioned earlier, they provide a taxonomy for MLS / DBMSs.
- These architectures are:
  - Integrity lock
  - Enable of mandatory security by the operating system
  - Kernel extensions
  - Secure subject
  - Distributed architecture
    - Partitioned approach
    - Replicated approach.

Integrity Lock Architecture(1)

- This approach uses an insecure DBMS (back end) with access to data from the database, an insecure interface (front-end) that communicates with the user, and a secure front-end application that uses checksums (Figure 6).
- The unsafe components are isolated from each other so that there is no communication between the two without the mediation of the secure filter.
- The DBMS is maintained at the system level (the highest level supported by the system).
- Multiple front-end instances are maintained, one instance per each user level.
- The secure filter is also maintained at the system level.

Integrity Lock Architecture (2)

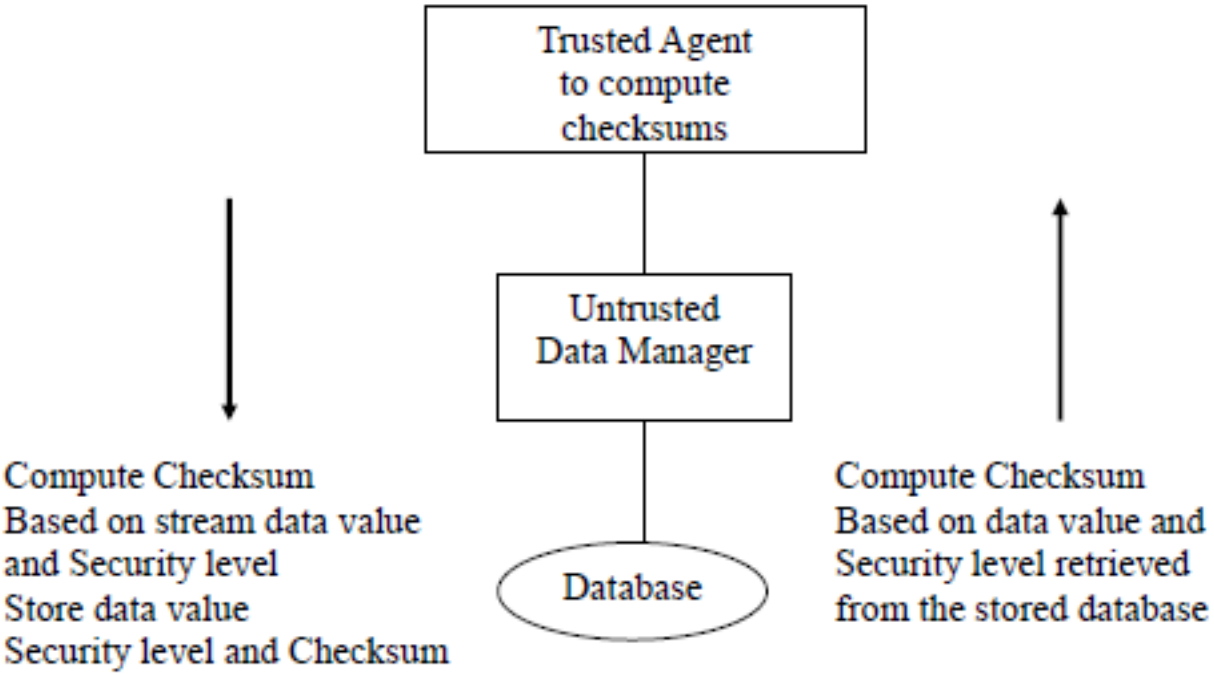


Figure 6. The Integrity lock architecture

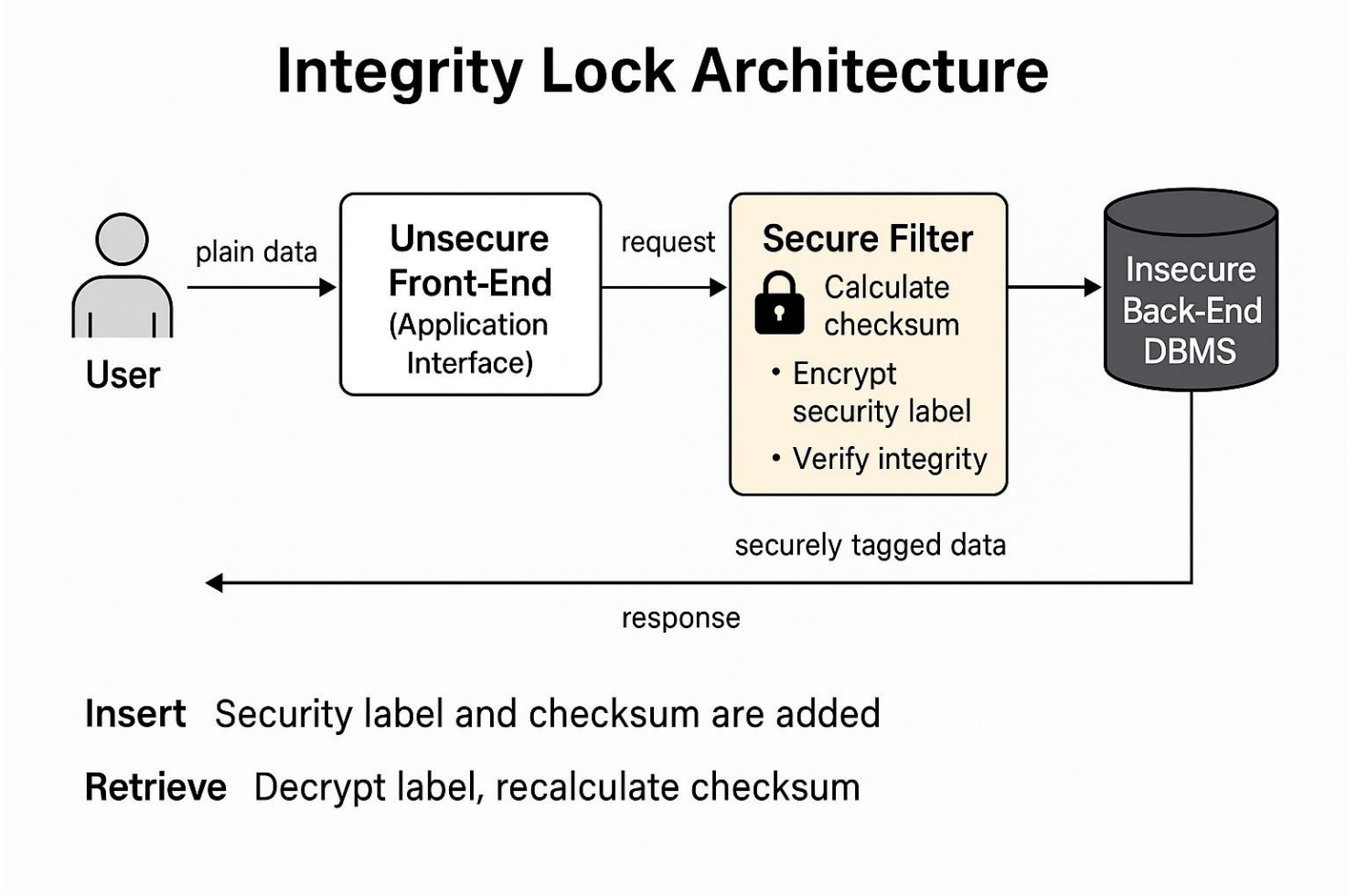
Integrity Lock Architecture (3)

- In this architecture, each tuple that is inserted in the database is associated with a security label and a checksum.
- The security label is encrypted and the data is unencrypted.
- The sums are calculated by the secure filter at insertion and recalculated at retrieval.
- Upon insertion, the secure filter calculates the sum and the insecure DBMS takes the data (for example, the tuples), the associated label, and the sum and stores them in the database.
- In the retrieval operation, the back end finds the data tuples and sends them to the secure filter that recalculates the sums based on the tuple and the found label.
- If the filter determines that the data has not been affected (modified), it transmits it to the user via the insecure front end.

Integrity Lock Architecture (4)

- The advantage of this architecture is that a small amount of additional secure code is required by the MLS/DBMS, and the performance is independent of the number of security levels involved.
- The disadvantage is that this approach is subject to interference threats.
- This threat occurs because the insecure back end can access secret data, encrypt it as a series of accessible data tuples, and transmit the encrypted tuples to the secure front end.
- Because the data tuples are accessible, the secure filter will not be able to detect hidden operations on the insecure DBMS.

Integrity Lock Architecture (5)



Access control through the operating system (1)

- This approach (Figure 7), also known as Hinke-Schaefer, uses the secure operating system to mediate access control.
- No mediation of access control is performed by the DBMS.
- Database objects (for example, tuples) are treated similarly to operating system objects (files).
- Thus, the tuples on the Secret level are stored in Secret files, and the Top Secret ones are stored in Top Secret files.
- With this approach, there is no DBMS that allows access to data from the database.
- There is an instance of the DBMS for each security level.
- The advantage of this architecture is that it is simple and secure.
- The downside is that performance issues increase with the number of security levels.
- This approach is also called the single-kernel approach.

Access control through the operating system (2)

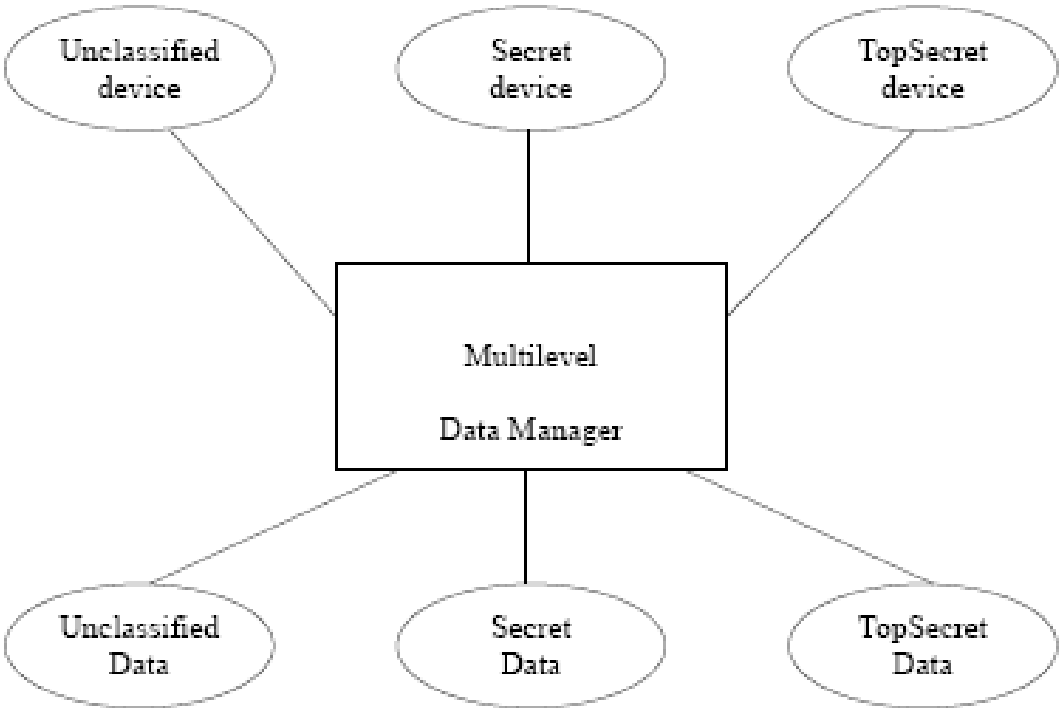
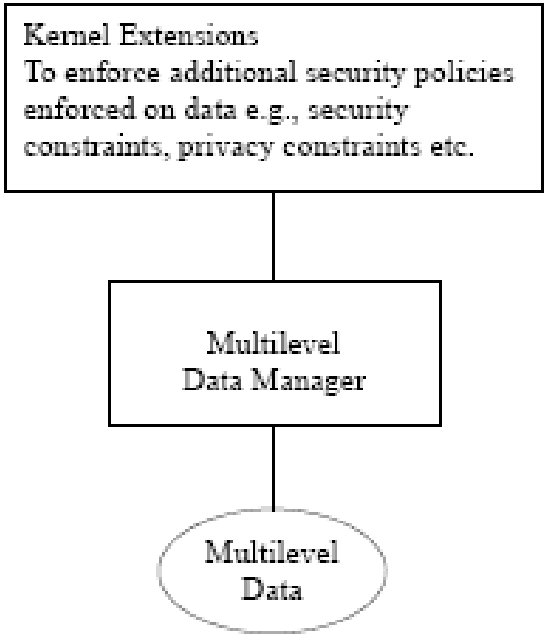


Figure 7. Mandatory access control through the operating system

Kernel Extensions Architecture (1)

- This architecture (figure 8) is an extension of the previous one.
- The operating system is used to provide basic MAC and DAC mediation.
- MLS/DBMS will supplement this access mediation by providing access control mediation.
- For example, MLS/DBMS can provide context-dependent DAC on views.
- This approach differs from that of the secure subject because the policies enforced by MLS/DBMS do not depend on those of the operating system.
- This approach has the same performance issues as the single-kernel approach
- However, because it provides more sophisticated access control mechanisms, it can meet certain real needs for access control.

Kernel Extensions Architecture (2)



Figur8 8. Kernel extensions Architecture

Secure subject Architecture (1)

- This architecture, also called dual kernel-based architecture, does not rely on the operating system to perform access control mediation.
- Thus, access to database records is mediated by the secure DBMS.
- The name of the architecture comes from the fact that the DBMS is usually a secure subject (or process) hosted on the operating system. Essentially, the DBMS can access the data in the database.
- The advantage of this architecture is that it can provide good security, and the performance is independent of the number of security levels involved.
- The disadvantage is that the code of the DBMS that mediates access must be secure. This assumes that the approach may require a large amount of secure code.

Secure subject Architecture (2)

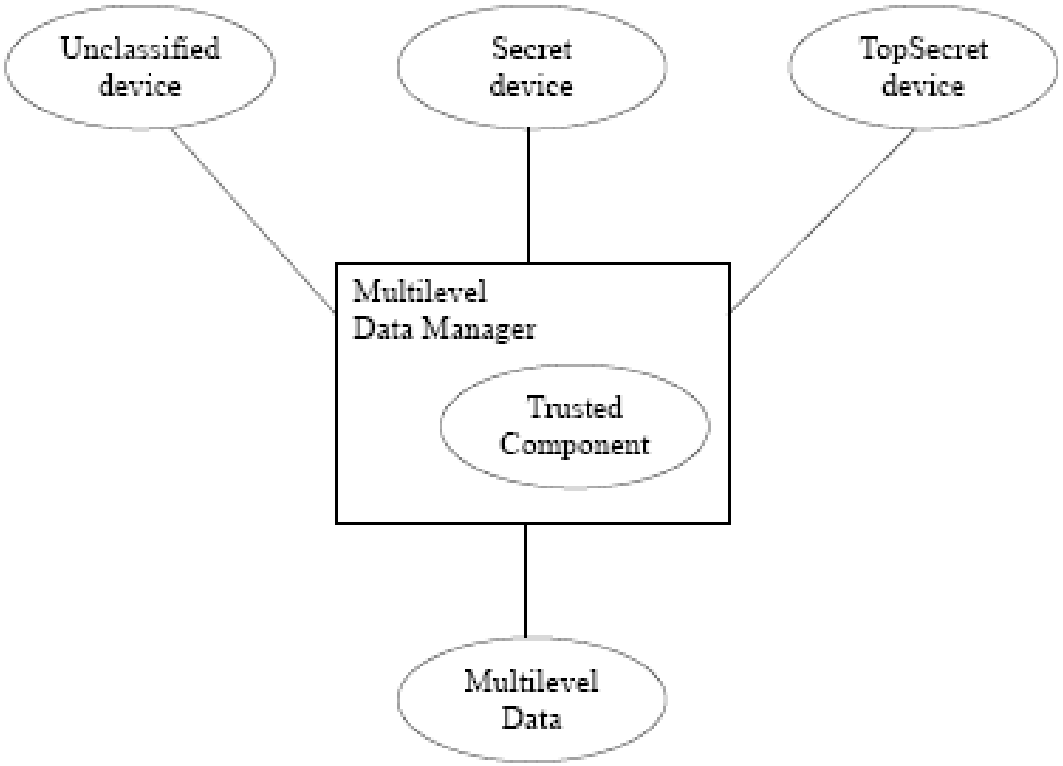


Figure 9. Secure subject Architecture

The Distributed Architecture (1)

- Within this architecture, there are multiple insecure back-end DBMSs and a single secure front-end DBMS.
- Communications between back-end DBMSs take place through the front-end.
- There are 2 main approaches to this architecture.
- In the first of these, each back-end DBMS has data at a certain level and operates at that level (Figure 10).
- This assumes that the back-end DBMS at the Secret level will handle Secret data, and the one at the Top Secret level will handle Top Secret data. This approach is called partitioned.

The Distributed Architecture (2)

- In the second approach (Figure 11), data from lower levels are replicated at higher levels.
- Thus, the Secret DBMS will handle Secret, Confidential and Unclassified data. This approach is called replicated.
- In the partitioned approach, the secure front end is responsible for ensuring that the request is directed to the correct back end, but also for performing the join operations on the data sent from the back-end DBMSs.

The Distributed Architecture (3)

- Because the query may contain information classified at a higher level than the back-end DBMS (for example, the values in the WHERE clause of the query), the problem of high signaling channels may arise in the partitioned approach.
- This occurs because queries can be sent to DBMSs that operate at lower levels than the user's level.
- In the replicated approach, the secure front end ensures that the query is directed to a single DBMS.
- Because only DBMSs that operate at the same level as the user are queried, this approach does not have the above problem.
- Furthermore, no front-end DBMSs are required to perform join operations. Because the data is replicated, the secure front end must ensure the consistency of the data maintained by the various DBMSs.

The Distributed Architecture (4)

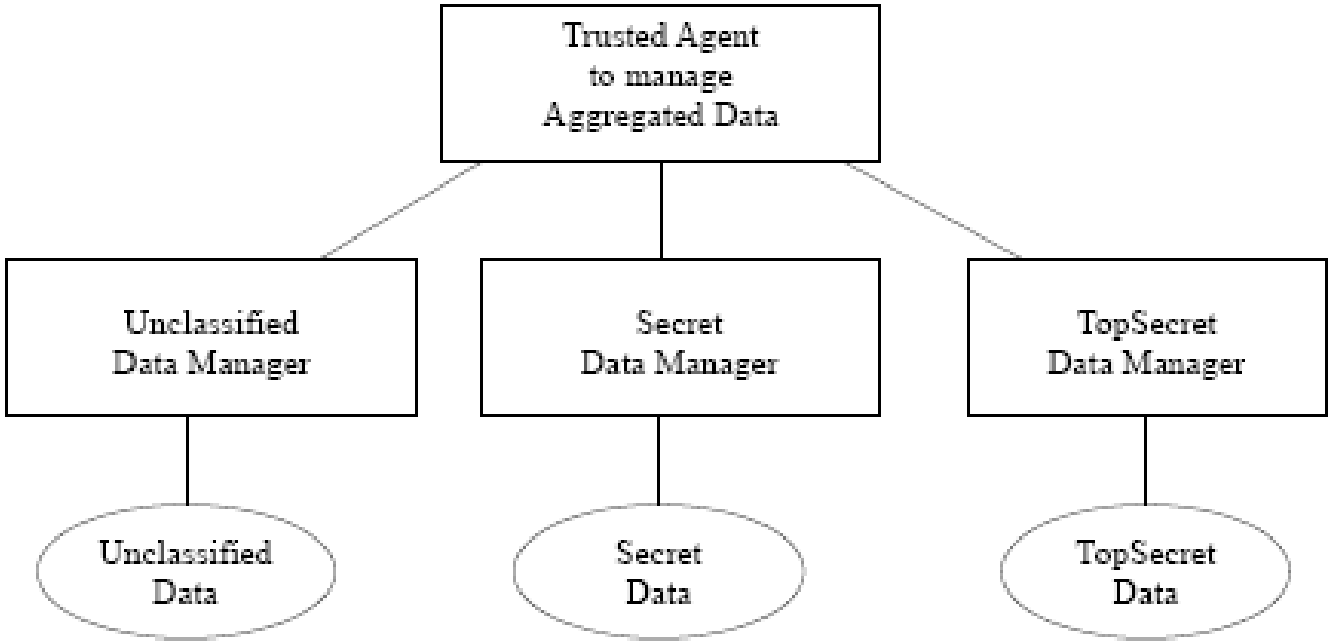


Figure 10. The distributed architecture - the partitioned approach

The Distributed Architecture (5)

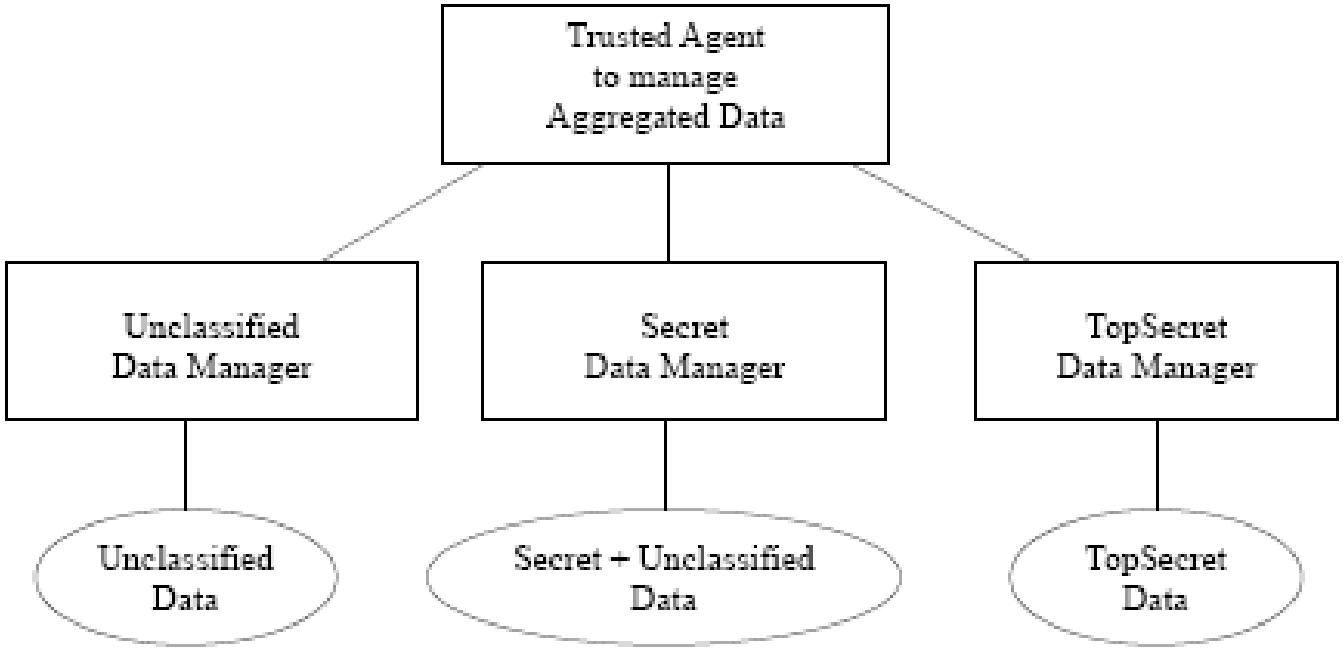


Figure11. The distributed architecture – the replicated approach