A Scaleable Extended DGSA Scheme for Confidentiality of Shared Data in Multidomain Organizations

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University.

By

Tarik Himdi

B.S. Computer Science, University of San Diego, San Diego, California, 1985
M.S. Computer Science, The George Washington University, Washington D.C., 1993

Director: Dr. Ravi S. Sandhu, Professor
Information and Software Systems Engineering

Spring 1998
George Mason University
Fairfax, Virginia

Acknowledgments

I wish to express my deepest gratitude to my advisor Professor Ravi Sandhu for his continuous guidance, advice, support, and encouragement during my pursuit of the Ph.D. program. He has been a source of great inspiration. I consider myself very fortunate and privileged to have benefited from his exceptional insight, experience and maturity in research.

I also would like to express my sincere appreciation to Professor Edgar Sibley for being the chairman of my doctoral advisory committee and for his time, support and suggestions during my dissertation.

I would like to thank Professor Mohamed Habib for serving as a member of my committee and for providing continuous encouragement and motivations during this pursuit.

I would like also to thank Professor Xiaoyang Wang for serving as a member of my committee and for his support and suggestions.

I wish also to thank my friends Wajdi Al-Jedabi, Dr. Sami Halawani, Dr. Adel Ajaji, Fouad Gasas, Dr. Nabel Bokahree, Eiz-Alden Baraka, and Gamar Monur for their supports and encouragement.

Finally, I would like to thank my parents, brothers and sisters for their love, my wife for her love and sacrifices during this effort, and children Hebba, Hanen, and Ahmed for their love and support.

Table of Contents

List of Tables

List of Figures

Abstract

A Scaleable Extended DGSA Scheme for Confidentiality of Shared Data in Multidomain Organizations

Tarik Himdi, Ph.D.
George Mason University, 1998
Dissertation Director: Dr. Ravi S. Sandhu

There is a need for data sharing between distributed heterogeneous systems in multidomain organizations (M-D.Os). However, there is a greater demand between those systems for data sharing under different security requirements. An M-D.O is an organization that consists of two or more sub organizations each of which is performing specific tasks using separate system(s). Governments are M-D.Os since there are different governmental organizations that belong to the same M-D.O.

In this dissertation, a government was chosen as an example of an M-D.O. The data about the security requirements of three government ministries, that need to exchange information, were collected directly by the author from the sources through interviews. Those ministries required three increasingly sophisticated security requirements on shared data: No-obligation access security, Multilevel security, and Chinese Wall security. Each requirement seeks confidentiality of shared data as the primary concern.

The requirements were analyzed by using a lattice model as the common denominator. Generally, it is important to show that it is possible to represent different confidentiality requirements (pertaining to information flow) in complex data sharing situations by using different lattice models. These or similar requirements can be expected to arise in a large class of M-D.Os.

The US Department of Defense (DOD)'s Goal Security Architecture (DGSA) is a recently proposed architecture that has been adopted by DOD. DGSA is designed to provide data sharing and data-transfer between distributed heterogeneous systems in an M-D.O. Analysis shows that pure DGSA cannot support any lattice model requirement. But Hilborn proposed a scheme that supports lattice model requirements by strengthening some DGSA rules. A major problem in his scheme is that it is not scaleable with respect to the size of the lattice. In this dissertation, an extended DGSA scheme that is capable of supporting lattice model requirements as well as being scaleable with respect to the lattice's size was developed and analyzed.

In summary, the contributions here are the following:

1) Definition and analysis of the case study's confidentiality requirements of shared data, and their representations in lattice models.
2) Extension of DGSA through sets of additions to the original inadequate architecture in order to develop a scheme that is capable of supporting lattice model requirements as well as being scaleable in size of the lattice.

1. Introduction

Most organizations rely extensively on computer technology when they start to create and implement new techniques and tools to improve their services. One area that requires research developments for new techniques is the security of *shared data* in a heterogeneous distributed environment.

In a large organization that uses heterogeneous distributed systems (later defined as M-D.O), there are demands for data sharing between these systems, but there are greater demands for preserving *confidentiality* of shared data. For example, if a large company is using two distributed systems (A and B), where system A is used by the research department in Los Angeles and system B is used by the training department in New York, there could be a need to share files about new products from system A to system B and files about training projects from system B to system A. But system A may require multilevel security (MLS) to access its files, while system B may require Chinese Walls security to access its files. Confidentiality of shared data under these two security requirements becomes a major concern of the two systems, because one system may fail to support the other's security requirement. For simplicity, security requirements that concern confidentiality are called in this dissertation "confidentiality requirements". In this dissertation, three confidentiality requirements (pertaining to information flow) in a complex data sharing situation are represented by lattice models.

In addition, a very well-known architecture, DOD Goal Security Architecture

(DGSA) [1] , is analyzed here with respect to applying any lattice model requirement. The reasons that caused pure DGSA to fail to support lattice requirements are identified here. Finally, the dissertation shows that DGSA can be extended to enforce an information flow lattice in a scaleable manner with respect to the size of the lattice.

1.1 Background of the Study

This section defines some terms that are used here.

Definition 1.1: ***Partial Order Set (POSET)***

is the pair (S,≤) where S a set and ≤ is a binary relation that is reflexive, antisymmetric, and transitive on S. [2]

For example, Figure 1-1 shows a partial order set S ={a,b,c,d,e,f}, where a ≤ b, a ≤ c, a ≤ d, a ≤ e, a ≤ f, b ≤ e, b ≤ f, c ≤ f, and all reflexive ones.

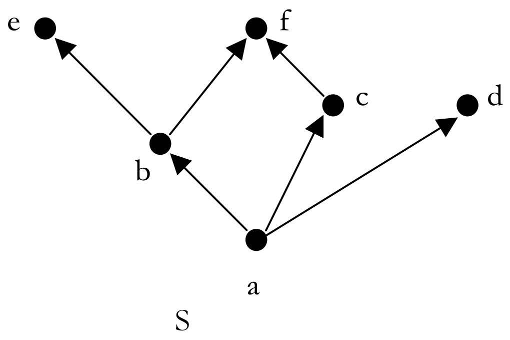


Figure 1-1: A partial order set

Definition 1.2 : ***Totally Ordered POSET***

is a POSET S where for each choice a and b in S, either a ≤ b or b ≤ a. [2]

For example, Figure 1-2 shows a totally ordered POSET T ={x,y,z,w}, where x ≤ y, x ≤ z, x ≤ w, y ≤ z, y ≤ w, and z ≤ w, and all reflexive ones. Thus a chain is a totally ordered POSET.

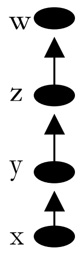


Figure 1-2 : Totally ordered POSET

<u>Definition 1.3</u>: ***A Maximal Chain of a POSET***

is a chain which is not properly contained in another chain of the POSET S. [2]

For example, the maximal chains in the POSET of Figure 1-1 are the chains: {a,b,e}, {a,b,f}, {a,c,f}, and {a,e}. Notice that the maximal chains are not the same size. The size of a maximal chain is measured by the number of elements it has. Thus, the maximal chain {a,b,e} has the largest size (the size here is 3), because it has the maximum number of elements any maximal chain may have.

<u>Definition 1.4</u>: ***A Lattice***

is a POSET S where every x and y in S has a unique greatest lower bound (x ∧ y) and a unique least upper bound (x ∨ y). The binary relation ≤ represents here, in addition, a can-flow relation on S. [2]

For example, Figure 1-3 shows a lattice, where Low is the greatest lower bound (glb) for A and B: A ∧ B= A meet B, and High is the least upper bound (lub): A ∨ B= A join B. In addition, for B and C, the glb is Low the lub is D.

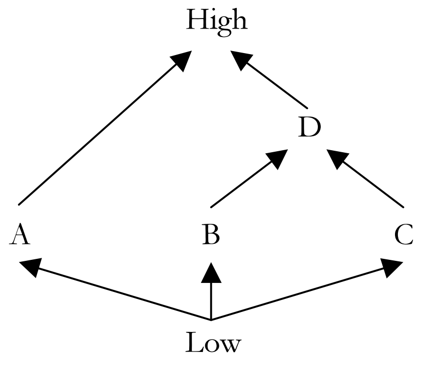


Figure 1-3: Lattice

<u>*Definition 1.5*</u>*: A Subset Lattice of m Elements*

is a special type of lattice S which is derived from a nonlinear ordering on the set of all 2^m subsets of S. The binary relation $\leq$ corresponds to set inclusion $\subseteq$, the least upper bound operator ($\vee$) to set union $\cup$, and the greatest lower bound operator ($\wedge$) to set intersection $\cap$. [3]

For example, Figure 1-4 shows a subset lattice of 2 elements (a,b). The empty set ϕ corresponds to the intersection of a and b, while {a,b} corresponds to the union of both.

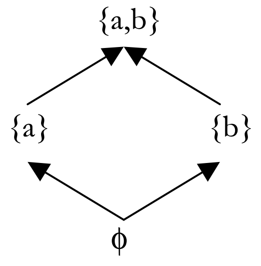


Figure 1-4: A subset lattice

A subset lattice of m elements has a total of 2^m compartments, where system-Low set (empty set) is connected to m subsets, each of which has only one element. As information flows up, the number of elements in the subsets increased by one until it reaches system-

High, which has m elements.

<u>Definition 1.6</u>: A Subset Lattice of m Elements and n Classes

is a special type of lattice which consists of n subset lattices, each of which is of m elements. Each subset lattice represents a class.

For example, Figure 1-5 shows a subset lattice of 2 elements and 2 classes. Each class is represented by a sub lattice consisting of a subset lattice of 2 elements. These sub lattices are connected at corresponding nodes via edges oriented in the same direction. Thus in Figure 1-5, the two subset lattices U, and C are connected via 4 edges: u/ϕ to c/ ϕ, u/{a} to c/{a}, u/{b} to c/{b}, and u/{a,b} to c/{a,b}.

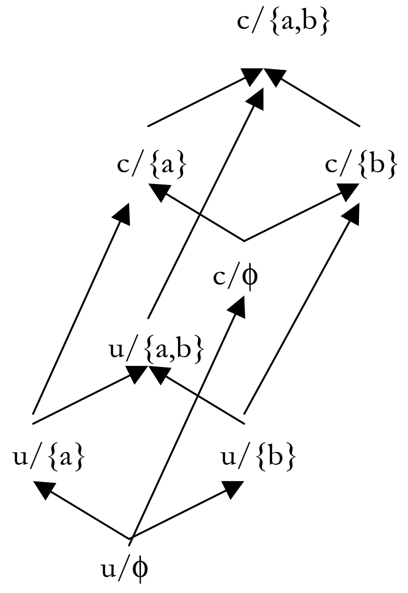


Figure 1-5 : Subset lattice of 2 elements and 2 classes
C class is higher than U class

<u>Definition 1.7</u>: Heterogeneous Distributed Systems

consist of computers with dissimilar operating systems and/or CPUs located in different geographical locations, where each has its own database(s) and may run different types of applications. [4]

Definition 1.8: Multidomain Organizations

The author's definition of a Multidomain Organization (M-D.O) is:

an organization with two or more sub organizations each of which is performing specific task(s) using separate system(s). All systems in an M-D.O are heterogeneous distributed systems.

A good example of an M-D.O is a government in which several ministries/ departments offer different services. Figure 1-6 shows a government as an M-D.O and its ministries as the sub organizations.

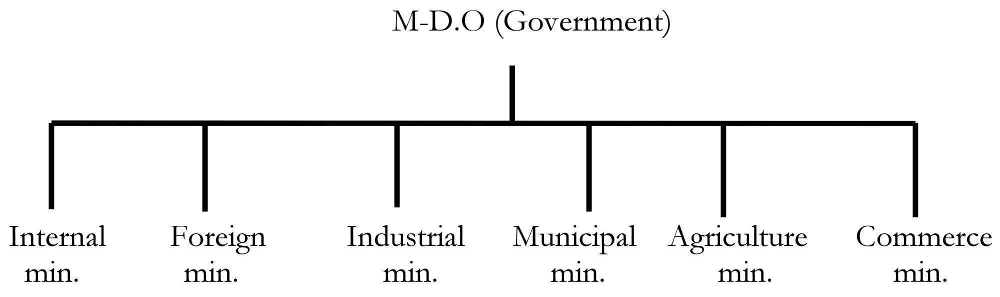


Figure 1-6: M-D.O with its sub organizations

In this dissertation, the Saudi government will be considered as an example of an M-D.O where separate government departments have different database systems, and they need to share data depending on specific security requirements (the dissertation here refers to this M-D.O as the Saudi M-D.O).

Definition 1.9: Shared Data in M-D.O

The author's definition of shared data in an M-D.O is:

databases/files/programs created by systems administrators of any sub organization in an M-D.O.

Such data could be accessed with Read or Write rights (or both) by local users of that sub organization as well as remote users from other sub organizations in the same M-D.O. Such access will remove the absolute ownership of this data from its creator (insert/ delete/ update) and will mark it as shared data.

1.2 Conceptual framework

This section will review related literature about lattice-based access-control models, because these will be applied in solving the security requirements of the case study. The relationships between confidentiality, access-control models, and M-D.Os are then discussed. Finally, brief descriptions of DGSA and a Hajj case study are given.

1.2.1 Lattice models

Lattice models were developed to handle information flow in computer systems [5]. In [6], Sandhu examined the following two lattice models that are used in government as well as commercial arenas:

1- Mandatory Access Control (MAC) in the Bell-LaPadula (BLP) model

2- Chinese Walls model

1.2.1.1 MAC in BLP model

The concept of MAC was first formalized by Bell and LaPadula [7]. They defined a model commonly called the BLP model. BLP takes a two-step approach to access control [6]. First, there is an access matrix D, which represents the access rights of subjects (programs running on behalf of users) to objects (files) in a system. The contents of such a matrix can be modified by subjects. But the operations (Read and Write) must be authorized by the mandatory access-control policy, over which users (subjects) have no control. Second, the mandatory access-control policy is expressed in terms of unchanged security labels attached to subjects and objects (each label contains two parts: subject's or object's security class + subject's or object's name) . The mandatory access rules given in BLP are as follows, where λ signifies the security label of the indicated subject or object:

- Simple security property: subject S can read object O only if $\lambda(S) \geq \lambda(O)$.
- *-property: subject S can write object O only if $\lambda(S) \leq \lambda(O)$.

Read access implies a flow from object to subject, hence the requirement that $\lambda(S) \geq \lambda(O)$. Write access conversely implies a flow from subject to object, hence the requirement that $\lambda(S) \leq \lambda(O)$.

With such rules in mind, it is possible for MAC to support MLS (Multilevel Security), where each user belongs to a security class (Unclassified (U), Confidential (C), Secret (S), Top Secret (TS)). In addition, every user may have different subjects to invoke. A user can have subjects with the same security class as his or lower. Thus, a user can access the same object with different access rights depending on the subject he invoked.

As shown in Figure 1-7, a lattice's category dominates another if compartments of the former are a superset of the latter. Hence, depending on its label, an S subject can Read/ Write the category that has a similar label to the subject's, can Read only some categories that are dominated by this label, while it can Write only to categories that dominate this label.

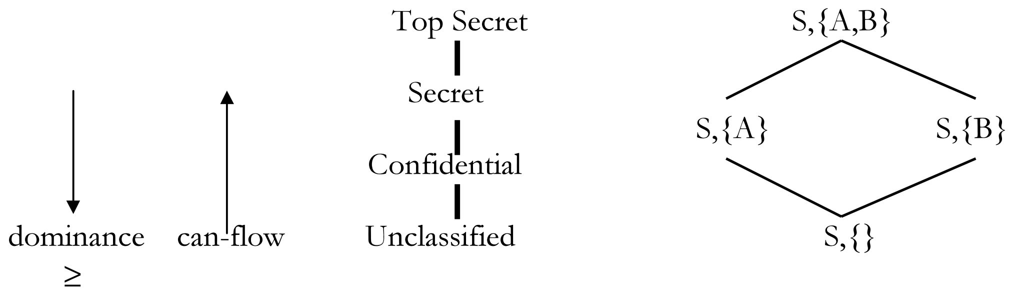


Figure 1-7: BLP model

1.2.1.2 Chinese Walls model

The Chinese Walls policy arises in the financial segment of the commercial sector, which provides consulting services to other companies. The policy was identified by Brewer and Nash [8]. The objective of the Chinese Walls policy is to prevent information flows that cause conflict of interest for a consultant [9]. If a consultant company has information about some banks and oil companies, then there are two conflict-of-interest (COI) classes: one for banks and another for oil companies. Any consultant can access information only about *one data set* from each class; for example, he/she can access information about one bank and one oil company.

Sandhu showed in his paper [10] that the Chinese Walls policy can be represented by a BLP model by maintaining a careful distinction between users, principals, and subjects. Each object is associated with an *n*-element *vector* [i1,i2,...,in] where each ik ∈ COIk or ik = ⊥ for k = 1..n and ⊥ = null . So if a system has four COI classes (n = 4), and an object contains information from company 5 in COI2 and company 7 in COI4, then it is labeled

[⊥,5,⊥,7]. In the previous example, if each COI class has two data sets, then the Chinese Walls model could have a lattice, as in Figure 1-8, where [BANK,OIL CO] is the object's vector which indicates what information each object has.

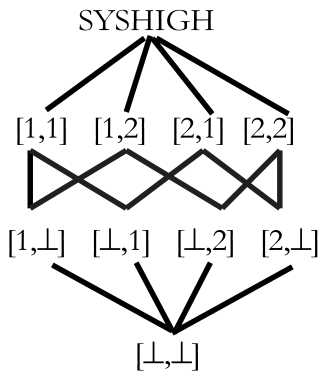


Figure 1-8: Chinese Walls model

SYSHIGH objects that violate the Chinese Walls policy can be accessed by the security administrator only.

1.2.2 Confidentiality in Multidomain organizations

Since information could be related in different systems of an M-D.O, the demand for data-sharing between those systems could be high [11]. The major security objective in sharing data between systems of an M-D.O is confidentiality, because:

- Creators of files that need to be shared lose absolute ownership, since other users may update the records of these files. This makes shared data even more valuable and requires prevention from unauthorized access and authorization violations.
- Systems use different hardware, operating systems, and database management systems (DBMS), some of which may not support all security policies and requirements. This

could lead to the use of inappropriate security measures to handle the missing policy or requirement, which may cause authorization violation [12].

For example, if system A requires multilevel access control on its shared files and system B does not support MLS, then system B may use an inappropriate method to assign security classes or labels to its users who need to access system A's shared files. Such a method could lead to authorization violations of shared data.

- Serious decisions could be taken depending on information gathered from a shared file or a combination of shared files. Thus, confidentiality of those files must be strongly maintained.

1.2.2.1 Security threats in multidomain organizations

In M-D.Os, the main threats that concern the security administrators and the security architects are:

1. Unauthorized access to the systems.
2. Authorization violation: A person authorized to use a system for one purpose uses it for another purpose.
3. Information leakage.
4. Intercept/alter: A communicated data item is changed, observed, deleted, or substituted while in transit.

The demand is high for a security architecture that helps M-D.Os perform data sharing and data transfer with reduced threats when using multiple security requirements.

1.2.2.2 Security architectures for confidentiality in M-D.O

In an M-D.O with different systems, each system could be applying a different access-control policy [13]. In order for those systems to share data without violating the confidentiality of any system or its shared data, a security architecture must be adopted by the M-D.O. This architecture must:

- Contain a communication network capable of allowing systems to communicate easily, regardless of each system's hardware and software.
- Invent or reuse data sharing mechanisms between systems.
- Support security requirements and policies of individual systems.
- Maintain confidentiality of shared data between systems.

1.2.3 DGSA

DGSA was developed by the Defense Information Systems Security Program. It is one of the few architectures that supports data sharing/data-transferring between distributed heterogeneous systems. The basic idea of DGSA is to create information domains between two distributed systems. An information domain consists of the following:

- A set of information objects.
- A set of members.
- A security policy that regulates access to the domain.

A domain's information objects are the databases/files/programs of one system that need to be shared with or transferred to other domains. A domain's members represent the systems' users who are allowed to access the domain's objects depending on the domain's security policy. It is shown here that DGSA cannot directly support the three confidentiality requirements of the Saudi M-D.O or any lattice model. However, it is possible to develop a set of extensions to it to support these requirements and all lattice models.

1.2.4 The case study: Why Hajj?

The case that will be used in this dissertation is about an M-D.O such as the Saudi Arabian government, where different ministries (Foreign (F), Internal (I), Hajj or pilgrimage (H)) are working together toward a common goal during the pilgrimage season.

The reasons for choosing Hajj as the case study in this dissertation are:

1. It is a good example of an M-D.O with distributed systems that need to share related data.
2. It is a real example where every ministry:

- is using real data.
- has real users.
- has real security requirements.
- applies real access-control polices and security models
- wants to share real data with the ministries.

3. The Hajj example has various security requirements of interest that provide different stages of complexity. These requirements concern confidentiality of data that will be shared between the three systems in this M-D.O.

The author traveled to Saudi Arabia and met with officials from the three ministries (F,I,H) and summarized their security needs. A detailed explanation about the case study will be given in Chapter 2.

1.2.5 Other examples

Another example of an M-D.O is the United States Government, which has the network between the US Navy system, Federal Drug Interdiction system, Central Intelligence Agency (CIA) system, and Federal Bureau of Investigation (FBI) system [1]. All four systems need to share and transfer data regarding illegal drug activities. To extend the security requirements to data sharing and data transfer between these systems, the security administrators of those systems might request MLS or Chinese Walls security on some files before they can be shared or transferred.

1.3 Statement of the problem

In some M-D.Os, it is possible to have a situation in which different security requirements that support confidentiality are needed in order to allow data sharing between different sectors of that M-D.O. When each one of those requirements pertains to information flow, it is important to know if it is possible to maintain data sharing using those requirements by lattice models.

Meanwhile, DGSA is designed to provide data sharing and data transfer between distributed heterogeneous systems in any M-D.O. But in its pure form, DGSA cannot support any security requirement that requires lattice model. However, Hilborn developed a DGSA scheme that solves this problem. One major problem of his scheme is that it is not scaleable with respect to the size of the lattice.

1.4 Summary of contributions

This dissertation shows that when different confidentiality requirements (pertaining to information flow) are needed in complex data sharing situations, it is possible to maintain data sharing under those requirements by using lattice models. In addition, the dissertation demonstrates that it possible to develop an extended DGSA scheme that is capable of supporting any lattice model requirement and is scaleable with respect to the lattice's size.

In summary, the contributions here are the following:

1) Definition and analysis of the case study's confidentiality requirements of shared data, and their representations in lattice models.
2) Extension of DGSA through sets of additions to the original inadequate architecture in order to develop a scheme that is capable of supporting lattice model requirements as well as being scaleable to the size of the lattice.

1.5 Organization of the dissertation

The dissertation consists of five chapters and one appendix. Chapter two considers the Saudi government as a real example of an M-D.O. It illustrates complete definitions of systems involved in Hajj and how they became parts of the Saudi M-D.O. The confidentiality requirements of those systems (No-obligation access security, MLS, and Chinese Walls security) are discussed briefly in this chapter.

Chapter three deals with the confidentiality requirements of the Saudi M-D.O discussed in chapter two. This chapter analyzes each requirement in depth. Then it shows how that requirement could be satisfied by building a lattice model for it. Finally, the chapter demonstrates the information flow polices using the three requirements for each system of the Saudi M-D.O.

Chapter four gives a complete analysis about pure DGSA and the reasons behind its failure to support lattice-based requirements of an M-D.O. In addition, the chapter examines Hilborn's scheme and its problems, then it introduces an extended DGSA scheme that is capable of supporting lattice requirements as well as being scaleable with respect to the size of the lattice. Finally, the chapter discusses applying the extended DGSA scheme to each requirement of the Saudi M-D.O.

The dissertation concludes in chapter five, while Appendix A illustrates DGSA technical specifications.

2. The Hajj case study

2.1 Introduction

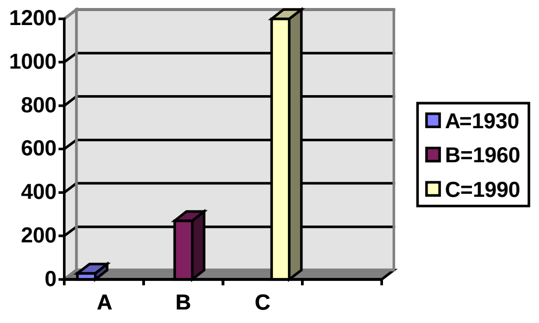
The Hajj, or pilgrimage, is one of the five required tasks in Islam. Every adult Moslem must perform the Hajj at least once in his/her lifetime [14]. The story of pilgrimage goes back to the early Islamic age, when prophet Mohammed, may peace be upon him, came back to Makkah with his followers after defeating their enemies. The journey was called "pilgrimage to Makkah."

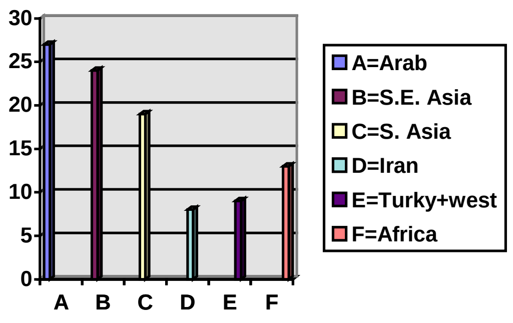
Since that time (630 AC), every year a large number of Muslims come from all over the world to Makkah in Saudi Arabia to perform Hajj in certain places during certain times. The government of Saudi Arabia annually spends millions of dollars on projects that will ensure the safety and comfort of all pilgrims. Three major government ministries create and process Hajj data separately in their systems. Sharing of Hajj-related data between these ministries is currently done manually. There are many benefits from sharing data electronically. For example, currently a user in each system enters the personal data of every pilgrim, but these data could be captured once at the F system (when a pilgrim applies for Hajj visa) and transferred later to I and H-systems. But due to the sensitivity of some data and the common requirement from officials of not sharing all data, there is a need for a trusted environment that provides interoperability between various government ministries while ensuring confidentiality of shared data. To study the feasibility of establishing such an environment, the author collected data regarding the confidentiality requirements of the three Saudi ministries directly from the sources through interviews.

This chapter is organized as follows. Section two provides a background about the services offered by the Saudi government during Hajj. Section three introduces the Saudi government as an M-D.O consisting of three sub organizations (F,I,H) which need to share data related to Hajj. Section four presents Hajj-data that each system in the Saudi M-D.O is willing to share and the three confidentiality requirements under which these data can be used. Section five concludes this chapter.

2.2 Services offered by the Saudi government during Hajj

For the past 13 centuries, pilgrims came to Makkah every year during Hajj month with no restrictions or visa or any kind of permit, and their total number averaged between 20,000 and 30,000 annually [14]. Most came with convoys that spent months traveling on sea and land. As the number of Muslims increased and methods of transportation made it easier and more affordable to travel, the total number of pilgrims increased 40 times, where it is currently averaging between 1,000,000 and 1,200,000 pilgrims per year [15]. Figure 2-1 shows this growth as well as the percentage of pilgrims by region.





Total: by 1,000 Percentage: by regions/1990

Figure 2-1: pilgrim total and distribution

To provide better services to pilgrims, in the 1970s the Saudi government established rules and regulations that keep the number of pilgrims in balance with the services available and solve the problem of pilgrims becoming illegal immigrants, especially after Saudi Arabia became a very attractive place to work due to the wealth from oil sales. Some of these rules and regulations are [16]:

1) Controlling the number of pilgrims from each country to 1,000 for each million of the Muslim population in that country (applied since 1986).
2) Regulating the entrance to Saudi Arabia for pilgrimage by requiring a visa for the Hajj period.
3) Taking the pilgrim's passport once he/she arrives at the airport/seaport and giving it back to him/her at departure day at the airport/seaport.
4) Limiting the travel of pilgrims in Saudi Arabia to only those places in the pilgrimage journey.
5) Providing well-trained guides (mutaufs) working at Hajj service offices to take care of pilgrims' needs.

6) Establishing special fire and rescue teams in crowded areas.

7) Increasing the number of police to control the traffic and ensure the safety of pilgrims.

8) Establishing special hospitals and clinics for pilgrims with contagious diseases.

Once a pilgrim arrives in Jeddah, he or she will go through immigration, where his or her name, passport number, and other information are entered into the immigration system. All arriving pilgrims are then transferred to the Hajj hall to be registered at the reception office, then they will be provided with buses to take them to Makkah.

2.3 The Saudi M-D.O

Three major distributed systems (Foreign ministry's system, Internal ministry's system, and Hajj ministry's system) need to share data about pilgrims and other related information that affect the security and safety of the entire operation. Each system is willing to share some of its files with the other two systems if a security architecture can be adopted that ensures the confidentiality of shared data. Some of the advantages that might benefit the Saudi government when such a security architecture is applied are:

1. Sharing data that might prevent terrorist attacks.
2. Solving the illegal immigration problem of pilgrims overstaying their visas.
3. Accessing statistics reports that could improve the services offered to pilgrims.
4. Determining the location and status of any pilgrim.

2.3.1 The Foreign ministry's system

Currently, the system administrators of the F-system are working to connect it via modems with every Saudi embassy's (E) system around the world. The F-system (which is located in Riyadh) is applying MLS. F-system has its own files related to Hajj such as "Haj-visas", which can be accessed by any Saudi embassy in the world. Figure 2-2 shows the plan for F-system's architecture.

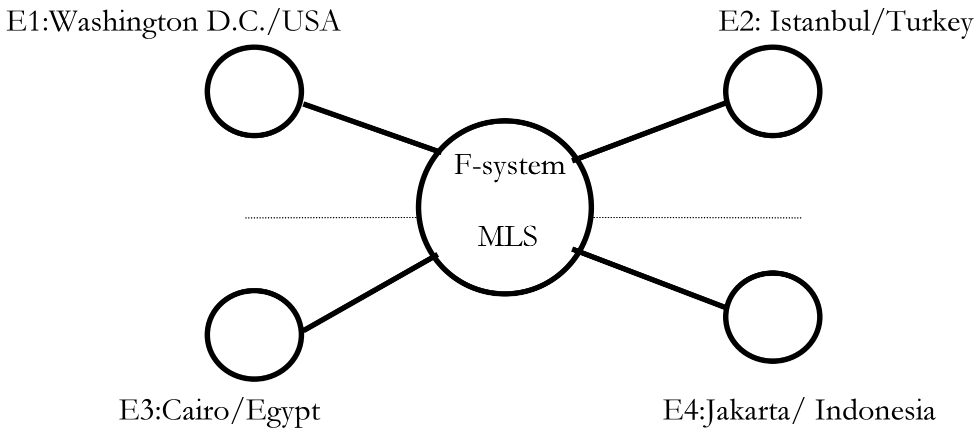


Figure 2-2: The F-system

2.3.2 *The Internal ministry's system*

The Internal ministry is the largest ministry in the Saudi government. The main responsibilities and services of this ministry include: police force, fire and rescue, intelligence services, secret services, immigration and naturalization, passports and citizenship, traffic control, vehicle registration and ownership, civil rights and disputes, and foreign labor permits. Figure 2-3 shows some departments of the Internal ministry.

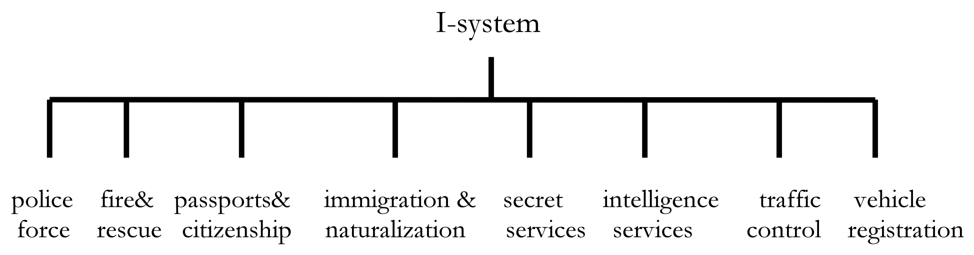


Figure 2-3: The Internal ministry departments

The Internal ministry's main system is currently at the National Data Center in Riyadh, connected to every department of the Internal ministry in every city of Saudi Arabia via modems. The main system is applying an MLS policy for access control. I-system has its own files related to Hajj, such as "Haj-In" and "Haj-Out", which can be accessed by an immigration officer at any Saudi airport or seaport. Figure 2-4 shows the I-system's architecture.

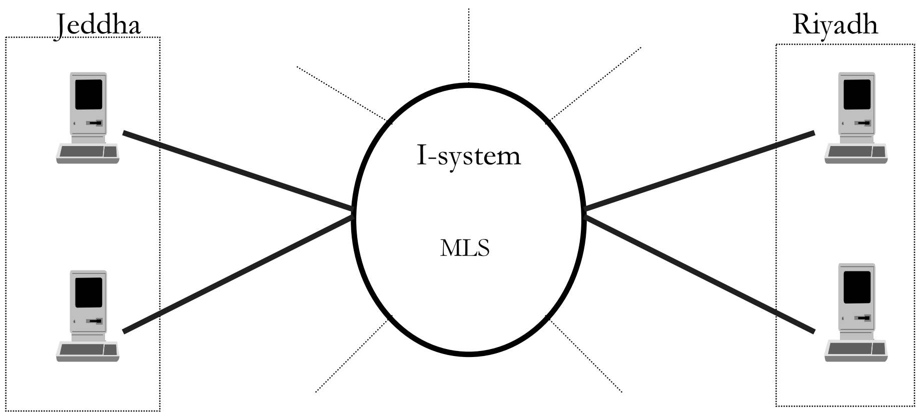


Figure 2-4: The I-system

2.3.3 *The Hajj ministry's system*

The Hajj ministry has many sub organizations working with it. Each sub organization has a different system. Some of these systems are located in Jeddah, while the rest are located in Makkah and Medina. The central computer of the H-system is located in Jeddah and is connected to servers in Makkah and Medina. The H-system is applying a MLS policy.

The system administrators of the H-system are working to connect it with the other sub organizations' systems via modems. Some of these sub organizations are [16]: registration (united agents office), transportation (public vehicles union), six guide establishments (mutauf) in Makkah, and one guide establishment (adela) in Medina.

Each system (H and its sub systems) has its own files related to Hajj. For example H-system has "Arrival&departure" file, which can be accessed by H's users, while the Arab guide establishment has "Guide-Est" file, which can be accessed by H's users as well as users from that establishment. Figure 2-5 shows the plan for H-system's architecture, while Figure 2-6 shows the six guide establishments in Makkah that serve pilgrims according to the geographical area of their nation:

- Arab Hajj-guide establishment: serving pilgrims from Middle Eastern countries.
- Southeast Asia Hajj-guide establishment: serving pilgrims from Indonesia, Malaysia, Philippines, China, etc.
- South Asia Hajj-guide establishment: serving pilgrims from South Asia, India, Pakistan, Bangladesh, etc.
- Iran Hajj-guide establishment: serving pilgrims from Iran only.
- Turkey and West Hajj-guide establishment: serving pilgrims from Turkey, Europe, North America, and South America.

- Africa Hajj-guide establishment: serving pilgrims from the rest of Africa (excluding Middle Eastern countries) such as Nigeria, South Africa, etc.

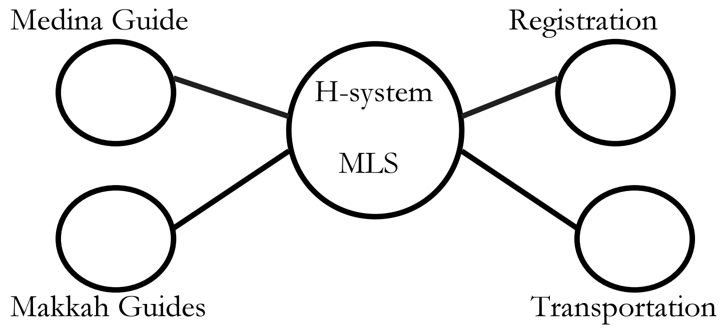


Figure 2-5: The H-system

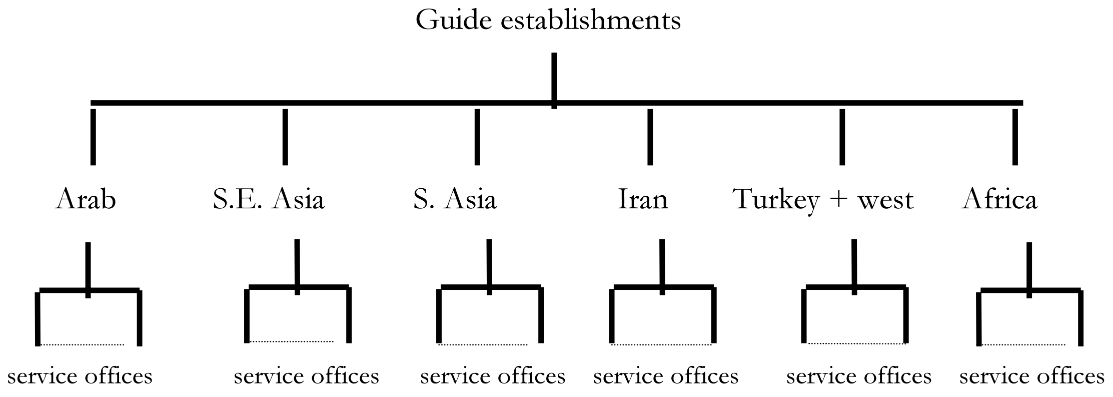


Figure 2-6: The six guide establishments in Makkah

Each establishment has many service offices. Each office serves 2,000-3,000 pilgrims. All guides in each office have been approved and selected by the Operation department in the headquarters of the establishment. Not anyone can be a guide, only those who inherit this job from their ancestors can be chosen to practice it [16]. Every establishment has a board of directors that manages different departments as follows:

- Operation for service offices.

- Financial.
- Transportation.
- Computer.
- Arrival and departure.
- Makkah-housing.
- Araft and Muna-housing.
- Guides-assistance.
- Administrative affairs.

The Operation department for service offices is responsible for supervising all service offices in the quality of services offered to the pilgrims. For example, the inspectors of the Operation department are looking to see if the guides in a service office are providing comfortable accommodations in Araft and Muna for their pilgrims.

Each service office has nine guides working together as one group:

1. The president
2. The vice president
3. The passports-registration member
4. The Makkah-housing service member
5. The Araft-housing member
6. The Muna-housing member
7. The transportation member
8. The reception member

9. The financial member

In order for guides in a service office to perform their duties for pilgrims, the establishment that the office belongs to pays the office S.R 150 ($40) for every pilgrim it serves, distributed over three installments (20%, 50%, and 30%). Each office has discretion to manage spending of these funds, but if duties of the office are not carried out, penalties will be imposed [16]. Some offices might lose money, and guides are obligated to pay the difference from their personal savings, while some office may gain profits that are distributed equally among the members. In other words, each service office is similar to a temporary company that exists for three months.

2.4 Multiple security requirement of the Saudi M-D.O

By considering the Saudi government as an M-D.O with three sub organizations (F,I, H), the confidentiality requirements of the over all system can be analyzed as the combination of the confidentiality requirements needed by F, I, and H-systems. Currently, each system is using MLS locally, with the four classical hierarchical security levels: Unclassified (U), Confidential (C), Secret (S), and Top Secret (TS).

Based on data collected by the author from Saudi officials, three increasingly sophisticated confidentiality requirements were identified: No-obligation access security, MLS, and Chinese Walls security. The three systems' administrators should agree on a security architecture that is capable of using different types of access-control models to meet these security requirements.

2.4.1 *First requirement: No-Obligation access security*

The first requirement deals with sharing files that do not contain sensitive information, but only certain systems can access them. It is defined as follows.

A user's subject can access a remote U file provided:

a) the user is assigned to access the remote system that has that file,

b) the user's local system has permission to access that remote file, and

c) U subjects can have Read/Write access to that file but C (and higher) subjects can only have Read access.

For example, say Fu-10 is a U subject from F-system who is assigned to access H system (condition a). The H-system has two U files (Haj-Missions and Housing) as shown in Table 2-1(a) column I. The F-system has permission to access the "Haj-Missions" file but cannot access the "Housing" file (condition b). Therefore, Fu-10 can access "Haj-Missions" file with Read/Write rights. Now if Fc-20 is a C subject from F-system who is also assigned to access H-system, then he may access the "Haj-Missions" file with only Read right (due to * property).

This is called the No-obligation requirement, because there is no obligation that requires the subject's security level to be upgraded if two or more U files are accessed simultaneously by that subject (i.e., U + U = U). For example, if the U subject "Iu-40" from the I-system is authorized to access H-system (condition a), and since H-system has two U files "Haj-Missions" and "Housing", both of which are accessible to I-system's users as shown in Table 2-1(a) column I (condition b), then Iu-40 can access both files with Read/ Write rights with no obligation to upgrade its security level to C, for instance. In other words, any combination of U files will have the security level as U.

Column I of Table 2-1 (a) represents the U files for the first requirement and the systems that can access them. For example, F-system has two U files "Haj-Attaché" and "Haj-Tourist", each of which can be accessed by users from F, I, and H-systems. Access by F-system users is implied and not shown explicitly in the table, whereas access by I and H systems is shown by the tag [I,H] attached to each file. The I-system has a single U file "Haj-Plan", which can be accessed by I and H-systems but not the F-system. As discussed in examples above, the H-system has two U files: "Haj-Missions", which can be accessed by users from H, F, and I-systems, and "Housing", which can be accessed by users from H and I-systems only. The tokens such as "*atc*" and "*tor*" attached to the respective files will be used as abbreviations for them later. This notation also applies to columns II and III. Note that access by a system to its own files is implied and not shown explicitly in Table 2-1(a). Table 2-1(b) gives a brief description of the contents of each file.

	I	II	III
	No-obligation/Unclassified	Multilevel-aggregation/C,S,TS	Chinese Walls
F-system	Haj-Attaché "*atc*"[I,H] Haj-Tourist "*tor*"[I,H]	Haj-Visas"*vis*"[I,H]+ Gov-Guest"*gus*" [I,H]	Haj- Deplom"*dpl*" [I,H] Secret-Comm "*scm*" [I,H]
I-system	Haj-Plan "*pln*"[H]	Haj-In "*hji*" [F,H]+ Haj-Out "*hjo*" [F,H]	Black List "*bkl*" [F] Non-Saudi in List "*nsl*" [F]
H-system	Haj-Missions "*mis*" [F,I] Housing "*hos*" [I]	Guide-Est "*gde*" [I] + Arrival&departure"*a&d*"[F,I]	Haj-Registration"*rgs*"[F,I] Transport "*trn*" [I]

at H-system "file" [F,I] = from H to F, and I (a)

	File Name	File Description: each includes *
I	Haj-Attaché	reports about agreements between Saudi officials and other Governments' officials about co-operation in organizing their pilgrims
	Haj-Tourist	data about tourist groups that organize journeys for Hajj
	Haj-Plan	data about plans for traffic control and fire fighting
	Haj-Mission	data about every nation's pilgrimage mission in Saudi Arabia
	Housing	data about houses rented to accommodate pilgrims in Makkah and Medina
II	Haj-Visas	personal data of a pilgrim granted Hajj visa
	Gov-guest	the names, nationality, and all necessary arrangements of government's guests who will perform Hajj
	Haj-In	data about a pilgrim's arrival flight, date, and port
	Haj-Out	data about a pilgrim's departure flight, date, and port
	Guide-Est	data about each establishment and its services offices, including the names of pilgrims each office serves
	Arriv&dept	personal data about each pilgrim arriving and departing Makkah and Medina
III	Haj-Deplom	reports about senior government staff and VIP from other nations who wish to perform Hajj
	Sec-Comm	reports of confidential letters between F central system and its subsystems regarding Hajj
	Black-list	data from intelligence resources about non-Saudi citizens who should not enter or leave Saudi Arabia
	Non-Saudi	personal data about every non-Saudi citizen who lives inside Saudi Arabia and information about his sponsor
	Haj-Regist	type and amount of payment received from each pilgrim. This database summarizes total revenue for each session.
	Transport	data about number of pilgrims who used Hajj transportation system and financial reports about this system

* Files' names used are not the original ones (b)

Table 2-1: Security requirement data table

2.4.2 Second requirement: MLS

The second requirement deals with sharing sensitive files, where confidentiality and MLS are the major concerns. These files are shown in column II of Table 1(a). Each of these files is individually labeled as C. The "+" sign between any two C files indicates that the combination of those two files will be S. Similarly, a combination of S files from two or more different systems will be TS. For example, the combination of the C files "Haj-Visas" and "Gov-Guest" is S. The combination of the two S files "Haj-Visas + Gov-Guest" and "Haj-In + Haj-Out" is labeled TS, because each one belongs to a different system.

The second requirement is defined as follows.

A user's subject can access a remote C/S/TS file provided:

a) the user is assigned to access the remote systems that has that file,

b) the user's local system has permission to access that remote file,

c) the subject and file labels must satisfy the standard simple-security and * properties (with write-up allowed), except that combinations of files may have a higher security level than their individual constituents as per the following rules:

I. a combination of two C files from a single system is S.

II. a combination of S files from two or more systems is TS.

Note that conditions a and b are identical to conditions a and b of the first requirement. Condition c is different and stipulates that confidential + confidential may equal secret and secret + secret may equal top secret. The reason for these rules is to prevent aggregation of sensitive information. Without such aggregation controls, the individual systems are not willing to share their data. Fortunately, the system administrators of the three systems do agree on the above uniform requirement.

For example, in column II of Table 2-1(a), the I-system requires that only users from I, F, and H-systems whose security clearance is S or above may access its S file "Haj-In+Haj-Out" with Read/Write rights. This is accomplished when the user invoked the S subject labeled "Haj-In+Haj-Out." C users or higher from those systems may access one of the C files "Haj-In" or "Haj-Out" with Read/Write rights. This is also accomplished when the user invoked the C subject labeled "Haj-In" or "Haj-Out", respectively. Note that this is as in the usual practice in multilevel systems, where a user is allowed to have subjects at any label dominated by the user's clearance.

2.4.3 Third requirement: Chinese Walls security

The third requirement deals with sharing special and sensitive files (Chinese Walls files) by special users (Chinese Walls users), such that a user can access only one file from each system. Moreover, these special users are not allowed to access files under the first or second requirements. Furthermore, TS users are exempted from this rule, because they are sufficiently trusted.

The third requirement is defined as follows.

A user's subject can access a remote *Chinese Walls* file provided:

a) The user is assigned to access the remote systems that has that file,

b) The user's local system has permission to access that remote file,

c) The subject is limited to a maximum access of one Chinese Walls file from each system, and the user cannot access any non-Chinese Walls file (under the first two requirements). TS users are exempted from condition c.

Again note that conditions a and b are identical to conditions a and b of the previous two requirements, but condition c is very different.

For example, in column III of Table 2-1(a), the I-system requires that any special user (non-TS) from the F-system (or I-system) who has access to the "Black-List" file of the I-system should not have access to the I-system's file "Non-Saudi in List." This requirement is motivated by the Chinese Walls model [8] to prevent aggregation of information. Such a requirement is important to prevent leakage and misuse of sensitive data. Otherwise, users (lower than TS) who access both files may modify records in the "Non-Saudi in List" file that have data about a blacklisted person.

Column III of Table 2-1(a) represents the special files (Chinese Walls files) for the third requirement and the systems that can access them. Each system has two Chinese Walls files, such that a non-TS user can access only one of them.

2.5 Conclusion

The three Hajj systems (F, I, and H) have required three different security requirements to maintain confidentiality of shared data between them. The first requirement deals with sharing non sensitive files, where certain systems (not all systems) can access them. This is accomplished by providing users' subjects with access to U files provided the three rules of the requirement are met. In addition, there is no obligation that requires the use of a higher subject than U in order to access more than one U file. Thus, a remote U subject may Read all U files of a system once it is permitted to access that system. This is the main difference between the first requirement and the second one. The second requirement deals with sharing sensitive files, and MLS must be applied to users' subjects when accessing C, S, and TS files. In the second requirement, the combination of two C files that belong to the same system is labeled S, while the combination of two S combinations is labeled TS provided that each belongs to a different system. Thus, a C subject cannot Read an S object, only S and higher subjects whose labels include that object may Read it (as in usual MLS practice). The third requirement is applying the Chinese Walls model where a special user's subject (Chinese Walls user's subject) may access only one Chinese Walls file from each system, while only TS users may access Chinese Walls file as well as U, C, S, and TS files.

As shown above, the three security requirements are increasingly sophisticated, and *confidentiality of shared data* is the major concern in all three. For example, every requirement emphasizes the confidentiality of certain types of files (the first requirement with U, second with C/S/TS, and the third with Chinese Walls) by limiting access to them with specific systems, specific types of users, and specific access rights.

3. Lattice models for the Hajj system

3.1 Introduction

Data sharing under different security requirements could appear in many different ways in Multi-domain organizations. For the Hajj case study, there is a need for data sharing under three different confidentiality requirements only. This chapter examines each requirement and finds the common denominator between all three.

This chapter is organized as follows: Sections: 3.2, 3.3, and 3.4 analyze the first, second, and third requirements respectively. In each section, the reasons for selecting a lattice model as a solution for that requirement are introduced. Section 3.5 presents the information flow policies for the Hajj network. The chapter is concluded in section 3.6.

3.2 No-obligation requirement

As discussed in chapter two, the major reason for selecting this requirement by systems involved in the Hajj's case study is the need for sharing insensitive files with some *but not all* systems. This requirement is called the no-obligation requirement because there is no obligation that requires the subject's security level to be upgraded if two or more U files are accessed simultaneously by that subject (i.e., U + U = U).

3.2.1 Requirement analysis

By considering the first column of the Hajj's case study data table, Table 3-1, five U files are presented to be shared by the three systems (F, I, H) under the No-obligation requirement. F-system has two U files (atc, tor) that both could be accessed by I and H users as well as local F users. I-system, on the other hand, has only one U file (pln) that could be accessed by H users as well as local I users. Finally, H-system has also two U files (mis, hos), where "mis" could be accessed by users from systems F and I, while "hos" is allowed to I users only; as above, local H users may access both files.

	No-obligation/Unclassified
F-system	Haj-Attaché "*atc* "[I,H] Haj-Tourist "*tor* "[I,H]
I-system	Haj-Plan "*pln* "[H]
H-system	Haj-Missions "*mis* " [F,I] Housing "*hos* " [I]

Table 3-1: First column of data table

Now, if every system has three U users—where the first may access one remote system, the second may access another, and the third may access both—then the access-control matrix of those users with the five U files will be as in Table 3-2.

Files / Users	atc/F: I,H	tor/F: I,H	pln/I: H	mis/H: F,I	hos/H: I
f1/I	r/w	r/w	---	---	---
f2/h	r/w	r/w	---	r/w	---
f3/i,h	r/w	r/w	---	r/w	---
i1/f	r/w	r/w	r/w	---	---
i2/h	---	---	r/w	r/w	r/w
i3/f,h	r/w	r/w	r/w	r/w	r/w
h1/f	r/w	r/w	---	r/w	r/w
h2/I	---	---	r/w	r/w	r/w
h3/f,I	r/w	r/w	r/w	r/w	r/w

f1/h= F-system user f1 who can access I-system only

Table 3-2: Access-control matrix

As shown above, a U user such as "f1" who is from F-system and is supposed to access U files of I-system only, cannot access I-system's file "pln" because his local system (F) has no permission to access that file. The only files he is allowed to access are his local system's files (atc, tor). On the other hand, the user "f2" who is supposed to access H system's file, may access only "mis" file because his local system has permission to access only "mis" file (no permission is granted to access hos file). In addition, "f2" may access his local system's files (atc, tor). Similarly, the user "f3" who is supposed to access I and H systems files, can access only "mis" file from H-system as well as his local system's files (as f2).

On the other hand, since F and H-systems allow I-system to access their four files, I-system user "i1" may access only F-system's files "atc" and "tor" in addition to his local file "pln" while "i2" may access only H-system's files "hos" and "mis" in addition to "pln". Moreover, the user "i3" can access all five files.

Finally, since F and I-systems allow H-system to access their three files, then H-system user "h1" may access only F-system's files "atc" and "tor" in addition to his local files "hos" and "mis" while "h2" may access only I-system's files "pln" in addition to "hos" and "mis." As "i3" the user "h3" can access all five files.

In the above access relation between users and files, the major problem is the Trojan horse problem. For example, a user from I-system such as "i3" may access the H-system's file "hos" and since he has Read/Write right (because he is a U user), he can read "hos" file and write it to "mis" file where he has Read/Write right, too. When a user from F-system such as "f2" accesses the H-system's "mis" file, he will access the contents of that file that now has data from "mis" and "hos" files. Such a disclosure is a violation of the access control matrix because "f2" was prohibited from reading the contents of "hos" file.

3.2.2 No-obligation lattice model

In order to prevent the Trojan horse problem in this requirement, one solution could be to create a lattice model for users/subjects and objects between these systems.
Since files are not labeled with security classifications (other than U) and similarly users are not labeled with security clearances (other than U, C, S, TS), the notion of labeled subjects and objects is introduced to facilitate the creation of a lattice.

3.2.2.1 General lattice model

The lattice model for this requirement is defined as the No-obligation general lattice model, Figure 3-1. This is a standard subset lattice on five compartments (with no system low). Figure 3-1 omits the dominance relationships in the lattice to avoid cluttering the diagram. The five compartments correspond to the five U files (objects) in Table 3-1. They appear as singleton categories in the bottom row of Figure 3-1. Categories above these correspond to simultaneous access to multiple objects as indicated. A category dominates another if compartments of the former are a superset of the latter.

The No-obligation requirement is enforced by the following lattice-access rules in context of the No-obligation lattice.

1) As in the usual practice in multilevel systems, a user is allowed to have subjects at any label dominated by the user's clearance.
2) A user can only have those subjects allowed by conditions a and b of the No-obligation requirement.

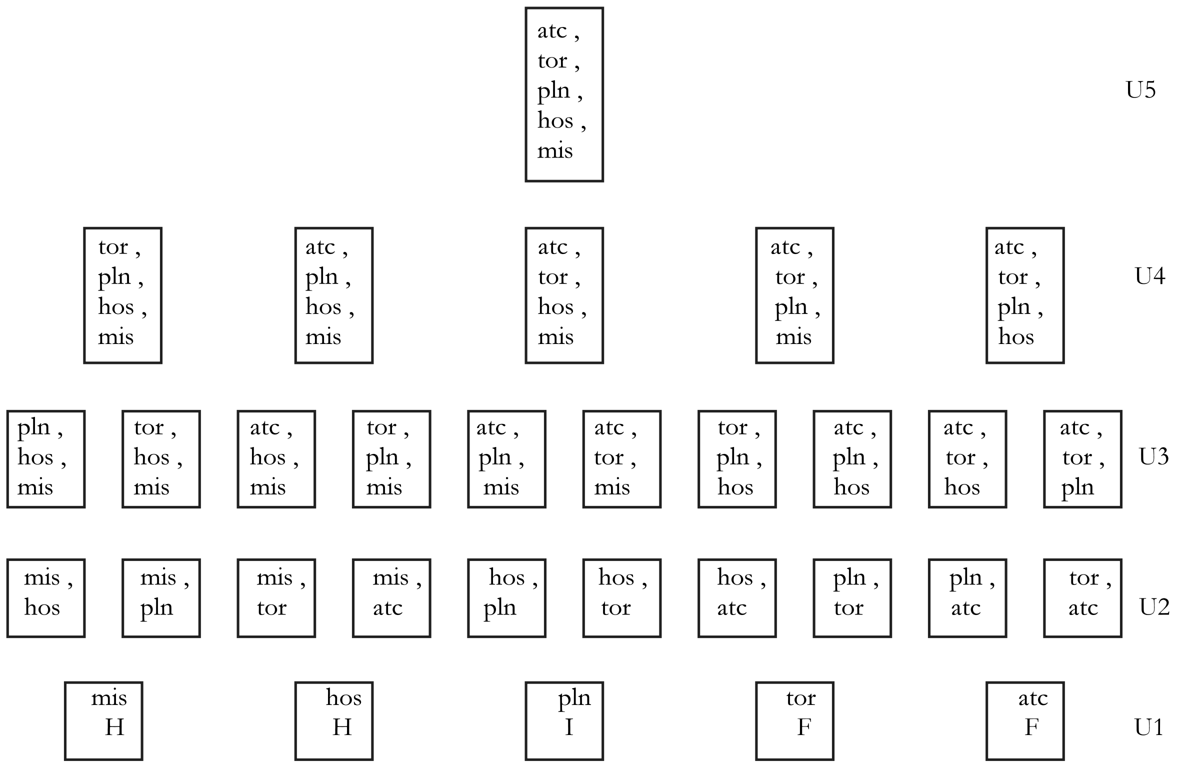


Figure 3-1: No-obligation general lattice model

Now if "i3" wants to Read/Write to "mis" file only, he must invoke his U subject **"mis"** (since he can invoke U subjects only). This subject allows him to Read/Write to the *"mis"* object in level U1. This object contains data from the "mis" file only. By using the same subject **"mis",** "i3" can access all objects in level U2 and above that include *"mis"* as part of their labels, but in this case, "i3" can Write only to those objects.
Similarly, in order for "i3" to Read "mis" and "hos" files simultaneously, he must invoke his U subject **"mis,hos".** This subject allows him to Read/Write to the object *"mis,hos"* in level U2. This object contains data from those two files. With the same subject, "i3" can Read only the objects *"mis"* and *"hos"* in level U1. In addition, he can Write only to the objects in level U3 and above that include *"mis,hos"* in their labels. In this case, those objects are: *"pln,mis,hos", "tor,mis,hos", "atc,tor,mis,hos"* etc. Thus, when "f2" accesses *"mis"* object now, he will not find any data from "hos" file.

3.2.2.2 Specific lattice model

Figure 3-1 presents a general lattice model if all five U files and their combinations are shared between the three systems. But since some systems access one file or null from others (F-system's users cannot access I-system's file), then *not* all lattice categories are going to be used. For example, no user will be accessing the object *"atc,pln,mis"* because F-system users can either access both F-system's files "atc" and "tor" (f1) or, in addition, they can access H-system's file "mis" (f2,f3), but they cannot access I-system's file "pln" while I system's users can access either all five files (i3) or only I-system's "pln" file with another system's files (i1,i2). Similarly, H-system's users can either access all five files (h3), or only H system's files " mis" and "hos" with another system's files (h1,h2).

With such a justification, only 12 categories are chosen from the 31 of the general lattice model to create the No-obligation specific lattice model. Figure 3-2 presents this new lattice model, which consists of the necessary categories that will be used between the three systems. Similarly, 12 subjects are chosen to access objects in these categories. Those subjects are available for the following users:

1. **atc**: for f1,f2,f3,i1,i3,h1,h3
2. **tor**: for f1,f2,f3,i1,i3,h1,h3
3. **pln**: for i1,i2,i3,h2,h3
4. **mis**: for f2,f3,i2,i3,h1,h2,h3
5. **hos**: for i2,i3,h1,h2,h3
6. **atc,tor**: for f1,f2,f3,i1,i3,h1,h3
7. **mis,hos**: for i2,i3,h1,h2,h3
8. **atc,tor,mis**: for f2,f3,i3,h3
9. **atc,tor,pln**: for i1,i3,h3
10. **pln,mis,hos**: for i2,i3,h2,h3
11. **atc,tor,mis,hos**: for i3,h1,h3
12. **atc,tor,pln,mis,hos**: for i3,h3

Notice that the users i3 and h3 can invoke any labeled subject to access the category with a corresponding label.

In order for a user to access a lattice category, he must invoke his necessary subject according to the lattice-access rules mentioned above (in section 3.2.2.1).

Tables 3-3, 3-4, and 3-5 present the access-control matrixes for F, I, and H-systems' users, respectively, with the subjects available to them and the access rights on the lattice categories.

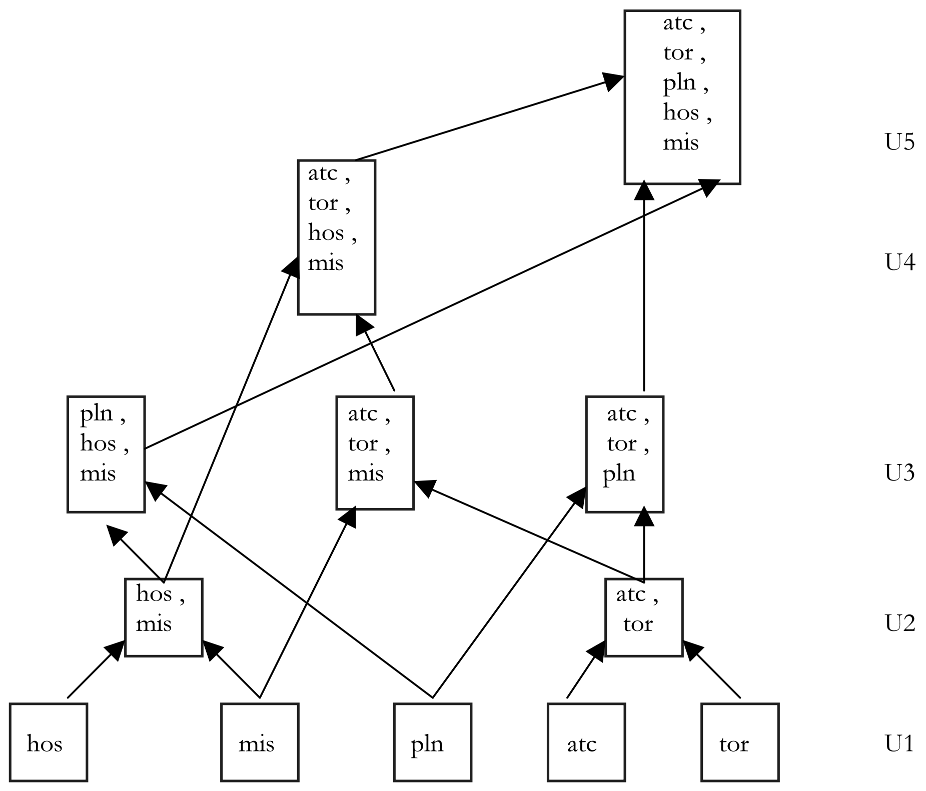


Figure 3-2: No-obligation specific lattice model

As shown in Table 3-3, only five subjects are available to F-system users. Where **"atc", "tor",** and **"atc,tor"** subjects could be invoked by all three users f1, f2, and f3, while **"mis"** and **"atc,tor,mis"** subjects could be invoked by f2 and f3 users only.

Now if f3 decides to invoke the subject **"atc,tor,mis"** for example, he can Read only the labeled objects: *"atc", "tor", "mis"*, and *"atc ,tor."* In addition, he can Read/Write to the labeled object *"atc,tor,mis"* and Write only to *"atc,tor, mis,hos"* and *"atc,tor,pln,mis,hos."* Similarly, if f3 obtains the subject **"atc"** (since he dominates it), he can Read/Write to the labeled object *"atc"* Write only to the ones labeled *"atc,tor", "act,tor,mis", "atc,tor,pln", "atc,tor, mis,hos"*, and *"atc,tor,pln,mis,hos."*

Objects / Subjects	*atc*	*tor*	*pln*	*mis*	*hos*	*atc, tor*	*mis, hos*	*atc, tor, mis*	*atc, tor, pln*	*pln, mis, hos*	*atc,tor, mis, hos*	*atc, tor, pln,mis , hos*
atc	r/w	-	-	-	-	w	-	w	w	-	w	w
tor	-	r/w	-	-	-	w	-	w	w	-	w	w
mis	-	-	-	r/w	-	-	w	w	-	w	w	w
atc,tor	r	r	-	-	-	r/w	-	w	w	-	w	w
atc,tor,mis	r	r	-	r	-	r	-	r/w	-	-	w	w

Table 3-3: Access-control matrix for F-system's users

On the other hand, in table 3-4, I-system users may access 10 subjects, where all three users may invoke the subject **"pln"** but only i3 may invoke the subject **"atc,tor, pln,mis, hos."** In addition, both users i1 and i3 may invoke the subjects **"atc", "tor", "atc,tor"** and **"atc,tor,pln"** while i2 and i3 may invoke the subjects **"mis", "hos", "mis,hos"** and **"pln,mis ,hos."** For example, when i1 obtains the subject **"atc,tor,pln"** he can Read/Write to the object labeled *"act,tor,pln"* Read only the objects *"act", "tor", "pln", "act,tor"* and Write only to the object *"act,tor,pln, mis, hos."*

Objects / Subject	*atc*	*tor*	*pln*	*mis*	*hos*	*atc, tor*	*mis, hos*	*atc, tor, mis*	*atc, tor, pln*	*pln, mis, hos*	*atc,tor, mis, hos*	*atc,tor, pln, mis, hos*
atc	r/w	-	-	-	-	w	-	w	w	-	w	w
tor	-	r/w	-	-	-	w	-	w	w	-	w	w
pln	-	-	r/w	-	-	-	-	-	w	w	-	w
mis	-	-	-	r/w	-	-	w	w	-	w	w	w
hos	-	-	-	-	r/w	-	w	-	-	w	w	w
atc,tor	r	r	-	-	-	r/w	-	w	w	-	w	w
mis,hos	-	-	-	r	r	-	r/w	-	-	w	w	w
atc,tor,pln	r	r	r	-	-	r	-	-	r/w	-	-	w
pln,mis,hos	-	-	r	r	r	-	r	-	-	r/w	-	w
atc,tor,pln,mis,hos	r	r	r	r	r	r	r	r	r	r	r	r/w

Table 3-4: Access-control matrix for I-system's users

Similarly, H-system's users may invoke 10 subjects as in Table 3-5, where all three users may invoke the subjects **"mis", "hos"** and **"mis,hos"** while only h3 may invoke the subject **"atc,tor,pln,mis,hos."**

Objects / Subject	*atc*	*tor*	*pln*	*mis*	*hos*	*atc, tor*	*mis, hos*	*atc, tor, mis*	*atc, tor, pln*	*pln, mis, hos*	*atc,tor, mis, hos*	*atc,tor, pln,mis, hos*
atc	r/w	-	-	-	-	w	-	w	w	-	w	w
tor	-	r/w	-	-	-	w	-	w	w	-	w	w
pln	-	-	r/w	-	-	-	-	-	w	w	-	w
mis	-	-	-	r/w	-	-	w	w	-	w	w	w
hos	-	-	-	-	r/w	-	w	-	-	w	w	w
atc,tor	r	r	-	-	-	r/w	-	w	w	-	w	w
mis,hos	-	-	-	r	r	-	r/w	-	-	w	w	w
pln,mis,hos	-	-	r	r	r	-	r	-	-	r/w	-	w
atc,tor,mis,hos	r	r	-	r	r	r	r	r	-	-	r/w	w
atc,tor,pln,mis,hos	r	r	r	r	r	r	r	r	r	r	r	r/w

Table 3-5: Access-control matrix for H-system's users

In addition, both users h1 and h3 may invoke the subjects **"atc", "tor", "atc,tor"** and **"atc,tor,mis,hos"** while h2 and h3 may invoke the subjects **"pln"** and **"pln,mis,hos."**

3.3 MLS requirement

As indicated in chapter two, the major reason for selecting this requirement is the increased *sensitivity of shared files* due to aggregation. Conditions a and b are identical to conditions a and b of the first requirement. Condition c is different and stipulates that confidential + confidential may equal secret, and secret + secret may equal top secret. The reason for these rules is to prevent aggregation of sensitive information.

3.3.1 Requirement analysis

The second column of the Hajj's case study data table, Table 3-6, represents the C files from systems involved. These files will be used under of the second requirement (MLS).

	Multilevel-aggregation / C,S,TS
F-system	Haj-Visas"*vis*"[I,H]+ Gov-Guest"*gus*" [I,H]
I-system	Haj-In *"hji"* [F,H]+ Haj-Out *"hjo"* [F,H]
H-system	Guide-Est "*gde*" [I] + Arrival&departure*"a&d"*[F,I]

Table 3-6: Second column of data table

As shown above, every system has two files to share (total of six). Each file is individually labeled as C. The "+" sign between any two C files indicates that the combination of those two files will be S. Similarly a combination of S combinations from two or more different systems will be TS. For example, the combination of the C files "Haj-Visas" and "Gov-Guest" from F-system is labeled S, while the combination of the two S combinations "Haj-Visas+Gov-Guest" from F-system and "Haj-In+Haj-Out" from I-system is labeled TS because each one belongs to a different system.

These files and their combinations are available to be shared between all three systems except H-system's file "Guide-Est" (and other combinations that involved this file), which is available for H and I-systems only. For example, F, I, and H users can access the H system's file "Arrival&departure" but only I and H users can access the H-system's file "Guide-Est". Thus, the combination "Gov-Guest,Haj-Out,Arrival&departure" will be accessed by users from F, I, and H-systems, while the combination "Gov-Guest,Haj-Out, Guide-Est" will be accessed by users from I and H-systems only.

The combinations of two or three C files where each is from a different system will be C, too. Similarly, the combinations of a S combination of a system with one C file or two, where each is from one of the other systems, is also S. Finally, the combinations of a TS combination with one C file from the third system is TS too. For example, the combinations "Haj-Visas,Haj-In", "Haj-Visas,Haj-In,Guide-Est" and "Gov-Guest,Haj-Out, Arrival& departure" are labeled C because each combination has, at most, one file from every system. Similarly, combinations like "Haj-Visas+Gov-Guest,Haj-In" and "Haj-Visas+Gov-Guest, Haj-In,Guide-Est" are labeled S because each combination has, at most, one S combination (+) from one system and one file from each other systems. Likewise, combinations such as "Haj-Visas+Gov-Guest, Haj-In+Haj-Out,Guide-Est" and "Haj-Visas +Gov-Guest,Hja-In+Haj-Out,Arrival& departure" are labeled TS because each one has minimum of two S combinations from different systems.

By analyzing this requirement, it is obvious that a lattice model is needed that specifies information flow between objects. Compartments of this lattice represent combinations of the six files (objects) that need to be shared under this requirement. In addition, an access-control matrix specifies access rights available for subjects to be used on the lattice's objects. Finally, attachments of those subjects to the involved users from the three systems are also needed.

3.3.2 MLS lattice model

In order to support the second requirement, suppose that every system has three types of users as follows:

- C users: where each can access one C file or a combination that
  holds two or three C files (one from each system).
- S users: where each can access one S combination that holds at least two
  C files of a system.
- TS users: where each can access one TS combination that holds at
  least two S files.

The lattice model for MLS will indicates the compartments that represent the six C files and their combinations. Figure 3-3 presents the lattice model for the second requirement, where abbreviations of the six files are used. It is very similar to a subset lattice on six compartments. There are $2^6 - 1 = 63$ compartments (omitting the empty one). The major difference between this lattice and the one in Figure 3-1 (no-obligation) is that all compartments there are U, while here some of the compartments (at L1C, L2C and L3C) are C, others (at L2S, L3S and L4S) are S, while the rest (at L4TS, L5TS and L6TS) are TS. This follows from condition c of the second requirement that certain combinations of files have a higher security level than their individual constituents as per the following rules:

I. a combination of two C files from a single system is S

II. a combination of two S files from two or more system is TS

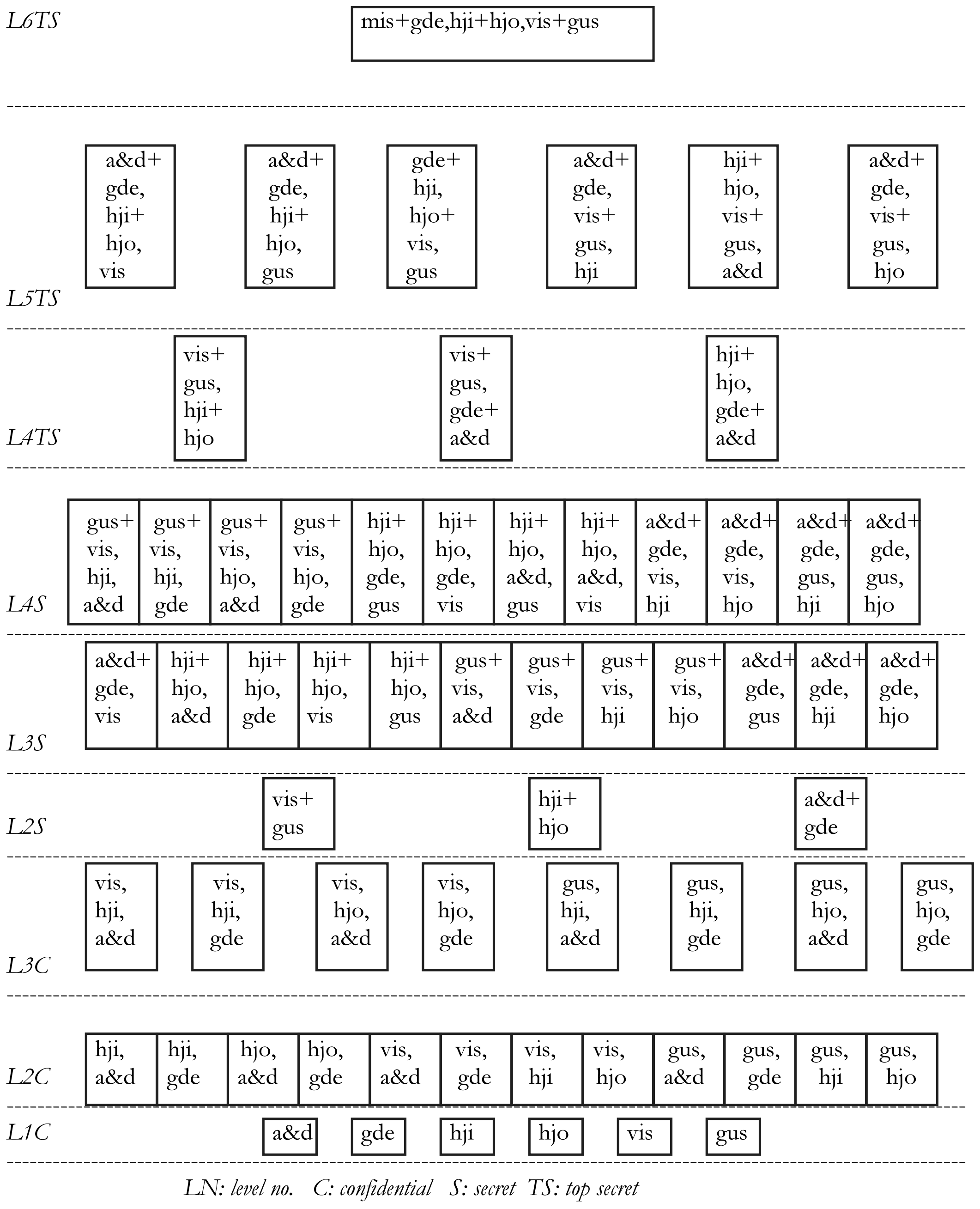


Figure 3-3: The lattice model for MLS

Information flow and category dominance are the same as in the previous lattice of Figure 3-1, that is, a category dominates another if compartments of the former are a superset of the latter. To emphasize the labeling rule for combinations of compartments it is shown that the lattice is going up from C to S to TS compartments. Note that all compartments of size four are S and all of size 5 or 6 are TS. Compartments of size 2 or 3 can be S or C. Nevertheless the information flow is exactly that determined by the superset relationship.

For example, a category at level L3C such as: "vis,hji,a&d" dominates the ones that have a subset of it as "vis,hji" and "vis" at levels L2C and L1C respectively. Similarly, a category at level L2S, such as: "vis+gus" dominates the ones that have a subset of it as "vis" and "gus" at level L1C only. This means that information may flow from the C category "vis" at L1C to the C categories "vis,hji" at L2C, "vis,hji,a&d" at L3C, and to the S category "Vis+gus" at L2S. Figure 3-4 shows the dominance and information flow relations between levels of the MLS lattice.

$$\begin{tikzcd}
 & \text{L6TS} & \\
 & \text{L5TS} \arrow[u] & \\
\text{L4S} \arrow[ur] & \text{L4TS} \arrow[u] & \\
\text{L3S} \arrow[u] \arrow[ur] & & \text{L3C} \arrow[ull] \arrow[ul] \\
\text{L2S} \arrow[u] & & \text{L2C} \arrow[u] \arrow[ull] \\
 & \text{L1C} \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

L3C: L = level, 3 = no. of files a category in this level may have
C = the security class of categories in this level

Figure 3-4: The dominance and information flow of the MLS lattice.

Next, it is important to define an access-control matrix that specifies the access rights for subjects on objects. Table 3-7 shows *some* of the C, S, and TS subjects/objects access relations. For example, the subject "**vis,hji**" is labeled C because it can Read/Write to the C object labeled "*vis,hji*" Read only the C objects labeled "*vis*", "*hji*" and Write only to the S object labeled "*vis+gus,hji,a&d*" the TS object labeled "*vis+gus,hji+hjo.*"

Objects / Subjects	*vis* /c	*gus* /c	*hji* /c	*hjo* /c	*gde* /c	*a&d* /c	*vis, hji*/c	*gus,hjo, a&d*/c	*vis+ gus*/s	*vis+gus, hji,a&d*/s	*vis+gus, hji+hjo*/ts
vis/c	r/w	-	-	-	-	-	w	-	w	w	w
vis,hji/c	r	-	r	-	-	-	r/w	-	-	w	w
gus,hjo, a&d/c	-	r	-	r	-	r	-	r/w	-	-	-
vis+gus/s	r	r	-	-	-	-	-	-	r/w	w	w
vis+gus, hji,a&d/s	r	r	r	-	-	r	r	-	r	r/w	-
vis+gus, hji+hjo/ts	r	r	r	r	-	-	r	-	r	-	r/w

c =confidential, s =secret, ts =top secret
r/w = Read and Write operations

Table 3-7: Part of the general Access-Control Matrix
for confidential, secret, and top secret subjects/objects

Finally, by attaching labeled subjects discussed above to users involved, the second requirement is fully satisfied. For example, if F-system has the following C, S, and TS users:

f11: is confidential to Read/Write the file "vis."
f12: is confidential to Read/Write the files "vis" and "hji."
f13: is confidential to Read/Write the files "gus", "hjo" and "a&d."
f21: is secret to Read/Write the files "vis" and "gus"
f22: is secret to Read/Write the files "vis", "gus", "hji" and "a&d."
f31: is TS to Read/Write the files "vis", "gus", "hji" and "hjo"

then Table 3-8 represents labeled subjects from Table 3-7 that could be invoked by those users. Notice that users such as f12 are supposed to invoke the subjects: **"vis", "hji"** and

"vis,hji" but since the subject **"hji"** is not included in Table 3-7, it is omitted from Table 3-8.

Users	Subjects
f11/c *	"vis"
f12/c *	"vis" "vis,hji"
f13/c	"gus,hjo,a&d"
f21/s	"vis", "vis+gus"
f22/s	"vis", "vis,hji", "vis+gus", "vis+gus,hji,a&d"
f31/ts	"vis", "vis,hji", "vis+gus", "vis+gus,hji+hjo"

* Users and subjects have one-to-many relationship, i.e., the subject "vis" that belongs to the user f11 differs from the subject "vis" that belongs to the user f12.

Table 3-8: Relations of some F-system users with subjects to invoke

By reviewing Table 3-6, if every system wants to declare *one* user who can access *a certain lattice compartment* with Read/Write right, then the total number of users every system may have is equal to the total number of compartments it is allowed to access.

For example, since I and H-systems can access all six C files and their combinations, then each system may have 63 users divided as follows:

26 C users divided as:
- 6 for the 6 compartments at L1C
- 12 for the 12 compartments at L2C
- 8 for the 8 compartments at L3C

27 S users divided as:
- 3 for the 3 compartments at L2S
- 12 for the 12 compartments at L3S
- 12 for the 12 compartments at L4S

10 TS users divided as:
- 3 for the 3 compartments at L4TS
- 6 for the 6 compartments at L5TS
- 1 for the only compartment at L6TS

But since F-system can access only five C files and their combinations (excluding gde and its combinations), then F-system may have only 31 (2^5 - 1) users divided as follows:

17 C users divided as:

- 5 from the 6 compartments at L1C
- 8 from the 12 compartments at L2C
- 4 from the 8 compartments at L3C

12 S users divided as:

- 2 from the 3 compartments at L2S
- 6 from the 12 compartments at L3S
- 4 from the 12 compartments at L4S

2 TS users divided as:

- 1 from the 3 compartments at L4TS
- 1 from the 6 compartments at L5TS

The only difference between this requirement (MLS) and the first one (No-obligation) is that objects here may have different security classifications (C, S, and TS), while in the first requirement, all objects there share the same security classification (U). Thus, it is not enough here for a user to satisfy conditions a and b in order to access all files of a system (as in no-obligation), but also condition c must be satisfied to enforce MLS. For example, in the first requirement, a U user such as f1, who accesses his local F-system only, may invoke the subjects **"atc"** and **"tor".** These subjects allow him to Read/Write to the objects *"atc"* and *"tor."* However, in this requirement, a C user such as f11, who accesses only his local F-system, may invoke only one subject **"vis"** which allows him to Read/Write to the C object *"vis."* Only an S user such as f21, f22, or f31 may invoke the subjects: **"vis**" and **"gus"** which allows him to Read/Write to both C objects *"vis"* and *"gus."*

3.4 Chinese Walls requirement

The third requirement deals with sharing special and very sensitive files by special users, such that a user can access only one file from each system. Moreover, these special users are not allowed to access files under the first or second requirements. Furthermore, TS users are exempted from this rule, because they are sufficiently trusted.

As shown in chapter two, conditions a and b are identical to conditions a and b of the previous two requirements, but condition c is very different. The main reason for selecting this requirement and introducing condition c is to prevent leakage and misuse of sensitive data. For example, some systems are using sensitive files that store common data (i.e., total number of pilgrimages), but each file may record it differently. Thus, these systems may identify these files as special sensitive files to check the credibility and accuracy of their data and sources. Hence, if a user has the privilege to access a combination of those files from the same system, he may then tamper with the common data in order to make it match in all files, which results in misleading information to higher management. Therefore, this requirement will prevent users from such acts and ensure credibility and accuracy of these files.

3.4.1 Requirement analysis

The third column of the Hajj's case study data table, Table 3-9, represents the special files (Chinese Walls files) for the third requirement and the systems that can access them. Each system has two Chinese Walls files such that a *special* user who is not a TS user can access only one of them.

	Chinese Walls
F-system	Haj- Deplom*"dpl"* [I,H] Secret-Comm *"scm"* [I,H]
I-system	Black List "*bkl*" [F] Non-Saudi in List *"nsl"* [F]
H-system	Haj-Registration"*rgs*"[F,I] Transport *"trn"* [I]

Table 3-9: Third column of case study data table

For example, the I-system requires that any special user (non-TS) from the F-system (or I-System) who has access to the "Black-List" file of the I-system should not have access to the I-system's file "Non-Saudi in List." Otherwise, users (lower than TS) who access the "Black-List" file might modify records in the "Non-Saudi in List" file, which have data about a blacklisted person. This means that any special user may access a maximum of three special files at one time, one file from each system.

As shown above, the Chinese Walls files for F-system are "dpl", "scm", while " bkl", "nsl" are for I-system, and finally "rgs", "trn" are for H-system.

The major problem here is the Trojan horse problem. For example, if the F-system identified two special users, say fs25 and fs26, who can access the following files with Read/Write:

fs25 access "dpl" and "bkl"
fs26 access "scm" and "bkl"

then fs25 can read data from "dpl" and write it to "bkl." Later, when fs26 accesses "bkl" he will read "dpl" data that exist in "bkl" now. Such an access is a violation of the third requirement, because fs26 is accessing data from "dpl" and "scm" simultaneously (both files from F-system). In order to prevent such an access violation, this requirement should be enforced by the use of a Chinese Walls lattice-based model as discussed in [2] to prevent aggregation of information.

3.4.2 Chinese Walls lattice model

In order to create a Chinese Walls lattice model, it is important to determine the number of data sets. As indicated in Table 3-9, every system has one data set that contains two Chinese Walls files. Therefore, objects with three-vector labels will satisfy the Chinese Walls policy. For example, the object labeled *[dpl,bkl,ϕ]* contains data from F-system file "dpl" I-system file "bkl" and null from H-system. Objects with four-vector labels or more are exempted from the Chinese Walls policy and considered as SYSHIGH. Figure 3-5 presents the lattice model for the Chinese Walls objects and their combinations as concluded from Table 3-9.

```
          [dpl,scm,rgs,trn]      [dpl,scm,bkl,nsl,rgs]      [dpl,scm,bkl,nsl,rgs,trn]
                 H                         F                            I
               CHN4,                     CHN5,                        CHN6        TS=Sys-High
-----------------------------------------------------------------------------------------------
                  [dpl,bkl,rgs]   [dpl,bkl,trn]   [ dpl,nsl,rgs ]   [dpl,nsl,trn]
                  [scm,bkl,rgs]   [scm,bkl,trn]   [scm,nsl,rgs ]   [scm,nsl,trn]
CHN3
-----------------------------------------------------------------------------------------------
        [dpl,bkl,ϕ] [dpl,nsl,ϕ] [dpl,ϕ,rgs] [dpl,ϕ,trn] [scm,bkl,ϕ] [scm,nsl,ϕ]
        [scm,ϕ,rgs] [scm,ϕ,trn] [ϕ,bkl,rgs] [ϕ,bkl,trn] [ϕ,nsl,rgs] [ϕ,nsl,trn]
CHN2
-----------------------------------------------------------------------------------------------
        [ dpl,ϕ,ϕ ]   [ scm,ϕ,ϕ ]   [ ϕ,bkl,ϕ ]   [ ϕ,nsl,ϕ ]   [ ϕ,ϕ,rgs ]   [ϕ,ϕ,trn]
          F,I,H          F,I,H           F,I            F,I           F,I,H         I,H
CHN1
-----------------------------------------------------------------------------------------------
```

Figure 3-5: The lattice model for Chinese Walls

There are six different levels in the lattice presented as CHNm, where m stands for that level number. For example, the level CHN2 is higher than CHN1 and lower than CHN3 to CHN6. It indicates that a subject with CHN2 level can Read/Write to an object at CHN2 level Read down and Write up, depending on its label. In addition, an object at CHN2 holds data from two files, while an object at CHN1 holds data from one file only. Hence, only objects from CHN1 to CHN3 satisfy the third requirement, while the ones at CHN4 and up are considered as SYSHIGH. Therefore, only three security classes are needed here: CHN1, CHN2, and CHN3 for users/subjects and objects. Objects at CHN4 and up are labeled TS and accessed by TS users/subjects.

Now if F-system has some users who need to access some files as follows:

f11: is a special user who should access "dpl"
f12: is a special user who should access "scm"
f13: is a special user who should access "bkl"
f25: is a special user who should access "dpl" and "bkl"
f26: is a special user who should access "scm" and "bkl"
f35: is a special user who should access "dpl", "bkl", and "rgs"

then the access-control matrix must be created that specifies the relation between those users' subjects and these files (objects). Table 3-10 presents such a matrix, where there are six subjects and objects. As shown each column represents an object that was presented in the lattice model above. In addition, every row represents a subject that can access some objects with certain access rights.

Objects / Subjects	*[dpl,ϕ,ϕ]*	*[scm,ϕ,ϕ]*	*[ϕ,bkl,ϕ]*	*[dpl,bkl, ϕ]*	*[scm,bkl,ϕ]*	*[dpl,bkl,rgs]*
[dpl,ϕ,ϕ]	r/w	-	-	w	-	w
[scm,ϕ,ϕ]	-	r/w	-	-	w	-
[ϕ,bkl,ϕ]	-	-	r/w	w	w	w
[dpl,bkl,ϕ]	r	-	r	r/w	-	w
[scm,bkl,ϕ]	-	r	r	-	r/w	-
[dpl,bkl,rgs]	r	-	r	r	-	r/w

Table 3-10: Access-control matrix

Now, a subject whose label is **[dpl,bkl,ϕ]** may Read/Write to the object *[dpl,bkl,ϕ]* in CHN2 level, Read the objects whose labels are *[dpl,ϕ,ϕ], [ϕ,bkl,ϕ]* in CHN1 level, Write to the objects whose labels are *[dpl,bkl,rgs]* and *[dpl,bkl,trn]* in CHN3 level, and Write to all objects in CHN4 and up have *"dpl"* and *"bkl."*

Notice that the object *[dpl,bkl,trn]* and other objects at CHN4 and up are not included in the access-control matrix.

In order to attach Chinese Walls users from F-system to subjects discussed above in the matrix, the following relations between users and subjects will be created:

f11: may invoke the subject **[dpl,ϕ,ϕ]** only.
f12: may invoke the subject **[scm,ϕ,ϕ]** only.
f13: may invoke the subject **[ϕ,bkl,ϕ]** only.
f25: may invoke the subjects **[dpl,ϕ,ϕ,], [ϕ,bkl,ϕ]**, and **[dpl,bkl,ϕ]**.
f26: may invoke the subjects **[scm,ϕ,ϕ,], [ϕ,bkl,ϕ],** and **[scm,bkl,ϕ].**
f35: may invoke the subjects **[dpl,ϕ,ϕ,], [ϕ,bkl,ϕ], [scm,bkl,ϕ],** and **[dpl,bkl,rgs].**

For example, the user f25 may invoke subject **[dpl,bkl,ϕ]** where he can Read/Write to the object *[dpl,bkl,ϕ]* , Read only the objects *[dpl,ϕ,ϕ]* and *[ϕ,bkl,ϕ]* , and Write only to the object *[dpl,bkl,rgs]*. Notice that an F-system user cannot access the object *[dpl,bkl,trn],* since F-system has no permission to access "trn" file.

3.5 Information flow policies for the Hajj network

From the above requirements, there are two types of shared data: one is "Multi", which deals with U, C, S, and TS files, while another is "Chin", which deals with Chinese Walls files. Thus, it is appropriate to have two types of users: *Multi,* who may access some or all Multi files (as TS users) and *Chin,* who may access some Chin files [17]. Hence users of type *Multi* will deal with categories of the lattices created for the first or the second requirements, where they can access (U) files for the first and (C, S, TS) files for the second. Users of type *Chin* will deal with categories of the lattice created for the third requirement, where they can access Chinese Walls files only (except for TS users who can access files of all three requirements).

By analyzing each system according to Table 2-1(a) and the above lattices, it is possible to create the information flow policy for each system. Figures 3-6, 3-7, and 3-8 show the information flow policies for F, I, and H-systems, respectively. For example, Figure 3-6 shows the access of F-system's users/subjects to those local/remote categories with Read/Write as follows:

1- All subjects that belong to multilevel users: U, C, S, TS have access to the following U files and their combinations with different rights depending on the subject's security class and its label: *"atc"* and *"tor"* (from F-system) and *"mis"* (from H-system).

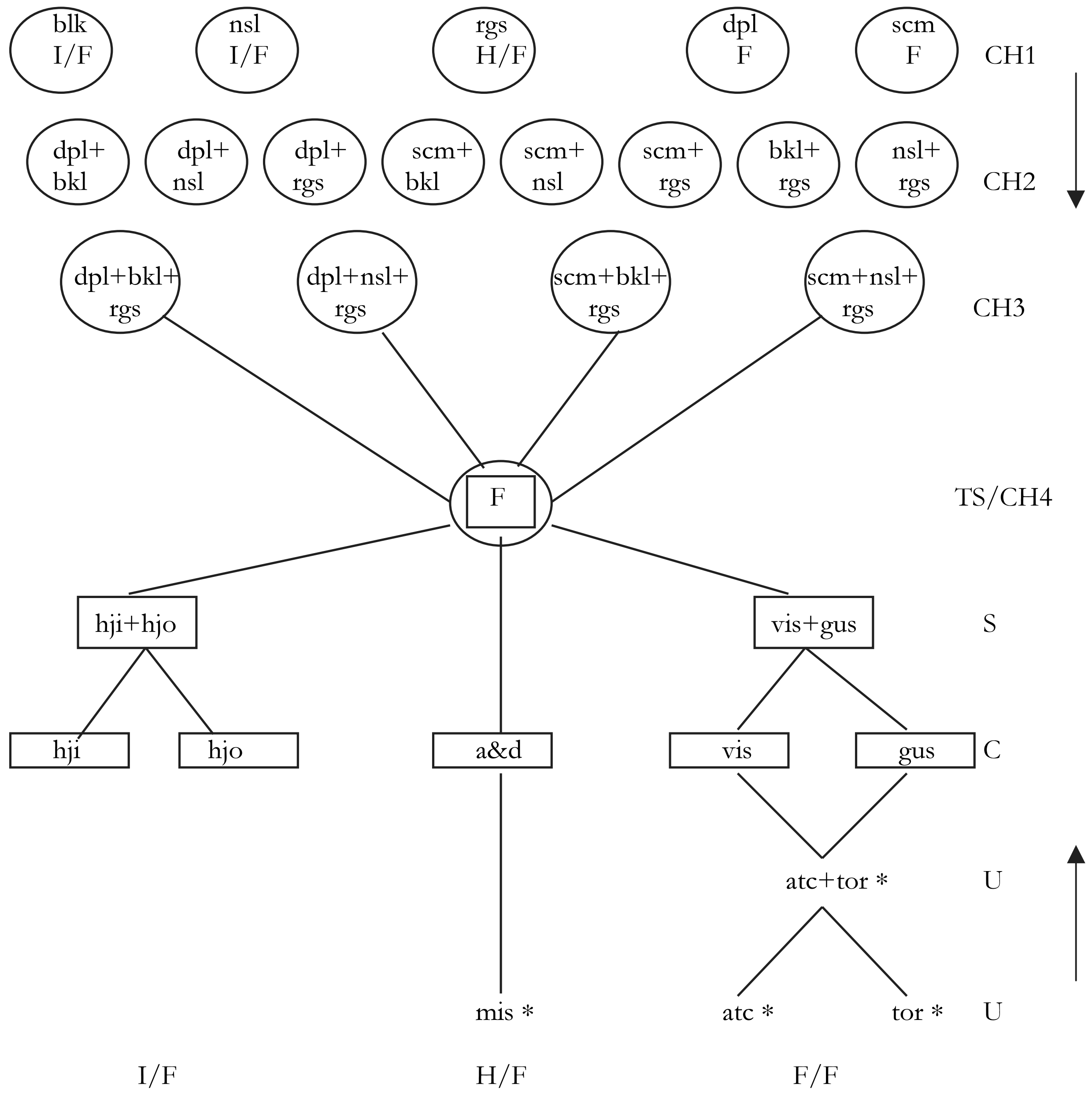
2- A C subject that belongs to a multilevel user: C, or S, or TS may access one or null local C category: *"vis"* or *"gus"* (from F-system). It may also access the single C category *"a&d"* (from H-system) and one of *"hji"* or *"hjo"* (from I-system).

3- An S subject that belongs to a multilevel user: S or TS may access one S category from each system. In this case, we have *"vis+gus"* from F-system and *"hji+hjo"* from I-system, while there is no S category from H-system.

4- A Chinese Walls subject that belongs to a special user may access Chinese Walls categories according to the subject's label.

5- A TS subject that belongs to a multilevel user: TS may access all S categories and below, as well as the Chinese Walls files *"bkl"*, *"nsl"*, *"rgs"*, *"dpl"*, *"scm"* and all combinations.

In Figure 3-6, the system high TS class is shown in the center of the diagram to make the diagram clearer. As before, we have omitted the dominance lines in the Chinese Walls categories from CHN1 through CHN3 to simplify the figure.



*=Unclassified file, = Confidential, Secret, Top Secret file, O = Chinese Wall file, ↑ = information can flow, I/F = I-system file accessed by F-system users.

Figure 3-6: F-system's information flow policy

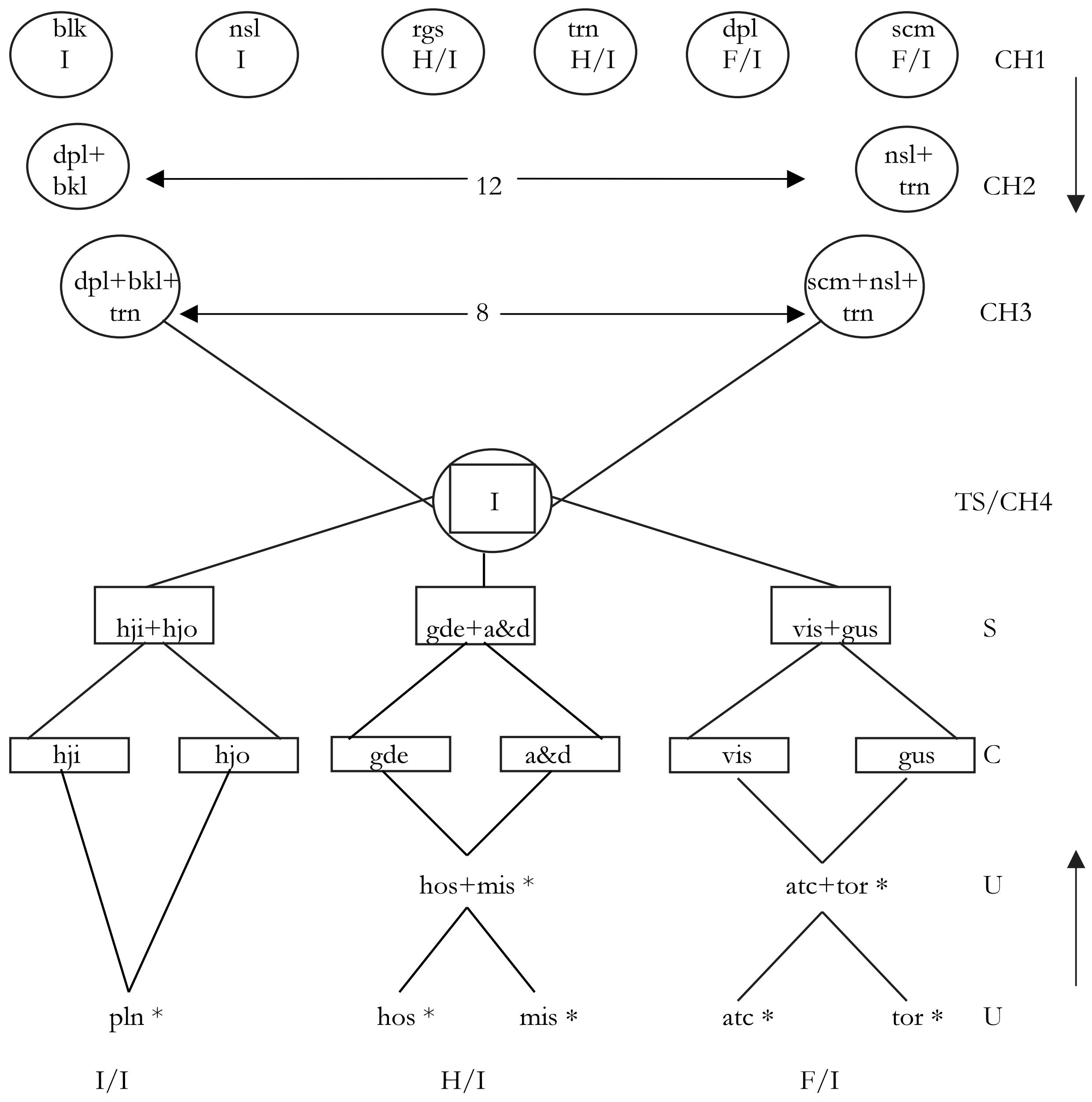


Figure 3-7: I-system's information flow policy

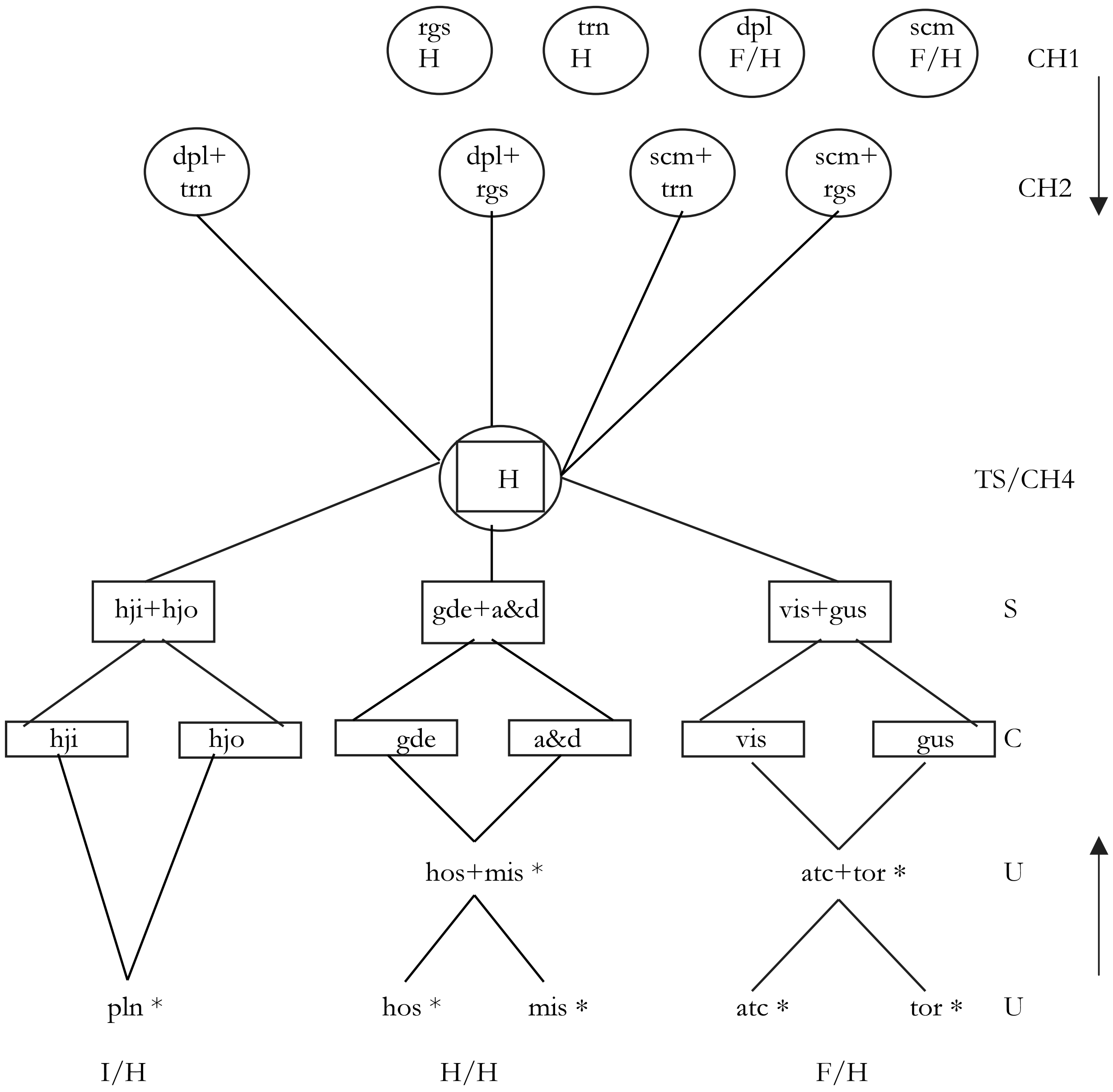


Figure 3-8: H-system's information flow policy

3.6 Conclusion

In this chapter the three confidentiality requirements for the Hajj system are analyzed. To solve the Trojan Horse problem in the first requirement, a lattice model was introduced as a solution. Obviously a lattice model was needed for the second requirement to accommodate multi level security. Finally to support the Chinese Walls model in the third requirement, a lattice model was also introduced as discussed in [10]. As a result, the common denominator between all three requirements is a lattice model. At the end of the chapter the information flow policies for Hajj network were examined. Those polices specify for each system the information flow under the three security requirements. Hence it is possible to represent confidentiality requirements (pertaining to information flow) in complex data sharing situations by lattice models.

4. A scaleable DGSA scheme: Object-Tag

4.1 Introduction

DGSA was designed to support data sharing/data transfer between distributed heterogeneous systems. The basic idea of DGSA is the creation of information domains between end systems to enable controlled sharing of data [18]. Each domain holds information objects that represent files that need to be shared or transferred between those systems. Users from the end systems are the domain's members who are allowed to access the domain's information objects depending on the domain's security policy. DGSA is described in further detail in Appendix A.

DGSA is considered a powerful architecture for M-D.Os. On the other hand, in its pure form, it fails to guarantee support for confidentiality requirements that use lattice models. This chapter analyzes reasons that prevent a pure DGSA scheme from supporting lattice models and presents two alternative solutions to this problem. One of those solutions is Hilborn's scheme. But this scheme is not scaleable with respect to the size of the lattice. However another solution is developed in this chapter called "Object-Tag scheme" that is scaleable with respect to the size of the lattice.

This chapter is organized as follows. Section 4.2 outlines DGSA and its rules, then introduces a lattice model in a simple MLS example. DGSA problems with regard to the lattice model of the example are presented in this section. At the end of the section, Hilborn's scheme and its problems are introduced. Section 4.3 introduces an extended DGSA scheme "Object-Tag scheme" which solves the problems of supporting lattice models as well as the scalability problem in Hilborn's scheme. With this scheme, an algorithm is developed which converts any partially ordered set (including a lattice) to this scheme. Section 4.4 examines the new scheme with the lattice models of the three confidentiality requirements in the Hajj system. This chapter is concluded in section 4.5.

4.2 Deficiencies of DGSA

Section 4.2.1 contains a brief description about DGSA and its rules. Then a simple MLS example is presented in section 4.2.2. The reasons that caused DGSA to fail in supporting the presented example are discussed in section 4.2.3. Finally, Hilborn's scheme and its problems are discussed in section 4.2.4.

4.2.1 DGSA

The DOD developed a security architecture (DGSA) to simplify data sharing/data transfer between its heterogeneous systems by using common information domains. An information domain [D = (O, M, P)] combines the following (see Appendix A):

- A set of (files, programs) *information Objects* (O) identifiable as belonging to the domain.
- A set of (human) *members* (M) of the domain.
- An information domain *security policy* (P) that includes:
  - the requirements for membership
  - the rules of access by members to information objects of the domain

- the rules of import and export of information from/to other information domains
- the required protection of the information objects of the domain.

The following summarizes the details of the security policy of an information domain:

a- All information objects in an information domain have identical security attributes.

b- All members of an information domain need not have equal access to its information objects (Read/Write/Read&Write).

c- A given member has identical access rights to all information objects in an information domain (Read all objects/write to all objects/Read&Write to all objects).

d- The security policy of a domain determines for a member the operations (create, remove, process, transfer, collect) and the information objects on which to perform those operations.

e- No information object belongs to more than one information domain.

f- Individuals may be members of more than one information domain.

g- Transfer of information between domains occurs only in accordance with the policies of both the *exporting* and *importing* domains (one-way or bi-directional transfer).

h- Transfer of information between information domains can be accomplished only by a member of both the exporting and importing domains.

i- Protection requirements for an information domain are stated independently of any other information domains.

Figure 4-1 shows DGSA structure of three systems (A, B, and C) that need to share data between themselves. Every system determines its own files to share with one or more other systems *and* users who should have access to those files. As illustrated in the figure, four different information domains are used between those systems for data sharing/data transfer (number of domains could be more or less, depending on many factors such as number of files and types of security policies). Each information domain has a set of information objects (files) that will be shared between any two systems or all three. In addition, each domain has a set of members (users from these systems) and a security policy that specifies types of access rights (Read or Write) and permissions of operations for the members to use (Transfer: Export or Import). For convenience, each user’s name will have the system's name as a prefix. For example, A1 and A2 are users from system A. B1 and B2 are users from system B etc. Also each domain is named after the systems involved, where only users from these systems are the domain's members.

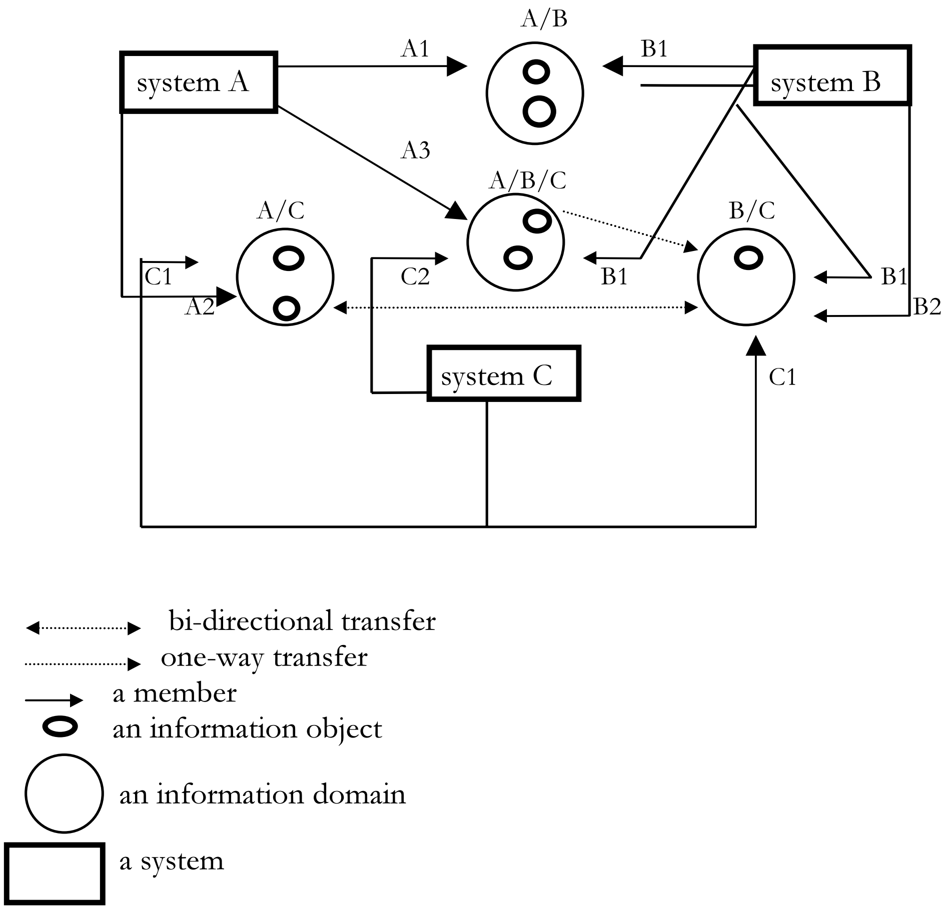


Figure 4-1: DGSA structure

For instance, the domain that contains files from A and B systems is named A/B domain, and only users from A system (such as A1) and B system (such as B1) are the members in this domain.

A common member (such as B1) may transfer information between domains where he is a member (such as A/B and B/C). According to rules "g" and "h" of DGSA, the type of transfer (one way or bi-directional) is determined by the security policies of the source and the destination domains. For example, a common member such as C1 may perform a bi-directional transfer between the domains A/C and B/C. While another common member such as B1 may perform a one-way transfer from A/B/C domain to B/C domain. As indicated in rule "g", in order for a common member to transfer information between the two domains, he must has an Export right for that information in the source domain and an Import right for the same information in the destination domain.

4.2.2 MLS example

To develop an example about distributed systems that need to establish MLS on shared data, suppose there are three systems (A, B, C) that need to share the following three files and their combinations: f1 (from system A), f2 (from system B), and f3 (from system C). Each system is applying the conventional security classes for its subjects and objects. The security classifications of f1, f2, and f3 are C. When any two of those files are combined, the security classification of the new file which contains such a combination is S. In addition, a TS classification is given to the file which contains all S combinations.

Figure 4-2 shows the lattice model for this example (a subset lattice of three compartments), where there are eight categories: one U (for pub object), three C (for f1/f2/f3 objects), three S (for f1,f2/f1,f3/f2,f3 objects), and one TS (for f1,f2,f3 object).

Table 4-1 shows the example's access-control matrix that contains subjects (rows), objects (columns), and access rights each subject has to perform on each object.

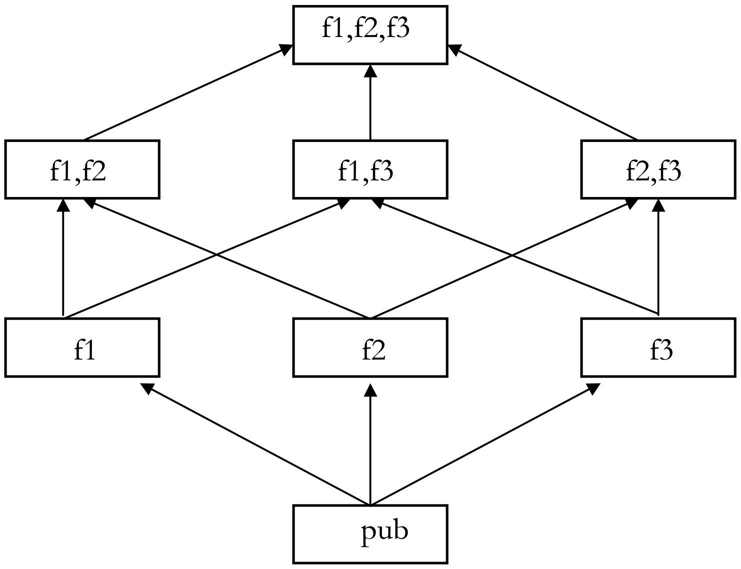


Figure 4-2: MLS lattice for the example

Objects / Subjects	pub/u	f1/c	f2/c	f3/c	f1,f2/s	f1,f3/s	f2,f3/s	f1,f2,f3/ts
pub/u	rw	w	w	w	w	w	w	w
f1/c	r	rw	-	-	w	w	-	w
f2/c	r	-	rw	-	w	-	w	w
f3/c	r	-	-	rw	-	w	w	w
f1,f2/s	r	r	r	-	rw	-	-	w
f1,f3/s	r	r	-	r	-	rw	-	w
f2,f3/s	r	-	r	r	-	-	rw	w
f1,f2,f3/ts	r	r	r	r	r	r	r	rw

u: unclassified , c: confidential , s: secret , ts: top secret , r: read , w: write , -: null

Table 4-1: Access-control matrix for MLS example

Every subject is labeled with an equivalent label to an object's, where it can Read/Write to it, Read down, and Write up.

A user from any system who wishes to Read or Write to any object in the lattice must be able to invoke the subject that allows him to access that object with such a right. For example, if system A has the following users:

Ac1 = who should be able to Read/Write to the local file "f1" only, and

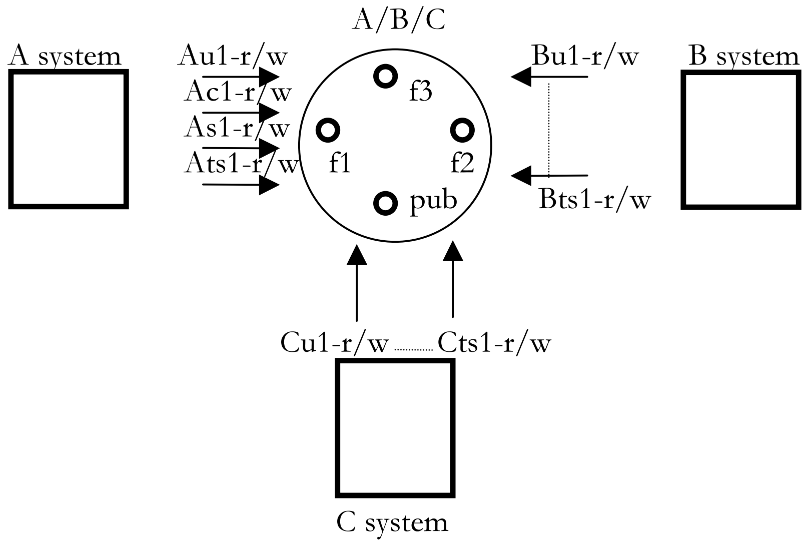
Ac2 = who should be able to Read/Write to the remote file "f2" only,

then Ac1 must have a subject similar to f1/c, while Ac2 must have one similar to f2/c.

4.2.3 DGSA problems

When applying DGSA to the above example, different designs could be accomplished. One design could be in creating one large information domain between the three systems. This domain contains all shared files as its information objects, and users from these systems are its members. Figure 4-3 shows this common domain (A/B/C) with its members and information objects.

Obviously, such an illustration cannot apply MLS because of *the violations in the security of the lattice model*. For example, the U member "Au1" can now access U, C, S, and TS files in the domain with Read and Write rights (according to DGSA rule c), while he is supposed to access only the U file "pub" with these two rights and can access the rest of the files with Write-only right.



Ac1-r/w= a confidential user from A system "Ac1" who can login with Read/Write right.

Figure 4-3: One way to implement the example

Another design could be in creating one domain for each lattice category, where every file (object) in each category becomes the only information object in a specific domain. Only users who can access that category with their rights become the domain's members with the same access rights. Moreover, each domain will be named after the file it contains. Figure 4-4 illustrates such a design, where four domains are created between system A and system B. For simplicity, only four files that represent the four categories: "pub", "f1", "f2", and "f1,f2" are used in these domains, where each is stored in a specific domain as its information object. Users from both systems who can access these categories are the members in these domains. Each domain's member has certain access rights depending on his security clearance and the above access-control matrix. For example, the C member "Ac1" may login to the "f1" domain with a Read/Write right, while he can login to the "pub" domain with a Read-only right and to the "f1,f2" domain with Write-only right.

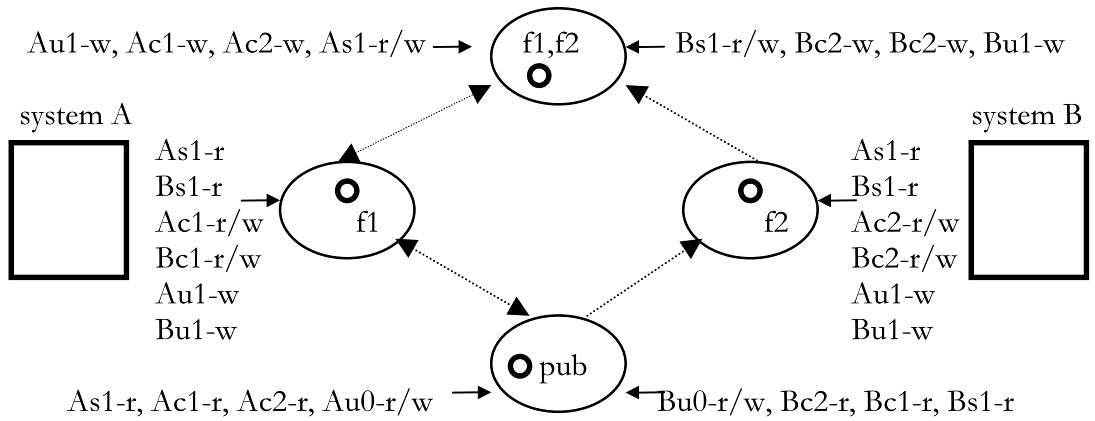

Au1-w, Ac1-w, Ac2-w, As1-r/w
f1,f2
Bs1-r/w, Bc2-w, Bc2-w, Bu1-w
system A
system B
As1-r
Bs1-r
Ac1-r/w
Bc1-r/w
Au1-w
Bu1-w
f1
f2
As1-r
Bs1-r
Ac2-r/w
Bc2-r/w
Au1-w
Bu1-w
pub
As1-r, Ac1-r, Ac2-r, Au0-r/w
Bu0-r/w, Bc2-r, Bc1-r, Bs1-r

Figure 4-4: Another way to implement the example

Two major problems are raised with this design:

1. Since members are human users, then there is no User * Subject relation.
2. Bi-directional transfer may result in a disclosure violation.

An example of the first problem is when a S user such as "As1" can only login to each domain with one uniform access. He can only login to the domain "f1, f2" with Read/Write right and the domains: "pub", "f1", and "f2" with Read-only right. But he cannot login to the domain "f1" with Read/Write right (by using a C subject for instance). This is happens because DGSA has not specified subjects for human users (members). Such a requirement is very important in the usual practice of MLS.

An example of the second problem is when a common member such as "As1", who is a member in the two information domains "f1,f2" and "f1", may transfer information of the object *"f1,f2"* from the first domain to the second. This is done because bi-directional transfer is allowed between those domains. Such a transfer will result in the disclosure of information by other members, who were not authorized to make such a disclosure. For instance, "Ac1" can now Read information from both objects *"f1,f2"* and *"f1",* while he is supposed to Read information from *"f1"* only.

Due to the problems discussed above, it seems that *pure DGSA with no rules modifications cannot guarantee the support of MLS or any lattice-based model requirement.* In order to provide such a guarantee, the rules of DGSA need to be strengthened or modified. Such modifications may include updating or deleting existing rules or adding new ones.

4.2.4 Hilborn's Scheme

In his paper, Gene Hilborn [19] showed that MLS can be applied using DGSA by defining the term *adjacency* and strengthening DGSA rule "h".

***Definition 4.1**: Adjacency*

Information domains are adjacent if the information flow between them is in one way and acyclic.

In addition, Hilborn strengthened the "h" rule to become:

h- Transfer of information between information domains can be accomplished only by a member of both the exporting and importing domains when there is an adjacency between those domains.

Hilborn then suggested a scheme for MLS using DGSA by the creation of domains according to the lattice model where the new "h" rule is applied. Each domain corresponds to a lattice category. To support the lattice in the example above (Figure 4-2), each domain contains one information object that presents the object (file) in the corresponding lattice category. Hence, information will flow between domains as it flows between the lattice categories (one way transfer). Figure 4-5 shows the scheme's structure, in which eight domains are created. Each domain is named after the object it contains. For example, the domain that contains the S object *"f1,f2"* is named "f1,f2".

Hilborn considered human users as members of a domain who can access the domain's information object. The access rights each member may have are different from one domain to another. For example, the C member "Ac1" may have a Read/Write right in the C domain "f1", a Read only right in the U domain "pub" and a Write only right in the S domains "f1,f2", "f1,f3" and the TS one "f1,f2,f3." In addition, this member has no access rights in the incomparable C domains "f2" and "f3" or the S domain "f2,f3." Meanwhile, the C domain "f1" has all U users as members with Write-only right, the C users: "Ac1", "Bc1", and "Cc1" as members with Read/Write right, the S users: "As1", "As2", "Bs1", "Bs2", "Cs1", "Cs2" as members with Read-only right and finally all TS users as members with Read-only right. The rest of C and S users are not members in this domain.

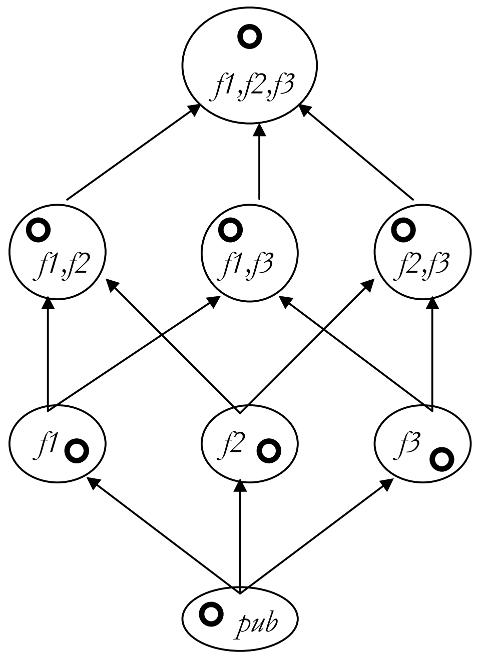


Figure 4-5: Hilborn scheme

Notice that Hilborn did not use the concept of subjects in his scheme, which is necessary in MLS practice as discussed earlier. As a result, a C user such as "Ac1" may not login as a U member in the U domain "pub" with a Read/Write right (as he is using unclassified subject), he can login only as a C member with a Read only right.

As shown in Figure 4-5, the information flow between domains matches the one between the lattice's categories. For simplicity, Hilborn considered any member who has at least a Read right in a domain may Export from that domain and any member who has at least a Write right in a domain may Import to that domain. Hence, a common member can transfer information from one domain to another if he has at least a Read right in the source domain and at least a Write right in the destination domain. For example, a user such as "Ac1" may login as a member in the C domain "f1", the S domains "f1,f2", "f1,f3" and the TS domain "f1,f2,f3". In this case, "Ac1" may Read/Write in "f1", and Write only in "f1,f2", "f1,f3" and "f1,f2,f3". Thus, "Ac1" may transfer information about the object *"f1"* from that C domain to those S and TS domains.

By applying the above scheme, DGSA can satisfy MLS or any other lattice-based model requirement. On the other hand as the number of shared files increases, the number of information domains will increase, too (the number of domains is equal to the number of lattice categories). For example, if the number of files increased from 3 to 10, then the number of domains will increase from 8 to 1,024 (2^{10}). Therefore a major problem of Hilbon's scheme is that it is not scaleable with respect to the size of the lattice.

4.3 Object-Tag scheme

Before discussing this scheme, it is important to introduce a solution for the User * Subject problem in Hilborn's scheme. The solution is by introducing ***roles*** as members of domains. A role is similar to a subject except that many users can invoke the same role. Therefore the user-role relationship is many -to- many, whereas the user-subject relationship is one-to-many. All users accessing a domain through the same role will all have identical rights within the domain. The concept of roles is used in this scheme rather than users as in standard DGSA or Hilborn’s. The security administrators determine which users can access which roles and which role can be a member in which domain, much as in standard DGSA they determine which users can access which domain. As with a user, the same role can be a member of multiple domains. This concept of a role is similar to the simpler form of roles in role-based access control [20].

By referring to the access control matrix discussed in section 4.2.2, Table 4-2 shows eight common members (roles) for users to invoke, while Table 4-3 shows users from the three systems who can invoke those members. Members are also labeled, and they have the same security classes (U, C, S, TS), where a user may invoke the members that he dominates. For example, the A system's S user "As1" can invoke four members with different access rights on the same objects. They are the S member **R4**, the C members **R1** and **R2** and the U member **R0**. While the C user "Ac1" can invoke only two members with different access rights on the same objects. They are the C member **R1** and the U member **R0.**

Objects / Roles	Pub/u	f1/c	f2/c	f3/c	f1,f2/s	f1,f3/s	f2,f3/s	f1,f2,f3/ts
R0/pub/u	rw	w	w	w	w	w	w	w
R1/f1/c	r	rw	-	-	w	w	-	w
R2/f2/c	r	-	rw	-	w	-	w	w
R3/f3/c	r	-	-	rw	-	w	w	w
R4/f1,f2/s	r	r	r	-	rw	-	-	w
R5/f1,f3/s	r	r	-	r	-	rw	-	w
R6/f2,f3/s	r	-	r	r	-	-	rw	w
R7/f1,f2,f3/ts	r	r	r	r	r	r	r	rw

u: unclassified , c: confidential, s: secret , ts: top secret , r: read , w: write , -: null

Table 4-2: Roles and objects access matrix

Users	Roles
Au1,Bu1,Cu1	R0
Ac1,Bc1,Cc1	R1,R0
Ac2,Bc2,Cc2	R2,R0
Ac3,Bc3,Cc3	R3,R0
As1,Bs1,Cs1	R4,R2,R1,R0
As2,Bs2,Cs2	R5,R3,R1,R0
As3,Bs3,Cs3	R6,R3,R2,R0
Ats1,Bts1,Cts1	R7,R6,R5,R4,R3,R2,R1,R0

A: A system , B: B system , C: C system
u: unclassified , c: confidential, s: secret , ts: top secret

Table 4-3: Systems' users and possible invoked roles

In general, the following three steps must be performed in order for a human user to login to a domain and access its information objects:

1) The user must dominate the member (R) in order to invoke it.

2) After the user invoked the member (R), he must specify for (R) which domain (D) to access.

3) After being authenticated by the domain (D), the member (R) is then logged to (D),

where it has specific access rights.

When these steps are completed, the user can access the information objects in the domain (D) with the rights of the member he invoked (R).

Finally, there are two major reasons behind choosing roles as members rather than subjects:

1. Less number of roles are needed than subjects. For instance, every TS user will need 8 subjects (1TS, 3 S, 3 C, 1U), a S user will need 4 subjects (1 S, 2 C, 1 U), a C user will need 2 subjects (1 C, 1 U), and a U user will need 1 subject (1 U) only. Thus, the total subjects needed for each system's users are 27 (8*1 TS + 4*3 S + 2*3 C + 1*1 U), and for all three systems will be 81 subjects (27*3 systems). While in contrast with roles, only 8 roles are needed for all users to share. When two or more users want to invoke the same role, multiple sessions could be created, where each is using a different copy of the same role. The worst scenario that could occur is that all users from all three systems are invoking all roles they dominate simultaneously. In this case the number of sessions created will be 81. However, the number of roles will only be 8.
2. Updates on the access-control matrix are easier to do with roles than with subjects. For example, if "As1" is upgraded from S to TS, then only the roles: **R3,R5,R6,R7** are added to him to invoke. But when subjects are used, new subjects are created for him,

which domains must recognize too.

4.3.1 Scheme methodology

One solution toward providing a scaleable DGSA scheme with respect to the size of the lattice is to develop one that minimizes the number of domains needed to the number of a lattice's levels available. Each domain corresponds to one level and it contains all objects that belong to that level. In order to support such a scheme, the following modifications to rules "a" and "c" of DGSA are necessary:

a- All information objects in an information domain have identical security attributes, but each has a tag attached to it that indicates the members who can access it.

c- A given member (role) has identical access rights in an information domain to information objects that have that member in their tags.

Therefore, when those domains are created, the information objects in each domain represent the objects in all sets with the same number of elements in a subset lattice. The relationship between each object and members who can access it is indicated in the object's tag. All members who can access a lattice category in a certain level become the members of the corresponding domain. In a subset lattice, since every member has a unified access right to one or more categories in a certain level, then that member can has the same right in the corresponding domain. By applying the new rules discussed above (a and c), a domain's member can then access with its right those information objects that include it in their tags only. That is why this scheme is called "Object-Tag ".

As in Hilborn's scheme, adjacency between domains must be supported in this scheme too. Therefore, information flows in one direction from a lower security level domain to higher one(s), and such a flow is acyclic. Figure 4-6 illustrates the scheme's structure, where four information domains are used to solve the MLS example under this scheme. For simplicity, each domain is labeled with the corresponding number of elements in the subsets of the example's subset lattice.

By considering the U category which contains the object *"pub"* as the empty set ϕ, then the first domain is labeled L0. This domain contains only one information object named "*pub*". The second domain is labeled L1. It contains three information objects named "*f1*", "*f2*", and "*f3"*, which represent the three objects in all C categories. Similarly, the third domain is labeled L2. It contains three information objects named *"f1,f2"*, *"f1,f3"*, and *"f2,f3"*, which represent the three objects in the S categories. Finally, the fourth domain is labeled L3. It contains one information object named *"f1,f2,f3"*, which represents the object in the TS category.

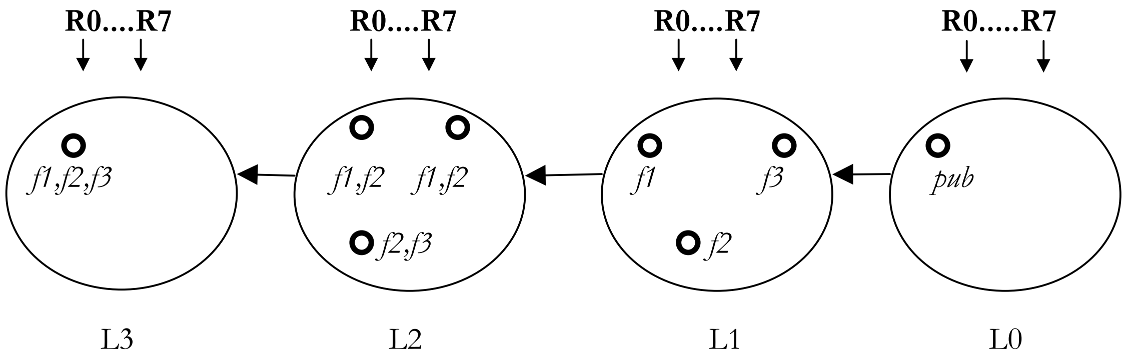


Figure 4-6: Object-Tag scheme

All members with corresponding security clearance to the domain's label may access the new domain with Read/Write, while members with higher security clearance may access it with Read only, and members with lower security clearance may access it with Write only. Therefore, all members (**R0** to **R7**) are accessing all domains (U to TS). But each member has a uniform access right in each domain. That right could be different than its right in other domains. For example, the member **R0** may access the U domain with Read/Write right and the rest of domains with Write-only right.

Each information object in a domain will have a tag attached to it that indicates which member can access it. For example, the U information domain contains one information object *"pub."* The tag attached to this object has the following members:

object	members
pub	**R0,R1,R2,R3,R4R5,R6,R7**

R0 = Read/Write

R1 to **R7** = Read only

where **R0** has Read/Write right and **R1** to **R7** has Read-only right. In addition, the C information domain contains three information objects *"f1"*, *"f2"*, and *"f3."* The tags attached to these objects have the following members:

object	members
f1	**R0,R1,R4,R5,R7**
f2	**R0,R2,R4,R6,R7**
f3	**R0,R3,R5,R6,R7**

R0 = Write only

R1,R2,R3 = Read/Write

R4,R5,R6,R7 = Read only

where **R0** has a Write-only right, **R1, R2**, and **R3** have Read/Write right, and **R4, R5, R6,** and **R7** have Read-only right. The S information domain contains three information objects with the following attached tags:

object	members
f1,f2	**R0,R1,R2,R4,R7**
f1,f3	**R0,R1,R3,R5,R7**
f2,f3	**R0,R2,R3,R6,R7**

R0,R1,R2,R3 = Write only

R4,R5,R6 = Read/Write

R7 = Read only

But in this case, **R0, R1, R2, R3** have Write-only right, **R4, R5, R6** have Read/Write right, and **R7** has Read-only right. Finally, the TS domain has one information object "f1,f2,f3". The tag of this object has the following members:

object	member
f1,f2,f3	**R0,R1,R2,R3,R4R5,R6,R7**

R0 to **R6** = Write only

R7 = Read/Write

where **R0** to **R6** have Write-only right and **R7** has Read/Write right.

Notice that any role with Read/Write right in any domain is included only in the tag of one information object. For example, in the C domain, **R1, R2,** and **R3** have Read/Write rights. But **R1** is included in the tag of *f1* only, while **R2** in the tag of *f2* only, and finally **R3** in the tag of *f3* only.

Hence, when a S user such as "As1" invoked the secret member **R4**, then **R4** can *currently* Read/Write to the information object *"f1,f2"* in the S domain, Read only "*f1*" and "*f2*" objects in the C domain, Read only "*pub"* object in the U domain, and Write only to *"f1,f2,f3"* object in the TS domain. In addition, if "As1" invoked the C member **R1** (since he dominates it), then **R1** can currently Read/Write to "*f1*" in the C domain (because **R1** is indicated there in the tag of *"f1"* only), Read only "*pub*" in the U domain, and Write only to *"f1,f2"*, *"f1,f3"* in the S domain and *"f1,f2,f3"* in the TS domain.

Since adjacency is supported in this scheme too, then information may flow from domains with low sensitivity labels to domains with high ones. Thus, a common member such as **R1** may transfer information of the object *"f1"* from the C domain to the information objects *"f1,f2"*, *"f1,f3"* in the S domain and the information object *"f1,f2,f3"* in the TS domain. Such a transfer can be accomplished successfully because **R1** can Read/ Write in the C domain, where it is included only in the tag of the object *"f1"*. Also it can Write only to the S domain, where it is included in the tags of the objects *"f1,f2"*, *"f1,f3"*, and the TS domain, where it is included in the tag of the object *"f1,f2,f3"*, too.

4.3.2 Scheme conversion algorithm and Theorems

To prove that the Object-Tag scheme is scaleable with respect to a lattice's size, it is important first to develop an algorithm that can convert any partially ordered set (including a lattice) to the structure of this scheme. Then it is possible to develop a theorem and corollaries which show that by using this algorithm the number of domains created for the Object-Tag scheme is equal to the size of the largest maximal chain in the POSET.

Algorithm 4.1 : convert POSET to Object-Tag scheme

4.1.1 Statement of the algorithm: Given any POSET, it is possible to develop the structure of the Object-Tag scheme. Each node in the POSET represents an element or an object where certain roles (members) can access that node. The access rights available are: Read/Write, Read only, and Write only. In the POSET, a given member can access more than one node. Each member who access a node can not have more than one access right to that node. As in the usual practice, a member who has a Read/Write access to a node may write up and read down to comparable nodes to that one in the POSET.

The following steps describe the algorithm:

1. Start with the lowest nodes in the POSET (nodes which have no children). Call this set CN (Childless Nodes).
2. Create a new domain.
3. With each node in CN, create a tag that consists of one field: "access members". In this field store the members who are accessing that node.
4. Every member who is accessing any node in CN becomes a member in the newly created domain. The domain's security policy grants that member an access right similar to the access right it has in the POSET with CN nodes.

5. Store the nodes in CN (with their tags) in the newly created domain, then delete them as well as the edges that connect them to their parents.
6. Make the newly created domain adjacent to all previous domains (if any).
7. Repeat steps 2 to 4 until the POSET is empty.

A more formal statement of the algorithm is given below:

```
Procedure convert partial order (POSET,Object-Tag);  // Input: POSET,
                                                     Output: Object-Tag scheme //
begin
 I=1;
 while POSET not empty do
 begin
   create new domain(i);   // new domain //
    let CN be the set of nodes that have no children;
   associate a tag with each node in CN;
   store in that tag member(s) who can access this node;
   consolidate members of all nodes in CN and make them the members of
   domain(i);
   copy the access right each member has with each node in CN and make the
   security policy of domain(i) associate this right with the corresponding
   member;
   consolidate the nodes in CN (with their tags) and store them in domain(i);
   delete the nodes in CN and the edges to their parents from POSET;
   if domain(i) is not the first make it adjacent to domains (i-1, i-2, ....);
   I = I+1;
 end;   // while POSET and Repeat until POSET is empty //
end .       // procedure //
```

4.1.2 Proof of correctness: To analyze the above algorithm, it is important to show that for a given POSET, how the algorithm satisfies the Object-Tag scheme, and how the domains in this scheme are created with their components (security policy, information objects, and members).

The while loop in the algorithm is repeated as many times as the number of created domains. With every iteration a new domain is created. The childless nodes which are now available in the POSET are stored in the new domain as the domain's information objects. Each object has a tag associated with it which holds the members who are accessing that object (node) in the POSET. In addition, during every iteration, the members who are accessing the childless nodes become the domain's members. The domain's security policy will grant each member the same access right as the one it has with those nodes in the POSET.

To prove that this algorithm is correct, it is important to show that algorithm satisfies the following point:

1) The flow of information between objects in domains is exactly the same as the flow of information between elements in the POSET.
2) The objects in domains are assigned proper tags.
3) The members of each domain can access that domain's objects with the same access rights as they access elements of the POSET.
4) The algorithm halts

First, given any POSET, its properties are: reflexive, antisymmetric, and transitive. Thus incomparable nodes of that POSET have no connection in between. As shown above, the algorithm consolidates incomparable nodes (starting with lowest ones) and stores them in a single domain, hence there will be no connection between them inside that domain too. As domains are created they are connected together in same way as information flow in the POSET. Thus the properties of reflexive, antisymmetric, and transitive are applied between those domains too. Therefore, comparable objects (nodes) must be stored in separate domains and the flow of information between those objects is the same as in the POSET. Notice that if two comparable objects are stored in the same domain, a common member of both nodes in the POSET can not have a uniform access to both nodes.

Second, since the algorithm creates a tag for each node and stores in this tag the members who are accessing that node in the POSET, then when the node is transferred to a domain, only members who used to access it will be in its tag. This is very important to the Object-Tag scheme, because this scheme allows only members who are in the object's tag to access that object.

Third, the algorithm consolidates members who are accessing the childless nodes in the POSET (with every iteration) and make them the members of the new created domain. Then due to the access right rules discussed above, every domain's member will have a single access right in a given domain. This access right corresponds to its right to the corresponding node(s) in the POSET.

Finally, the while loop in the algorithm is repeated until the POSET is empty. On every iteration, the nodes without children are removed from the POSET, so eventually the POSET will become empty and the algorithm will halt.

4.1.3 Theorem 4.1: **Given any POSET, the number of domains created by using algorithm 4.1 is equivalent to the size of a largest maximal chain in that POSET.**

Proof: To prove the theorem, we use induction on the size of a largest maximal chain. Let this size be: max-GR, and let the number of domains created by applying algorithm 4.1 be: f(GR).

1) Basis step: If max-GR =1, then f(GR) =1, and the theorem is true.

2) Hypothesis step: assume the theorem is true for max-GR= x, so we have: f(GR) = x.

3) Induction step: let max-GR = x+1, we need to show that: f(GR) = x+1.

When algorithm 4.1 is applied to GR, the childless nodes in all maximal chains will be consolidated and stored in one domain. Then they are deleted from their parents giving a new POSET GR1. Thus max-GR1 = x. By induction hypothesis f(GR1) =x, so f(GR) = x+1.

Example 4.1: Figure 4-7 (a) illustrates a POSET with 6 nodes (a,b,c,d,e,f) and the information flow is from low to high. Suppose that the following roles have these access rights:

R1: Read/Write : a, Write: b,c,d,e,f

R2: Read/Write : b, Read : a, Write : e,f

R3: Read/Write : c, Read : a, Write f

R4: Read/Write : d, Read: a,

R5: Read/Write : e, Read: a,b

R6: Read/Write : f, Read: a,b,c

In addition, suppose that there are three users as follow: User1 can only Read/Write the node "d", while User2 can only Read/Write the node "e", and finally User3 can only Read/Write the node "f". Thus those users can invoke the following roles in Table 4-4:

User1	R1,R4
User2	R1,R2,R5
User3	R1,R2,R3,R6

Table 4-4: Users/Roles relations

Since the size of a maximal chain with the largest size in this POSET is 3 (a-c-f or a-b-e or a-b-f), then the number of domains needed by applying algorithm 4.1 will be also 3. Figure 4-7 (b) illustrates the new scheme after applying the algorithm to the POSET.

The first domain has only the node "a" as its information object, and roles R1 to R6 as its members. All roles are included in the tag of "a", where R1 has Read/Write right while R2 to R6 each has Read only right.

The second domain has the nodes: "b", "c", and "d" as its information objects, and the roles R1 to R6 as its members. R1 has Write only right, while R2,R3,R4 each has Read/Write right, and R5,R6 each has Read only right. R1 is included in the tags of all objects, while R2,R3, and R4 each is included in the tag of one information object: "b", "c", "d" respectively. Finally, R5 is included in the tag of "b", while R6 is included in tag of "c".

The third domain has the nodes: "e", "f" as its information objects, and the roles R1,R2,R3,R5, and R6 as its members.

The roles R1 to R3 each has Write only right, while R5 and R6 each has Read/Write right. R1 and R2 are included in the tags of "e" and "f", while R3 is included in the tag of "f" only. Meanwhile, R5 and R6 each is included in the tag of one information object: "e" and "f" respectively.

There is no information flow between nodes inside a domain since they are incomparable. However, information may flow from node "a" in the first domain to all nodes in the second and third domains. In addition information may also flow from some nodes in the second domain to some nodes in the third one, where it flows from node "c" to node "f" and from node "b" to the nodes "e", "f".

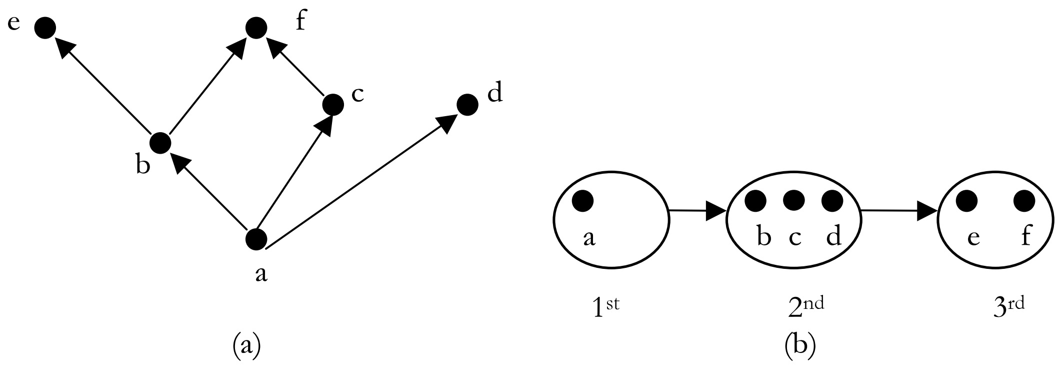


Figure 4-7: Theorem 4.1 example

The above example shows that the number of domains needed as a result of applying the algorithm is equal to the maximal chain of POSET. The algorithm works as follow:

1) The node "a" is the only node that has no children.
2) Thus "a" is stored in the first domain, then it was deleted with edges to its parents (b,c,d).
3) The nodes "b", "c", and "d" become the only nodes with no children and they are incomparable.
4) Thus they are stored in the second domain, then they are deleted too.
5) Last the nodes "e" and "f" become the only nodes with no children and they are incomparable.
6) Thus they are stored in the third domain, and when they are deleted POSET will become empty and the algorithm ends.

Hence the number of domains needed is 3 which equivalent to the size of the maximal chain with the largest size in POSET.

4.1.4 Optimality discussion: As shown above, the result from converting any given POSET to the Object-Tag scheme is n domains, where n is the size of the largest maximal chain in that

POSET. In addition, comparable nodes are stored in different domains. This means that every node in the largest maximal chain will be in a different domain starting from 1 to n. Thus it is impossible to write a new algorithm that can convert that POSET to the Object-Tag scheme with less than n domains. Hence algorithm 4.1 is the optimal solution by contradiction.

4.1.5 Corollary 4.1: **Given a subset lattice of m elements then the size of a largest maximal chain is m+1. Thus the number of domains created by using algorithm 4.1 is also m+1.**

Proof: All maximal chains in a subset lattice have the same size starting at system-low (which is a compartment with zero element) and going to system-high (which is a compartment of m elements). At each step in any maximal chain, the number of elements in the upper compartment is exactly one more than in the lower compartment. Therefore the size of the largest maximal chain (or any maximal chain) is m+1.

Example 4.2: Figure 4-8 (a) illustrates a subset lattice of 2 elements (a and b), where the total number of compartment are 4 (2^2). The empty set ϕ (system-Low) has no element, while its parents, the 2 subsets {a} and {b}, each has one element. The system-High set here has 2 elements {a,b}.

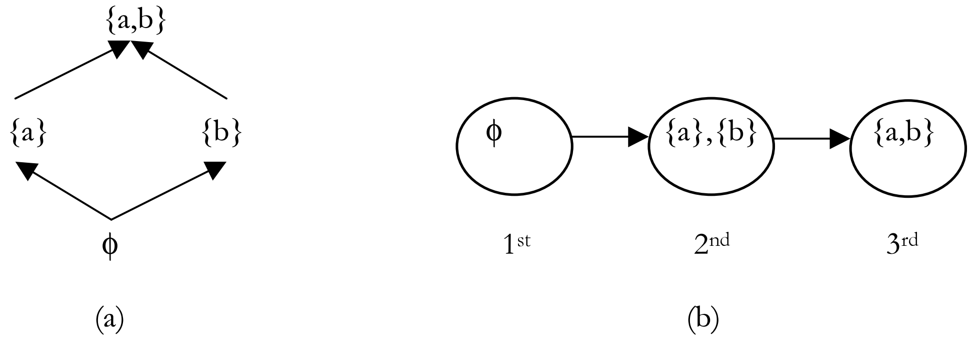


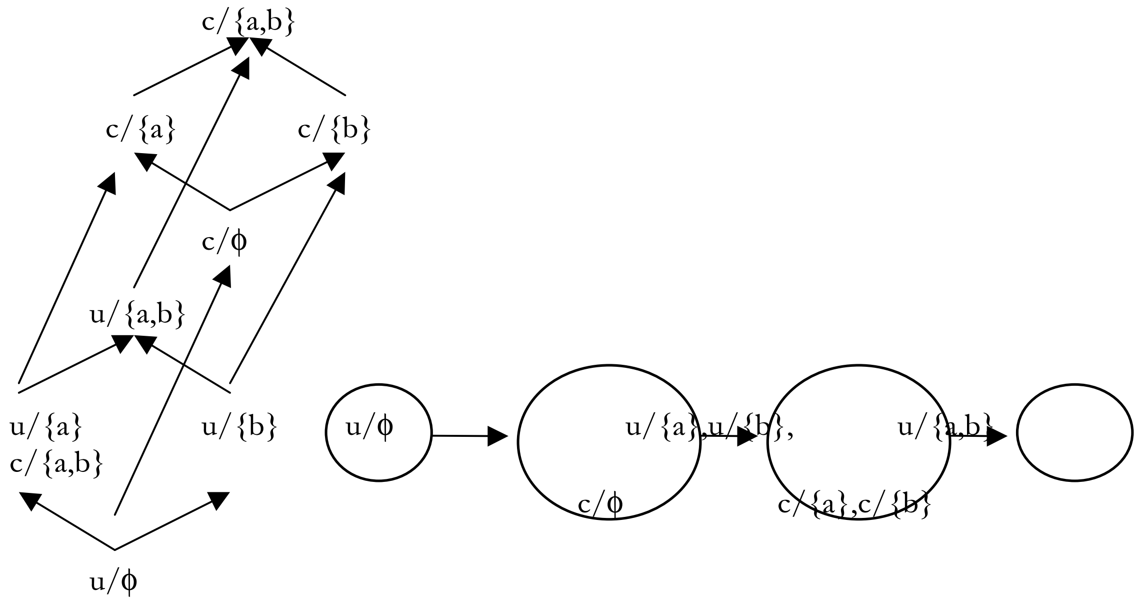
Figure 4-8: Corollary 3.1 example

When algorithm 4.1 is applied to the above lattice, the empty set ϕ is the first set which has no children and it is incomparable to none, thus it is stored in the first domain. Then it is deleted from its parents. Now the subsets {a} and {b} become the only sets which have no children and they are incomparable to each other, thus they are stored in the second domain. Then they are deleted from their parent. Finally the system-High set {a,b} become the only set with no children and it is incomparable to none, thus it is stored in the third domain. When the system-High set is deleted from the lattice, the lattice became empty and the algorithm ends. As a result, only three domains are created to convert a subset lattice of two elements to the Object-Tag scheme. Figure 4-8 (b) illustrates this conversion.

4.1.6 Corollary 4.2: **Given a subset lattice with n classes and m elements, then the size of a largest maximal chain is n+m. Thus the number of domains created by using algorithm 4.1 is equivalent to that size.**

Proof: All maximal chain in a subset lattice with n classes and m elements have the same size starting at system-low and going to system-high (which is system-high in the most upper class or subset). Hence, to go up in a maximal chain, either the class is increased by one (move from lower to higher class), where the total number of moves is n-1, or the number of elements is increased by one in the subset lattice, where the total number of moves is m+1 (corollary 4.1). Thus the size of a maximal chain in a subset lattice of n classes and m elements is equal the sum of both totals = n-1 +m+1= n+m.

Example 4.3: Figure 4-9 (a) illustrates a subset lattice with 2 classes (u and c) and 2 elements (a and b). The figure shows that there are 2 sub-lattices each of which is a subset lattice of the 2 elements a and b.



(a) (b)

Figure 4-9: Corollary 4.2 example

When algorithm 4.1 is applied to the above lattice, the empty set u/ϕ is the first and only set which has no children and it is incomparable to none, thus it is stored in the first domain, and deleted from its parents. Now the subsets u/{a}, u/{b}, and the empty set c/ϕ become the only sets with no children and they are incomparable to each others, thus they are stored in the second domain, and deleted from their parents. Now the subsets u/{a,b}, c/{a}, and c/{b} become the sets with no children and they are incomparable to each others, thus they are stored in the third domain, and deleted from their parent (the system-High set). Finally, the system-High set c/{a,b} is the only set left which has no children and it is incomparable to none, thus it is stored alone in the fourth domain and then deleted. When the system-High set is deleted, the lattice become empty and the algorithm halts. As a result the number of domains created is four, which is: n+m. Figure 4-9 (b) shows the resulting domains.

4.3.3 Scheme advantages and disadvantages

The extended DGSA scheme "Object-Tag" can enforce the MLS example or any lattice model requirement and is scaleable with respect to the size of the lattice. By using this scheme, the number of domains will be equal to the size of the largest maximal chain in the lattice, no matter how many shared files are used.

In comparison with Hiborn's, there will be a tremendous saving in the number of domains. In this scheme, each domain may have all objects that belong to the same level of a lattice, while in Hilborn's scheme each object in a lattice required a separate domain.
A major disadvantage of "Object-Tag" scheme is the effort required to update an object's tag once the number of members increased. This problem may occur due to an increase in the number of shared files. For example, if the number of shared files increased from 3 to 5, then the new subset lattice will have 2^5 compartments (32). In addition, the size of the largest maximal chain in the new lattice will be 6 (m+1). Hence the number of domains will be 6. With such an increase, the number of members will increase too (may be **R0** to **R31**). Such an update will consume a great effort in updating the tags of existing objects and creating a new tag for each new object.

4.4 The three requirements with the new scheme

In this section, the Object-Tag scheme is applied to each lattice model that represents a security requirement discussed in chapter three. Notice that all roles/objects access matrixes discussed later are similar to the subjects/objects access matrixes discussed in chapter three.

4.4.1 *First requirement with Object-Tag*

After creating the specific lattice model for the No-obligation requirement, Figure 3-2, it is possible now to apply the Object-Tag scheme to this requirement. Table 4-5 represents the access matrix for selected roles on selected objects, while Table 4-6 represents the users who can invoke those roles.

Objects / Roles	*atc*	*tor*	*pln*	*mis*	*hos*	*atc, tor*	*mis, hos*	*atc , tor , mis*	*atc , tor , pln*	*pln, mis, hos*	*atc, tor, mis, hos*	*atc, tor, pln, mis, hos*
R1/atc/u1	r/w	-	-	-	-	w	-	w	w	-	w	w
R2/tor/u1	-	r/w	-	-	-	w	-	w	w	-	w	w
R3/pln/u1	-	-	r/w	-	-	-	-	-	w	w	-	w
R4/mis/u1	-	-	-	r/w	-	-	w	w	-	w	w	w
R5/hos/u1	-	-	-	-	r/w	-	w	-	-	w	w	w
R6/atc,tor/u2	r	r	-	-	-	r/w	-	w	w	-	w	w
R7/mis,hos/u2	-	-	-	r	r	-	r/w	-	-	w	w	w
R8/atc,tor,mis /u3	r	r	-	r	-	r	-	r/w	-	-	w	w
R9/atc,tor,pln /u3	r	r	r	-	-	r	-	-	r/w	-	-	w
R10/pln,mis, hos/u3	-	-	r	r	r	-	r	-	-	r/w	-	w
R11/atc,tor, mis,hos/u4	r	r	-	r	r	r	r	r	-	-	r/w	w
R12/atc,tor,pln, mis,hos/u5	r	r	r	r	r	r	r	r	r	r	r	r/w

Table 4-5: Roles/objects for the first requirement

Users	Roles
f1	R6,R2,R1
f2	R8,R6,R4,R2,R1
f3	R8,R6,R4,R2,R1
i1	R9,R6,R3,R2,R1
i2	R10,R7,R5,R4,R3
i3	R12,R11,R10,R9,R8,R7,R6,R5,R4,R3,R2,R1
h1	R11,R8,R7,R6,R5,R4,R2,R1
h2	R10,R7,R5,R4,R3
h3	R12,R11,R10,R9,R8,R7,R6,R5,R4,R3,R2,R1

Table 4-6: Users/roles for the first requirement

By reviewing corollary 4.1 and applying the Object-Tag scheme to support the No-obligation requirement, five domains will be created between the three systems. Each domain represents a "U" level in the lattice model. U1 information domain contains five information objects: *"atc"*, *"tor"*, *"pln"*, *"mis"*, and *"hos."* U2 information domain contains only two information objects: *"atc,tor"* and *"hos,mis"* while U3 information domain contains three information objects:*"atc,tor,pln"*, *"act,tor,mis"* , and *"pln,hos,mis."* Finally, U4 and U5 information domains each contains one information object: *"atc,tor,hos,mis"* and *"atc,tor,pln, mis,hos"* respectively. Figure 4-10 illustrates the Object-Tag scheme for the No-obligation requirement.

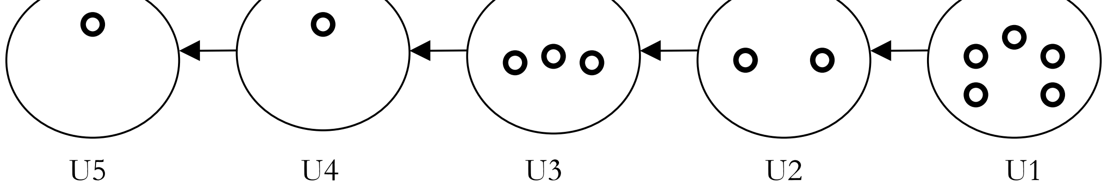


Figure 4-10: First requirement with Object-Tag scheme

4.4.2 Second requirement with Object-Tag

It is possible now to apply the Object-Tag scheme to the MLS lattice, at Figure 3-3, which represent the second requirement. Table 4-7 represents the access matrix for selected roles on selected objects, while Table 4-8 represents the users who can invoke those roles.

Objects / Roles	*vis* /c	*gus* /c	*hji* /c	*hjo* /c	*gde* /c	*a&d* /c	*vis, hji*/c	*gus,hjo, a&d*/c	*vis+ gus*/s	*vis+gus, hji,a&d*/s	*vis+gus, hji+hjo*/ts
R1/vis/c	r/w	-	-	-	-	-	w	-	w	w	w
R2/vis,hji/c	r	-	r	-	-	-	r/w	-	-	w	w
R3/gus,hjo, a&d/c	-	r	-	r	-	r	-	r/w	-	-	-
R4/vis+gus/s	r	r	-	-	-	-	-	-	r/w	w	w
R5/vis+gus, hji,a&d/s	r	r	r	-	-	r	r	-	r	r/w	-
R6/vis+gus, hji+hjo/ts	r	r	r	r	-	-	r	-	r	-	r/w

Table 4-7: Roles/objects for the second requirement

Users	Roles
f11/c	R1
f12/c	R1,R2
f13/c	R3
f21/s	R1,R4
f22/s	R1,R2,R4,R5
f31/ts	R1,R2,R4,R6

Table 4-8: Users/roles for the second requirement

By reviewing corollary 4.1 and applying the Object-Tag scheme to support the MLS requirement, six domains will be created between the three systems. Each domain represents all domains in a given level. L1 information domain contains six information objects that represent the information objects in the six information domains at L1C in the previous two schemes. Similarly, L2 domain contains 15 information objects that represent the information objects in all domains at L2C and L2S in the previous schemes. Figure 4-11 illustrates the Object-Tag scheme for the MLS requirement, where the domains:

L1 contains six information objects = L1C

L2 contains 15 information objects = L2C+L2S.

L3 contains 20 information objects = L3C+L3S.

L4 contains 15 information objects = L4S+L4TS.

L5 contains six information objects = L5TS.

L6 contains one information objects = L6TS.

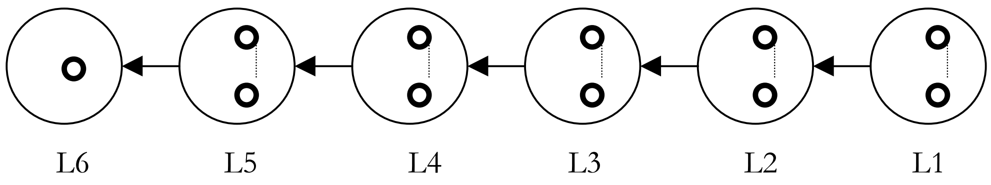


Figure 4-11: Second requirement with Object-Tag scheme

4.4.3 *Third requirement with Object-Tag*

After constructing the lattice for the Chinese Walls requirement, Figure 3-5, it is possible to implement the Object-Tag scheme for the third requirement. As discussed in chapter three, only objects from CHN1 to CHN3 satisfy the third requirement, therefore the total number of objects used to support the third requirement is 26 out of 63 (categories in CHN1 + CHN2 + CHN3). The total number of objects associated with each level is as follows:

CHN1: = 6 objects
CHN2: = 12 objects
CHN3: = 8 objects

Total: 26 objects.

Table 4-9 represents the access matrix for selected roles on selected objects, while Table 4-10 represents the users who can invoke those roles.

Roles \ Objects	*[dpl,ϕ,ϕ]*	*[scm,ϕ,ϕ]*	*[ϕ,bkl,ϕ]*	*[dpl,bkl, ϕ]*	*[scm,bkl,ϕ]*	*[dpl,bkl,rgs]*
R1/[dpl,ϕ,ϕ]	r/w	-	-	w	-	w
R2/[scm,ϕ,ϕ]	-	r/w	-	-	w	-
R3/[ϕ,bkl,ϕ]	-	-	r/w	w	w	w
R4/[dpl,bkl,ϕ]	r	-	r	r/w	-	w
R5/[scm,bkl,ϕ]	-	r	r	-	r/w	-
R6/[dpl,bkl,rgs]	r	-	r	r	-	r/w

Table 4-9: Roles/objects for the third requirement

Users	Roles
f11	R1
f12	R2
f13	R3
f25	R1,R3,R4
f26	R2,R3,R5
f35	R1,R3,R4,R6

Table 4-10: Users/roles for the third requirement

By reviewing corollary 4.1 and applying the Object-tag scheme to support the Chinese Walls requirement, three domains will be created between the three systems. Each domain represents a "CHN" level in the lattice model which is lower than SYSHIGH. All objects associated with CHN level are consolidated and contained in the corresponding level domain. For example, the object [scm,bkl,ϕ] that is presented in CHN2 level on the lattice is represented in CHN2 domain. Figure 4-12 illustrates the Object-Tag scheme for the Chinese Walls requirement.

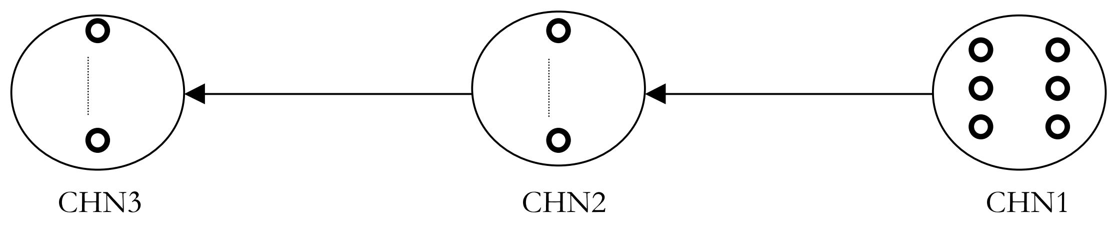


Figure 4-12: Third requirement with Object-Tag scheme

CHN1 information domain contains six information objects that are presented in the six domains at CHN1 level in the previous two schemes. Similarly, CHN2 contains 12 information objects, while the CHN3 domain contains eight information objects. The total number of objects in all three domains is 26, as in previous schemes.

4.5 Conclusion

Due to the existence of some DGSA rules, pure DGSA cannot support lattice models directly. Hilborn developed a DGSA scheme that supports these models by strengthening some DGSA rules. His scheme defined adjacency and modified rule "h" accordingly. Then he proposed the creation of domains according to the lattice model where each domain corresponds to a lattice category. Hence domains became related in transfer as the information flows in a lattice model (one way). Therefore, for the example illustrated in Figure 4-2, each domain contains the object which is presented in a given lattice compartment. Every human user can access the domain's object depending on his access right on that domain if any. Hilborn's scheme succeeds in enforcing DGSA to support lattice models, but it is not scaleable in the size of the lattice, since for a large lattice, a large number of domains are needed (one domain for each lattice's category).

The Object-Tag scheme solves the scalability problem of Hilborn's scheme. The number of domains here are equal to the size of the largest maximal chain in the lattice. In addition, rules a and c were modified so that each object has a tag associated with it. Only the members (in this case Common Programs) who are indicated in the object's tag may access it with their access rights. As a result, this DGSA scheme supports lattice models and it is scaleable with respect to the size of the lattice. By applying this scheme to the three lattice models of the three confidentiality requirements respectively, the scheme succeeds in supporting those lattice models and is scaleable with respect to the sizes of those lattices.

5. Contributions and Future Research

5.1 Introduction

The Hajj case study is a real-life case that been chosen as an example to demonstrate that confidentiality requirements (pertaining to information flow) in real-world complex data sharing situations can be represented in lattice models. The three confidentiality requirements (No-obligation, MLS, Chinese Walls security) of the Hajj example are represented by lattice models, where the case study's systems (F,I,H) can share together separate data under different requirements.

Meanwhile, DGSA is a powerful architecture that distributed systems of an M-D.O (such as: F, I, H of the Saudi Hajj M-D.O) may use to share and transfer data between them. The architecture concentrates mainly on the security of transferred data between domains, and the security of data in domains. Some DGSA's rules prevent it from directly supporting any lattice model requirement. It is important to develop a modified or extended DGSA scheme that can support lattice models and minimizes the number of domains created.

The Object-Tag scheme introduced by the author supports any lattice model and it is scaleable to the lattice's size. When this scheme is applied to the Saudi Hajj example's requirements, it maintain the lattices in a reasonable number of domains. As shown in chapter four, the Object-Tag scheme is scaleable with respect to the size of any partially ordered set including a lattice.

When applying the Object-Tag scheme to the lattices of the Saudi Hajj example, only 14 domains are needed for the three requirements: 5 for the first, 6 for the second, and 3 for the third. Figure 5-1. While if Hilborn's scheme is used, 101 domains will be needed: 12 domains for the first requirement, 63 for the second, and 26 for the third. The Object-Tag scheme used a much smaller number of domains, perhaps even a minimum number because of inherent information flow requirements.

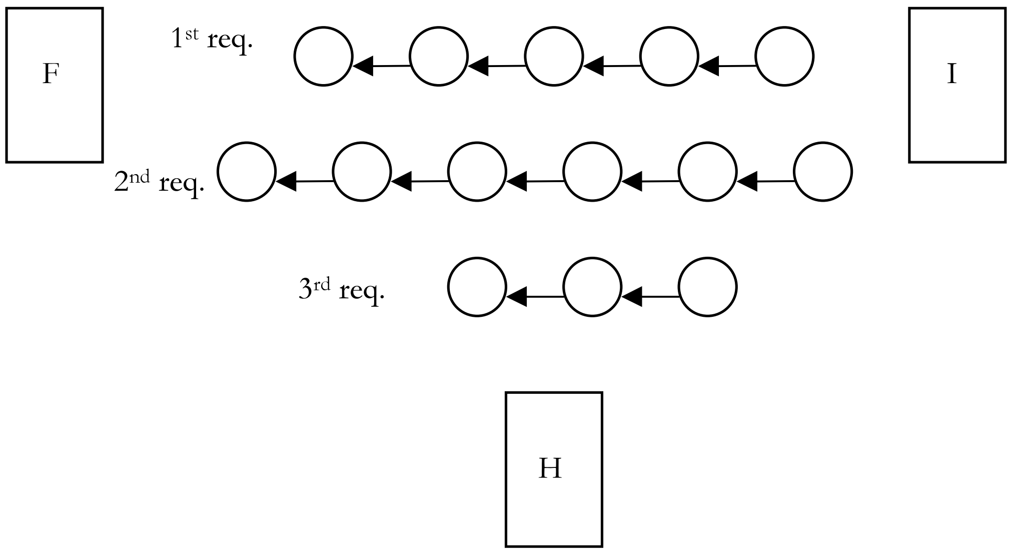


Figure 5-1: Domains required under the three requirements for the Hajj example

In general, with any example of an M-D.O similar to the Hajj the following steps should be followed in order to apply DGSA with the Object-Tag scheme to different lattice model requirements:

1) Create the lattice model for each requirement.
2) Identify the users from each system and the files they need to share, taking into account that some users may access files under different requirements.

3) Specify the information flow policy for each system by determining the objects from the various requirements' lattices that users from that system may access.
4) Identify the Common Programs under each requirement and the users who can invoke them, taking into account that new users may be added, and some roles may access different objects under different requirements.
5) Support the Object-Tag scheme for each requirement by applying corollary 4.1 together with algorithm 4.1 to each requirement's lattice. At the end, each requirement should have a set of domains. Each domain contains a set of information objects with tags, members (roles) who can access those objects, and a policy which regulates data-access and data-transfer.

In summary, the central contributions of this work are the following:

1) Definition and analysis of the case study's confidentiality requirements of shared data, and their representations in lattice models.
2) Extension of DGSA through sets of additions to the original inadequate architecture in order to develop a scheme that is capable of supporting lattice model requirements as well as being scaleable in the size of the lattice.

5.2 Future research

Future works related to the dissertation are as follows:

1- Developing another DGSA scheme that can reach the same goals as Object-Tag scheme.
2- Developing other DGSA schemes which may solve different scalability entries with respect to shared data and systems, such as large size of files, rate of updates between

systems, large number of users, and frequent shutdown.

3- Solving other security requirements with DGSA, such as security groups.

4- Applying other architectures to the Hajj example, such as SINTRA [21] , [22] , or CORBA [23].

5- Presenting the information flow policies of the Hajj example in object-oriented systems [24] , [25].

Appendix A: DGSA

This appendix gives a brief overview of DGSA (DOD Goal Security Architecture) based on [1].

A.1 Introduction

DGSA was developed by the Defense Information Systems Security Program. It is defined in volume 6 of the DOD Technical Architecture Framework for Information Management.

A.1.1 Purpose

Consideration is being given to the protection of information and system assets as part of the total view of the missions, threats, performance, extensibility, usability, and cost of implementations.

The DGSA will provide the basis for the development of security products and mechanisms that may be chosen by security systems engineers and integrators. DGSA is designed to provide data sharing and data transfer between distributed heterogeneous systems.

A.1.2 Architectural Types

Information system architectures range in definition and occur in sequence from abstract views to specific views of what is to be developed. The types are abstract, generic, logical, and specific, each of which is described below.

A.1.2.1 *Abstract Architecture*

An abstract architecture begins with knowledge of the requirements and defines corresponding functions to be performed. It defines principles and fundamental concepts that guide the selection and organization of functions.

Abstract security architectures cite principles, fundamental concepts, and functions that satisfy typical security requirements.

A. 1.2.2 *Generic Architecture*

The development of a generic architecture is based upon the abstract architectural decisions. It defines the general types of components and allowable standards to be used and identifies any necessary guidelines for their application.

A generic security architecture proceeds from an initial allocation of security services and functions and begins to define the types of components and security mechanisms that are available to implement the security services with particular strengths.

A.1.2.3 *Logical Architecture*

A logical architecture is a design that meets a hypothetical set of requirements. It serves as a detailed example that illustrates the results of applying a generic architecture to specific circumstances. The only difference between a logical and a specific architecture is that the specific requirements are real, not hypothetical.

A.1.2.4 *Specific Architecture*

The objective of any system architect is to accomplish a level of design specification such that components can be acquired to implement the system. The specific architecture addresses components, interfaces, standards, performance, and cost. Specific security architectures show how all the selected information security components and mechanisms, including doctrine and supporting security management components, combine to meet the security requirements of the specific system under consideration.

A.1.3 Process

Development of the DGSA is achieved through an iterative process, cycling through the following steps until a satisfactory generic security architecture (or architectures) is accomplished:

- Perform requirements analysis
- Create structure for the architecture
- Make security service allocations
- Select security components and mechanisms
- Perform interdependency analysis (evaluation).

A.2 Security Architecture's Specification

A.2.1 DGSA Security Requirements

The four security requirements are:

1. Multiple Information Security Policy Support,
2. Open Systems Employment,
3. Appropriate Security Protection,
4. Common Security Management.

A.2.1.1 *Multiple Information Security Policy Support*

Although most current information systems support only one information security policy at a time, there has long been a desire by users to operate simultaneously at multiple sensitivity levels or under multiple security policies (e.g., by using multilevel secure systems) on a single device (e.g., workstation, outboard protocol device). Requirement 1 above recognizes that support for multiple security policy operation must become more common.

A.2.1.2 *Open Systems Employment*

DOD information systems must be open in the sense that potential connectivity among them is always supported, even if a particular request for communication is denied because of a security policy decision, given that users operating under different security policies may need to share components, and that complex policies for sharing and transferring information among users operating under different security policies must be supported.

A2.1.3 *Appropriate Security Protection*

What constitutes appropriate security protection, in part, is affected by the security protection provided by the communications system that is used among distributed systems.. The only service that should be assumed from a common carrier communications system is availability.

A.2.1.4 *Common Security Management*

Like the open systems requirement, security management appears to be concerned with operational issues, but it actually provides the foundation for many of the security mechanisms that implement the security services chosen to satisfy the other security requirements. To ensure that distributed information processing is properly supported, the DGSA must address common security management. This commonality will allow security administrators to manage, in a uniform manner, systems that operate under multiple security policies.

A.2.2 Security Views

To accomplish the security service allocations, several views and concepts are presented that support the DGSA.

A.2.2.1 *Abstract Information System Architecture Security View*

The DGSA target architecture is, first, simplified into an abstract perspective, defined as *local subscriber environments* (LSEs) that are connected to one another by a *communications network* (CN). Figure A-1 illustrates this first security view of the information system.

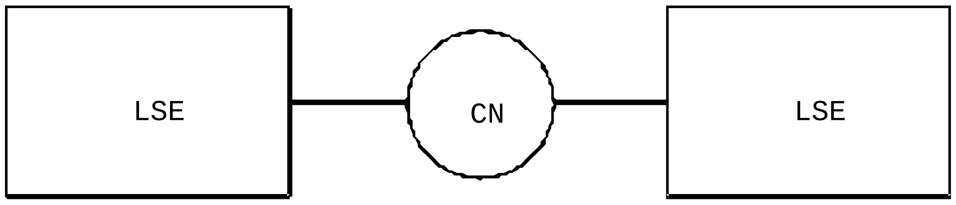


Figure A-1. Abstract security perspective

The LSEs include all devices and communication systems under user (organization) control. The CN provides communication capabilities that allow LSEs to share information.

A.2.2.2 *LSE Generic Security View*

The first generic information system architecture security view concentrates on LSEs, which may be composed of open systems and *relay systems* (RSs), as described in ISO 7498, and *local communications systems* (LCSs) within a given operating environment. Open systems accessed directly by users are conventionally referred to as *end systems* (ESs) and appear throughout the rest of the DGSA. The *transfer system* includes the communication protocols integrated into end systems and relay systems and the interconnecting LCSs and CNs. LCSs are controlled within LSEs, but CNs are not controlled within LSEs. Figure A-2 illustrates this generic LSE security view.

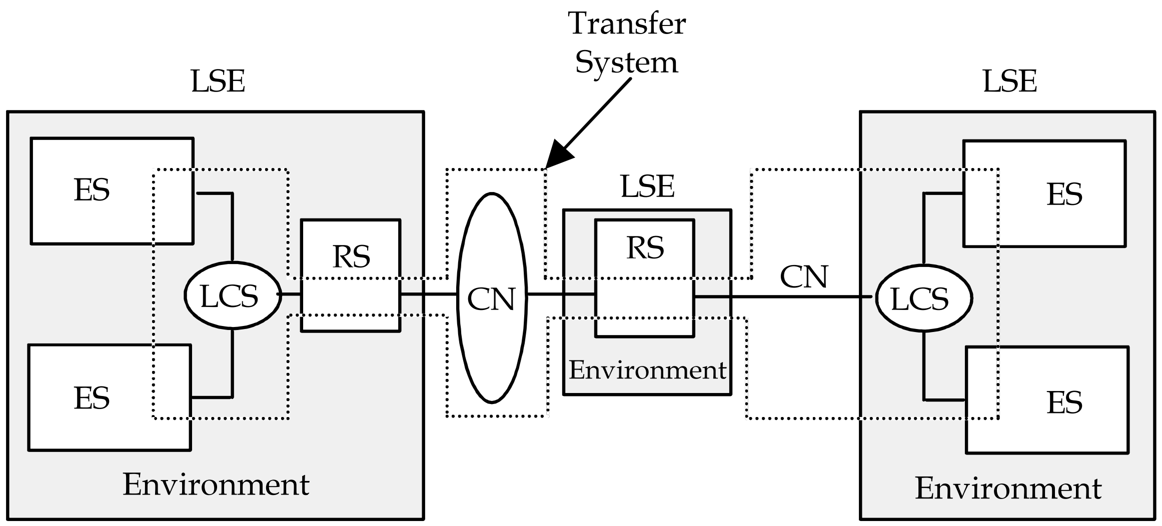


Figure A-2: Security view of LSEs

A.2.2.3 *CN Description*

Figure A-3 shows two types of network elements. In one type of network element (e.g., CN_b), both ends of the (private) link are within LSEs and can be protected with complete traffic flow security (TFS). In the other type of network element (i.e., CN_a), one end of each link terminates in the commercial control zone, and address information is needed by the switches in the commercial control zone, which limits the extent of TFS possible. One consequence of providing complete TFS between two LSEs is that the link cannot be used for any other purpose and, thus, creates a closed system. Given the importance of requirements for open systems, apparent requirements for complete TFS between LSEs must be examined very carefully. Figure A-3 provides examples of the network element variations that are consistent with the generic security view.

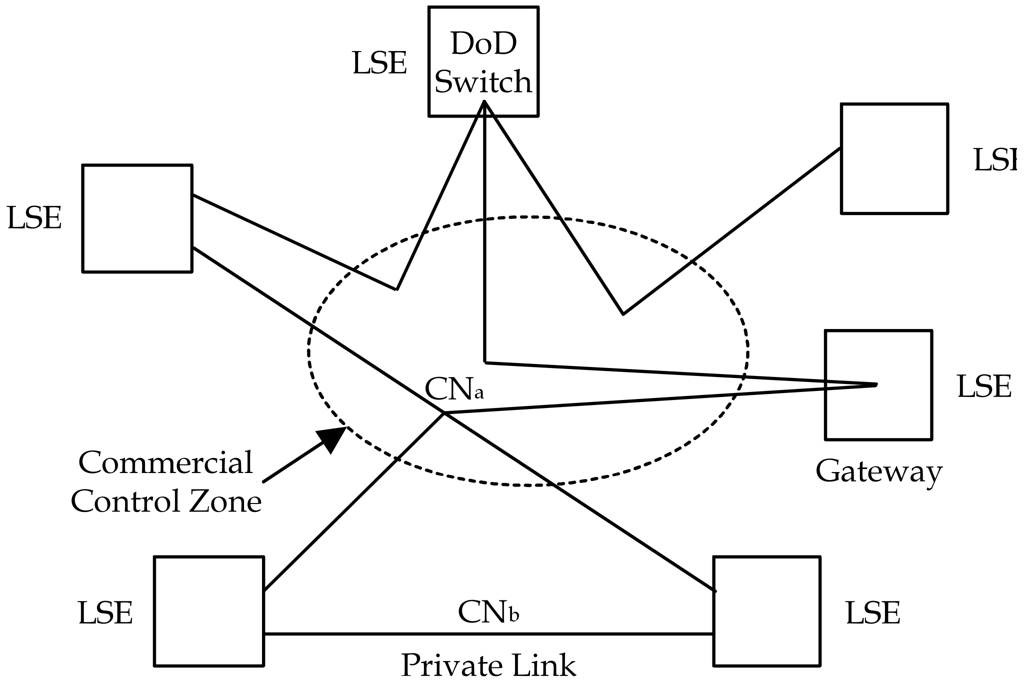


Figure A-3: Example variations of network elements

A.2.3 Security Concepts

The objectives for security in such an environment are to maintain open and distributed capabilities and yet be able to establish and enforce dynamic mission and information security policies. A simple characterization of such an environment is that resources and information may be shared or isolated as desired.

Several security concepts are set forth, here, that support these objectives: information domains, strict isolation, interdomain information-sharing and transfer, multidomain information objects and policies, absolute protection, uniform accreditation, and security management. These security concepts are discussed below.

A. 2.3.1 *Information Domains*

The management of information is accomplished by individuals and groups of people who create, collect, process, categorize, store, transfer, and communicate particular information. The value of that information and, therefore, the required protection of that information is determined by the group. The group determines the conditions for authorized access to the information and the conditions for individuals to become members of the group. Three elements are necessary for this concept to be employed:

- A group must have a defined membership.
- Information objects must be uniquely identified within the domain of the group.
- The security policy regarding the protection of and access to the information objects must be known and agreed to by the membership.

An *information domain* is a set of users, its information objects, and a security policy. An information domain security policy is the statement of the criteria for membership in an information domain and the required protection of the information objects. Information domains are not hierarchically related, and they may not implicitly or explicitly imply a sensitivity related to multiple categories of sensitivity.

Each information domain is identified uniquely. The unique identification indicates (directly or indirectly) the sensitivity of all the information objects in an information domain. Any security-relevant attributes of information objects in an information domain must be the same for all information objects in the information domain, that is, there must be no security-relevant distinction made among the information objects in an information domain. Members of an information domain may have different security-related attributes.
For example, some members might have Read-only permission for information objects in an information domain, while other members might have Read and Write permissions. Since all information objects in an information domain have the same security-relevant attributes, a user who has Read and Write permissions in an information domain has those permissions for *every* information object in the information domain.

A.2.3.2 *Strict Isolation*

Information systems that support multiple information domain security policies must adopt a protection strategy that provides a basis for satisfying all of them. One such strategy, termed *strict isolation*, is to isolate one information domain from another, except when there is an explicit relationship established. Under this strategy, an information system must provide mechanisms that maintain separation of information domains in ways that are satisfactory to each of them.

A.2.3.3 *Interdomain Information Sharing and Transfer*

Some mission requirements will necessitate the sharing or transfer of information objects among information domains. The establishment of new mission functions, new mission area relationships, or new organizations are examples of events that can create requirements for information-sharing and transfer.

The simplest method of sharing information is to accept new members into an existing information domain and to grant access privileges to them. Where a need exists to share some, but not all of the information objects in one or more information domains with members of other information domains, a new information domain may be created to contain the shared information objects. The new information domain, like any other information domain, requires a security policy. The members of the new information domain may or may not be members of the information domains from which its information objects were obtained.

Information objects can be transferred between two information domains only in accordance with established rules, conditions, and procedures expressed in the security policy of each of them. The transfer can be accomplished only by a user who is a member of both the sending and receiving information domains and, if required by the information domain policies, has been granted the appropriate privileges (e.g., "release authority").

The transfer of information objects between information domains may be implemented as a move operation (in which the information object no longer exists in the originating information domain) or as a copy operation (in which the information object exists in both information domains).

A.2.3.4 *Multidomain Information Objects and Policies*

The missions of most organizations require that their members operate in more than one information domain. To carry out their mission, users may need to process information objects from several information domains concurrently. Often, a user may have a *perception* that a collection of information objects from different information domains is a single, composite information object. Such a composite information object is referred to as a *multidomain information object*. This perception must be achieved without combining real information objects from different information domains to create real multidomain information objects.

A.2.3.5 *Absolute Protection*

Since open systems may consist of an unbounded number of unknown heterogeneous LSEs and it may be necessary to communicate with any of them, system security architects must have a rational basis for protection decisions in such an environment. In this environment, it is no longer possible to rely upon the assurances provided by physically separated networks or cryptographically isolated LSEs. Information domains must rely on the protections afforded by a heterogeneous collection of LSEs. The concept of *absolute protection* is set forth to provide a framework for achieving uniformity of protection in all information systems supporting a particular information domain. It directs its attention to the problems created by the interconnection of LSEs that provide disparate strengths of security protection.

A. 2.3.6 *Uniform Accreditation*

The concepts of information domains, strict isolation, and absolute protection work together to provide the basis for achieving a high degree of uniformity in accreditation. Each LSE is evaluated against the security policy of each information domain that it supports. The objective is to have equivalent protection in all LSEs that support a given information domain.

A. 2.3.7 *Security Management*

Security management is a particular instance of information system management. Security management is central to the proper operation of protected LSEs and their component parts, both separately and jointly. Within the DGSA, security management performs many critical functions in support of its requirements, security service allocations, and security concepts.

Security management is concerned with all aspects of protection of LSEs, protection within LSEs, and protection among LSEs. For example, security management includes the control of physical access to facilities, the maintenance of information domain security policies, the invocation of particular security mechanisms in end systems, and the control and protection of communications between LSEs.

A. 2.4 Security Services

The DGSA assigns certain fixed security service allocations to doctrinal security mechanisms. For the designer of more specific security architectures, choices must be made regarding security service allocations and types of security mechanisms.

An LSE and its components must satisfy each of the information domain security policies for which it is accredited.. The LSE is the principal location for direct implementation of doctrinal security mechanisms, but local security mechanisms may rely upon remote systems to provide initial capabilities and life-cycle support .

The security services of ISO 7498-2 (authentication, access control, data confidentiality, data integrity, and nonrepudiation) are extended in their definitions for use in the DGSA beyond communications. The DGSA also includes availability among the security services. Security services are implemented in LSEs in the form of physical, personnel, and administrative security mechanisms.

A.2.4.1 *Mechanisms for Identification and Authentication*

Authentication of the claimed identities of individuals, as individuals or as members of a group, is a typical security policy requirement. Authentication mechanisms provide varying degrees of credibility that such claims are correct. Authentication responsibilities are often shared between doctrinal, hardware, and software mechanisms.

The use of keys with locks, passwords, or cipher lock codes authenticates identity only to the extent of the probability that the presenter is a valid holder of the object or information.

A. 2.4.2 *Mechanisms for Access Control*

Access-control mechanisms enforce security policy requirements for the isolation of assets and information from people and their agents. Access-control mechanisms also permit authorized access to assets and information. The first line of protection for access control is through mechanisms that control access to the facilities (e.g., buildings, rooms) containing the end systems, relay systems, and LCSs. Key, combination, and cypher locks are common mechanisms for controlling access to facilities.

The next line of protection for access control involves the responsibility of hardware and software features of the end systems and relay systems. Access-control mechanisms can also contribute to the provision of confidentiality, integrity, and availability services; independent aspects of these services are presented in the following sections.

A. 2.4.3 *Mechanisms for Confidentiality*

Confidentiality mechanisms satisfy security policy requirements to protect information from unauthorized disclosure. The major applications of doctrinal confidentiality mechanisms in LSEs involve video displays, printing devices, sounds, and non-video electromagnetic emanations.

A. 2.4.4 *Mechanisms for Integrity*

Integrity mechanisms respond to security policy requirements to protect information and other system assets from unauthorized modification. The major applications of doctrinal integrity mechanisms in LSEs involve the correctness of end system and relay system hardware and software and the correct functioning and use of other doctrinal security mechanisms.

A. 2.4.5 *Mechanisms for Non-Repudiation*

Nonrepudiation mechanisms support security policy requirements for proof of delivery and proof of origin of information transactions.

A. 2.4.6 *Mechanisms for Availability*

Availability mechanisms in communications networks and LSEs satisfy security policy requirements for availability of communications and processing resources.

A.3 Summary

DGSA is an architecture that could be implemented between distributed heterogeneous systems to share and transfer data between them. It used information domains as the mechanism between systems that enables each system to share and transfer data to other systems.

An information Domain [D = (O, M, P)] combines the following:

- A set of *information Objects* (O), identifiable as belonging (files, programs) to the domain.
- A set of (human) members (M) of the domain.
- An information domain *security policy* (P) that includes:
  - the requirements for membership
  - the rules of access by members to information objects of the domain
  - the rules of import and export of information from/to other information

domains

- the required protection of the information objects of the domain.

The following list summarizes the details of the *security policy of an information domain*. See Figure A-4.

a- All information objects in an information domain have identical security attributes.

b- All members of an information domain need not have equal access to its information objects.

c- A given member has identical access rights to all information objects in an information domain

d- No information object belongs to more than one information domain.

e- Individuals may be members of more than one information domain.

f- Transfer of information between domains occurs only in accordance with the policies of both the exporting and importing domains.

g- Transfer of information between information domains can be accomplished only by a member of both the exporting and importing domains.

h- Protections requirements for an information domain are stated independently of any other information domain.

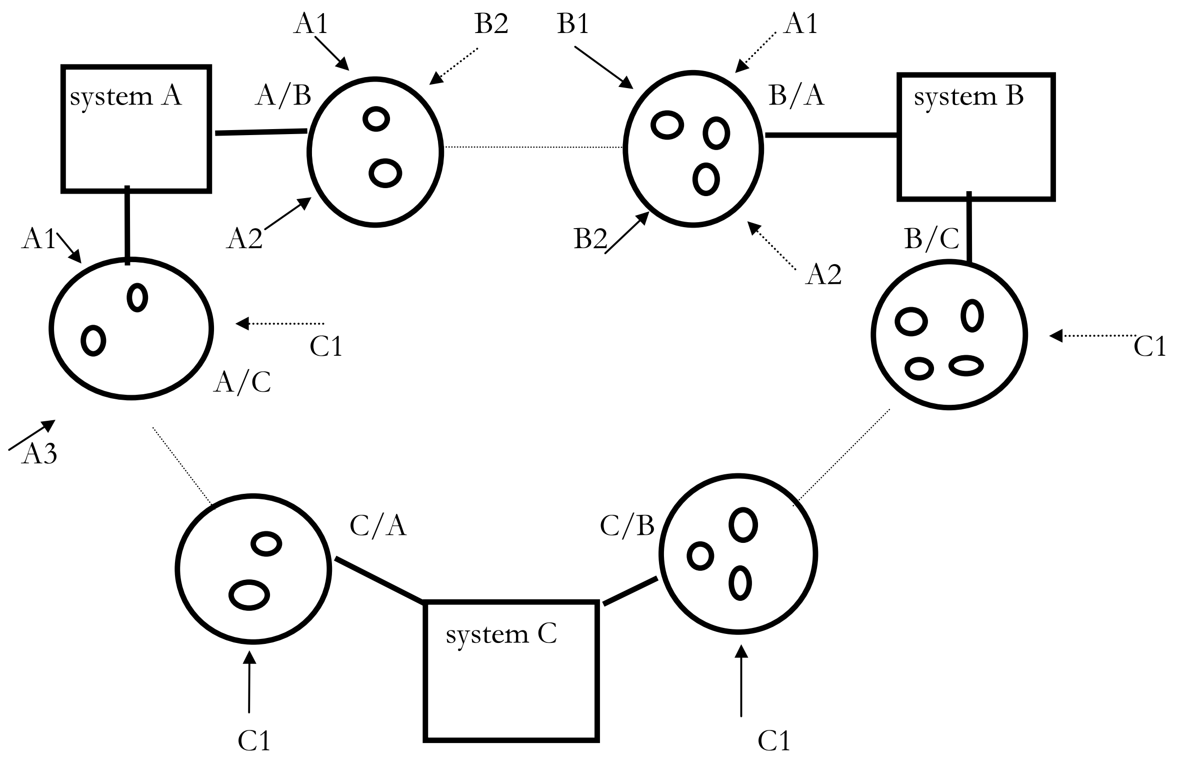


Figure A-4: DGSA architecture

The architectural design of DGSA specifies that if distributed systems with heterogeneous databases desire to share data together, then each system:

1- must have a mutual agreement with every other system; this agreement will create a mutual policy between the two systems.

2- must specify, according to the mutual policy, the information domains that will be created between the two systems, their security policy, the types and names of information objects that will belong to each information domain, and the users who will access these information objects in each information domain according to the information domain security policy.

3- specifies, according to the mutual policy, the location of each information domain and the methods of communication to access it.

Bibliography

Bibliography

[1] Department of Defense (DOD) Goal Security Architecture (DGSA), Center for Information System Security Program, Version 1.0, August 1993.

[2] Discrete Mathematics, Second Edition, by K. Ross and C. Wright, 1988.

[3] Cryptography and Data Security, by Dorothy Denning, 1982.

[4] Data & Computer Security, Dictionary of Standards Concepts and Terms, by Dennis Longley & Michael Shain, 1987.

[5] A Lattice Model of Secure Information Flow, by D.E. Denning, Communication of ACM, 1976, vol 19, no.5, pp.236-243.

[6] Lattice- Based Access Control Models, by Ravi Sandhu, IEEE Computer, November 1993.

[7] Secure Computer Systems: Mathematical Foundations and Model, by D.E. Bell and L.J. LaPadula, M74-244, Mitre Corporation, Bedford, Massachusetts, 1975.

[8] The Chinese Wall Security policy, by Brewer and Nash, IEEE Symposium on Security and Privacy, 1989, pp. 215-228.

[9] Extending the Brewer-Nash Model to a Multilevel Context, by C. Meadows, IEEE Computer Society Symposium on Research in Security and Privacy, Oakland, CA, May 7-9/1990, pp.95-102.

[10] Lattice-Based Enforcement of Chinese Walls, by Ravi Sandhu, Computer & Secuirty, 11/1992, pp. 753-763.

[11] Security in Open Systems, Special Publication 800-7, by U.S. Department of Commerce, National Institute of Standards and Technology (NIST).

[12] Modeling Multidomain Security, by J. Vazquez-Gomez, Proceedings of the New Security Paradigms workshop 1992-1993 ACM SIGSAC pp 167-174.

[13] Enforcing Complex Security Policies for Commercial Applications, by L. Kao and R. Chow, Proceedings of 19th Annual International Computer Software Applications Conference (COMPSAC95), Dallas, TX, 9-11/August/1995.

[14] Hajj in Islam, by M. Zani, Al-Manhel Magazine, Special edition May-1995.

[15] At the Service of Allah's Guests, by the Information Affairs of the Information Ministry, Special publication 1993.

[16] Pilgrimage Organizing Instructions, by the office of the Minister, Special publication of the Hajj Ministry 1994.

[17] Lattice Based Models for Controlled Sharing of Confidential Information in the Saudi Hajj System, by Tarik Himdi and Ravi Sandhu, Proceedings of the 13th Annual Computer Security Applications Conference, San Diego, CA, December 8-12/1997, pp 164-174.

[18] Applying The DoD Goal Security Architecture as a Methodology for the Development of System and Enterprise Security Architectures, by T. Lowman and D. Mosier, Proc of the 13th Annual Computer Security Applications Confernce, San Diego, CA, December 8-12/1997, pp. 183-193.

[19] Information Domains Metapolicy, by Gene Hilborn, proceedings of the 18th National Computer Conference, Baltimore, MD, October 1995, pp.27-36.

[20] Role-Based Access Control Models, by R. Sandhu, E. Coyne, H. Feinstein, and C Youman, IEEE Computer, Volume 29, Number 2, February 1996, pp. 38-47.

[21] Confidentiality in a Replicated Architecture Trusted Database System: A Formal Model, by O. Costich, J. McLean, and J. McDermott. Proc of the Computer Security Foundations Workshop VIII, IEEE Computer Society, 1994, pp.60-65.

[22] Design documentation for the SINTRA global scheduler, by M.H. Kang and R. Peyton, Naval Research Laboratory Memo Report 5542- 93-7362 (1993).

[23] CORBA Security Dreaft (Common Object Request Broker Architecture), by OMG, version 0.2, document no.95-9-1, September 1995.

[24] Information Flow Control in Object-Oriented Systems, by P. Samarati, E. Bertino, A . Ciampichetti, and S. Jajodia, IEEE Transactions on Knowledge and Data Engineering, vol 9, no.4, July-August/1997, pp. 524-538.

[25] Distributed Information Systems from Client/Server to Distributed Multimedia, by E. Simon, 1996.