EFFICIENT RUNTIME SECURITY SYSTEM FOR DECENTRALISED DISTRIBUTED SYSTEMS

Akeel A. Thulnoon

A thesis submitted in partial fulfilment of the requirements of Liverpool John Moores University for the degree of Doctor of Philosophy

July/ 2018

ABSTRACT

Distributed systems can be defined as systems that are scattered over geographical distances and provide different activities through communication, processing, data transfer and so on. Thus, increasing the cooperation, efficiency ,and reliability to deal with users and data resources jointly. For this reason, distributed systems have been shown to be a promising infrastructure for most applications in the digital world.

Despite their advantages, keeping these systems secure, is a complex task because of the unconventional nature of distributed systems which can produce many security problems like phishing, denial of services or eavesdropping. Therefore, adopting security and privacy policies in distributed systems will increase the trustworthiness between the users and these systems. However, adding or updating security is considered one of the most challenging concerns and this relies on various security vulnerabilities which existing in distributed systems. The most significant one is inserting or modifying a new security concern or even removing it according to the security status which may appear at runtime. Moreover, these problems will be exacerbated when the system adopts the multi-hop concept as a way to deal with transmitting and processing information. This can pose many significant security challenges especially if dealing with decentralized distributed systems and the security must be furnished as end-to-end. Unfortunately, existing solutions are insufficient to deal with these problems like CORBA which is considered a one-to-one relationship only, or DSAW which deals with end-to-end security but without taking into account the possibility of changing information sensitivity during runtime.

This thesis provides a proposed mechanism for enforcing security policies and dealing with distributed systems' security weakness in term of the software perspective. The proposed solution utilised Aspect-Oriented Programming (AOP), to address security concerns during compilation and running time. The proposed solution is based on a decentralized distributed system that adopts the multi-hop concept to deal with different requested tasks. The proposed system focused on how to achieve high accuracy, data integrity and high efficiency of the distributed system in real time. This is done through modularising the most efficient security solutions, Access Control and Cryptography, by using Aspect-Oriented Programming language. The experiments' results show the proposed solution overcomes the shortage of the existing solutions by fully integrating with the decentralized distributed system to achieve dynamic, high cooperation, high performance and end-to-end holistic security.

Acknowledgements

It is a pleasure to thank my Mother and Father, who made this thesis possible and gave me everything they have got in order to support me to reach this stage.

I offer my regards and blessings to my wife who supported me in all respects during the completion of the project. She has always been behind me.

I am heartily thankful to my director of study Dr. Kashif Kifayat, whose encouragement, supervision and support from the preliminary to the concluding level enabled me to develop an understanding of the subject. His help and guidance have kept me focused and enabled me to successfully complete this project.

I am grateful to my supervisors Dr. Bo Zho and Dr. Cristopher Carter for their help and guidance.

I would like to give sincere thanks to Professor Dhiya Algoumily, who offered me many words of encouragement, support and advice throughout my project. I would like also to thank Professor Qi Shi for his support and help.

I would like also to thank my friend Mohamed Abdlhamed who support and encourage me in different stages of this work. I would like to thank my colleagues and friends Ibrahim, Mohamed, Omer, Ahmed, Younis and Wajdi, for their friendship and advice throughout the PhD period.

At the finishing of this work, I would like to acknowledge and thank the Department of Computing at LJMU and the wonderful staff. With particular thanks going to Tricia Waterson, Ian Fitzpatrick, Neil Rowe, Steven Thompson, Paul Cartwright.

Lastly, all the thanks and regards to the Ministry of Higher Education and Scholarships department in my beloved country IRAQ that funded this study entirely. Continued thanks to the University of Anbar and College of Computer and IT, who allowed me this opportunity.

Contents

List of Figures

List of Tables

List of Publications

A. A. Thulnoon, B. Lempereur, D. Aljumeily, Q. Shi “Using Aspect Oriented Programming to Enforce Privacy Preserving Communication in Distributed Systems”, Second International Conference on Internet of Things, Data and Cloud Computing (ICC’17)

A. A. Thulnoon, K. Kifayat “Enforcing Access Control Models in System Applications by using Aspect- Oriented Programming: A Literature Review,” 10th International Conference Development in eSystem Engineering, France 2017.

Chapter One
Introduction

This chapter presents a brief introduction to security and privacy concepts in terms of the distributed system. It also provides introductions about the solutions components to address these security and privacy challenges, access control and cryptographic. This chapter highlights the current problems of distributed systems' security and motivation of this research. It also presents the programming paradigm, Aspect-Oriented Programming (AOP) the tool that we used to modularize the privacy and security concerns for the proposed solution. The chapter also outlines the project's aims and objectives, novel contributions, and finishes with the thesis structure.

1.1 Distributed System Security Challenges and Solutions

Coulouris [1] defines a distributed system as "one in which hardware or software components located at networked computers communicate and coordinate their actions only by passing messages". Systems that are scattered over geographical distances and provide different activities through communication, processing, data transfer etc are called distributed systems [2]; for example, web services offering to deal with a huge distributed system as single resource. There are many different concepts associated with distributed systems, including distributed file systems, distributed object-based systems, distributed Web-based systems etc. All of these systems have been built to fulfil the following objectives [3] :

- Transparency
- Openness
- Reliability
- Performance
- Scalability

This thesis starts with the focus on the security of distributed systems based on how the system face different types of threats. These threats could be in the following forms:

- *Interception*: An illegal attempt by unauthorized parties trying to access the service or certain data to make copies or just corrupt the information, thus disabling the system or service [4].
- *Interruption*: Destroying the Services or data and converting it to be unavailable, unusable, Examples of interruption threats include denial of service attacks, deletion of data, and corruption of data [5].
- *Modification*: Manipulating and changing the data in an unauthorized manner, thus changing the service to perform a different function from that originally intended [5].
- *Fabrication*: Generation of additional data or activity that would normally not exist. Examples of fabrication include adding an entry to a password database and replaying previously sent messages [6].

Besides these threats, there are different types of attacks that may cause serious damages on the systems workings like attacking the distributed system channels. For example, *eavesdropping* [7] when the attacker obtains copies of the transmitted messages, thus sniffing some sensitive information. *Denial of Service* [8] when the attacker floods the system channel with messages that can cause the service to stop.

To address all the above security issues, security and privacy concepts must be adopted. Although, these concepts are presented as the non-functional side of the distributed system, however the naturalism of the functional side (sending, receiving and processing) needs to adopt these concepts to save the information. According to the definitions, all distributed entities will be working together as one entity to accomplish the required tasks, thus sharing same data between them while preserving the locality for each individual entity. For example, the function performance of a machine relies on the results of sub-functions performance on the other machine e.g. Web Service Composition (WSC) [9]. This cooperation makes the data vulnerable when an unauthorized entity obtains access to sensitive information.

To investigate the security issues in distributed systems, there are three main fundamental security dimensions which should be applied to measure the strength of a system's security and the defense against different types of attacks. These dimensions are as shown in the following:

1. *Confidentiality*

The method that shows the ability to protect information from disclosure or exposure for those who are not authorized to access it [10]. In other words, this concept ensures that the access to the information is limited to authorized people only, who can handle and change according to the level of authorization [11]. There are some measures used to protect the confidentiality of information that are as follows:

- Information classification
- Secure document storage.
- Application of general security policies.
- Education of information custodians and end users.

Confidentiality can be compromised by the loss of a laptop containing data, a person looking over our shoulder while we type a password, an email attachment being sent to the wrong person, an attacker penetrating our systems, or similar issues [10].

2. *Integrity*

It is the ability to prevent information from modification or deletion by an unauthorized party [12]. This might include the unauthorized modification or deletion of data, or it could mean an authorized, but undesirable change or deletion of data. To achieve integrity, we might need to have the methods to protect against the unauthorized modification to data, but also need the ability to reverse authorized changes that have to be undone [10].

3. *Availability*

The ability for the authorized user to access the data when they need it without interference or obstruction, and the systems that provide this can appropriately resist or recover from attacks [11].

In order to deal with all these listed security issues in distributed systems, one needs to consider different types of cryptography and access control mechanisms.

1.1.1 Cryptography

A mechanism used to convert the plain file into unreadable file, thus transferring data safely between system channels in a way that the attacker cannot understand [13]. In addition, cryptography will prevent any unauthorized system entity from displaying the data.

1.1.2 Access Control

One of the accepted security solutions is Access Control (AC), it's a mechanism that is used to protect the information by controlling who can access the information, or even pieces of information, and where and when [14]. In addition, AC will give the authority to the authenticated person, or machine to perform one of the actions read, write or delete and this will be different in regards to the type of authority that the authenticated person has [15].

1.2 Privacy Preservation

Privacy preservation is an essential concern especially for applications that deal with sharing data such as healthcare, security, financial and other applications that deal with sensitive data [16]. Many governments, corporations and organizations desire to create an interface that amalgamates data sources to achieve a high level of knowledge base to reach accurate results and make a right decision [17]. The aggregation of data sources however will put the privacy of the data stored in these sources in a precarious position. Many methods, algorithms and models can be used to achieve privacy for applications that share their resources; however, the most important two main methods used for enforcing privacy are anonymity and cryptography [18].

1.3 Aspect-Oriented Programming

Aspect-Oriented Programming (AOP) is a software paradigm, that supports separation of concerns (SoC) [19], which play a major role in software evolution. In computer science, SoC is the procedure of separating a computer program into distinct features that overlap in functionality as little as possible. SoC can be achieved through modularity of programming and encapsulation with the aid of data hiding. Researchers have explored many methodologies in order to assess the reusability of Object Oriented (OO) software systems. AOP aims to modularize crosscutting concerns in an application, which can not be modularized using traditional approaches such as Object Oriented Programming (OOP). By using an aspect oriented approach, concerns like security and privacy can be isolated, resulting in the increased maintainability and reusability of the system [20].

1.4 Motivation

Dealing with the naturalism of the distributed systems is still an open challenge, because of the variety of requirements, environments, ability of expanding, updating, loosely–coupled problems and more. Indeed, all these problems represent the functional side of the distributed system; however, non-functional concerns like security and privacy protection will add more load to facing the above challenges. Adding privacy and security concepts to network system applications will increase the trust between the user and these applications, leading to an increase in the number of users willing to use them. Unfortunately, obtaining privacy and security assurances in a distributed system remains difficult; typically, in some places in distributed applications, there are no trusted relationships among participants of the networked system. Nonetheless many distributed systems need to be trusted because they handle sensitive and private data, such as clinical data, financial information, business-to-business transactions and joint military information [21].

The proposed solution deals with decentralized distributed systems(DDS), in which each individual node within the system works autonomously. Each node has both statuses (client and server) at the same time, so can send requests and receive requests

or response as well as perform its own function. The features of these systems are that sometimes there are no direct connections between nodes, so if a node sends a request, this request might do a multi-hop between different nodes to reach the right process. Furthermore, the processing of the requested task might require some nodes to process the task, called processing nodes, and data might be transmitted between some intermediate nodes which work as bridges to deliver data to the intended nodes, called bridges (hop) nodes. However, security concerns have emerged as a consequence of this topology like cooperation will be restricted or sometimes impossible, because according to the security concepts, data transmitted between system nodes might be restricted, especially if these nodes have different levels of security. One of the most widely accepted security models is based on a multi-level security system, in which system nodes will be classified according to their security clearances, and provides the authority to access data only if the latter has an appropriate classification. Keeping the cooperation principles with these systems is extremely difficult, because messaging between nodes which have the same security clearance might be impossible if data needs to transmit between some intermediate nodes which have been classified with greater security clearance.

One of the main concepts in distributed systems that adopt multiple-level security policy is to prevent data transmission from high clearance entity to lower clearance [22]. However, in some urgent cases, there is allowance for the data to transmit from a high security clearance point to a lower one after review of the data by security guard device or a human resource, to ensure there is no spill of sensitive data from high to low. This operation requires the use of specialized devices or Meta software between these nodes to work as security guards. Indeed, the last solution should be done after clustering the system nodes into sub clustering. Each one holds the same security clearance nodes, the nodes within the same cluster are connected in low security measures, and the data transmitted between the clusters will be controlled by security guards. This process is inadequate to deal with the systems that have dynamic naturalism as well as facing any incident change during running times, e.g updating security methods, auto-reclassifications of nodes and information.

1.5 Aim and Objectives

The aim of this study is to propose a security mechanism that can achieve security concepts for the decentralised distributed systems that works dynamically in real time, while preserving the performance of the whole distributed system network.

To achieve this security system, the proposed solution should be able to fulfil a high level of cooperation through allowing bidirectional connections between nodes, which might have different levels of security clearance, and in addition, eliminate the need to use Trusted-Computing Based (TCB) [23], which sometimes represents one of the major working conditions with systems based on the Multi-level Security System. The proposed solution should also allow software to be more flexible and able to be understood by separating the security concerns into distinct parts which makes it easy to understand and execute, and separates the security from the core functionality. Moreover, the system should permit data sharing across system sites while at the same time preventing the sites from sharing private data directly, and keeping the data in a protected environment during transmissions and sharing processing.

The main objectives of the thesis are as follows:

1. To perform detailed background research in the area of distributed system security.
2. To research literature in the field of Aspect-Oriented programming with security solutions, access control, cryptography and intrusion detection.
3. To research literature in the field of security solutions based on MLS without using AOP.
4. Based on our investigation, we identify the challenges, which surround the distributed system security. In addition, Identifying the gaps in the existing solutions, that proposed to address the security challenges in research area.
5. To develop an integrated security solution called 3AC_AOP which is a combined between three components; access control models, cryptography and data sanitization to ensure high integrity and confidentiality of the data forwarding and processing through the distributed systems entities.

6. To propose a technique for automatically reviewing the data that are transmitted between nodes that have different levels of security clearance.
7. To propose a dynamic technique that improves the performance of the data transmission and processing, and is integrated with the AOP technique.
8. To perform extensive evaluation of the 3AC_AOP by checking that all constituent techniques work successfully.
9. To compare 3AC_AOP against existed solutions, to measure the success and performance of 3AC_AOP.

1.6 Contributions and Novelty

This work introduced several novel approaches; in an attempt to tackle this, firstly, the work considered the implications of introducing end-to-end security [24] into systems designed without it. Point-to-point security is relatively straightforward, but gives a significantly weaker result, since it assumes the trustworthiness of all intermediate nodes [25]. As we will see in the literature review, some progress has been made on how to achieve security and privacy concepts by solving a specific problem, but questions remain, especially about the generality of these solutions. The contributions and novelty of the thesis are as follows:

1- Access control model.

This thesis presents an access control model modularized in both Object-Oriented Programming (OOP) and AOP. This model is a composition of three access control models:

1- Attribute –Based Access Control (ABAC).
2- Identity-Based Access Control (IBAC).
3- Multi-Level Security (MLS).

All these models are gathered to form a powerful model, starting from ABAC as the abstract level of the model and IBAC as the intermediate level, ending with MLS as a core level.

2- *End- to- end security*

The proposed solution deals with end-to-end security after detection of the security requirements to accomplish this task. We have combined between access control policies and cryptography algorithms to ensure the security and integrity of data transmitted between system nodes as well as processing data inside the authorized node.

3- *Privacy-preservation methodology.*

The proposed solution allows each node to follow the privacy-preservation methodologies of dividing the files according to the sensitivity level of the information. This division will be translated to aspect pointcuts and advices in order to integrate it with the access control model.

4- *Domination relationship*

The proposed system has employed a domination relationship, which represents a special case in MLS systems in order to achieve high cooperation between system nodes. This relationship works from high to low level by granting a high security clearance node to send a request to low-level nodes to perform certain missions on certain data. This supports the concept that not all information in high-level nodes is classified as being high sensitive.

5- *AOP security guard*

Adaptation of the domination relationship between system nodes requires filtering data between nodes. This thesis presents a novel approach of injecting security guards between system nodes called AOP security guards. This guard is a bidirectional automatic guard injected between nodes which have different security clearance levels. This side of the proposed solution is based on dynamic aspect-oriented programing, to arise only if data is forwarded from high to low level node, and in cases of processing only.

6- *Dynamic Multilevel Security System Clustering*

In chapter 6 we present a brief explanation about how we can utilize AOP to create a dynamic clustering between the nodes in the distributed system. This clustering is based on different security levels in which the neighbour nodes within the same level will be in the same security cluster in real time. Thus, decrease the cryptography processes between individual nodes and make it between clusters.

7- *Designed to face incident changes of access control policies*

The proposed solution is designed and developed to face all changes of access control policies that might occur during runtime when data is processed or transmitted between system nodes. Although the changes of the policy may impact on the information and thus cannot guarantee the response of the sent request, the proposed methodology guarantees the proper response, even if changing policies by keeping the source node ID and applying data sanitization methods on the receipt data.

1.7 Outline of the Chapters

This thesis is divided into 7 chapters, each covering a specific area of the project work. The following outline sections provide an overview for each of the chapters to guide the reader through the report.

Chapter 1 Introduction: This chapter outlines the fundamental elements of this work firstly, starting with a brief introduction of these elements following with a short clarification of the problem and motivation, followed by aims and objective of this research. The novelty and contributions part are also presented in this chapter. Finally, this chapter finishes with the thesis structure section.

Chapter 2 Background: This chapter deals with the proposed system's elements in detail. In fact, there are three main pillars that the proposed system is based on. These

pillars are access control, cryptography and Aspect-oriented programming (AOP). With access control, the author presents this part through dealing with some main access control models like RBAC, MAC, and DAC. We are dealing with ABAC, IBAC and MLS which are considered as the basis of the proposed system. In the cryptography part, we deal with cryptography algorithms of types symmetric and asymmetric. The final part of this chapter is focused on aspect-oriented programming, through highlighting the components of this language tool, how it works, and how to apply it to the code body.

Chapter 3 Related Works: This chapter navigates with three main security fields that have been designed and developed by Aspect-oriented programming languages. These fields are access control, intrusion detection, and cryptography. Although these files share the same concept of ensuring the security in application systems, they differ on the technical side of how to apply to these systems. This chapter has concentrated on the varieties and methods which have adopted AOP as a tool to insert the security and privacy concept on different systems.

Chapter 4 Proposed System: This chapter focuses on the proposed solution by dealing with access control model, cryptography model, and AOP security guard model. All methodologies, algorithms, and designs are presented in detail to show the benefit of using these models and what are the positive impacts of aggregating all these models to produce a powerful security and privacy model.

Chapter 5 implementation of the proposed solution: This chapter details the technical side of the proposed solution. It starts with a clarification of the distributed system that has been adopted, on which to apply the proposed solutions. Afterwards, we explain how to convert access control models from Object-Oriented programming to be treated as pointcuts and aspect advice with Aspect-Oriented Programming, followed by the cryptography method and AOP guard. This chapter includes snapshots of the original code and of AspectJ codes for all methodologies, as well as many algorithms which give the proposed solution more generality if we try to apply them by using different programming languages.

Chapter 6 Evaluation and comparing with existing solutions: This chapter shows the evaluation results of the proposed solution. Both OOP and AOP are used as a base for comparing the run time performance. Finally, this chapter finishes with comparing studies between proposed system results, and existing system.

Chapter 7 Conclusion and future works: This chapter covers the whole project and reviews the findings. It also outlines future work that can be done to improve the project.

Chapter 2
Background

This chapter provides a background of the fundamental elements which represent the infrastructure of the proposed system. Firstly, this chapter begins with the environment that we used to apply the proposed solution which is distributed systems and especially decentralized distributed system (DDS). The second part of this chapter focuses on security and privacy methodologies, starting with a brief explanation of the security concepts and ending by dealing with the most accepted solutions of security, which are cryptography and access control. With cryptography, this chapter presents the main algorithm types, and with access control models presents the main access control models as well as a detailed explanation of the access control models which are the basis of the proposal access control model. Furthermore, the data sanitization method is presented in this chapter. Finally, the aspect-oriented programming is explained in detail including most topics related to this powerful language tool.

2.1 Distributed system

Many definitions have been used to define distributed system concepts, the most acceptable to the author is the definition by A. Tanenbaum and M. Steen [26] when they defined the distributed system as:

"*A distributed system is a collection of independent computers that appears to its users as a single coherent system*."

This is exactly what the thesis is based on. Each node (machine, computer) is a fully autonomous node which has the ability to run and implement its own function independently, in addition to communicating with other nodes in the system. The intercommunication between system nodes enables the users of each node to feel that he/she works with only a single system. In practice, there are various layers between the user and the system being worked, these layers are called middleware and will be hidden from the users, making these systems relatively more easy to use [26].

In spite of the fact that each machine in a distributed system is an independent machine, this does not prevent the fact that some distributed system's machines are controlled by the same server to act and use the facilities of the connected machines to achieve the system goals. This will create a kind of confusion over the concept of independence, and these systems are called a centralized distributed system. Some distributed systems however, have enjoyed high independence though their machines have been designed to hold both statuses (client/server) all together, to create what is called a decentralized distributed system. There are four fundamental objectives which should be available in distributed systems and these are: resource accessibility, transparency, openness and scalability [26].

2.1.1 Types of distributed systems

1. Distributed computing system

This type of distributed system is used to handle complex tasks because of its high performance systems. The computational task will be divided into sub tasks and each will be processed by one or a group of machines to achieve a high level of efficiency [27]. For example, cluster computing networks, where each connected node within the clustering has the same operating system and shares the hardware, in order to create a high or super speed local-area network however, with grid computing, which is the second example of distributed computing, the structure is quite different. The individual nodes or groups will be constructed as a federation of computer systems in which each system might follow different domains, and thus might be dealing with various hardware and software [26].

2. Distributed information system

This type is utilizing the distributed computing systems to control and run data resources between the communication machines. Indeed, it is a combination of software that runs the data and hardware that runs the data storages and telecommunication network [28]. All these fundamental elements are gathered to prop up the cooperation, coordination, decision making and more distributed system objectives[29]. Generally, these systems represent the backbone of our digital life by

controlling most applications around use in automated or even manual processes; for this reason human resources are considered as a component of these systems associated with data, software, hardware, process and networking [30].

2.1.2 Naturalism of Distributed System Communications

1. Decentralized Distributed System

It is a distributed system in which the control and management will be distributed over nodes [30]. Decentralized systems provide a high level of autonomy for the system nodes to deal with their own data and software, without taking into account the impact on other system nodes. In other words, each node or a certain group of nodes will be under their own responsibilities, and own business destiny; for this reason, these systems are considered of high reliability. Moreover, DDS are more flexible and scalable than centralized distributed systems and any fault in any system node will not stop the whole system working [31]. The most observed example of these systems is unstructured peer-peer network such that each node within the network has connections with one or some nodes (neighbours). Some algorithms are used to control sharing files and searching tasks between the peers, and the data and queries will be flooded between the distributed peers to accomplish a specific task. Enforcing security policy over these systems is a major mission, facing scale-up problems, and autonomy for each node to change and update their software and policies will demand the security procedures to keep up with the changes. Figure 2.1-B- shows an example of these distributed systems.

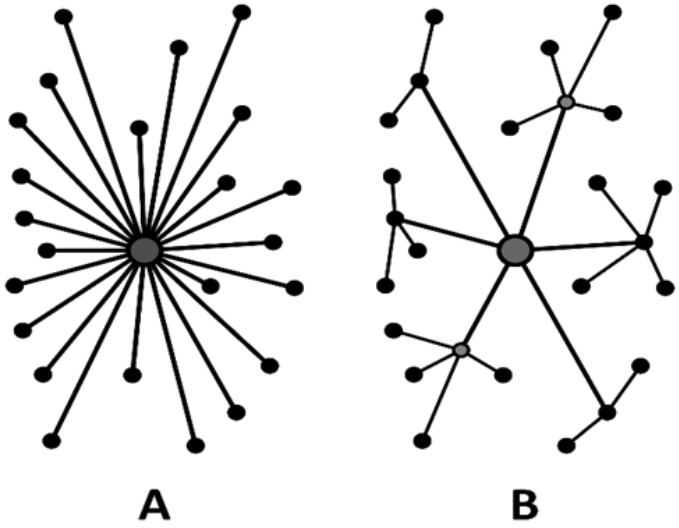


Figure 2.1. A- Centralized Distributed System, B- Decentralized Distributed System

2. Centralized Distributed System.

Contrary to the Decentralized Distributed System, these systems show more dependence on a central controller [31]. It is most popular because it is the easiest to be controllable, manageable, and maintainable because running and controlling tasks will be located in the central core. Many websites and enterprises adopt it, for example Facebook, Twitter and some other chatting applications in which there are servers controlling the communications and messaging between clients. A simple example of this type is a socket program (client-server) and multiple clients with one server, in which the server works as a communication bridge to satisfy the required messages and communication between clients. Applying security within these systems will be easier if comparing with the decentralized one, because the majority of these policies will be held in the server and the clients should follow any changing or updating in future. Figure 2.1A shows how the terminals are connected with the central point, to control the messages and communication services.

2.2 Distributed System Security

In order to increase the trustworthiness between distributed system nodes and the users within the distributed system, the availability of security principles should be considered. Although adding a security represents the most challenging task because of the difficulty of controlling and running different security policies throughout the distributed system's nodes. Security concerns in distributed systems have two main parts. The first part concentrates on securing the communication channels between system nodes. The best mechanism to deal with this part is to use cryptography algorithms to ensure the integrity of data transmitted between communication channels. The second part is the authorizations, in which accessing data resources will be granted only to the nodes which have the right authentication to use these resources. The mechanism that is used to deal with this part is called access control, and the access to data resources will be controlled by different access control policies. Recently, the research direct to the way that use cryptography to enforce access control or mixing between them[32], [33].

Instead of cryptography algorithms, access control models are the most popular solutions to deal with distributed system security, but this will add extra heaviness upon the system as well as the variety of applying these solutions on distributed systems. For this reason, many different access control models and various cryptography algorithms are adopted just to decrease the negative impact of using these optimal security solutions. In different sections within this chapter we deal with access control and cryptography in more detail, to prepare the entry to the design chapter which shows how we adopted these solutions to enforce security and privacy concepts in distributed systems.

2.3 Cryptography

Cryptography is a mathematical method applied to a plain text (clear, organized text) to transmit it to a cipher text (unclear, disorganized text). *Cryptology* is classified into two subjects. *Cryptography which deals with designing of the cryptosystem, and in*

contrast cryptanalysis used to break the cryptosystem [13]. One definition of cryptography [34] is "Cryptography is the hiding of and exchanging of information or data which is not in readable form over a public or private network". In my opinion, we partially agree with this definition when the author said exchanging the information would make it unreadable. However, to say "hiding of" does not make sense, because to hide anything we need a cover, and the cover must be clear and as visual as possible to prevent the phishers from doubting that there is something which has been hidden by this cover; this is what is called *Steganography*.

The historical background of cryptography dates back centuries to when Julius Caesar's letters were encrypted, by writing D for A, E for B and so on. After that, the Arabs generalized the idea to monoalphabetic substitution [13]. In the past the main reason for using this art is to protect the transmitted messages between sender and receiver from any kind of attacker or eavesdroppers especially in military cases.

Presently, the main reason is still with an additional extra motivation to face the challenges of data protection in WWW. We have listed some objectives that cryptology includes to create them:

Confidentiality: is to ensure data remains private and the eavesdropper cannot understand the content of it. This principle, applied to both transmitted message and stored data is to protect them from illegal access [35].

Integrity: To ensure that the receiver will receive the message without any modification and alteration which can happen through message transmission [36].

Authentication: To ensure that information came from the authenticated party. In other words, the receiver will recognize that the information is coming from the expected source and not from somebody else. This possibility is equivalent to a signature [35][36].

Non- repudiation: It proves that the sender has really sent the message that the sender denies having sent [35][36].

2.3.1 Type of Cryptography Algorithms

There are two main common types of cryptography 1) Secret Key Cryptography which is also known as Symmetric Key Cryptography and 2) Public Key Cryptography which is also known as Asymmetric Key Cryptography. In the next subsection, we will discuss each type individually and review the advantages and disadvantages of each.

1. Symmetric Cryptography

What distinguishes this type is both sender and receiver sharing the same key. In other words, the sender encrypts the message by using a key and sends the encrypted message to the receiver, which in turn decrypts it by using the same key. This type is known as a classical cryptography in which the key may be identical or there may be a simple transformation to go between the two keys [37].

Given an alphabet A we define A^* to be the set of all strings over A. In order to define a cryptosystem, we require a collection of sets:

A = plaintext alphabet	A′ = ciphertext alphabet
M = plaintext space	C = ciphertext space
K = (plaintext) keyspace	K′ = (ciphertext) keyspace

Where M is a subset of A^*, C is a subset of A'^*, and K and K′ are sets which are generally strings of fixed finite length over some alphabets (e.g. An or A′n). A cryptosystem or encryption scheme is a pair (E,D) of maps

$E : K \times M \longrightarrow C$

$D : K' \times C \longrightarrow M$

such that for each K in K there exists a K′ in K′ such that

$D(K',E(K,M)) = M$

for all M in M. We write EK for the map $E(K, \cdot) : M \rightarrow C$ and similarly write DK′ for $D(K', \cdot) : C \rightarrow M$. With this notation the condition on E, D, K and K′ is that $DK' \circ EK$

Indeed, Symmetric key algorithms are quicker than asymmetric key algorithms and most commonly used for encryption [36]. It is easier however to break them than the asymmetric ones. Moreover, there is a problem of the distribution of the symmetric key to be shared between Alice and Bob [38]. To solve this problem, a trusted key distribution center (KDC) has been suggested to manage the key distribution process. Figure 2.2 shows the symmetric Cryptography and Key Distribution.

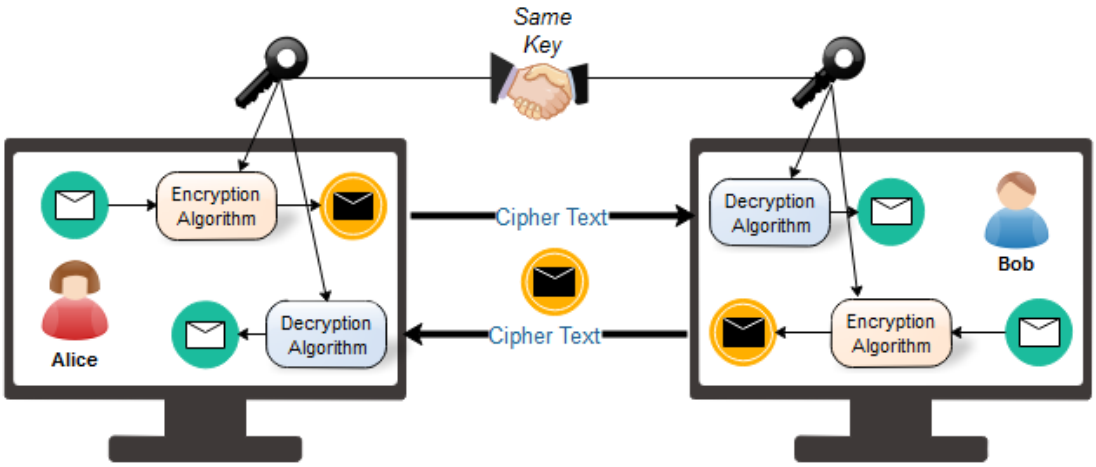


Figure 2.2.Symmetric Cryptography and Key Distribution

2. Asymmetric Cryptography

Also known as public key cryptography, this refers to the concept of this kind of cryptographic algorithm which depends on key pairs: one public key and one private key; both are required for the encryption and decryption process respectively[36][38]. Diffie and Hellman in 1976 came up with this method to fill the weakness gaps of symmetric cryptography, which are the distribution of the key and the concept of a digital signature [39].

The principle of this type of cryptography is: the sender encrypts the message using the receiver public key (published key), while the receiver decrypts the message using his private key (hidden key). The mathematical definition of this type is:

DEFINITION

1. A set $\mathcal{K}$ called the key space whose elements are called keys.

2. A rule by which each $k \in \mathcal{K}$ is associated with a trap-door one-way function E_k with domain M_k (the plaintext space) and range C_k (the ciphertext space).

3. A procedure for generating a random key k ∈ K together with a trap-door d for Ek and the inverse map Dk : Ck −→ Mk such that Dk(Ek(m)) = m, for all m ∈ Mk [40].

Figure 2.3 shows Asymmetric Cryptography and Key Distribution.

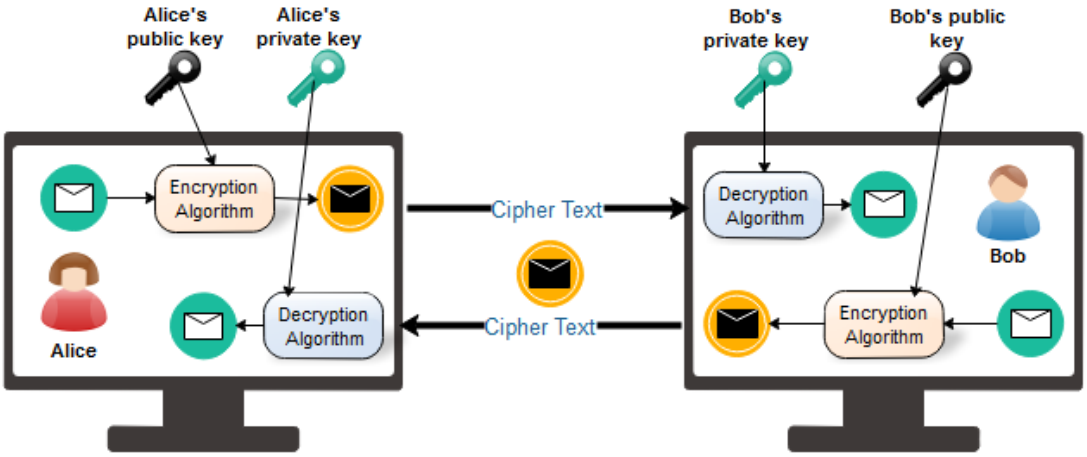


Figure 2.3. Asymmetric Cryptography and Key Distribution

2.4 Data Sanitization

Data sanitization is a method used to clean and remove data that has been classified as highly sensitive from the documents, to produce clean documents which can be released after making sure the documents are free of sensitive information. Data sanitization has different shapes and methodologies even if all sharing the same objectives, and is how to keep sensitive information on the safe side. Some of these methods are implemented normally, like when someone has a document which includes sensitive information, he/she would use a correction ink or any bold dark colour pen to strike off the sensitive information characters before sending to publishing. These methods are however not 100% practicable because, by using some chemical liquid,

the receiver can just clean the covering bold to distinguish the original information. The best way to solve this problem, is to do what we mentioned, as well as using sticky tape over the data without sending the original, but photocopy the result document and send the cloned version. In this case, the secret risk to discover the sensitive data will be minor especially if we use a very dark colour pen [41].

In a digital world, data sanitization has sanitization methods not too much different from the manual ones but it's a more difficult task. Redaction, which is one of the computational sanitization methods and provides a way to sanitize information in different aspects, for example in the context of security protection using cryptography algorithms to protect the information [42]. In the context of privacy preservation using anonymity methodologies, as well as data masking, has been adopted in Oracle Database since 11g as shown in Figure 2.4, or can erase data from a hard disk as well.

Similarly, as in manual methods however, there are some anti-sanitization methods used to retrieve sensitive information after sanitization is done. Unfortunately, some software products' features can be used as a tool to recover (anti sanitization) the sensitive information after sanitization. For example, about what happened in May 2005 when a US military report was talking about the death of an Italian secret agent called Nicola Calipari [43]. This report had been published after the name had been sanitized using commercial software and the publishing version was PDF format. Unfortunately, the publisher after a while discovered that the black portion (an agent's name) can be removed by just copying this part and pasting into the word processor. For this reason, dealing with such software should be more careful to discover all powerful and weak points before using this as a reliable sanitization software [44].

In the proposed solution, a sanitization method has been adopted to do the filtering and removing of sensitive information from the data which has been transferred to the node, classified as lower than the sender nodes.

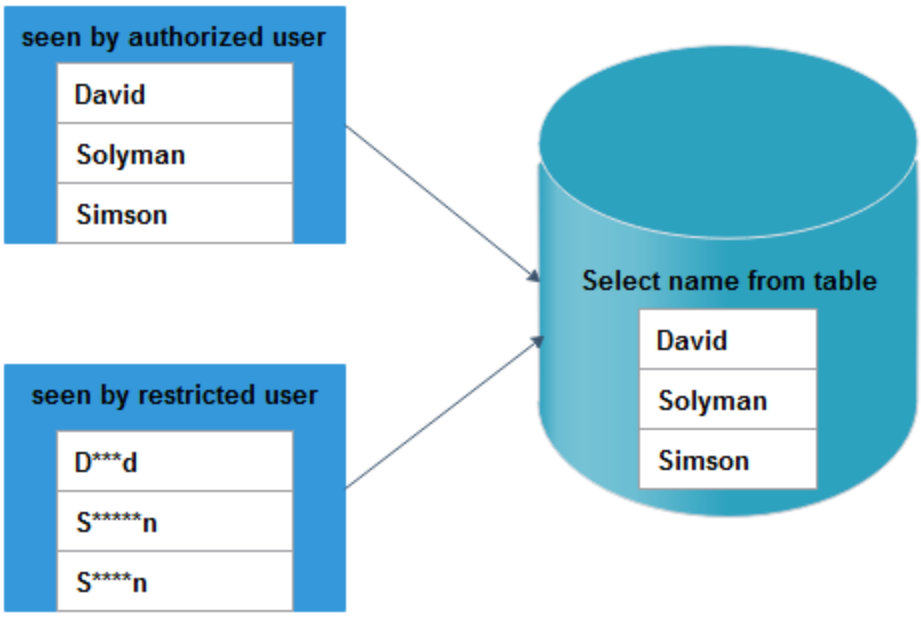


Figure 2.4. Data anonymization

2.5 Privacy

Privacy enforcement is an essential issue especially for applications that deal with sharing data such as health care, security, financial and other applications which deal with sensitive data [16]. V. Safanov [45] used the common definition of privacy, "Privacy is the ability of an individual or group to keep their lives and personal affairs out of public view, or to control the flow of information about themselves". He considered that privacy is one of the main pillars of trustworthy computing (TWC). Many governments, corporations and organizations wish to create an interface that collects databases to achieve a high level of knowledge base to find accurate results and help in making a right decision [17]. The aggregation of databases however, will put the privacy of data stored in the database in a precarious position. Many methods, algorithms and models can be used to achieve privacy for applications, which share their resources. The most important two methods used however for enforcing privacy are anonymization and cryptography [18]. The highlight topic that privacy preservation revolves around is Privacy Preserving Data Publishing (PPDP), which is a method for for publishing data in an untrusted environment, while at the same time keeping it

practically useful while individual privacy is preserved [46] [47]. Figure 2.5 shows data collection and data publishing in database system.

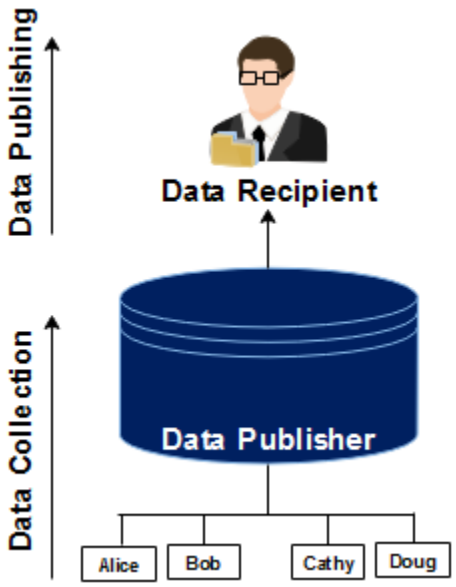


Figure 2.5. Data collection and data publishing [48]

2.5.1 Tables and Attributes

A database can be represented as a table or collection of tables which either have a relationship between them, or are stored separately in a storage device. These tables have the ability to save the information in different files and formats. The dominant purpose of a database table however is to store binary information. The attributes (columns) of database tables are the linchpin which is surrounded by privacy issues. All the privacy preservation methods agreed to divide the attributes of the table into four different types of attributes, taking into consideration the probability of all of these types not being present in the same table.

Identifiers (I): There are some attributes that fully and distinctly lead to identify a person. These attributes, also called identical attributes, include SSN, passport number and full name; such attributes are removed before publishing. In some cases, [49], more

than one identifier is required to uniquely identify the individual. For example, the name “John Smith” is a popular name and appears hundreds of times in the searching of public telephone directories in a certain city, however, combined with a telephone number, the individual can be more easily identified uniquely [49].

Quasi-identifiers (QID): Types of attributes used with some extra knowledge to do some statistics, analysis and linking process with the anonymized table in order to uniquely identify individuals. These attributes are not less dangerous than the identifiers, because they are used to narrow the range of searching and thus, release the information for individuals or groups. Example of these attributes are, postcode, gender, age, religion and so on. Sweeney [49]’s study showed 87% of the US population can probably be directly identified by using only the QID (zip code, gender, age). The major dilemma here is the data publisher’s decision to establish which attributes should be treated at quasi-identifiers, because adding too many attributes will impact the data utility, while too little will increase the risk to the privacy of the individuals [49].

Sensitive attribute(S): These attributes have a sensitive value for the individuals, for example, diseases, salary and so on [50].

Non-sensitive attributes (NS): after having classified the table into identifiers, quasi-identifiers and sensitive attributes, the rest of the attributes represent the non-sensitive attributes [51].

For the general form of PPDP, the data publisher has a table of the form

D(Identifiers, Quasi-identifiers, Sensitive-attribute, Non-sensitive-attributes) [52]**.**

2.6 Access Control Models

Access control (AC) which may also be referred to as authorization, is a mechanism used to coordinate and control the interactions between the users and data resources in

a way that only authorized users are allowed access to these resources [53]. Access control has been broadly used to organize the dealings with different sides in our life, for example controlling the access of employees to their offices within enterprises when the employer's ID card represents the access clearance of the employers and so on. In a context of computer and system application, access control represents the backbone of much security software such as in database management systems, where the need for access control is essential to give authorization for the user to access only what they are allowed to access, otherwise the access will be denied. In some systems which are based on working on sensitive data, like the healthcare system, access control is very important to preserve the privacy of patient's records and thereby increase the recommendations of the system. By using user name and password or sometimes with more security steps, the user can have access to data resources to do his/her own authorization actions, according to access control policy [54].

There are many access control models that have been used in past decades and currently. In this section we will discuss briefly the main access control models: Role-Based Access Control (RBAC), Discretionary-Based Access Control (DBAC), Mandatory-Based Access Control (MBAC), and finally we finish this section with Attribute-Based Access Control (ABAC) and Identity-Based Access Control (IBAC) which represents the pivot of the proposed work.

2.6.1 Role-Based Access Control (RBAC)

RBAC is an approach based on restricting access to data resources according to the roles of the user, which gives him/her a privilege that authorizes them access to the permitted resource only. The user's role is the main concept of this model and can be seen as a user's job within an enterprise or organization. Giving permission by assigning privileges to the user is totally based on the user's job duties or qualifications. Roles is the intermediate layer between the users and data resources [55]. Permission is updatable by increasing or decreasing the limitation according to changes in users' roles. Roles can be seen as a group of transactions associated with data items in which each role is assigned by an individual's organisation membership. The RBAC concept

can be represented as a many-to-many relationship between subjects and data resources [56].

What distinguishes this model from another is it is the easiest way of achieving integrity and availability of the system by explicitly controlling access to resources, as well as how access can occur [57]. RBAC was founded to decrease the administration difficulties which occur when dealing with large and commercial organizations, when sometimes using DAC or MAC may not be an appropriate option to apply access control policy [58]. Hu Vincent and others described RBAC in detail through present RBAC terminologies which are Object, Operations, Permissions, Roles, User, Group, Constraint, Session and Role Hierarchy. Objects, user(s), permissions and roles represent the pillars that the model is based on, and the rest of the terminologies are used to organize the model working [59].

For example, in the context of the organization, when the user *U* tries to access an object *O* to do a specific operation *OP*, then the procedure will take these steps:

1- U will open a session with the role in order to map him to his assigned roles

2- Roles will assign u to a certain privileges pr.

3- According to this assignment, will detect *U*'s permission and the ability to perform the operation op upon objects.

Figure 2.6 shows the interactions of RBAC's components.

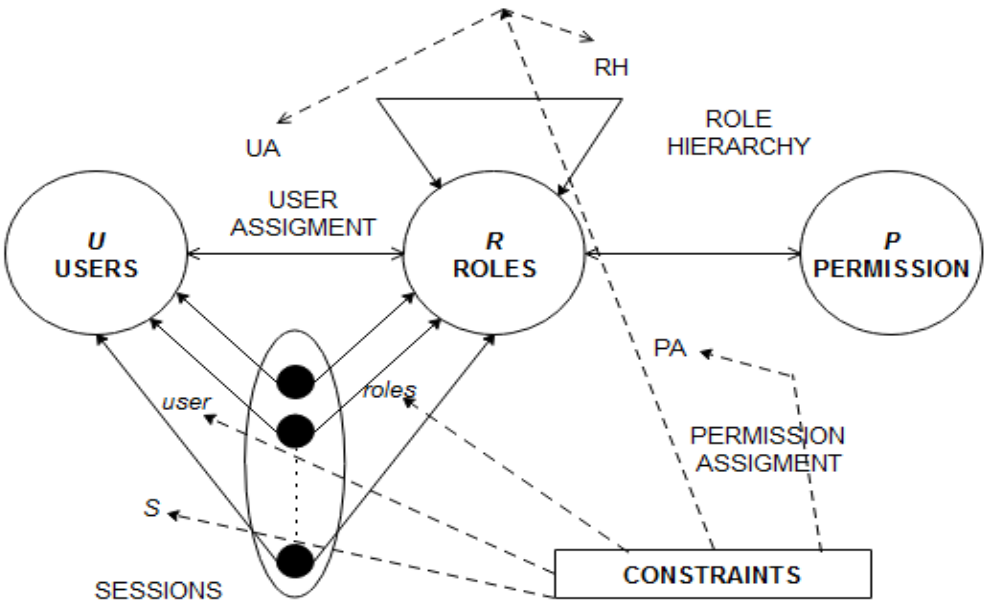


Figure 2.6. Role-Based Access Control (RBAC)[60]

2.6.2 Discretionary Access Control (DAC)

DAC is one of the common access control models. This model adopts an object's ownership relation in which the owner of the object is responsible to grant access permissions to the subject, thereby the subject with a given discretionary access to a data resource has the ability to pass the access to another subject [56]. Many operation systems have adopted this principle of access control model like UNIX, Windows2000, and FreeBSD. In fact, this mode is based on implementing the Access Control Matrix, which can be seen as a three-dimensional matrix where rows are subjects, columns are objects and the mapping between them represents the permission that the subject has over the object [57]. Despite the high reliability of DAC, the latter may be vulnerable to an intensive risk, coming especially from the user. For example, the user can violate the other's right, like the classical "chmod 777" which gives permission to anyone in Unix/Linux system. Moreover, the problem of transactive read access, when a user A has a permission to read a user B's file because the latter has gave him this permission, there is not 100% assurance that user A is trusted. In other words, user A can copy a file's contents and save it in a different name, thus giving the other user a right to access this file, considering that user A is the original owner [58].

Finally, this model is not designed for dealing with big systems which have large numbers of users and objects, the Access Control Matrix will be huge and this leads to the system maintenance becoming enormously difficult, because it deals with a large number of users having varying access rights to their own resources [57].

2.6.3 Mandatory Access Control (MAC)

MAC is widespread in systems which have adopted Multi-level Security access control as a working base. Basically, this model is associated with the Bell-LaPadulla Model [61]. This model restricts the access allowance based on the subject's clearance and object's classifications. In other words, the security for both subjects and object will be divided into four main levels: Top-secret (TS), Secret (S), Classified (C), and Unclassified (U) in which TS > S > C > U [1]. The security level which is related to a subject is called clearance and that which is associated with objects is called classification. Accessing from subjects to objects will be controlled in a way that prevents a low level subject from viewing a high-level object, thereby preventing information from spilling [58]

Data flow in this model is one-directional from low level to high level and not vice versa. In this model users have no authority to organize the access to data like DAC, but the security policy of this model is totally controlled by the security policy administrator [62]. In our view, the strategy of the inventory dealing with data by only a security policy administrator will increase the reliably of the system. Through restricting the access law to one person, this is better than scattering this law between different system users, which may lead to a rise in tangling as well as exposing the data to illegally disclosed risks.

The main principles of this model can be seen through two properties [62]:

Simple Security Property: restrict reading by a subject *s* to an object *o* only and only if the security level of subject is greater than or equal to the security classification level of an object, $SL\ (s) \geq SL\ (o)$ *where SL=Security level.*

*-Property: restrict writing by a subject *s* to an object *o* only and only if the security level of subject is lower than or equal to the security classification level of an object, $SL(s) \leq SL(o)$.

For more information about this model, the reader can see the Multi-level Security section, where this model is discussed in more detail.

2.6.4 Attribute Based Access Control (ABAC)

ABAC is the model where the direction of accessing rights to the data resources is totally restricted to the attributes of objects (requestor), subjects (distention), service and sometimes even the attribute of system environment [56]. The National Institute of Standard and Technology (NIST) SP 800-162 [59] has defined Attribute Based Access Control (ABAC) as:

> *" **Attribute Based Access Control (ABAC): An access control method where subject requests to perform operations on objects are granted or denied based on assigned attributes of the subject, assigned attributes of the object, environment conditions, and a set of policies that are specified in terms of those attributes and conditions** "*

Given the importance of this model in the proposed system, we will be dealing with it in more detail than others because it is the kernel of the proposed system. The core of this model incudes some categories used to classify model attributes see Figure 2.7, these categories are [63]:

> ***User Attribute***: this is the attribute related to the subject, and may include name, age, office number, job title…and so on. In fact, these attributes are divided into two sets, static attribute like name and gender, and dynamic attributes like age, office number, job title and so forth.
>
> ***Object Attributes***: Attributes of data resource of the system. May include the attribute of meta-data of the objects like the file creator, modify date, file size and so on, or the attribute of the contents like the columns address in database tables.

Environmental Attributes: These attributes come from measuring the state of system environments currently; for example, CPU usage, day of the week, current time and so on.

Connection Attributes: these attributes express the current connection session of the user like IP address, physical location.

Administrative Attributes: Attributes have been set manually by an administrator. These attributes will be enforced to the whole system, for example a threat level, minimum trust level and the like.

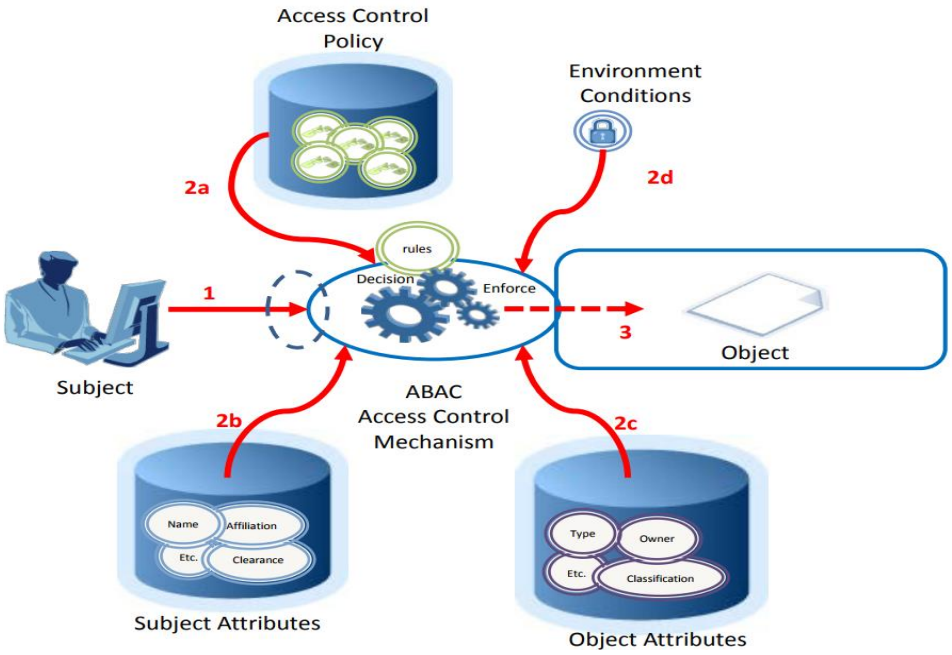


Figure 2.7. Attribute-based access control model[59]

Other aspects of the component of the model are [62]:

users (U), subjects (S), objects (O), user attributes (UA), subject attributes (SA), object attributes (OA), permissions (P), authorization policies, and constraint checking policies for creating and modifying subject and object attributes. In which U associates with a set UA and S is created by U to do some actions in the system. Each S in turn is

associated with a set of SA. O is the data resource which needs to be protected, Each O is also associated with OA. Giving P from the system administrator to the subject to access required objects is totally dependent on the relationship between these two components and what the matching level of their attributes are.

2.6.5 Identity-Based Access Control (IBAC)

IBAC is an access control model which is used to restrict access to the data source only to the authorized user or group of users who have the intended identity. The identity here means the id of the user or job title or what service provider, or anything that makes the user unique from other users or group of users; different from other groups if we take the nature of the work as a basis for identification. IBAC has been defined by the National Institute of Standards and Technology (NIST) in source: SP 800-53; CNSSI-4009 as [64]:

"Access control based on the identity of the user (typically relayed as a characteristic of the process acting on behalf of that user) where access authorizations to specific objects are assigned based on user identity"

In the proposal system we used Identity-Based Access Control (IBAC) as the second layer between the Attribute-based access control (ABAC) and multilevel security (MLS) to achieve a high level of accuracy through directing data to the intended nodes only in distributed application environments.

2.7 Multi-Level Security (MLS)

Despite many of the security policies having been suggested in the past and present, the Multi-Level Security (MLS) policy remains the pioneer in terms of its adaptation by the highly sensitive system. The Department of Defence (DoD) in U.S [65] has adopted this type of security policy for having a high capability to deal with both the data on one side and the users on another side. MLS systems depend on data by giving it different labels according to its degree of sensitivity, and users through giving them different clearances according to their sensitive position within the enterprise. Through these divisions, access to the data will be restricted according to the data and the user

who wants to deals with it. Therefore, even the functions (read/write) will be restricted according to users' classifications.

According to DoD and some other organizations which adopt MLS, the user will be assigned to one of the four clearances (Top Secret, Secret, Classified, Unclassified); in contrast, data will be assigned to one of the four labels (Top Secret, Secret, Classified, Unclassified). The general idea of these classifications is that each user who has a specific clearance can have access to the data which has the same level or lower. The function of this access is (read/write) in the case where user and data have the same level, and moving to read only in the case where a user has a higher level than data, as shown in Figure 2.8.

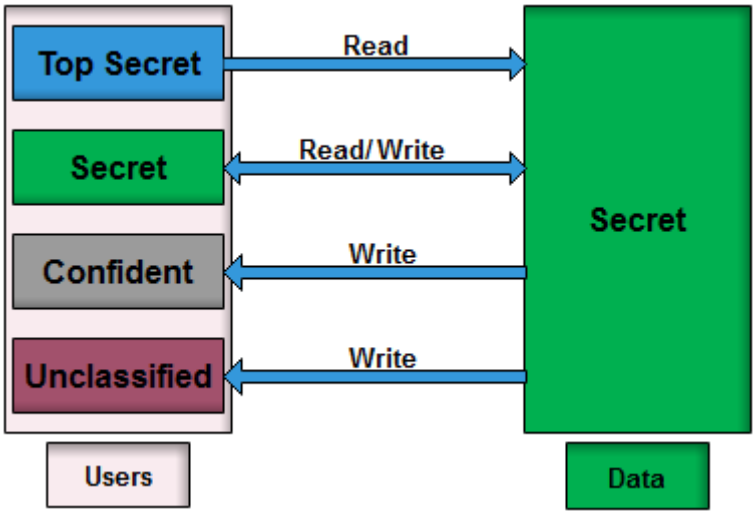


Figure 2.8. Function permissions between the users and data in MLS

Although this type of security policy is popular among other policies, this does not prevent the presence of a dark side. Indeed, there are many problems that can be represented through restriction of data flow between systems entities and poor data integrity; there is no 100% guarantee that the data will not be leaked, as well as the possibility of attacks (i.e. Trojan horse)[65]. In this chapter, we review MLS models, deal with the classification of DoD for both data and users, and discuss the drawbacks of these models and what is the most acceptable solution.

2.7.1 Security classification and clearance

As we mentioned before, the object side of the MLS models will take one level of security (T, S, C, U). Sometimes the unclassified level does not appear in security levels set on the ground, in that if the information does not occupy any from the first three security levels, we already consider it as unclassified. The simple hierarchy of security levels is shown in Figure 2.9. The arrows in the Figure refer to the direction of data flow of MLS: form "low level" to "high level" and not vice versa.

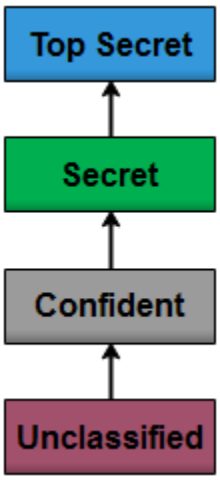


Figure 2.9. Security and Classifications Levels

The Government of Canada (GOC), see Table 1 has classified information in a sensitive category, where an unauthorized reveal can be defined in terms of national injury, and applying the Protected type where the injury is defined according to a person or organization [66].

Table 1 GOC information Sensitive [66]

Information Sensitivity	**Classified (i.e. National interest)**	**Protected (i.e. Individual or Organization)**
Unauthorized disclosure could cause exceptionally grave injury	Top Secret	Protected C
Unauthorized disclosure could cause exceptionally serious injury	Secret	Protected B

Unauthorized disclosure reasonably expected to cause injury	Confidential	Protected A

Clearance Level refers to the trustworthiness level which has been given to a person with a security clearance, or a computer system which processes classified information, or a storage device which has been physically secured for storing classified information.

Classification level refers to the level of sensitivity of information. These levels will be associated with the information in different formats like document or computer files. Disclosure of this information could cause a national disaster, according to the sensitivity level of this information.

Security level is a term referring to either a clearance level or a classification level [67]

The sensitivity of the information will be labelled and mapped to an appropriate domain, which represents the security level of the object. Domains can be broken down by caveats, which are levels of sensitivities restricted to access by specific groups or categories. According to GOC, domains are classified as Top Secret, Secret, Designated and Unclassified [67].

2.7.2 Bell-La Padula Model

This is a distribution of a "mathematical model of security in computer systems". In the early seventies of the last century, Len La Padula and David Elliot Bell were asked by the MITRE Corporation to produce a report entitled "Secure Computer Systems". They produced a security model called after them: the Bell-La Padula model [67]. This model is widely used in MLS systems, because of its potential to cover the fundamentals of the access restrictions that are used in MLS systems [68].

The basics of the Bell-La Padula model's working falls into two terminologies: *subjects* which represent processes and programs (active elements of the system that execute actions), dealing with *objects* which form information resources like files, I/O devices, messages and so on. Both subjects and objects carry security labels, in subjects these

are called clearance levels and in objects they are called classification levels of information. Permissible access is controlled by the relationship and the matching between subject's clearance and object's classification, therefore the access matrix will show how the subject is allowed to get the object according to the level matching relationship [69]. The operations which are to be applied by subjects to objects is called the access mode and include {read, write, execute, append}. MLS access restrictions are enforced by the Bell-La Padula model, through implementing the following properties and rules:

- ***Simple Security Property:***

An access mode {read} can only be done if a subject clearance is higher than or equal to the object's classification. For example, a subject with a Classified clearance cannot access (read) an object with a Secret or Top-Secret classification. This rule is called "No read-up" [70]

- ****-Property:***

An access mode {write} can only be done if a subject's security clearance is lower or the same as the object's classification. It is a way of preventing write-down actions happening when these are allowed. For example, a subject with a Top-Secret clearance cannot write an email to an object with a Secret classification. This rule is called "No write-down" [71].

Now both properties are obvious: the simple security property prohibits users from reading data with classification over the top user's clearance "read-up", at the same time it prevents "read down" which means reading data which has the same classification or lower than the user's clearance [68]; while *-property will restrict the higher clearance users from passing their sensitive data (high classification) to the users who don't have suitable clearance [68]. Enforcing these two properties will help to protect systems from a Trojan horse, such as leaking information or OS viruses [71].

The two principles "No read-up" and "No write-down" can be represented by the domination relationship as follows [72]:

We call class A as dominating Class B if the security clearance of Class A is greater than or equal to security clearance of Class B.

We call class A as being dominated by Class B if the security clearance of Class A is lower than security clearance of Class B.

Figure.2.10 shows the accessing operation from subjects to objects using different security levels.

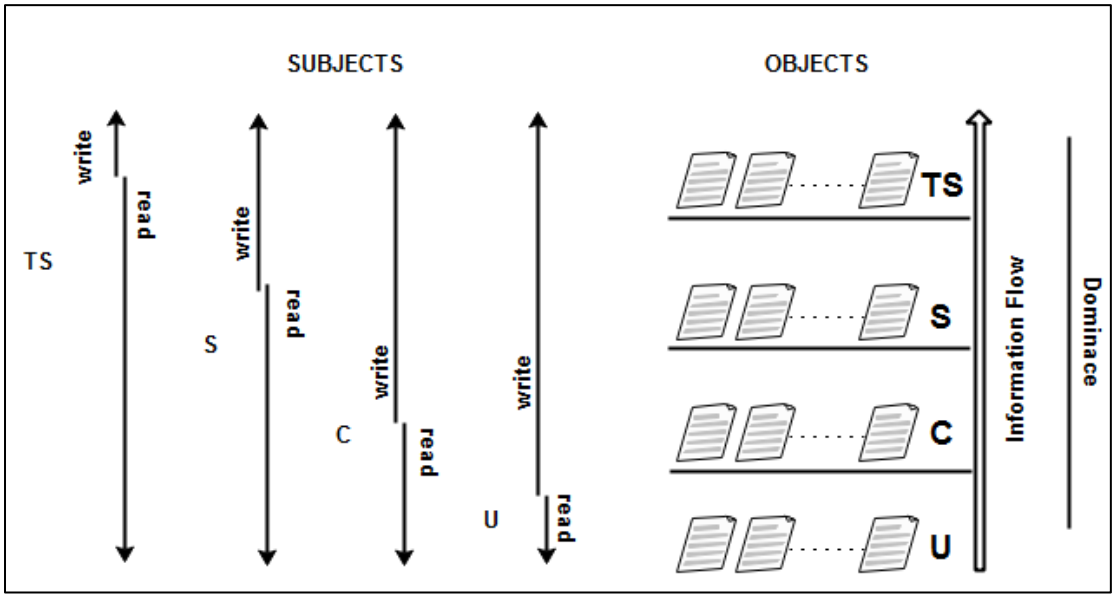


Figure 2.10. Information Flow and Domination Relationship in MLS [60].

Although both properties are used in BLM, both properties are also in need of support by two extra rules and properties, to control the data flow between MLS system applications. These properties are:

Strong Tranquility Property*:* which mean no changing of security label will happen during system work [73].

Weak *Tranquility Property**:* which mean no changing of security labels in a way that are inconsistent with the defined security properties [73].

2.7.3 Biba Model

Sometimes the security model focuses on one of the security principles (validity, confidentiality and integrity) at the expense of the others. For example, BLM

concentrates on confidentiality at the expenses of integrity, because there is a "No write down" rule but no "No write up" rule. In this case for example, the subject with a Secret clearance can write to a Top secret classification object; therefore, this may lead to corrupting the system object if the lower subjects try to distort higher security objects [74].

The Biba model addresses this problem by using two simple rules:

- ***Simple Integrity Axiom:***

A subject with a specific clearance cannot read data at a lower classification level. This is called "No read down". This will protect the integrity of the system through restricted access of the subject to information at a lower integrity level[1] and thus prevent bad information from moving up from lower integrity levels [74].

- ****Integrity Axiom:***

A subject with a specific clearance cannot write data at a higher classification level. This is called "No write up". This will protect the integrity of the system through preventing passing information up to a higher integrity level [74].

2.7.4 Lattice –Based Access control

This type of access control is used to control the security in complex environments. There are lower and upper limits implemented by the system to organize the relationship between the subjects and objects. Depending on the need of the subject, the lattice model allows reaching the higher and lower data classification of the object, and the security clearance that is assigned to the object. Access operations are based on two bounds, greatest lower bound (GLB) and least upper bound (LUB) which are used by the subject to access the object on their lattice position[75].

[1] Integrity level is a terminology which has two meanings: integrity level of a subject which means the trustworthiness level of the subject and integrity level of an object meaning the trust level that can be assigned on the information stored in the object [74]

2.7.5 Restriction Marking

In order to strengthen the restriction and make the subject achieve the appropriate objects only, some organizations have introduced new markings to classify their subjects and objects more deeply. These markings play an important role associated with the hierarchical security levels to customize the general meaning of MLS's concepts. In the following we review some of these markings and show how they affect the accessing restriction, while at the same time keeping the main principles of MLS rules.

- ***Compartments***

  This is additional marking associated with the security level for both subjects and objects. For example, if a file with a specific security level has one or more compartments, in addition the subject's security level must meet the file's security level, and the subject should include the same compartments of the file to achieve full accessing, or otherwise the subject won't be able to read the file [68]. This type of marking is used widely in the relational directed graph called a lattice [68]. For example, suppose we have two objects Obj1 and Obj2, both have a security clearance Secret and Top Secret respectively, and let C have a compartment set such that C={Black, White, Grey}; the data flow between system and object must be treated with the flowing formula:

  Let A→B is a data flow relationship, such that we say that information is flowing from A to B and this is accomplished if $B_{compartments}$ subset of $A_{compartments}$ as illustrated in Figure 2.11.

"Need-to-know" (NTK): is the ability applied when the subject has been granted access to a piece of information to perform some processing on it. This concept is enforced with Mandatory or Discretionary Access Control [76].

"Right-to-know" (RTK): is the general ability of a subject to get partial or total access to a specific object, according to matching between the subject's clearance and object's classification. This concept is widely used with Mandatory Access Control [76].

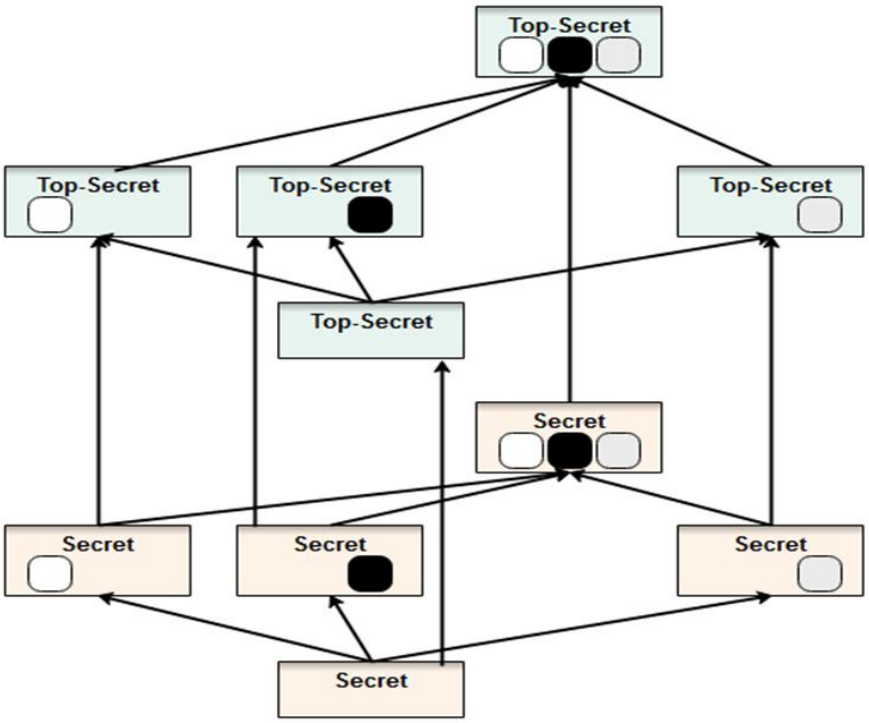


Figure 2.11. Compartments in MLS

2.7.6 Reference monitoring

This is a concept based on the relationship that allows system entities called subjects to make reference to other passive entities called objects, depending on the access authorization between both. A security kernel is used to prove the concept of reference monitoring through ensuring that every change comes on authorization, or any reference by subject to object must go through Reference Monitoring [78]. T. Jaeger [79] defined the reference monitoring concept as: a design requirement used to organize the referencing mechanism in the system and control the accessing between subjects associated with specific authorization levels and the ability to run some operations (read, write…) on the objects within the system, as illustrated in Figure 2.12

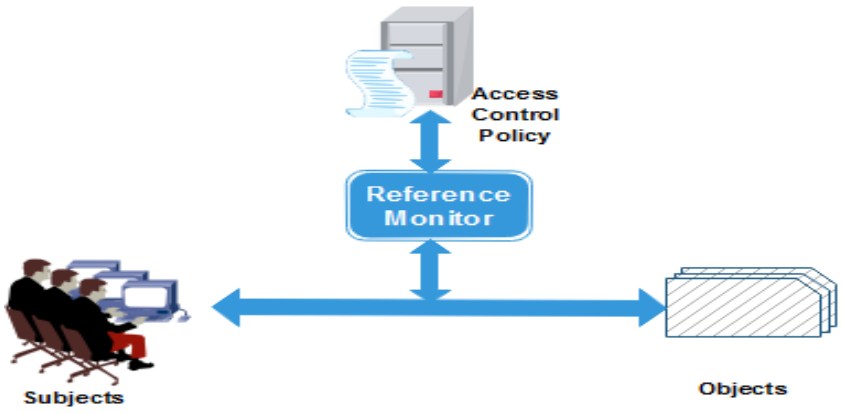


Figure 2.12. Reference Monitoring

2.7.7 Trusted Computing Based

This is a collection of all security and protection mechanisms within a computer system and this includes hardware, software, controls and processing; all these elements are combined to be responsible in enforcing the security policy over the system [78]. The TCB is responsible for satisfying both integrity and confidentiality of security requirements as well as monitoring four fundamental system functions [80]:

- ***Input/output operations***: is a monitoring of I/O operations through previewing the outermost and determining which appropriate security protection it may need.
- ***Execution domain switching****:* is monitoring of different invocations between entities working in different security domains.
- ***Memory protection****:* is a memory reference monitoring in order to ensure integrity and confidentiality of storage device.
- ***Process activation****:* is a monitoring of different system processing actions like registration, process status information and access file lists as these activations are considered vulnerable points in multiprogramming environments.

2.7.8 Multi-level Security Problems

Despite precautions taken by MLS systems, this does not however guarantee 100% that the information will not be leaked. Indeed, these systems may be suffering from some intended or non-intended misuse which leads to serious ramifications through threatening national security in terms of the government's systems, or lost files or even breakdown of systems at the level of enterprises. In fact, most of these disasters (so to speak) are caused by data flow between system entities or sometimes any flaw in the system software, even if it is tiny, which may put the whole system under threat. In the following, we will review some of these problems briefly, and we will be concentrating on the cross domain problem as representing the heart of the proposed system.

1. Covert Channel

In the DoD Trusted Computer System Evaluation Criteria, the "Orange Book [81] defines the covert channel as "*any communication channel that can be exploited by a process to transfer information in a manner that violates the system's security policy*". It is a mechanism of information flow by channels that were not designed for communication and not controlled by the security policy of the system, and thus may allow data flow from high to low level [82].

For example, a low level subject creates a file F1 at its own level. If a high level subject has access to this file it either upgrades the security label of the file or leaves it the same. If a low level tries to read the file, this case has two probabilities, 1) success, for disclosing the high level action if the latter did not change the file security level, thus one bit of information will flow from high to low, or 2) failure if security upgrade has been applied to file.

There are two types of covert channel [83]:

- *Storage channels* involve the direct/indirect writing of object values by the sender and the direct/indirect reading of the object values by the receiver.
- *Timing channels* involve the sender signalling information by modulating the use of resources (e.g. CPU usage) over time such that the receiver can observe it and decode the information.

There are some channel terminologies related to covert channels, like a side channel where the sender leaks the information unintentionally and only the receiver wants the communications to succeed. The steganography channel is a collusion between sender and receiver to hide the communication in a way that the observer cannot realize whether the communication has happened or not. Finally, a subliminal channel, where, using a cryptography algorithm in *a covert channel, in this case, the communication will be undetectable* [84].

2. Polyinstantiation

This problem arises when the low level has knowledge about the high level. For example, let's suppose that a high-level object tries to create a file with the name salary, and an object with low-level security try to create a file with the same name. In this case, if the MLS system prohibits the low object to do this, then it will cause leaked information, because the low object will know that there is a file called salary in the high object; at the same time if the system allows the low user to create the file, then we have a problem, two files have the same name [82].

3. Cross-Domain

Before talking about cross-domain we need to understand what is the information domain? The Information domain can be described as a silo or system entity which has information labelled by a certain security level [85]. Cross-domain security deals with data transformation between system entities which have in advance been assigned to varying security, authority and mutually trusted levels [86]. For example, at the enterprise level, cross-domain flow arises when the manager shares information with the heads of departments, and the latter share this information with employees. This problem worsens when there is a need to share information between networks related to different organizations or even governments. This sharing between different security levels exposes the information to risk, through increasing the likelihood of sensitive information being spilt to a specific destination with a certain security level which is not allowed to obtain that information.

A trusted solution for this problem is called the Cross-Domain Solution (CDS), which has been defined clearly by the Committee on National Security System (CNSS) in

Canada as “A form of controlled interface that provides the ability to manually and/or automatically access and/or transfer information between different security domains” [87].

CDS falls into three types:

- ***Access Solution***

This is represented as a subject’s ability to access to read and manipulate information from domains which have different security levels. The ideal solution is to prevent the overlap data between different domains preventing information spill between separate domains [67].

- ***Multi-level Solution***

This is different from above solutions which are based on domain separation. In this solution, all information will be stored in one domain, and dealing with these data will be controlled by using Mandatory Access Control (MAC) [67]. Applying this solution is expensive.

- ***Transfer Solution:***

This is the ability of information to transfer between varying domains with respect to data sensitivity and security policy of domains, in order to prevent violations even if by accident [67].

2.7.9 Guards

A guard is described as a combination between hardware and software used to ensure security of data transfer between different information domains. The main function of the guard is to do an inspection upon information to prevent leakage of sensitive information to a wrong domain [85].

- **Low to high guards** Sometimes called a one-way filter in which the data is transmitted in one direction only, from the domain classified as low to the other domain classified as high. The working of this type of guard totally depended on the concept of the Bell-La Padula model to prevent the spill of information

in the high domain. This is achieved through applying “read down”, write up” policy. Thereby, the users cannot read information from a domain classified higher than their own, simultaneously they cannot write information to a domain classified lower than their own. The way that information flows from low to high however will not guarantee the integrity of the data being transferred. This type of guard is popular in the environments that do not need periodic checking of file integrity in case of file manipulation or corruption. For example, weather data which is desired by a pilot or ship at sea, at the same time this data is frequently updatable, there is no need to check the integrity each time [85].

- **High to low guards** as with the previous guard, this is a one-way data transmission, but in this type the information is being transmitted from the domain classified as high to the domain classified as low. This guard is quite complicated when comparing with the low to high guard. Many precautions need to be taken into account to prevent the likelihood of disclosing the sensitive data from high to low domain. For this reason, some other auxiliary processing is required to address this problem, like using human review to ensure that the information is free from any sensitive information, or using data sanitation methods or sometime there is a need to use both. Indeed, this type of guard has been designed to ease transmission of unclassified data from a high domain to a low domain [85].

- **Bidirectional guards** in this type of guard, the data will be allowed to transfer in both directions from high to low and from low to high simultaneously. This guard in much complex than the others, because it needs to gather conditions and restrictions for both the previous types to create a balance and controller for the data flow between different domains. To achieve this, some security components are required to be embodied in this guard, like virus scanners, intrusion detection devices, file type checkers, trusted operating systems and so on [85].

2.7.10 Types of Data Transfer Review Process

In this section we concentrate on the main types of review and monitoring data transfer between different security domains.

1. Human Review

This is the most reliable and simplest review process which has adopted human resource. In this process, the human operator will be within the domain that is classified as higher to observe and filter any data flow from high to the low domain. The human operator is responsible for ensuring the integrity of information, by performing the required sanitation process or security insurance, at the same time he/she works as a judge to approve or deny the transmission as shown in the Figure 2.13 [67].

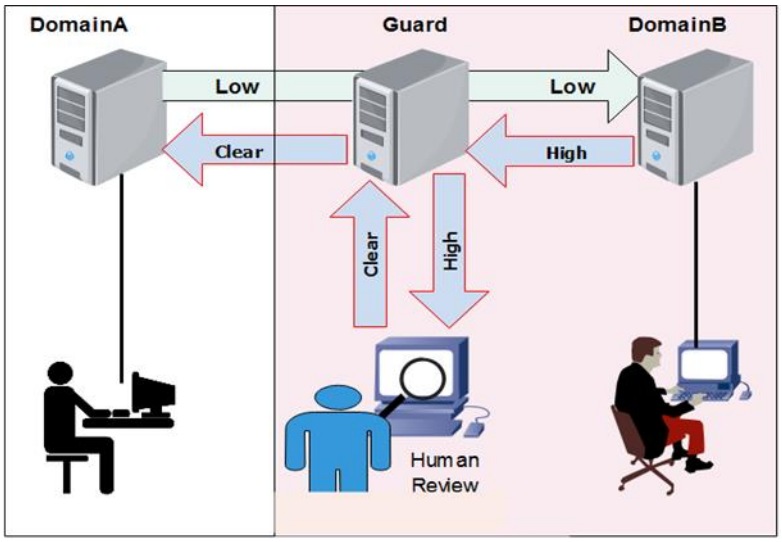


Figure 2.13. Human Review

2. Automatic Review

This type of review process has a mode to address the human limitation. Human senses have limitations to observe the contents of some digital multimedia, like audios, videos, images all of these media and more, which make content inspection by humans to be a mission which is almost impossible. There are many features belonging to this review model like being fast, scalable and consistent and also which has the ability to check the data flow is free from obscene language, and steganography in media files in context of data flow from low to the high domain; while at the same time preventing

sensitive data from being disclosed from the high to the low domain as illustrated in the next Figure 2.14. This mode of review is ideal for the static environment with structured, high volume and traffic data flows.

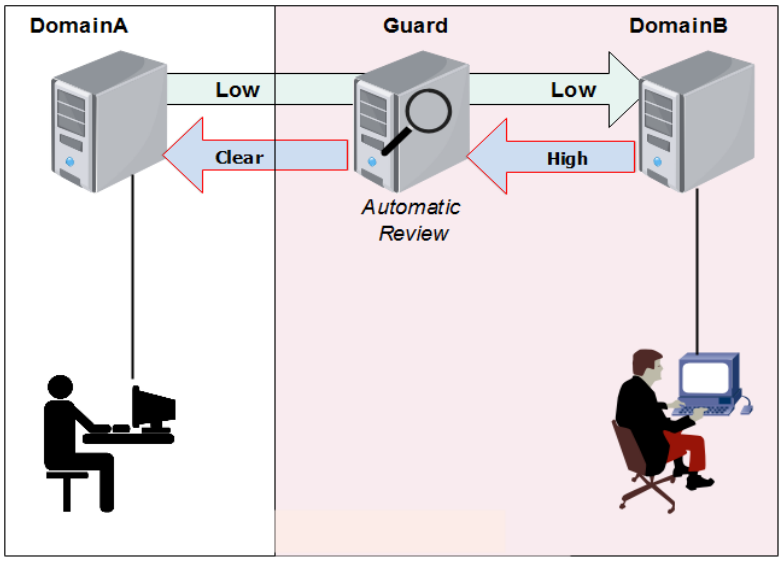


Figure 2.14. Automatic Review

3. Hybrid Review

This mode of review is designed to deal with an environment which has adopted a dynamic, high volume, high traffic and unstructured data flow. To address all of these issues, human review and automatic review are grouped to achieve high data flow integrity. Instead of rejecting the content by the automatic filter because of unintended error, the content will move to the human operator for more inspection and to see if they can correct the error to approve or reject the content as the final decision, as shown in Figure 2.15.

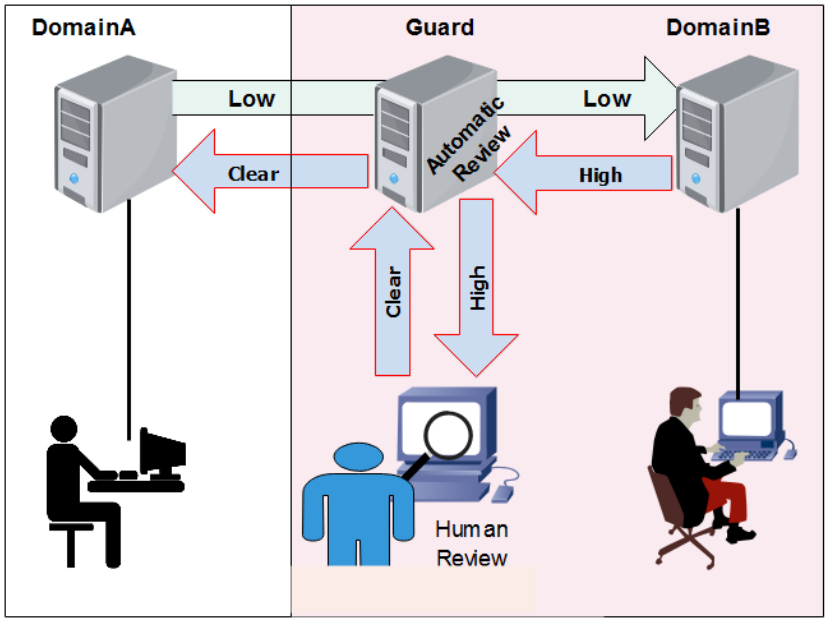


Figure 2.15. Hybrid Review

2.8 Aspect-Oriented Programming

Software developers had realized that there are some problems and concerns that are not well identified or executed by traditional programming methodologies. Moreover, some concerns do not represent the functional objective of application software, for example, bank applications offer to their clients some services (functions) like “online money transactions”, “withdraw”, “ display statements” and more other functions. All of these functions embody the base of the service interfaces that the service providers (bank application) offer to the client by their service interfaces. However, there are some concerns that non-functional tasks must be associated with the functional tasks to enhance the system working through, for example, coordinating the access to the system operations, adding security to the application system, updating a specific function in the system and so on. Adding or updating these concerns in system applications represents a difficult task for system developers because they need to track all program code to detect where these concerns methods should be inserted. At the same time, they have to define which piece of code needs to be applied for different policies. All of these concerns accumulate in the problem of scattering and tangling of system software.

A widely suggested but underused solution to these problems is that of Aspect Oriented Programming (AOP). AOP represents a high-level mechanism that deals with modularization of crosscutting concerns [88]. “It is a technology for separating crosscutting concerns into single units called aspects.” Each aspect is used to add a new behaviour to the methods or constructor or field through calling or execution of additional code [89]. Adding aspects of a program helps to cut across the OOP part by using aspect elements such as join points, pointcuts and aspect advice. All of these components establish the ability to change the behaviour of the program to meet user requirements, and at the same time supporting software developers by representing software code at a high level of abstraction, making it easier to modify and update. Figure 2.16 illustrates how a security and logging concerns crosscutting the main code and how this crosscutting modular by using AOP technology.

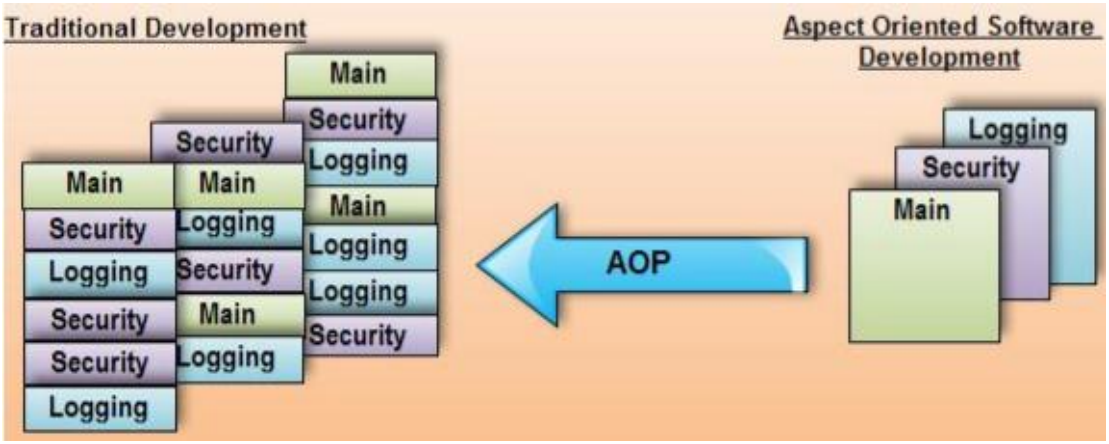


Figure 2.16. OOP and AOP Development

To understand AOP more clearly we will explain the main AOP terminologies:

1. **Concern**: A concern is a specific purpose, goal, concept or area of interest [90]. As stated by Laddad “A credit card processing system’s core concern would be processing payments, while its system-level concerns would handle logging, transaction integrity, authentication, security, performance, and so on” [91].
2. **Cross-cutting concerns**: Many such concerns – known as cross-cutting concerns– tend to affect multiple implementation modules [91]. As defined in [16] cross-cutting concern is a behaviour that may across a scope of piece of

the software. It may be simple behaviour running in some software classes or it may be more complicated through applying restrictions that totally affect the software working. From a technical point of view, a typical software system contains some core concerns and system-level concerns [90]. For example cross-cutting concerns are "security (authorization and auditing, logging and debugging, synchronization, persistence and more.

3. **Aspect**: The modular representation of a cross-cutting concern. A concern may cross-cut one or more components; security and logging are examples of cross-cutting concerns [92]. An aspect defines a pointcut and advice, and is compiled by the aspect compiler, such as the AspectJ compiler, in order for concerns (both dynamic and static) to be woven into existing objects (to interweave). Through separating aspects, crosscutting relations can be handled easily [90].
4. **Join point**: A join point is a point where a concern will cross-cut the main code. Join points can be at method calls, functions, constructors etc. Join points are defined generally, and useful for identifying problem points in code [93]. A specific join point is a precise execution point in the program, for example, a method in a class. We take join points to be an abstract concept; for our purposes it's not necessary to define them precisely [90].
5. **Pointcut**: Tells the aspect compiler when it should match a join point [93]. Essentially, it is a structure for the capture of join points. In contrast, a point-cut needs to be defined in an aspect [90]. A pointcut represents the specific aspect implementation that will be associated with a specific method [92].
6. **Advice**: The actual code that will be executed when the control flow reaches the join point [92]. In AOP you can specify the advice code to execute before, around or after the join point.
7. **Weaver**: The engine that weaves aspects along with their respective functional components. There are two main weaving kinds: Static and Dynamic. In the next subsection, we will expand on the main differences between them [94].

In the following Figure, we put all AOP terminologies together to be clearer and we will talk in semi detail about these terminologies in AsepctJ section.

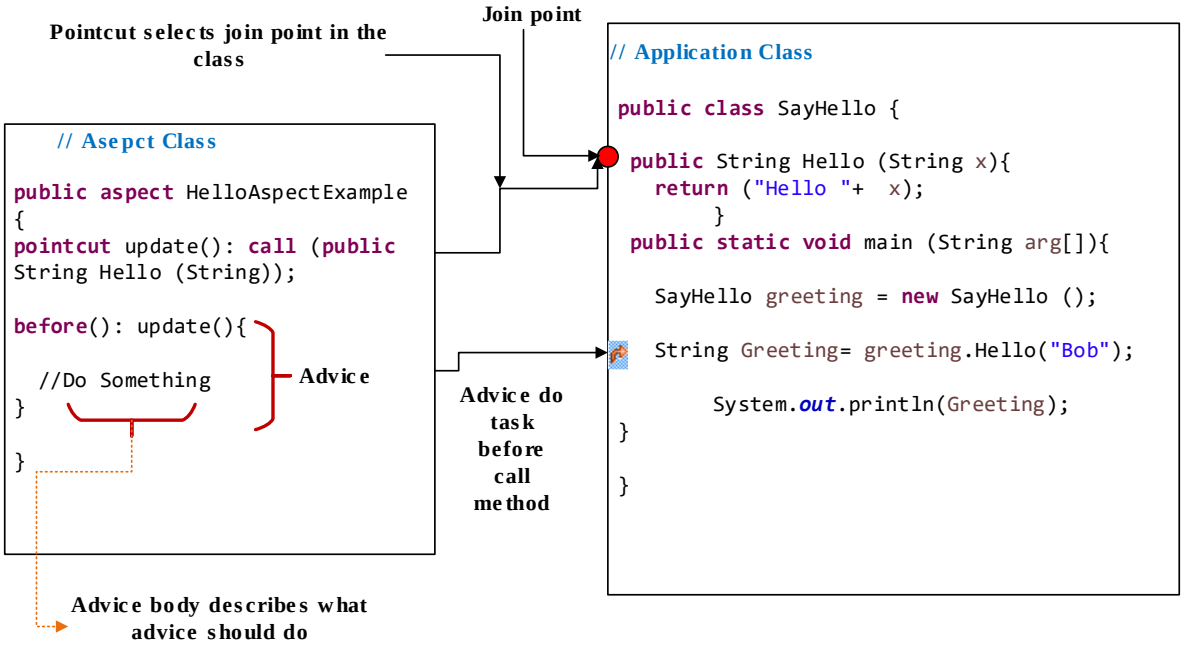


Figure 2.17. AOP example

Because of the importance of the various kinds of weaving used throughout this report, we discuss them in more detail in the next subsection.

2.8.1 Static and Dynamic Aspect Oriented Programming

AOP provides two kinds of weaving; *static* weaving and *dynamic* weaving. In the following we discuss each type separately.

1. Static Weaving

Static weaving consists of combining the aspects' and components' functionality prior to application execution. This combination consists of inserting calls to advice in the component's code. It causes little performance penalty because all the code is combined and statically optimised before its execution. Since the application is woven at compile time, any functionality to be adapted at runtime requires the application to be stopped, recompiled, rewoven and restarted from scratch, often losing persistency in the process. For this reason, this kind of weaving is not used in applications that require a high level of runtime adaptation; dynamic weaving is used instead [94].

2. Dynamic Weaving

As explained by Fletcher and Akkawi, “Dynamic weaving achieves the separation of concerns by separating the properties of the system such as logging, security, scheduling, etc., from the functionality of the system, it then weaves them together at run time to achieve the overall application system in order to achieve dynamic adaptability at run time”. During run time the aspect will be added or removed dynamically without the need to recompile the code each time [92]. For example, “dynamic weaving has been used in handling Quality of Service (QoS) requirements in CORBA distributed systems, managing web cache pre-fetching, balancing the load of RMI applications, and changing the control policy of distributed systems” [95].

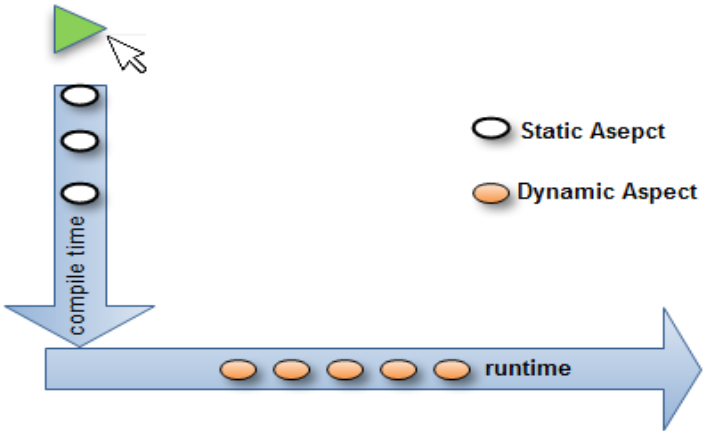


Figure 2.18. Static and dynamic weaving through program execution.

Both static and dynamic weaving can be adopted in a software system simultaneously, but with weaving applied at different times: static weaving at compile time and dynamic at running time as shown in Figure 2.18 Dynamic aspects may be more appropriate for a software program under development, because it offers to the developer the ability to measure and debug a program's behaviour without needing to change the source code after each execution or debug cycle. However, if the program has been designed so that no runtime adaptation is required, it is preferable to change the dynamic aspects to the static aspects to provide a higher level of performance at a run-time [95].

Sometimes the program developer needs to identify a specific condition, modifying any existing join points or debugging any pieces of the program that come in contact with the aspect before execution. Weave-time declarations give the opportunity to achieve this. Figure 2.19 shows the relationships between these concepts in an AOP system [96].

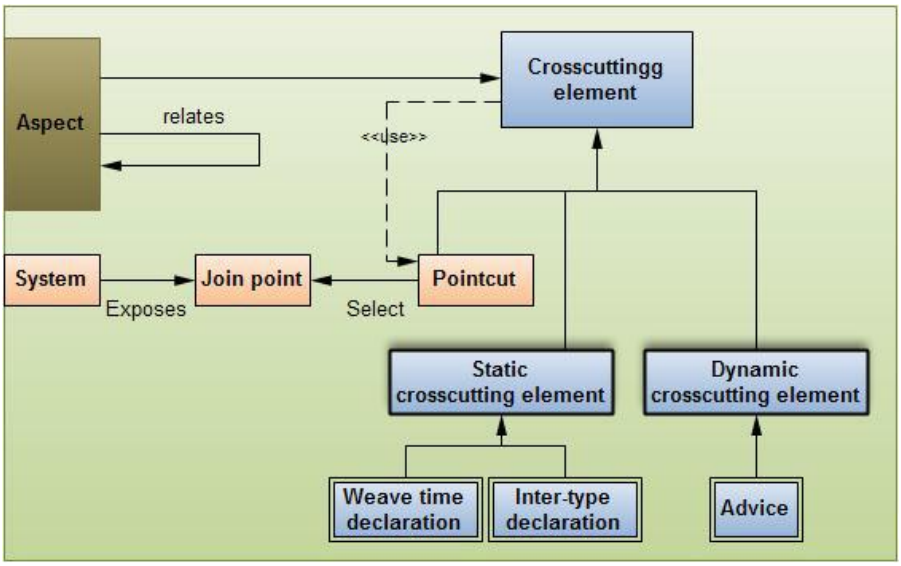


Figure 2.19. Generic model of an AOP system [96].

2.8.2 AspectJ

In this section we will consider AspectJ, which represents a platform for AOP implementation using the Java programming language. Originally developed at Xerox PARC [97], AspectJ offers a simple approach for dealing with aspect oriented extensions in Java by providing obvious modularization of cross-cutting concerns [98]. It works as a pre-compiler that generates class files that can be adopted by Java byte-code programs, as shown in Figure 2.20. By using AspectJ, the potential to modify and update application software has become possible without needing to change the application source code [97], [98]. It also provides further benefits, so there's no need for the end user to install anything special to run the programs except the Java Virtual Machine and AspectJ tools [97]. In Java there are also two further interesting approaches for applying AOP: JBoss AOP and AOP Spring. Moreover, AOP is used in many other languages, including AspectC for C, Aspect C++ and FeatureC++ for C++, and Sprint .Net for the CLR languages [99]. "AspectJ adds to Java just one new

concept, a join point, and a few new constructs: pointcuts, advice, introduction and aspects" [20].

For the work described in this report, we use the AspectJ Development Tools for Eclipse (AJDT) [100]. They constitute an open source Eclipse Technology Project that provides the required tools to develop and run AspectJ applications. We discuss the AspectJ concepts briefly as follows:

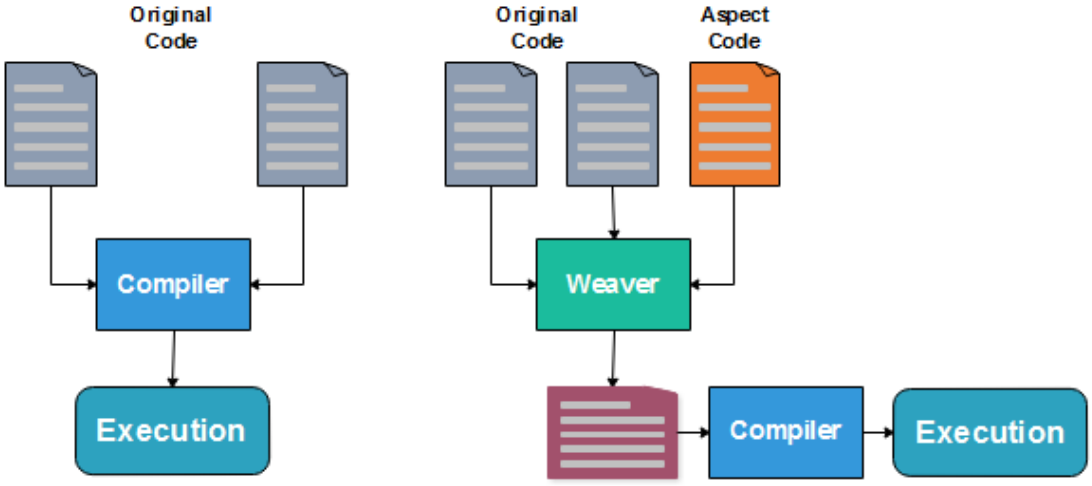


Figure 2.20. a) Java compiler without Aspect, b) java compiler with aspects.

- **Join Points**

Consider the following Java class:

```
class Coordinate
{
private int x, y;
Point(int x, int y) { this.x = x; this.y = y; }
void setX(int x) {this.x = x; }
void setY(int y) { this.y = y; }
int getX() { return x; }
int getY() { return y; }
}
```

If any code calls a method such as *setX(5)* the program will match the name of the method and type of argument (int). If the matching result is true then the output is (private int x =5). This happens for all methods and constructors in a Java program and follows the base: *"When something happens, then something gets executed"*[101]. OOP provides several kinds of "things that happen" which we refer to as join points. Join points consist of things like method calls, method executions, object instantiations, constructor executions, field references and handler executions [101], as shown in Table2.

Table 2 Join points [102].

kind	signature	this	target	args	Bytecode shadow
Method-execution	method	ALOAD_0 or none	Same as this	Local vars	Entire code segment of method
Method-call	method	ALOAD_0 or none	From stack	From stack	Invokeinterface, invokespecial *(only for privates)*, invokestatic, invokevirtual
Constructor-execution	constructor	ALOAD_0	Same as this	Local vars	Code segment of <init> after call to super
Constructor-call	constructor	ALOAD_0 or none	None	From stack	Invokespecial *(plus some extra pieces)*
Field-get	Field	ALOAD_0 or none	From stack	none	Getfield or getstatic
Field-get	Field	ALOAD_0 or none	From stack	From stack	Putfield or putstatic
Advice-execution	None	ALOAD_0	Same as this	Local vars	Code segment of corresponding method
Initialization	Corresponding constructor	ALOAD_0	Same as this	Complex	Requires in-lining of all constructors in a given class into one
Static-initialization	Typename	None	None	None	Code segment of <clinit>
Pre-initialization	Corresponding constructor	None	None	Local vars	Code segment of <init> before call to super, this may require in-lining

Exception-handler	Typename of exception	ALOAD_0 or none	None	From stack	Start is found from exception handler table. *(only before advice allowed because end is poorly defined in bytecode)*

- **Aspect Advice**

Code that is written in the aspect class and modifies the behaviour of the Java class at a certain join point. The general form of an AspectJ advice is:

[strictfp] AdviceSpec [throws TypeList]: Pointcut {Body}

Where ***AdviceSpec*** is one of

- *before(Formals)*
- *after(Formals) returning [(Formal)]*
- *after(Formals) throwing [(Formal)]*
- *after(Formals)*
- *Type around(Formals)*

Here *Formal* refers to a variable binding such as those used for method parameters, of the form *Type Variable-Name*, and *Formals* refers to a comma-delimited list of formal [74]. Figure 2.21 shows the applying of aspect advices on a method.

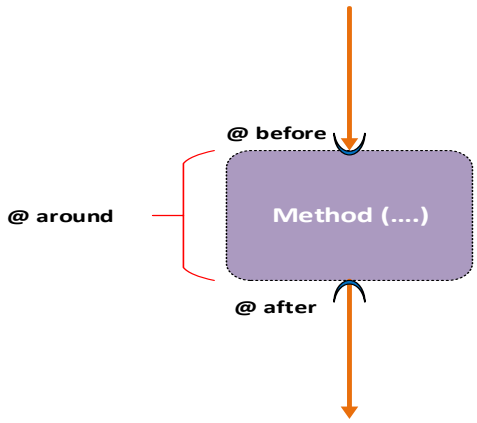


Figure 2.21. AspectJ Advices

2.9 Summary

This chapter has covered the background information about the fundamental components of the proposed system in order to clarify the main objectives and motivation when we deal with these components in the following chapters. It has also reviewed the security and privacy preservation challenges that might face any system developers when trying to apply these policies upon their system. Access control has obtained the largest amount of explanation in this chapter because it represents the core of the proposed as well as dealing with all sides that revolve around the access control. Besides concentrating on ABAC and IBAC as the first and second layer of the proposal system, this chapter presented the MLS model (third layer) in depth, by dealing with topics which are related to MLS like, compartments, Trusted-Based computing, domination relationship, reference monitoring and security guards. All these titles open the gate to understanding the proposed access control model.

Cryptography algorithms, privacy preservation and data sanitization are also presented in this chapter and we will show how to employ them to support the model in chapter 5. Finally, the last part of this chapter details the suggested programming tool, Aspect-Oriented Programming (AOP) through dealing with all topics which are associated with AOP.

Chapter 3
Related Works

This chapter provides a review on enforcing Aspect-Oriented Programming (AOP) in three objects: Access Control (AC), intrusion detection, and cryptography. We will be concentrating in depth on using AOP in AC models through exploring how we can exploit AOP to modules of different access control models. This will be done by separating these models from their core system in order for the easiest system to work as well as to increase the modularity, manageability, and flexibility of the system. At the end of the access control section, we will focus on applying AOP to deal with the systems that are based on the Multi-level security or (authority) concept, through using Mandatory Access Control (MAC) or other similar models. We will review the gaps in these models and look at what ways can be used to enhance these systems, and this will be work's contribution. This chapter presents some current methodologies which are used to apply access control models and deal with the MLS system without using AOP.

3.1 AOP and Access Control

Access control and other security solutions, which are considered as a non-functional service, are used to increase the reliability of the software by applying security concepts. The aim of this section is to review the capacities of using Aspect-Oriented Programming to modules and separates these non-functional services from the system code, thereby getting rid of scattering and tangling problems which arise when mixing between functional and non-functional service codes in the same program body. We will deal with some proposals, methodologies, and ideas that have leveraged AOP language to enforce Access control models in their systems. Regardless of the date of publishing these papers and the proposed ideas, this study focused on the objective of using AOP in such a way that we don't need to change the original code of the software, and therefore will not insert additional burdens on the system working. The following

subsections are divided according to access control model used to be modularized by AOP:

3.1.1 AOP with General Access Control Models.

This subsection deals with different types of access controls (not the main AC models) which are been designed under the fundamentals access control conditions and according to the system requirements.

K. Chen et al. [103] Proposed Fine-Grained Access control for web applications supported by aspect oriented programming, the Model-View-Controller (MVC) and Apache Struts framework have been adopted to structure the web application (see Figure 3.22). Their model depends on the interaction between a user and a web application through a sequence of access tuples of three main elements: < *user, function, data* > which means a user request to execute the function on a specific type of data object.

Their dealings with AOP fell into three stages:

1- Choosing the pointcuts: by selected Action class in Struts framework to be a target to aspect pointcuts because Action class is the class that is dealing with the user request and intermediate result as well as the response.
2- Constricting the Aspects: the constriction of their model concentrated on dividing the aspect code into two parts: generic part realized by abstract aspects and rule specific part realized by concrete aspects.
3- Aspects factory: they built different authentication codes to be available to all aspects in their framework to be like a factory of authentication objects and called AAAspectit. Afterwards, building two access control aspects called pre-checking dealing with common access cases, while post-filter uses it if the access constraint needs to refer to data attributes. Finally, they employ the AOP advice to enforce all the above policies.

K. Chen et al. [104] leveraged AOP technology to develop a privacy-ware Access Control framework through modular privacy preferences of the person (PII) available to the aspect responsible, in order to protect the personally identifiable information

(PII) from unauthorized access. Their proposal extended from [103] by utilizing inter-type declaration of AspectJ as

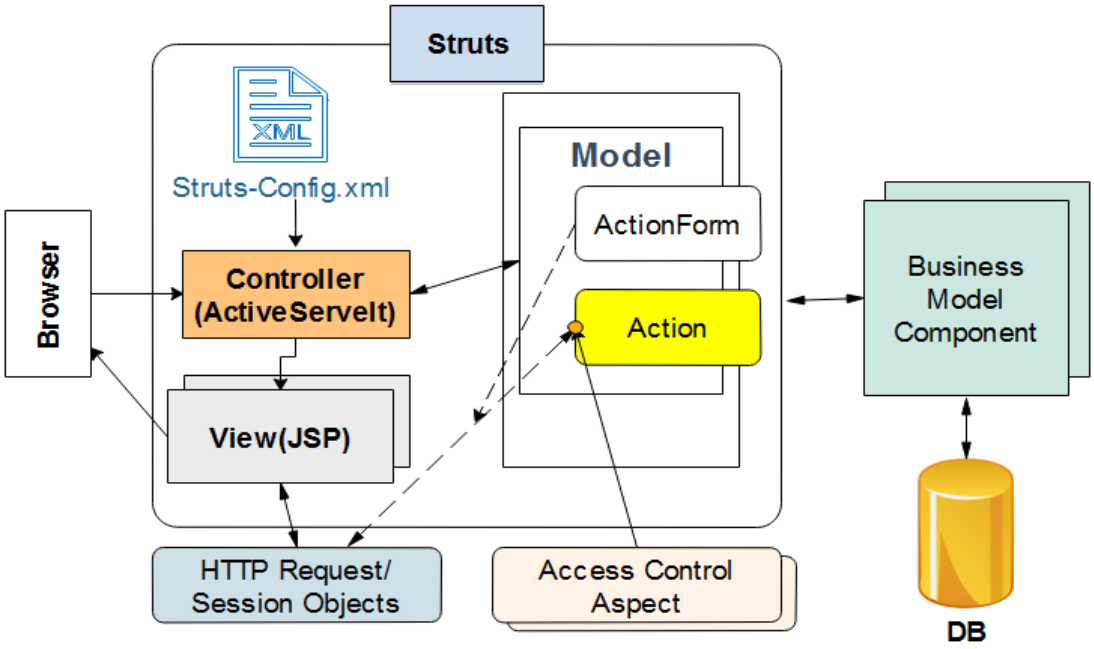


Figure 3.22. Struts-Based Web application architecture [103]

well as AspectJ advices. As in [103] Model-View-Controller (MVC) and Apache Struts the framework had been adopted to structure the web application. The pointcut targets the execute method of user action because it provides all the information needed to evaluate the access control constraints. The privacy–aware access control rules take the form" *allow or deny action on data categories by user categories for certain purposes consented by the data subject under certain condition*". By comparing with the traditional access control method, they add the privacy aware rule, referred to as privacy preferences through extending the rules required by two kinds of additional information "*the action and the purpose consented by the data subject*". The authors benefited from AOP characteristics through separating the management of privacy preferences from the application and linking them to the access control aspects by a preference factory. Both static and dynamic aspects are adopted in their proposal, such static tasks add a privacy preference field to all classes that included data subject's PII

to protect. While the dynamic one is conducted by an object construction advice, which is triggered right after any object is instantiated from those classes with PII. Thus, the advice will ask the privacy factory about a proper privacy preferences object which is matching the requested PII object and associates the preference object with the PII object. Note that each data subject specifies its privacy preferences regarding its PII in a consent form which is collected and managed by the privacy preferences management module. The preferences aspect will be invoked in both static and dynamic form to run the retrieval of the data subject's privacy preferences and associate them with the requested data.

F. Yang et al. [105] proposed AspectKLAIM which represents the extension of KLAIM. KLAIM is a language specially designed to program distributed systems consisting of several mobile components that interact through multiple distributed tuple spaces or databases. The actions and their permissions will be given as aspects, and by having a fixed set of such aspects the access policies specify when an action is permitted, thus governing the execution of the proposed net. The proposed aspect allows the user to trap both the action and the processes to be executed in future. The authors's point of view is focused on the consideration of "*whom should be trusted by preventing the means for expressing trust in terms of how data is actually used*".

K. Chen et al. [106] proposed the analysis of access control approaches from two different dimensions: 1) is the granularity level which concerns user requirements, 2) is the implementation technology adopted by application developers. With 1 we used three tuples <user, function, data> to model the interaction between users and application system. With 2 we divided the implementation technology for access control into three different levels: hard-wired, adopted and configuration level. Afterwards we made the intersection between 1 and 2 to find that most use of AOP will be in the adopted level for all users, functions and data. The authors used MVC Apache Struts framework to represent the proposal. They considered that the execute method will be a proper join point in order to catch and control the forwarding mapping and Http Req, and Resp. K. Chen et al defined three aspects: "authentication", "Precheck" and Postcheck" to capture the code structure for enforcing fine-grained access control, which is represented by granularity levels namely user, function and data.

R. Toledo el at. proposed a modular access control called ModAc, to modularize both using and supporting of access control through restriction aspects and scoping strategies, even in the existence of untrusted aspects [107]. The authors deal with access control, including privileged execution and first-class permission context (*like in java, class- privileged execution allows a trusted entity to hold responsibility for a certain action, while first-class permission allows the programmer to capture a set of permissions at a certain point and restore it later on*). The restriction aspect is used to ensure proper resource protection, and works by adhering to a different, dual pattern: *the pointcut selects access to a sensitive resource* but the aspect advice immediately disallows the access by not proceeding with the primitive operations. Whilst scoping strategies are used to ensure that the aspect only *sees* forbidden access through permitting fine-grained access control over the scope of the deployed aspect and specified by two propagation functions, a *call stack* specifies how an aspect propagates along with method calls and a *delayed evaluation,* to specify whether or not an aspect is "captured" in the object when they are created. In this way they modularized the proposal as an aspect: untrusted aspects cannot inhibit access control and trusted aspects are able to see any join point. Their implementation relied on R^0 and ZAC library.

T. Scheffler et al. [108] proposed an approach that uses "sticky privacy policies" written in the eXtensible Access Control Markup Language (XACML) to control the access rules for protecting resources. They present a privacy scheme associated with a reference monitoring implementation using the Java Security Framework. In their solution, they mix XACML and the Java Security Framework by implementing a client-side reference monitor. They developed a theme park location scenario that highlights privacy protection issues based on three main bases: localisation, privacy settings and service architecture. In their approach, they allowed for users to get benefits from the application without needing to share private information directly. The authors fixed some problems in their approach through using JBoss, which represents an implementation of AOP for Java. This was achieved by implementing a security layer that enforces the data-owner's defined privacy policies for protected resources. The reference monitor uses a set of task-specific advices to monitor access to the

resource and interrupt it if necessary. Before granting access to a resource, an advice evaluates its sticky policy and if the evaluation is not successful, the advice prohibits access by cancelling the method invocation that is trying to do it.

3.1.2 AOP With Role-Based Access Control

X. Li et al. [109] introduced conditions for adding to the role-based access control permission in order to enhance the traditional concept through minimizing the number of permissions. They completely separate their concept of access control from the application logic by using aspect oriented programming which allows access control to be integrated into a legacy 3 tier information system, without the need to change the application program. Their approach can be summarized as follows: when the user with role *r* wishes to perform a specific operation *op* on a certain data *dr1* of a table *t*, the success of this operation will be totally dependent on specific conditions. The application will scan the permission table for the entire (r, *op*, *t*, *somecondition*) if any of these conditions doesn't match, then the operation *op* will be denied otherwise it will be allowed. The usage of AOP is enacted by:

1- The access control developer implements all methods needed to perform access control in a special module, called *aspect*. For instance, an update-check method.
2- The access control developer must first decide which methods in the application programs require access control. These methods are defined as *join points*.
3- Then, he/she has to group join points that require the same type of access control (e.g., update-check). These groups are called *pointcuts*.
4- Finally, he/she has to indicate what access control actions have to be performed for the join points in the pointcut and when these actions should be performed (e.g., before, after or instead of the execution of the join point method). This is called an *advice*.

They used AspectJ as the most popular AOP tool and applied their proposal on Laboratory information system (LIMS).

I. Ray et al. [110] proposed Role-based Access control model aspect as patterns using UML diagram templates through composing between the aspect oriented model (AOM) approach which deals with access control concerns, referred to an *aspect* and functionality application referred to as a *primary model*. Before the composition the aspect model must be instantiated in the context of the application domain. This is obtained by binding elements in the aspect model to elements in the application domain, therefore producing the context-specific aspect model. Afterwards, the aspect models are produced for each part of the primary model and woven into the base model using composition directives.

J. Pavlich et al. [111] they proposed a formal framework for security software applications that is able to support the automatic translation of a new UML artifact through translating the role-slice access control policy into an aspect oriented programming enforcement code. The authors have shown the power of using AOP to intercept every call to the set of classes in which access needs to be controlled, and grants or denies access, depending on the permissions stored in the policy database. They submitted their proposal formally to enforce it by any programming languages.

C. Braga [112] deals with RBAC, proposed model driven architecture (MDA) [113] approach and shows how to transfer the code generator of the access control policy from SecureUML, an RBAC modelling language, to the language of Aspect for Access Control(AAC). The code generator for access control policies represents the transformation contract which specifies the relationships between the abstract syntax of the SecureUML and AAC and constrains the two languages. Their aspect language especially concentrated on defining pointcuts in the application body at any place needing to do a certain function, for example create method in Text Report Configuration (TRC) class and apply *before* advices to gathering with permission for a given method, thus dealing with access control processing as a precondition. The author used a metamodel to represent the language and to specify the transformation.

M. Hazaa et al. [114] used aspect for a design for CORBA access control supporting the RBAC model. They divided their design into three phases:

- Main concern Base design: The main concern of their study is to realize a CORBA AC mechanism that supports RBAC0. Afterward, any modification, reused in the future will be accounted for by AOD.
- Aspect1:Role hierarchy (RH) they aggregated the roles that need to be included in the base design in role hierarchy concern. By using aspect orientation design (AOD) they simplify the tangling and scattering which rely on regular modification and updating and reusing of some roles to modify the base design of the RBAC0, to produce RBAC1,through transfer of the crosscutting concerns into aspects.
- Aspect2: RBAC2 allows security administrator to set static separation of duty constraints on the assignment of users to a role. In other words, the security administrator puts the new function that should be executed every time in a static aspect.
- RBAC3 : finally combines role hierarchy and static constraint to produce RBAC3. This combination shows the advantage of using AOD through applying dynamic (RH) and static aspect with a minor modification of the main system.

S. Kallel et al. [115] deals with the concept of delegation in access control, which represents the core that allows users to assign all or part of their permission to other users. They combine between the formal method of delegation used in RBAC and AOP to enforce their delegation policies. They used TemprolZ as a formal language to represent the RBAC model as well as delegation and revocation models. Their approach has three steps,1: the specification step (security policies and their corresponding constraint using TemporlZ), and verification step, to verify the consistency of the system specification using theorem proving and 2: the implementation step which used java to implement the functional codes which are not including any logic for authorization. 3: the enforcement step, in this step the developer uses an aspect generator tool to generate the security module by translating the formal policies from TemprolZ to the aspect oriented programming language ALPHA (which is an AOP which uses a pointcut language based on logic queries). This language used a subset of Prolog queries for pointcut expression. Each delegation operation will use

one aspect to enforce the pre-condition and delegation constraints. The pointcut constituted by ALPHA predicate "Call" to intercept calls to the method in the functional code which holds the formal delegation operation. In the advice body each Z constraint is translated to a conditional statement by using a Z-based java package.

P. Colombo et al. [116] proposed an approach called PuRBAC (Purpose and Role-based Access control) which is a java application which operates in between relational DBMS in order to govern the execution of both SQL queries based on purpose, and role based privacy policies. In their methodology, they exploited AOP techniques for dealing with the dynamic features of relational DBMS environment, and to do precondition evaluation filtering at running time.

J. Pavlich et al. [117] proposed a role slices method to provide an abstract to collect information on the security of roles that cut across all the classes in an application. Afterwards they transform the role slices into the application code using aspect oriented programming to capture the access control policy (authorized or prohibited) which is already defined by role slices.

The system works by using an aspect-oriented code to control the access control to the method, through checking whether the presented method is denied for the active role (which the current user has when logged in) and raises an exception if that occurs; otherwise the method is allowed to execute. In other words, the AOP advice will be the link between the database which holds the security permission (role slices) and the user who logs into the system. To summarize the working of the system it includes: security policy database and access control aspects which include role slices to represent the security method and aspect advice that is woven at the pointcuts, defined by role slices and the security database.

A. Mourad et al. [118] proposed an aspect-oriented approach for the dynamic enforcement of web service security, based on the synergy between AOP and business processing execution language (BPEL) of the composed web service. The author used Role-Based Access control applied on Flight service (RBAC-FS) to ensure the authentication and authorization for accessing the web service resources. The result shows how the propsed model can separate the security concerns totally from the web

service composition, and how we can apply these concerns in a specific join point in the BPEL execution body.

3.1.3 AOP With Organization-Based Access Control

M. S. Idrees et al. [119] they exploited the aspect oriented concept to enforce their security policy as Organization Base Access Control (Or-BAC) in dynamic form. Their motivated problem revolves around how an evaluation can dynamically modify security enforcement with respect to the new rule, especially obligation rules and how to manage the obligations and changes during the runtime. Their contribution is showing how different knowledge modelled in the security policies can be extracted and translated into security aspectual knowledge that is used to define appropriate security aspects. They applied the security policy as a set of rules (permission, prohibition, and obligations). Afterwards the module will be responsible for taking a decision (allow/deny). They deal with aspects in a generation phase which is based on a translation process which translates the concepts used to define a security rule (role, activity, view) to the concept used to define the aspects, such as aspect type, advice code and pointcuts. Thus, the functional requirement to enforce the security policy will be dynamically associated. In this case, any changes at running time will be accounted through weaving /unweaving aspects. They proposed "Context Awareness" which is used to monitor any change in the system during running time.

S Ayed et al. [120] proposed a framework based on the INYER_TRUST project with a security model based on the (Or-BAC) to govern the security policy based on AOP. This framework presented the whole architecture which describes the loop of security aspect generation and weaving, as well as depending on the AOP approach to dealing with this loop and making a dynamic control of security requirement.

S. Ayed et al. [121] proposed a security framework architecture through modularized (Or-BAC) by using AOP approach to enforce security policies dynamically on distributed systems. They deal with access control and usage control as a security requirement to making a link with the deployments of these policies by translated to

aspects. The authors combined various security modules (security policy modeling, policy engine, policy interpreter, Aspect generation and context awareness) to by linking them to clarify the security policy cycle. This cycle will be established during the running time and be able to face any security changes during run time. Aspect generation is the database of all security policies received from policy interpreter and modular by aspect. This generation will be divided into modules: Generic Aspect Generation module which deal with the general security policy part which not related to the AOP framework. While the second module is Concrete Aspect Generation module which is generate during run time and has related to the AOP framework. The access control aspect will be generated according to set of permissions relaying on the access policy and not depend on prohibitions. The aspect will follow the tuple <subject, action, object> which are coming from policy interpreter module. Thus, control the accessing by allowing the authorized user (subjects) to perform a specific action on the objects (data) in a modularization way by using AOP.

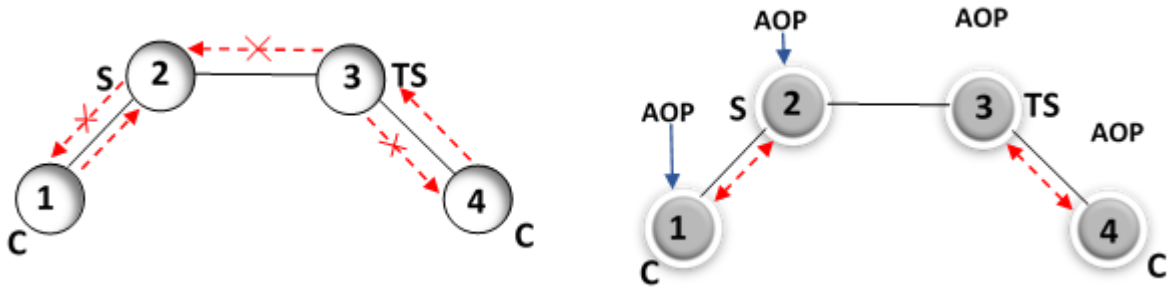
3.1.4 AOP with Multilevel Security System

Despite all the reviewed solutions, they succeeded to perform their objective's mission by a separate access control concept from the body of their systems by using AOP, and thus to achieve the main objectives of what the AOP was designed for. Interactions between the system code and AOP code however, to use the latter as the core part to control the access and build a simple solution working in a system, has already been designed based on the Multi-level security concept.

García *et al.* [94] considered the adaptation of security measures of distributed systems, even when their sizes and arrangements changed. This is achieved without compromising global security, and while attempting to improve flexibility and ease in dealing with distributed systems. The proposed approach uses dynamic AOSD to implement security mechanisms in distributed systems when the system is running, without requiring its execution to be stopped or interrupted. They submitted solutions for two common security problems in distributed systems: (i) access control and data flow and (ii) encryption of transmissions. In this way, the distributed system is able to

adapt to security measures when required, and can vary in size and arrangement without compromising security. Their scenario revolves around the ability of flow data between distributed system nodes which have different levels of authority. The authors created an AOP shield around each individual node used to give allowance to the data to access the node or just be discarded. They used this scenario to enhance the access model, working on a system based on Multi-level authority by allowing any source node with a specific security level to send information, labelled with the same node's level, to any node in the distributed system which has a higher or same level of the source node security. Regardless of this transmission, they passed intermediate nodes before approaching the distention node as illustrated in Figure 3.23, there is a data flow comparing between two distributed systems.

The first one: (a) is based on object oriented programming concept, shows that the Node 1 cannot send information to Node 4 even if they have same level of security because data cannot flow from Top Secret (Node 3) to Classified (Node 4). However, if a node is shielded by AOP language, the data flow will be handled in an abstract level as seen in –(b)- the level will be checked before entering the data to the node. In this case, data will flow freely between system nodes.



-a- **-b-**

Figure 3.23. -a- OOP distributed nodes, -b- AOP+OOP distributed nodes, (TS, S, C) refers to (Top-secret, Secret, Confidential) security level respectively.

They used a real client-server FTP scenario to test the proposed system. They show that dynamic aspects can be used in order to cipher all messages exchanged between the clients and the server when increased communication security is needed, while

reverting to the default channel when the exchanged information can be transmitted in the clear. We are going to discuss this paper in more depth in the next chapter.

R. Ramachandran et al. [122] deal with Multi-level security systems based on the Bell-La Padula model (BLP) using Aspect-Oriented Programming. Their scenario discusses the payroll system which deals with the managers supposed as high security clearance, and employees who are supposed as low level. The interaction between high and low level will be implemented by using AspectJ. The pointcuts intercept each read or write operation. Thus, the manager will be able to view or modify the payroll database while the employee will be restricted according to what the pointcuts allow him to use. They use JFTPd which is an FTP server implemented in java, to evaluate their proposal.

U Huseyin et al. [123] they developed Vigilies as a firewall to apply a Fine-Grained Access Control (FGAC) on the MapReduce system. They augmented the cloud's front-end API by implementing Vigilies as a middleware layer to work as reference monitoring. Vigilies deals with the MapReduce jobs as untrusted to prevent the probability of the user who submitted this job having suspect intentions. They used AOP to enforce the Vigilies security policy on Apache Hadoop by an injected aspect into three pointcuts: 1) *initialization aspect* used to intercept the *initialized ()* method, 2) *predicate aspect* used to intercept the *nextKeyValue()* method and 3) *modification aspects* used to intercept the *getcurrentKey()/ getcurrentValue()* methods. They used static AspectJ to evaluate their proposal.

K. Padayachee [124] they explore the ability of enforcing multi-level security concepts upon systems by implementing aspect-oriented programing language. They concentrated on a simple mode which works as a pre-authorization model where a decision is made before allowing access to the data. They claim that they enhanced working of [122] by two sides, the first one they used the extended aspect-oriented programming to circumvent around the problem of java API, which is not able to permit getting back directly the file name which is already intercepted by read and write pointcuts. The second problem is embedding the user's clearance with the original implementation, the authors separate this clearance totally from the original data, thus the access control model will be totally separated from the original program. Their

Aspect classes have two pointcuts methods called *Read* and *Write* used to intercept all read and write operations associated with the subject trying to access the object to perform the operation. They used *around* advice instant of *before* advice to proceed the accessing rights, only after meeting all the authorization and obligation conditions.

A. M. Hernandez et al. [125] proposed approach is based on combining between history and future sensitive policy in a distributed system. They focused on the concept *looking to the future* by using multilevel access control policies as suited to past analysis of how the system reached its current state. The Bell-La Padula model is presented to extend the AspectKB framework in order to allow to express policies that *look to the past*. The aspect is to trap an action, the response will be considered, granting access only if the security level of the subject is not lower than that of the object. This proposal gives some power to access information according to the past performance of the system.

S. M. Khan et al. [126] proposed a novel implementation approach, SilverLine *(Secure-information flow verification in lined), that enforced Mandatory Access Control (MAC) and information flow security policies on untrusted Java jobs binaries in Hadoop cloud.foot,* which exploited an AOP to elegantly specify, implement, and put in-line reference monitoring (IRM) into untrusted jobs without access to a job source code. They used aspect-weaving as a pre-processing step to rewrite automatically untrusted java binaries before passing them to the cloud. To enable aspects in the Hadoop environment, AspectJ JAR and aspects are distributed to all nodes in the cloud. SilverLines' aspect weaver will intercept the submitted jobs at the cloud edge and in-lines in the IRM. The aspect weaver may reside on any node inside the cloud, or may be deployed on a separate machine outside the cloud. The resulting self –monitoring binaries are then dispatched to the cloud for execution.

3.2 AOP Intrusion Prevention, Cryptography and Privacy

3.2.1 AOP Intrusion Prevention

E. Kajo-mece et al. [127] the proposal system is based on three parts: 1) Call *WebAppInputFilter* which is used to filter users' inputs through using the AspectJ advice that controls the validation process. 2) Validator used to validate against XSS and SQL injection attacks and this is done by AOP devices as well. 3) Finally the encoder which is used to encode the dangerous characters by converting them to their decimal equivalent to make them harmless.

The main idea is based on the user's inputs being validated before use as a part of the query. By the way, the authors focused on user's input without the partial SQL statement defined by the developer, in order to speed up the processing, especially when they considered that this part is already trusted. The mechanism of the system is summarized when the aspect captures the user input string and sends it to the first analyser (Syntactic Validator). If the string is not dangerous it is passed on to the second validation step (Semantic Validator). If the string is dangerous it is sent to the encoder. It encodes the dangerous characters and the result is passed to the Semantic Validator. If the string is not considered dangerous, it is passed on to the web application as a legitimate request. If it is considered dangerous, it is erased. JMeter is used for performance evaluation.

G. Hermosillo et al. [128] the authors used AspectJ and Jboss AOP to design a security aspect called AproSec for detecting SQL injection and Cross Site Scripting (XSS). Their proposal application AProSec (aspect security for web application server(WAS)) is used by intercepting and validating all the requests from the clients to the WAS and from WAS to the database server. The authors considered the SQL and XSS to be in the same aspects advice after they divided the advice into two validation parts: 1) HTTP request parameters and 2) DB queries. During the implementation, several syntaxes should be validated: double and single quotes, SQL injection, and XSS. HTTP request should validate the parameter value to void code injection and invalid HTML tags to

prevent XSS. For DB queries, the validation is made through analysing the query string to prevent “always true” comparisons, semicolons and comments. They used AspectJ, Tomcat for web application and MySQL as the database manager.

Z. Zhu et al. [129] proposed a model-based aspect-oriented framework for building intrusion-aware software systems. The authors started the design by identifying the vulnerable points in the target system and specifying the probable attacks that may exploit the system vulnerabilities. They used Aspect-Oriented Unified Modelling Language (UML) to model the attack scenarios and intrusion detection aspects. Their framework consists of five stages: identify vulnerabilities and attacks; model attack scenarios using UML; generate the intrusion detection aspect (IDA) code using an aspect code generator; weave aspects into the target system; test and deploy the integrated system.

M. Coates et al. [130] proposed AppSensor for intrusion detection. Their methodology depends on using some metrics to detect malicious use through studying and filtering user behaviours. They put some factors in order to distinguish between the suspicious (not if the user is attacker or just misuser of application) and the “Attacker” which is the real attack. Their application is integrated into business, presentation and data layers. This proposal has two modules the “detection module” to detect the attacks and malicious use and the “response module” to give an appropriate response for the detection at that time. They used AOP to inject their solution into the application system.

V. Schiavoni et al. [131] proposed an annotation toolkit that allows building DoS resistant component-based systems. The solution mechanism is able to handle the robustness concern as a separated and modularized but yet integrated aspect of the system. In typical component based applications, each component exposes a service. The key idea is to annotate such services and use the annotations as a means to detect an attack. They used Aspect-Oriented Programming techniques for modularizing and separating the implementation of annotation processing. The proposal implementation relied on the using of AOP techniques with java 1.5 annotation and implement it within Fractal, which is a java-based component model and provides an Architecture

Description Language (ADL). They show an improvement over a low level approach through focusing on a comment-based system, with which it was possible to provide a general mechanism to detect DoS attacks.

The proposal of *K. Padayachee et al.* [132] is dealing in how to use AOP techniques for monitoring the information flow between objects and how to detect vulnerabilities and misuse detection of this information. They considered a server application comprising of three classes *Server*, *Session* and *Account*, where there was vulnerability in that the server allows a malicious client to avoid getting charged for his/her connection time. They used AspectJ advices to intercept the vulnerabilities in order to detect any misuse of data flow, in which these advices can take a decision to permit the information flow or not, after examining the given message and classification of the sender and receiver. For example if source object (A) sends data to destination object (B), the flow will be intercepted by AOP and tested if it violates the policy of information flow. If any abnormal behavior the aspect will perform a specific action to deal with, otherwise the flow will proceed smoothly.

G. Georg et al. [133] proposed a methodology based on aspect oriented modelling (AOM) to design a secure application system. They separate between the implementation model called the *primary model* and attack and security mechanism which are localized in a different model. Their approach is focused on the impacts of the attack (aspect attack), after applying it to the primary model, and indicating whether the primary model may be compromised, then the proper security mechanism (aspect security) is used against this attack. This proposal has two types of aspect: generic aspect and context aspect. The first one is used to represent attack patterns and security protocols, while the context aspect is used for instantiating the generic aspect and both are modelled by UML models. The execution of this proposal can be summarized by two stages: the first stage, the attack aspect applies on a primary model to produce the *misuse model*, the latter will be analysed to determine if the protected resources are compromised by attack. For the second stage, if the results are unaccepted, then the aspect security will integrate with the primary system to produce the *security treated*

model. They used Alloy Analyser because it is easy to use and has been used for verifying many real-world applications.

J. M. Horcas et al. [134] proposed an approach based on the Interoperable Trust Assurance Infrastructure (INTER-TRUST) framework to deal with enforcing security policies in a dynamic form at running time. They used Montimage monitoring tools (MMT) security properties to formally specify security objectives and attack behaviours related to the application or protocol under test. This proposal has concentrated on two sides of the objectives, the first one is dealing with the dynamic deployment of security policies, while the second is used for dynamic monitoring of vulnerabilities through testing of the operation phases. The first objective is achieved by proceeding with the security specification through the Aspect Generation which connects with the Aspect Weaver, which in turn is connected with the repository of the security aspect. By designing a correct correlation between the security policies aspect and security specification, the application will be able to capture the modification of the security at running time as well as detecting some kinds of attacks. The second objective is the performance of monitoring and this is done through a defined set of vulnerabilities point in their approach that may break the correlations and defines a set of the kind of attacks which can affect these points. The authors evaluated their approach by applying the proposed system on two real case studies: The Intelligent Transportation System (ITS) and e-voting system.

SQL injection and XSS web attacks are used as examples of attacks that might be prevented this way. Hermosillo *et al.* [135] deal with SQL injection and XSS web attacks through design, and implement a security aspect called AProSec to harden a website against these attacks. Their design is based on a mixture of both AspectJ and JBoss using AspectJ at compile time to validate and filter the user information, and by implementing an SQL analyser to intercept and validate all the database queries before they are processed. Moreover, they used JBoss to weave aspects at runtime. They established the advantages of their approach by testing it with a vulnerable online bookstore, and their achieved objectives through preventing any query that contains a

commentary inside it, or any statement that is always true being passed to the database manager.

L. K. E. Mece [136] also suggested defending web services against SQL and XSS web attacks., The difference with this approach however, is that it can abstract through this system and will analyse the user input directly before it is used as a part of an SQL query, and the SQL validator checks the presence of SQL keywords in the user input. This processing will help check if there is any malicious injection.

K. Kawauchi et al. [137] suggested an aspect detect cross-site scripting. Their solution depends on sanitizing, i.e changing special characters for quoted ones, the input information being submitted by clients to web applications. They considered the scenario of servlet-based web applications. When information is submitted to a servlet, one of the subjects which occurs consists in determining whether it comes from an end user, or whether it occurs from a different servlet which delegates the request by means of the transfer mechanism supplied by the servlet container. In the latter case, data is assumed to be trustworthy as it simply grows from another section of the application. In such cases, the sanitizing can be skipped to be able to save computation time. To accomplish this, the authors propose to extend the syntax of the AspectJ pointcut language with another construct to detect data flows: the servlet input is sanitized if, and only if, it is written back to the servlet output stream.

G. Fan et al. [138] focused on service authorization, implementation traceability, data protection and fault handling through proposing a formal aspect-oriented approach used to analyse secure service composition. They used Petri[1] (*Petri net is a mathematically based technique for modelling and verifying software artifact*) net for formalising their model and describe the behavioural features of service composition. Through the integration between the advantages of using AOP and Petri net, they have shown how this integration reflects good results to observe the behaviour of service composition. Their proposal has two main processes:

Implementation phase: used Petri net for the modelling tool and AOP to separate crosscutting concerns and core concerns of the system. They integrated these two modules into a complete model.

Analysis stage: analysis of the security and fault handling of service composition by using the operation of Petri nets.

The result of this paper achieved security service composition and to reduce the effect of the single Web service's fault on service composition as much as possible.

P. Falcarin et al. [139] focused on ensuring that the software is not maliciously tampered with prior to and during the execution. They used an aspect to encapsulate a function that is used to generate an idiosyncratic signature which is associated with data transmission. Their proposal is based on the TrustedFlow^{TM} protocol which in turn is based on the cooperation between Trusted Flow Generator (TFG), Trusted Tag Checker (TTC) and some network interfaces (e.g. firewall, gateway) as well as Message Authentication Code (MAC) all working cooperatively to detect the tampering software.

H. Ulusoy et al. [140] proposed TrustMR, in order to detect attacks with a high probability while minimizing the overheads, TrustMR decomposes MapReduce tasks into smaller computations by means of aspect-oriented programming and replicates a subset of these tasks to verify the integrity of computations. TrustMR initiates multiple replicated map tasks on the replicated input splits. Some outputs of the map phase are randomly selected at runtime, and replicated map tasks only generate these key–value pairs. The results of replicated and original map tasks are verified at a map verifier by using a voting system. The results of replicated and original reduced tasks are also verified in the same manner as a reduce verifier.

P. Falcarin et al. [141] proposed an approach to dealing with secure messaging in Client-Server application. Their methodology is based on using Aspect Oriented programing tools in both client and server to control the transmitted messages from client to server, as well as to provide evidence to the remote server that the client code is authentic. The prototype of implementation is called TrustedFlow is held in the chatting server and contains three main aspect components: Aspect Manager, Aspect factory and Code Checker, while the client side holds the Tag Generator aspect. Through the cooperation between client and server aspects, the latter will be able to evidence that the code which has been sent by the user is authentic by checking the tag

values after each sending. They used the PROSE platform for the implementation part. Their system limits the possibility of some attacks like Discovery, disablement and replacement.

3.2.2 AOP Privacy

K. Chen et al. [142] designed an Aspect-base privacy management framework used to collect and manage patients' preferences independently yet can integrate with Health Information System (HIS) to support patients' privacy. The proposed framework is based on three main components: action purpose manager, privacy aspect and patient preference manager. The privacy aspect interacts with the Policy Decision Point (PDP) as the join point for advice weaving. The main task of the privacy aspect is to monitor the result of PDP and perform the enforcement and audition of the patient preference if necessary. The result summarizes that if the PDP grants an access request, the privacy aspect will then take responsibility to ensure that the intended use of the data matches those consented to by the patients.

C. Hankin et al. [143] used Belnap Logic to deal with aspect-oriented coordination language AspectK in order to apply security policy in each location, and then combine the relevant security policies when an interaction between locations takes place. Their framework is based on a four-valued logic for solving the conflicts such as: the value *tt* is interpreted as permitting the access, *ff* is interpreted as denying the access, ⊥⊥ is interpreted as missing information, and TT is interpreted as conflicting information. They are "attaching" aspect advices to each location to make the system more understandable and scalable.

P. Yu et al. [144] deals with the implementation of privacy-aware services in a Platform as a Service (PaaS) context. Their privacy enforcement mechanisms use AOP such that the aspects can be manipulated in the process and at the platform level. In this scenario, they adopted three main bundles managed by the cloud providers (JDBC Wrapper, Annotation Detector and SQL Filter) and one additional bundle (Policy Handler) providing privacy translation. All of these bundles increase scattering and tangling problems, thus they used AOP to address this and for better modularization.

C. Vanden et al. [145] introduced the Privacy Injector which also relies on AOSD to modularize and encapsulate privacy enforcement. The approach is based on a privacy metadata tracking part and a privacy policy enforcement part. Each piece of collected personal data will be associated with privacy metadata in the system, and any operations in the system should work only in compliance with what the metadata dictates. The proposed architecture manages the data using the sticky policy paradigm to enforce the privacy rule on the data before disclosure.

3.2.3 AOP Cryptography

H. Mestiri et al. [146] used SystemC and AspectC++ together to design an AOP-based system-level fault injection/detection environment to evaluate the robustness of the cryptographic design against fault injection attack. The fault injection/detection system has three AOP modules: fault controller specialization (FCS), fault injector specialization (FIS) and fault analysis specialization (FAS). FCS is a state controller that drives the synchronization between the other models. FIS is used to specify the injecting faults in times and locations and finally, FAS provides a report about the effects of the faults on the functional design. To show the capability of their solution they made a comparison between pure SystemC and SystemC with AspectC++ when applying the proposal fault injection/detection system into two types: single faults and multiply faults. The results show that the AOP does not have significant impact on both simulation time and size of the executable file.

A. A. Thulnoon et al. [147] utilized AOP features to control cryptography algorithms that used to ensure security and privacy of distributed system works based on choreography network. They proposed a super node called Judgment Node (JN) to control the processing starting from the source node to destination node. The main functionality of this node is to divide the distrusted nodes according to trustworthiness level to trusted and untrusted node as well as divide the sending file into portions according the process that required to finish the task. This file will be encrypted using different keys and algorithms (symmetric and asymmetric). JN will distribute the public and private keys to each node in the system as well as the symmetric key. Encryption/decryption operation are totally controlled by AOP (encrypt before sending

and decrypt after receiving). The scenario her, JN will create a balance of using cryptography algorithms be selecting the proper routing path to do the process and try to select the trusted path. If the processing routing path's nodes are trusted, then symmetric algorithm will be considered. However, if there are any untrusted nodes in the processing path then AOP will change working form symmetric to asymmetric algorithm because it more hard to be broken.

3.3 Non AOP Multi-Level Security Solutions

In this section we will review some of the projects that have been done by organizations and companies which are based on MLS concept to handle their data, and we will be concentrating on how they dealt with the "cross-domain" problem which is the major pivot of the proposed solution.

1- AXIOMATICS [148] is a company located in Stockholm, Sweden. Its premier vendor used Attribute-Based Access Control (ABAC) as a dynamic authorization adopted by more than 500 companies in a variety of fields such as healthcare, finance, manufacturing, and federal government agencies.

   AXIOMATICS presents "**SmartGuard™ for Big Data 1.1**" to show the high flexibility and the true full ability and dynamic behavioural variety of Attribute-Based Access Control (ABAC) when dealing with big data. This guard stratifies fine-grained access control principles for the data centre in Hadoop by using SQL-on-Hadoop engines HIVE and HAWQ. The basis of the guard working can be summarized through the following points:

   1- An application sends an access request to the data stored in Hadoop.
   2- An intercept agent which is appendixed with the application will intercept the SQL query and send it to SQL Transformer.
   3- Depending on the authorization policies which are associated with the application which sent the SQL query, the SQL transformer will do modifications to the query according to these policies.

4- According to the access policies, the SQL filter service will do the filtering and masking of the data which has been classified as sensitive.

5- Afterwards, the modified SQL query will return back to the SQL Transformer which is forwarded to the Application.

6- Now the Application sends the SQL query to a data store with the associated policies and rights to access only to the right data as can be seen in Figure 3.24

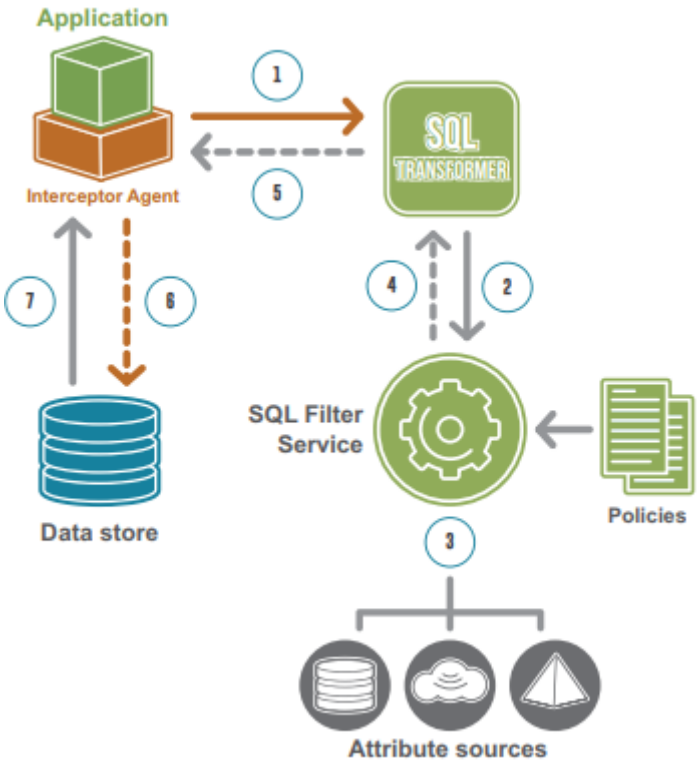


Figure 3.24. SmartGuardTM [148]

Although this guard has features like applying fine-grained access control for big data, masking the sensitive data, automatic modification of SQL query and applying all of these in dynamic manner, this guard however is a specialist only with environments that deal with databases and SQL queries. Moreover, this guard uses the Many-to-one relationship when one or more applications send SQL requests to access the data store. As has been mentioned before, the guard deals with the clients who want to access the data store, by ensuring that he/she will have access only to the permitted data. This

guard forgot the side of probability processing and flowing of data between distributed applications, and what sudden changes will happen to the access policy upon this data.

2- Trusted RUBIX™ [149] is the outcome of results of a collection of researches carried out by Infosystems Technology, Inc. (ITI) to achieve high assurance database software for clients who work in sensitive environments and need integrity and confidentiality for their data. The general model of Trusted RUBIX™ consists of three mandatory access control layers: 1) The abstract layer is Attribute-Based Access Control (ABAC), 2) the intermediate layer is Type Enforcement/RBAC (SELinux), 3) and the internal layer is Multilevel Security (MLS), so we can say the information is shielded by three Mandatory Access Control (MAC) policies as can be seen in Figure 3.25.

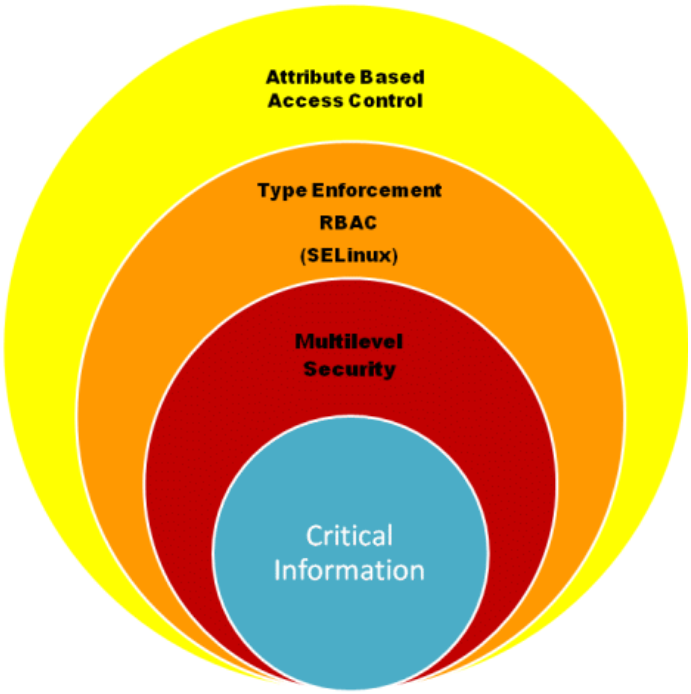


Figure 3.25. Trusted RUBIX Model [149]

For the cross-domain solution, the developers of Trusted RUBIX™ have adopted the Trusted RUBIX Security Markup Language (RXSML) policy to apply security policies. In fact RXSML policy consists of four major sub-policies (xdomain-select, mac_check, deny and xdomain-open), associated with two policy sets called

(table-obj and open-obj). In the following we describe the function of each policy individually as well as the two policy tasks:

The mac-check policy: applying OS-Mac security concept to control access to the data resource. The work basis of this policy is to use MLS principles by activating the relationship between the subject clearance (domain) and object classification (security label).

The deny policy: this policy uses "Catch-all" policy to enforce denial for the operations, when no more policies associated within the policy sets are permitted in the operations.

The xdomain-open policy: This policy allows the user in domain1 to open a session with domain2 to use data subject in domain2 and has been created by the user in domain1. In this case, domain1 and domain2 have the same security classification.

The xdomain-select policy: this policy will focus only on the "SELECT" operations from subject to object and both in different domains (even if the domains have the same security clearance but in different compartments). If the decision is permitted, then the subject will be able to have access to the selected parts of the object.

The open-obj policy: this is the set of all policies that is required for the subjects in different domains.

The table-obj policy: this is the set of all policies that is required for the objects in different domains.

To sum up the Trusted RUBIX^{TM} is how to deal with the cross-domain solution. By applying MLS, the subject in one domain has the right to access data only if it was in a different domain, but has the same security label. They ignored the possibility if the data need to flow from high to low level node.

3- Raytheon SimShield^{TM} [150] Raytheon is a company specialising in technologies related to defence, civil governments and cybersecurity solutions. Raytheon SimShield^{TM} is a commercial-off-the-shelf solution, allowing bi-directional data flow between two different security domains through providing a smart guard that

controls access to the data. Compared with the previous two solutions, SimShield™ has a comprehensive meaning to deal with the data in different formats like text, images, and video and so on. Moreover, it is presented as a two-way communication and sanitization of data flow. SimShield™ depends on the Trusted Operating System to create the trusted bridge that is used for ensuring security for data moving between different domains as illustrated in figure 3.26.

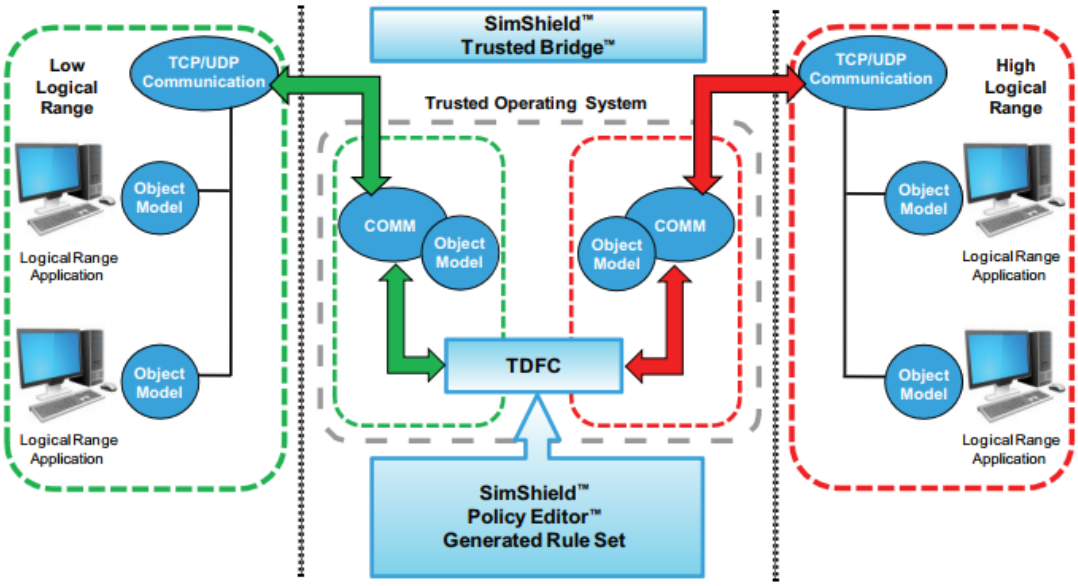


Figure 3.26. SimShieldTM [150]

4- Forcepoint [151] is a company dealing with cybersecurity solutions for different companies and governments. Forcepoint provides high speed guards that can be located between different domains to ensure the right access to data resources. As in Raytheon SimShield™, Forcepoint deals with varieties of data file types with more speed. This type of guard has the capability to do automatic sanitization on data before access by the subject to achieve high assurance security. Most consumers of this software set it up between two big networks to ensure security by filtering as well as to ensure high speed of performance as shown in Figure 3.27

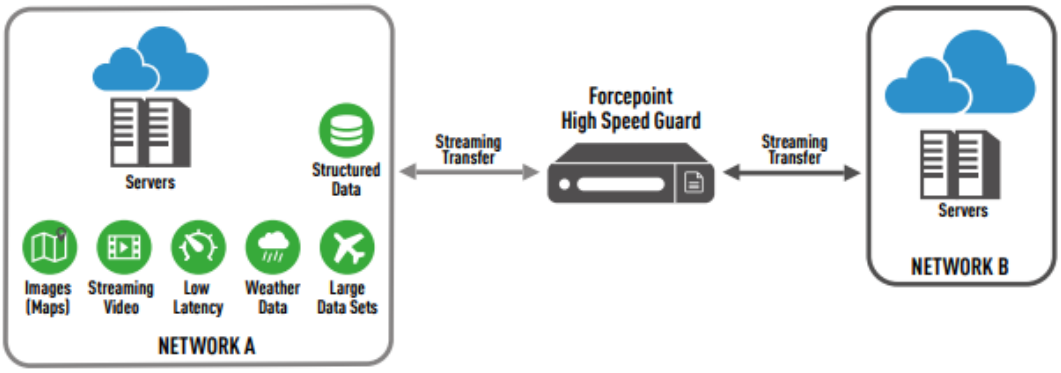


Figure 3.27. Forcepoint Security Guard [151]

The last three ways have presented high level solutions of CDS. However, through my understanding of their work I have arrived at the general idea that all these solutions are based on how the subject in the domain can access data in different domains (different here does not necessarily mean different in security label but also different in compartments), like in Raytheon SimShield^{TM} when there are two different domains and each has two communicated subjects (PC) trying to send and receive information between the domains; they need to set up a smart guard to do the sanitization of the sensitive data and files. The drawback of this guard is they need a trusted operating system to build a trusted bridge between the domains. If we see Forcepoint in depth, in spite of it providing a high speed guard, it is the same position between two different networks. Indeed, these big projects and others, and sharing with some other proposed papers which are not mentioned here, the general concept of the systems that use MLS as a basis of working, is dependent on the clustering concept in which all the system entities (nodes) will be clustered according to the security clearance that they belong to. In this case, the security guard will be in the middle way between the clustering groups only, and the data will move more freely between nodes related to the same cluster; high security outside cluster barriers and low security inside as can be seen in Figure 3.28.

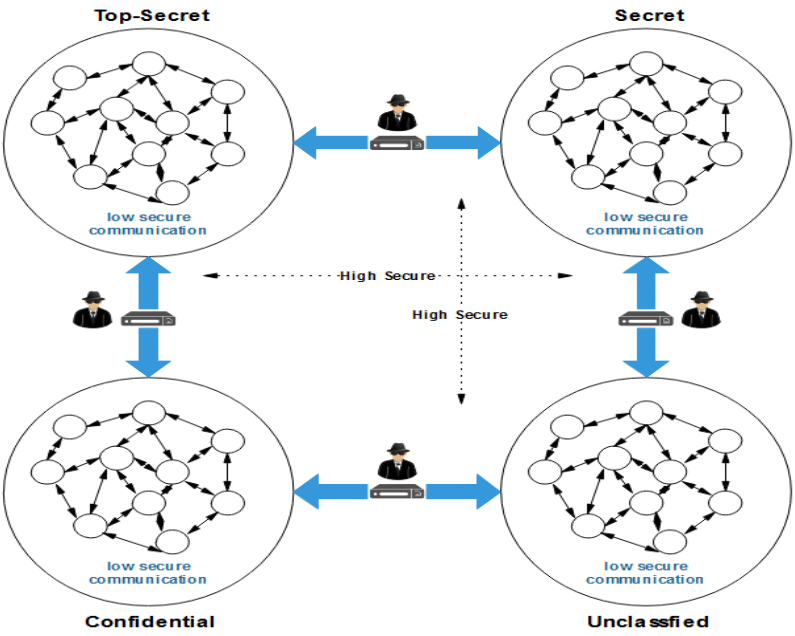


Figure 3.28. MLS Clustering

In fact, this is not practical with the enterprises that wish to upgrade their software to base on the MLS concept to ensure more security and integrity of data, since adding or updating the security side of the software, it is not necessary to change major concepts and principles of the software or start again from scratch. What the researcher wants to say here, is that not all enterprises or governments wish for the system to enforce what they should do. On the contrary, the enterprise or government should enforce what the system should do, and work to be more satisfactory.

Moreover, these projects forgot the intermediate processing and what will happen with the data and labels and what will happen if the data needs to complete the processing after sanitization.

3.4 Summary

This chapter has presented a review of related works which adopted Aspect oriented language as a basis for their solutions. It has provided a general perception of using AOP concepts in four major areas in security: Access Control Models, Intrusion Prevention, Privacy and Cryptography. This outline has shown the effect of the Aspect-Oriented technique for modularizing security concerns and changing the execution

behaviour of existing codes with only the minimal need to change existing software implementations. It has also reviewed how the AOP tool can be used to modulate the same problem but in different respects of view, for example, Role-Based Access Control has been modulated in different forms, but all give the same objectives, and this belongs to the power of this tool to deal with these concerns with more flexibility and maintainability.

During the reviewing the related works, it can be summarised that there are no current existing method can cover the requirement of DDS security In the following we review some of weaknesses points in current works:

Lack of implementation: in some proposals, the authors are satisfied by just building a framework to simulate their solutions using UML and some formal languages. We argue that because we apply the solution practically we will discover some gaps which were not taken into account at the time of framework design.

Apply partial solution: some papers provided a partial solution or modelling. For example, in the context of Access Control models, some papers take a part of the module and model it using Aspect-oriented language and ignore many roles related to this model.

Lack of Scalability: some papers do not care about the extension problems and what suggested solutions should be taken, as well as not working in distributed environments.

This chapter presented the inadequacies of the existing solution to deal with the security requirements of DDS. This is due most of them adopt point-to-point security and this sufficient for DS only. In addition, with the solutions that used MLS concept, they have weaknesses to deal with all model and DS requirement. This chapter shows why the proposed solution in this thesis is important to address the security weaknesses of the distributed systems in general and decentralised distributed systems in specific.

Chapter 4
3AC_AOP Model

Traditionally, allowing or denying access to the data resources is required as a direct controlling from the administrator to design the structure to give an access decision according to some roles and conditions. This scenario however is inadequate for some environments especially with those which are based on the distributed system concept. Most of the papers that we explored in the related work chapter focused on the relationship between the client and servers, service consumer and service providers and between the user and administrator. They identified their solutions to deal with the user who wants to access the data resource by granting him or just decline his access. If look carefully at these solutions, most of them forgot the whole figures of distributed system environments and what may impose on the data when flowing between the system entities. This motivated us to propose an access control model called (3AC_AOP) which aggregated the Attribute-Based Access Control, Identity-Based Access Control and Multi-Level Security (MLS) to fulfil the security gaps in distributed systems. These models will be modularized by using Aspect-Oriented Programming language to deal with all access control issues with more flexibility and manageability.

This chapter presents the proposed access control model 3AC_AOP and how it can deal with weakness points of the existing solutions. Besides, AOP cryptography aspect is presented as a method to achieve data integrity. The chapter explains how we utilized AOP to build AOP automatic security guard associated with 3AC_AOP model to ensure high cooperation between system nodes synchronized with achieving high security. To show the advantage of using the DDS environment, this chapter has presented a comparison study between two different processing in distributed system, Separation of Duty (SoD) and Service Function Chain (SFC) which is the adopted process. Finally, this chapter provides the analysis of how the proposed system works.

4.1 3AC_AOP Model

3AC_AOP is a security mechanism combined three fundamental security solution methods, access control (Attribute, Identity and MLS)-based access control models, cryptography (AES symmetric algorithm) and security guard. All these components are modularized by AOP to produce 3AC_AOP security model as illustrated in Figure 4.29.

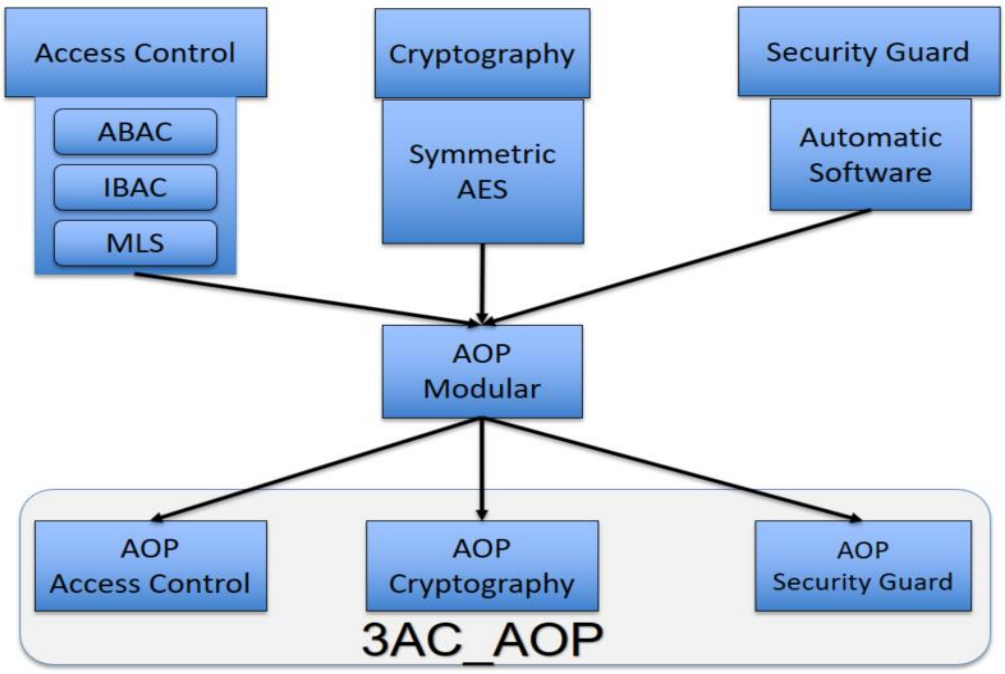


Figure 4.29 3AC_AOP security model components

Before diving deeply into model's components, it is important to have a general understanding of the model and the interactions between the model's parts. In general, there are four main types of nodes according to the processing requirements as following:

1- *Source node*: the node that the process request initiated from. Access control conditions are set in and the information (messages and files) are classified and tagged according to information sensitivity.
2- *Bridge node*: or hop node is the node where the receipt access conditions do not match with the node's accessing conditions. The processing with this node is just to re-send the encrypted information to the next node.

3- *Processing node*: is the node where the receipt access conditions match with the node's accessing conditions. This node will perform its own function and delete the current attribute.

4- *Destination node*: is the latest node in the processing path. According to the source node's request and after the final process is done in this node, the processed request is sorted in the node or will be sent back to the source nodes as a response. Figure 4.30 shows theses nodes according to their missions.

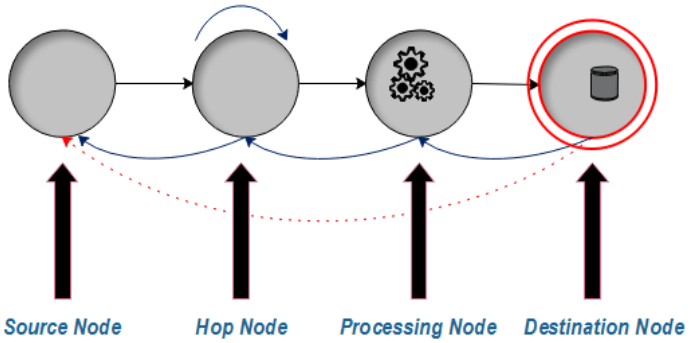


Figure 4.30. Types of Distributed Systems Nodes

Each node in the distributed system can have one of the above nodes types. Figure 4.31 shows 3AC_AOP model, the process starts from the source node by establishing a new request, the information associated with access control conditions will be sent to the next node. AOP encryption pointcut will intercept the sending process by encrypting the access conditions and the information separately. Before the information is received by the next node, the AOP decryption pointcut will decrypt the receipt information sequentially, by decrypting the accessing conditions first to implement the authority, if granted then decrypt the information otherwise re-encrypt the access conditions and re-send it with the information to the next node. If the access is granted then the node will perform its own function and delete the executed attribute. This process will continue until the attributes list is empty. There are two cases discussed in this model, the first one, is sending a request to the destination nodes while the second case is sending the request and waiting for the response. The second case just includes dealing with the file, however, if the sending information is messages, then only the first case will be applied. Figure 4.32 presents the flow chart of the proposed model in detail.

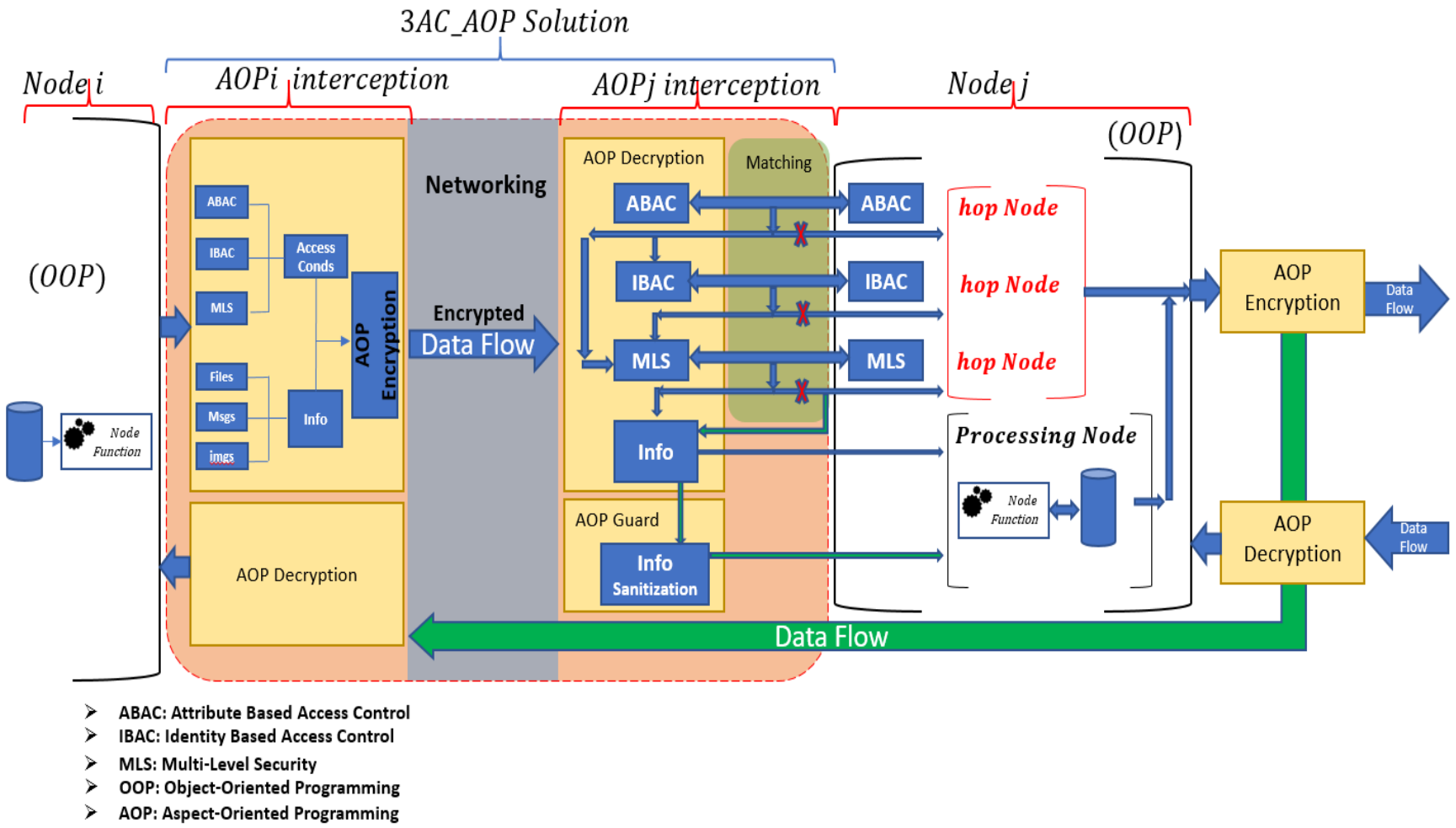


Figure 4.31. Design of 3AC_AOP Model

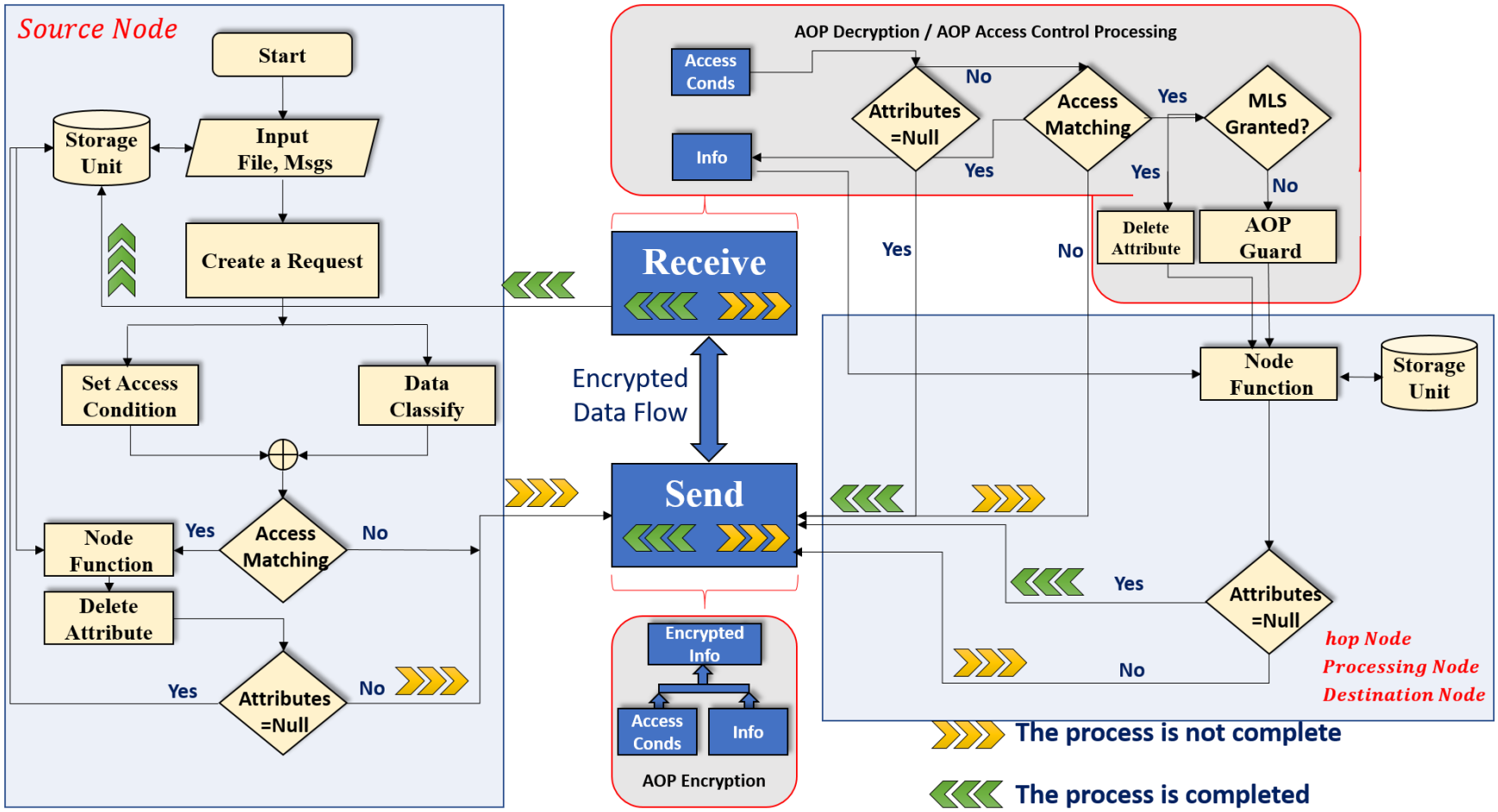


Figure 4.32. 3AC_AOP Model Flow Chart

4.2 3AC_AOP Access Control

As mentioned before, the proposed Access control is comprised of three access control models, Attribute-Based, Identity-Based and Multi-Level Security access control models. Figure 4.33 shows the orders of the models starting from ABAC as the abstract level, followed by IBAC as optional (use in special cases) as intermediate level and ending by MLS as a core level of the model.

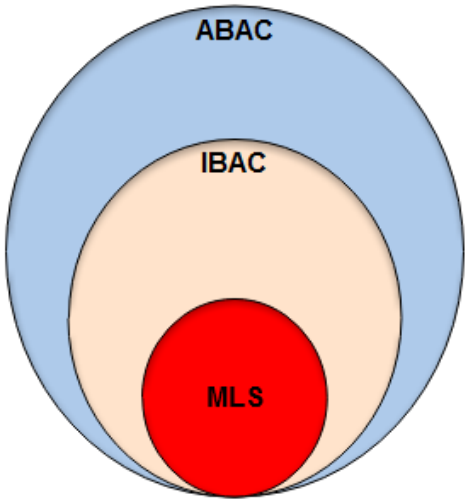


Figure 4.33. Proposal Model

1. Attribute-Based Access Control (ABAC)

As mentioned in section 2.7.5 (background chapter), in the MLS system some organizations have been adding "compartments" as extra elements to strengthen the access restrictions when they composite with Multi-level security concepts. These compartments can be considered as services and functions associated with both subjects and objects. 3AC_AOP mode is dependent on separating these compartments from the MLS's body and using them as attributes in ABAC.

There are different perspectives to deal with ABAC, some of these views use attributes as services and others deal with these attributes as approaches, and all of these models are correct, see Figure 4.34. The ABAC part of the model is based on the definition by NIST 800-162 [59].

"*An access control method where a subject requests to perform operations on objects are granted or denied based on assigned attributes of the subject, assigned attributes of the object, environment conditions, and a set of policies that are specified in terms of those attributes and conditions*"

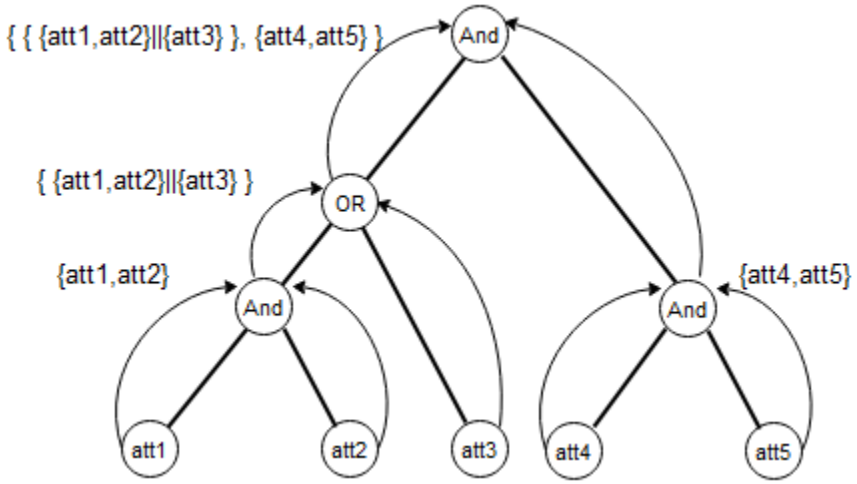


Figure 4.34. Attribute-Based Access Control

2. Identity-Based Access Control (IBAC)

This part of the model is optional and will be controlled by the source node which has the freedom to use this part of the model or not. In other words the naturalism of making a decision of using this section of the model or not is based on what the source node needs to do on its own sending data. For instance, assume an enterprise administrator wants to send information to any user having associated with attributes {att1,att2} as can be seen in Figure 4.35, where the data has been sent to only users id1 and id2 who associated with {att1,att2} and ignored the other users.

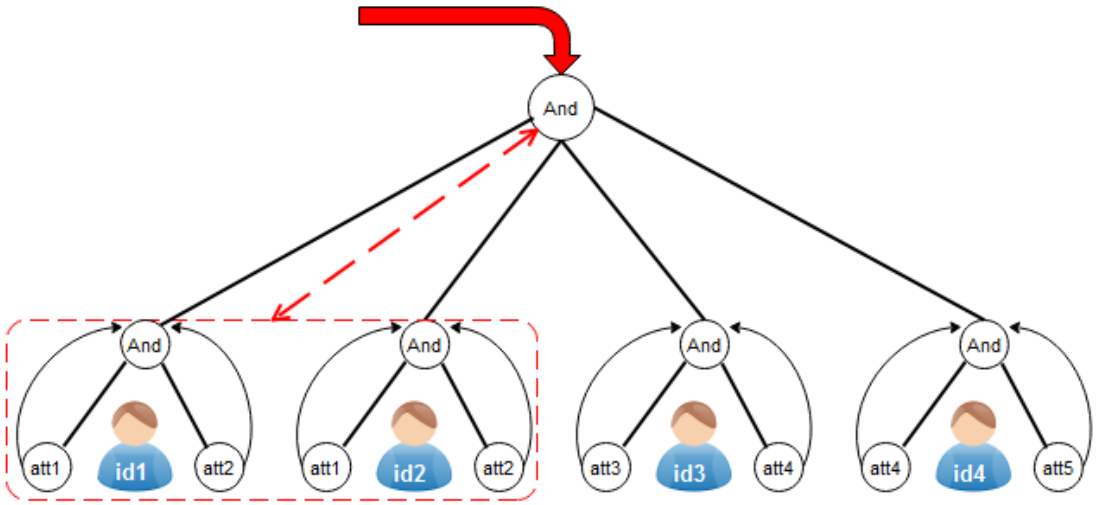


Figure 4.35. Identity-Based Access Control

On the other hand, if the source node activates the IBAC model, then the information will flow only to a specific user who has been selected by the source node as demonstrated in Figure 4.36 where the data goes only to user id2, even if he/she has the same attributes as user id1.

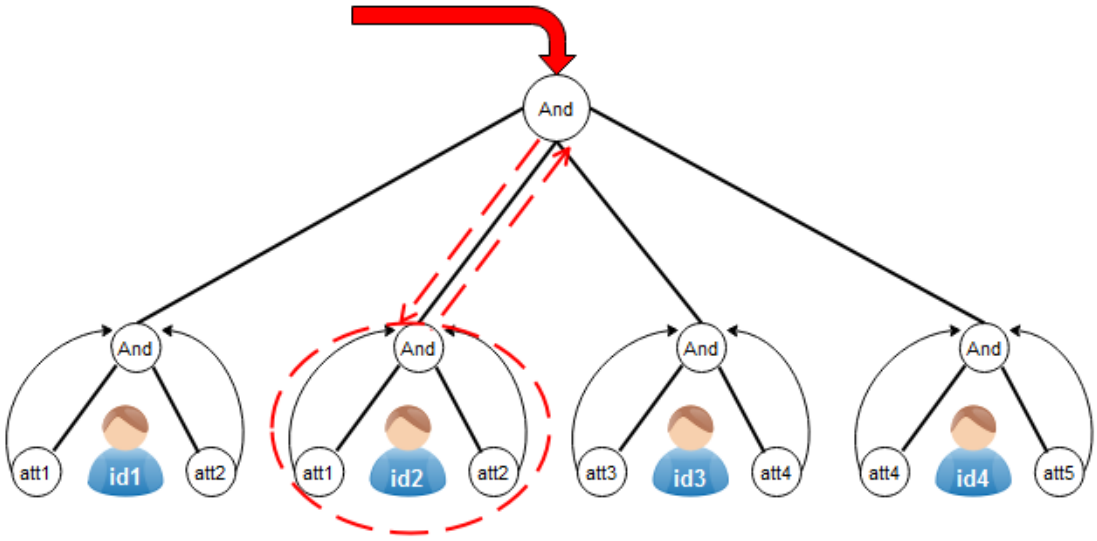


Figure 4.36. Identity-Based Access Control

Making a decision is a responsibility of the source node, whilst AOP has the responsibility of controlling the access of this data to the right direction according to the decision taken, as we will illustrate in the next lines in this chapter.

3. Multi-Level Security System

In this part of the model we will use the Bell-La Padula model and how it deals with both subjects and objects to control the access to the data. Furthermore, this model will be supported by dominance relationship. This relationship focuses only on subjects, regardless of objects, as it is considered to relate totally on subjects that own them.

The dominance relationship is:

- ***We say A is dominate B IFF security label of A is Bigger than or equal of B***
- ***We say B is dominated By A IFF security label of B is Lower than of A***

The order of subject clearances are:

Top Secret > Secret > Confidential> Unclassified

The order of sensitivities of data are:

TS > S > C > U

Data Attribute

This is what the source subject associates the sending data with the function name that needs to be accomplished by another subject.

Subject Attribute

This is what the subject can offer for functions and services.

Sample intersections between data attributes (compartments) and subject's attribute are:

Data attribute	Subject Attribute	Intersection
{A,B,C}	{A,B,N}	{A,B}
{A,B}	{B,C,D}	{B}
{A,B,D}	{X,Y,Z}	{θ}

Rule

Let D= Data and Sub=Subject.

1- Low to ---

- If D{d.atts} ∩ Sub{s.atts} ≠ {θ} then

  If D{d.atts} ⊆ Sub{s.atts} then

  If D{d.atts} ∩ Sub{s.atts}) = {d.atts}then

  allow all D

  else

  If (∃ a ∈ {d.atts}) ⋀ (a ∈ {s.atts}) then

  allow only a

  Else

  Decline

2- High to –

- If D{d.atts} ∩ Sub{s.atts} ≠ {θ} then

  If D{d.atts} ⊆ Sub{s.atts} then

  If (D{d.atts} ∩ Sub{s.atts}) = {d.atts}then

  If Sub.lable=Data.label then

  If Sub.id= D.id then /* optional*/

  Allow all D

  If Sub.lable< Data.label then

  If Sub.id= D.id then /* Mandatory*/

  Do sanitation of D then

  Allow only the intersection set

  Else

  If (∃ a ∈ {d.atts}) ⋀ (a ∈ {s.atts}) then

  If Sub.lable=Data.label then

  If Sub.id= D.id then /* optional*/

  Allow all a

  If Sub.lable< Data.label then

If Sub.id= D.id then /* Mandatory*/

Do sanitation of D then

Allow only a

Else

Decline

4.3 3AC_AOP Operations

After talking about the elements of the system, this section concentrates on the operations that are enforced on data by both source nodes or processing nodes according to what the AOP is designed for.

- **Node labelling**: Each node in the distributed system will be associated by an integer number between (1-4). This number represents the security clearance label of the node such that {4, 3, 2, 1} correspond {Top-Secret, Secret, Confidential, Unclassified} respectively.
- **Data labelling**: As explained in section 2.5.3, the data will be divided into four parts according to the sensitivity of the information. ***D (Identifiers, Quasi-identifiers, Sensitive-attribute, Non-sensitive-attributes)*** Data labelling will be {4, 3, 2, 1} respectively. The source node or the owner of the data will assign these labels to his own data as follows:

Identifiers → label = 4

Quasi-identifiers → label = 3

Sensitive-attribute → label =2

Non-sensitive-attributes → label = 1

Identifiers ∪ Quasi-identifiers → label =4

Identifiers ∪ Sensitive-attribute → label =4

Identifiers ∪ Quasi-identifiers ∪ Sensitive-attribute → label =4

Quasi-identifiers ∪ Sensitive-attribute → label =3

In this case any piece of information will be labelled according to the level of sensitivity as well as if this piece is associated with another piece of information and the latter has a higher label, then all the mixers will be labelled as the higher label as demonstrated in the Figure 4.37.

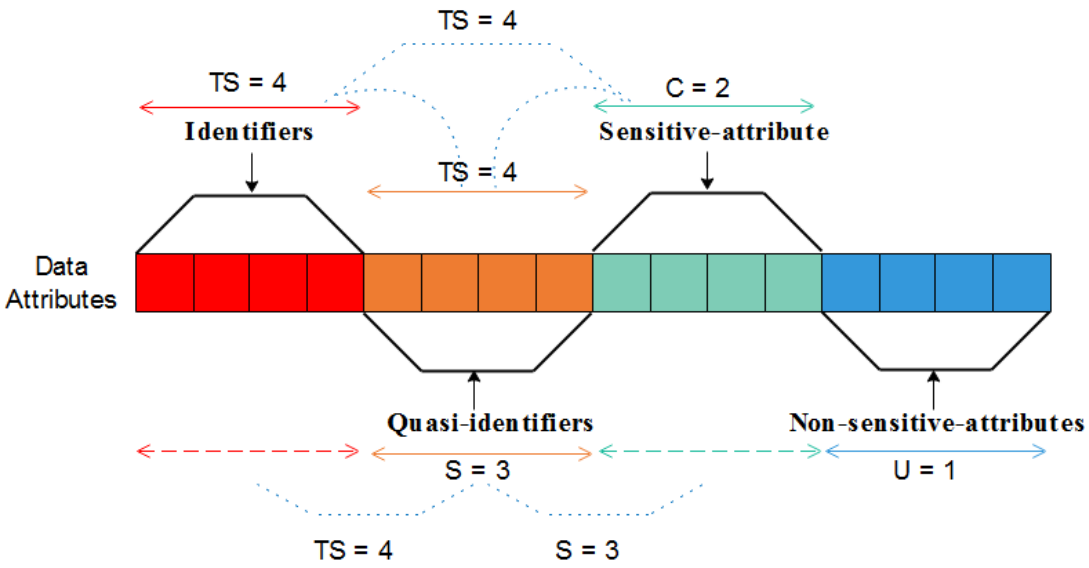


Figure 4.37. Privacy Preserving Division

4.4 3AC_AOP Cryptography

To ensure the integrity of the data transmission between system nodes, a symmetric cryptography algorithm has been adopted to accomplish this mission. The encryption process will be included both data, and access conditions individually. The process starts after authorization and authentication is granted, then decrypt the data file, otherwise keeping it in encrypted form and sending it associated with encrypted access conditions to the next node in the system.

The encryption and decryption processes will be controlled totally by AOP where they intercept the encryption process before the data leaves the source node and decrypt the data after the success of all authentication and authorization processes and before going to the receipt node as shown in Figure 4.38.

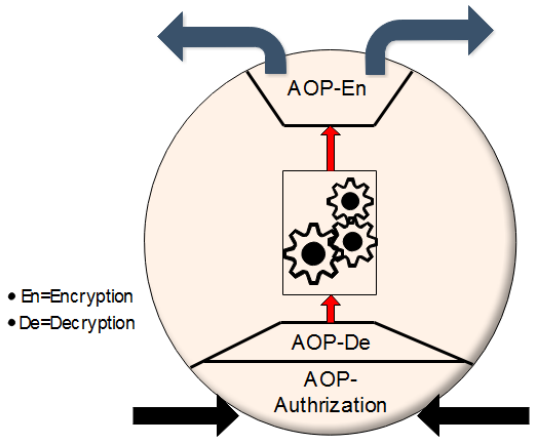


Figure 4.38. Cryptography

4.5 3AC_AOP Security Guard

AOP security guard is used to ensure the integrity of the data during the processing, and moreover to prevent the low level clearance node reviewing the sensitive information that has been sent by the high level node. We argue that not all information in high security nodes has a high security level, otherwise it would not have been treated with the information coming from low level nodes. With the purpose of achieving high levels of cooperation between system entities and high security assurance, we include the sanitization method to deal with the data that is going to be processed inside the node.

Similarly with the cryptography methods, the data sanitization method will be totally controlled by AOP in which it will be the intermediate phase between AOP-authorization process and AOP- encryption process. Because the sanitation process will only be used when data flows from high to low levels of security, the dynamic AOP security guard will be adopted to do this task. In other words, if the network is working in a correct form and data flows just from low to the same or to a high level, then the AOP which is used for the sanitization will be off and will not occur just if the flow was from high to low.

Dealing with sanitation methods falls into two ways:

1- Temporary sanitation (intermediates nodes):

In this method, AOP will erase the sensitive data temporarily by keeping them in a temporary storage device before the rest of the data goes to the node. After the intended node did the process on the clean data and before sending it to the next node, AOP advice will append the sensitive information with the processed data, to recreate the whole data file before going to the encryption process also by AOP. This type of sanitation method is used only with the intermediate processing nodes (the nodes that need to do process on data before approaching the destination node as can be seen in Figure 4.39.

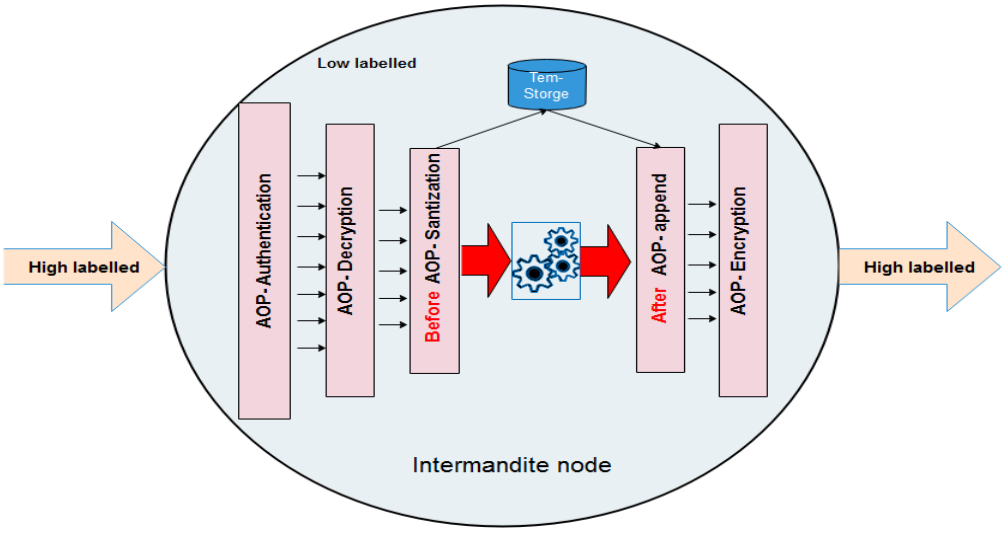


Figure 4.39. Temporary sanitation

2- Full sanitation (destination node):

This type of sanitization method has two faces:

1- If the destination node is the final processing station and the source node did not ask for a response, then the sensitive data will be erased totally and deleted by the AOP sensitive data remover as shown in Figure 4.40.

2- If the destination node needs to respond to the source node request then AOP will work like in point1. All these scenarios will be explained at the end of this chapter.

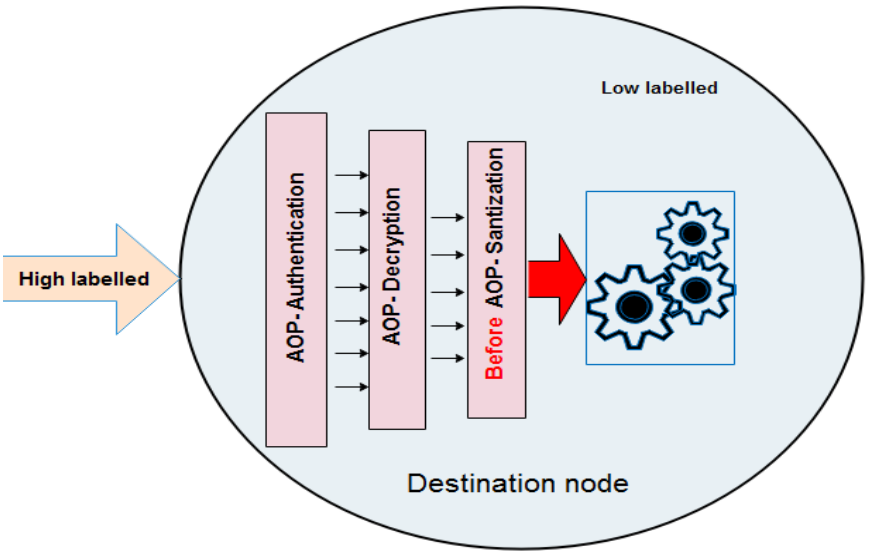


Figure 4.40. Full sanitation

4.6 Declassification:

Customarily, declassifying security labels of information is a complex task because it could have serious consequences, especially for the system using MLS principles as the basis of work. In this model, the domination relationship gives the higher classification node the ability to declassify their data only as necessary. In our view, declassifying data does not require that all data files will be declassified, but only the piece of data that the source node wishes to process in the lower node. Indeed, the declassification process is considered as the next process after the sanitation process, and it is controlled by AOP pointcuts.

4.7 Separation of duties and service function chain

This model provides a flexible handling of the data. The major challenge, however, is how to control the accessing and security policy when the data flows and processes between distributed nodes. For this reason, this model and the outcome result concentrate on how to do sequential processing on data rather than separate processing tasks between different nodes at the same time. In the following example, we show the benefits of using the sequential process beside the separation of duty:

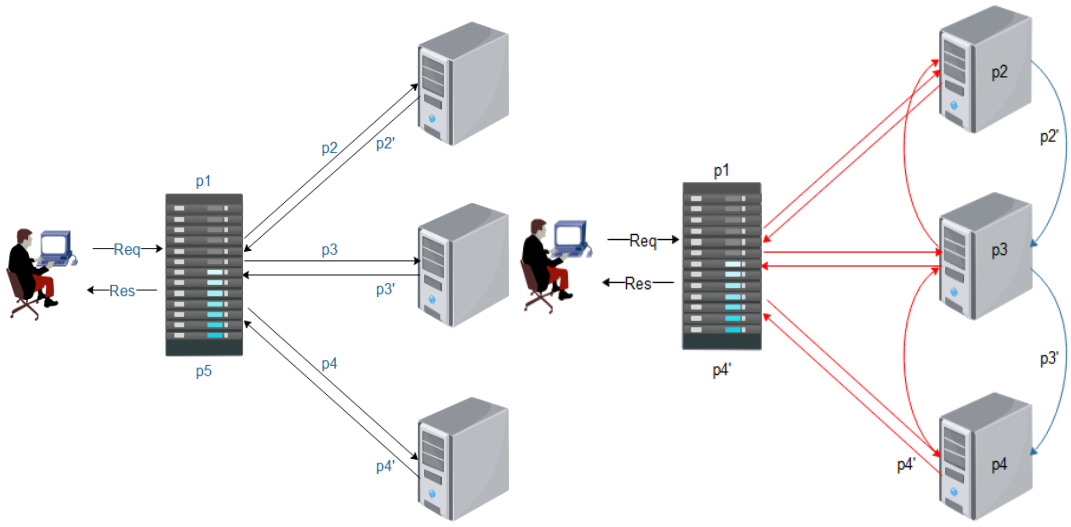
Let's suppose a client sends a query as a table with 10 names and some other identification information to checking:

"How many of these names, suffering from heart disease, get a benefit from city council and have no crime record".

Traditionally, the main server will divide this query into three parts depending on separation of duties concepts. Firstly, "*suffering from heart disease*" this part of query will be sent to the Healthcare system server. Secondly, "*get a benefit from city council"* this part will go to the city council server and finally "*have no crime record"* this part is a police officer server's task.

Let's suppose each record in the table needs 1s for processing and matching, then the result simply will be:

$3_{\text{servers}} \times 10_{\text{sec}} = 30_{\text{sec}}$ is the only time needed to process in all servers and follows with the time that is needed to handle the results in the main server before returning the response to the client, as can be seen in the Figure 4.41–a-.



-a- **-b-**

Figure 4.41. -a- Separation of Duty (SoD), -b- Sequential Processing

In comparison, with the sequential processing concept as be seen in Figure 4.41 –b-, the latter has many features which make it more efficient and reliable, because only the first server in the processing path will receive the whole data. While the remaining two

servers will receive only the outcome results from the first server, in which the second server will receive the first server's outputs and the third server will receive the second server's result, and so on. Thus this reduces the execution time and there is no processing required from the main server, as the result from the final server is the same as the client's response. Let's return back to our example, the execution time can be calculated by doing an addition operation between execution time for each server as follows:

Running time = (execution time –Server1)+ (execution time –Server2)+ (execution time –Server3)

As explained above, server2 will use the output of server1 and so on. Let's suppose that each server ignores 2 records as there is no matching, then the total time will be

Running time = $(10_{sec}) + (8_{sec}) + (6_{sec}) = 24_{sec}$

To sum up, the sequential process (sometimes called service function chain) is more efficient compared with the separation of duty process. In fact, there is no matching between their execution times except if there is a comprehensive matching between the income and the output of each server. Even then, the sequential process still has the foreground because the result from the last server will not need any other maintenance in the main server before returning the result to the client.

All these previous factors prompted us to choose the sequential processing system as a basis to enforce the proposed to ensure the security and privacy of information in both data flow and data processing in distributed system environments.

Figure 4.42 shows the whole figure of the proposed distributed system working and illustrated how the data processed and travel between system nodes.

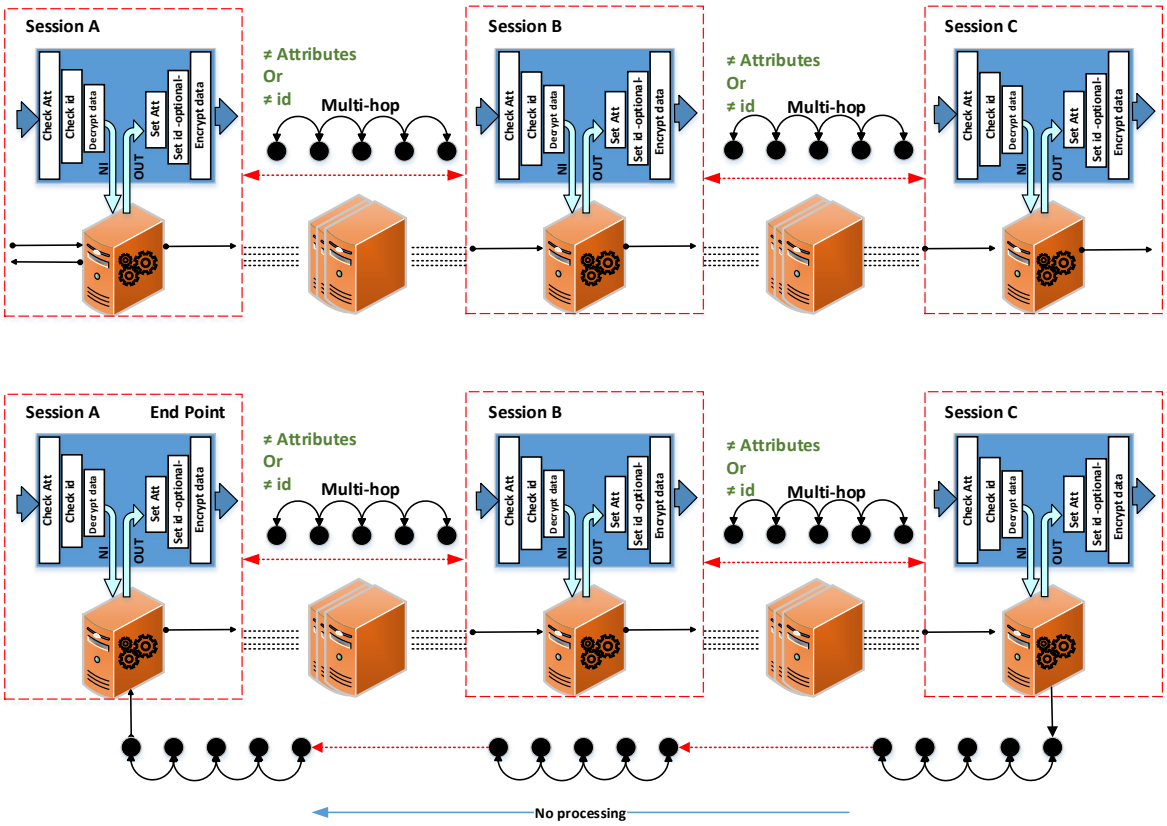


Figure 4.42. The Final Model

4.8 System analysis

Despite the increasing consolidation between access control models becoming more common recently, the execution and running of those consolidations however remains a hard task to do, and most of these models are placed in the proposed systems box. The main reasons for combining between access control models is to strengthen the features of each model at the same time reducing the weakness points which are associated with some models. There are many challenges which should be solved before doing the combinations between access controls:

1- Are the consequences of the consolidation meeting the customers' (companies') requirements?

2- Is the combination able to keep the features of individual models or will they be removed to produce a new hybrid feature?

3- What side effects will occur if the final access control model is applied to existing software?

4- Do the companies or customers need to change their software and follow what the new software with access control will dictate to them?

5- Does the new model take into account the dynamic behaviour of the distributed system?

6- Is the scalability of the model able to meet that of the company's software?

7- How does the model deal with the software's users, as trusted, untrusted or mix (need TCB)?

8- What attention should be paid for converting from an insecure system to a secure system, and adopting Access control as a security basis?

9- Is the consequent model able to satisfy the security requirements (C, I, and A) [152]?

10- How much should the cost be if the organization uses high cost software or depends on new hardware to do the task?

11- Does the new model affect the performance of the software?

In fact, all of these points should be taken into account before trying to combine between two or more access control models. Moreover, applying the consequences model practically is totally different from dealing with it as a framework, because in practice the reality will be simulated, and the real outcomes will be studied to produce a final version free from errors ACAP.

In the following we will explore the features and the capability of the model and we will show how it deals with the above concerns and what suggested solutions are to be adopted:

1- The combination of Access control models (Attribute, identity, MLS)- based Access control is done in a sequential manner. In other words, the access restrictions and conditions will start from the Attribute model and forward down to the Multilevel Security model. In case the access condition does not satisfy any model layer except the last layer then the access will be declined. Otherwise, access to the information will be allowed, after taking all precautions to prevent any types of intended or unintended penetration as the task would require at that time. We utilised AOP to design these models separately to prevent the occurrence of any tangling during the design and processing, after that we combine them together by using AOP as well.

2- We applied the access control model by using AOP. This helps to keep the original software the same, just as AOP has selected some pointcuts to intercept the execution of the software by injecting the model to control access to the information, without affecting the original software. Indeed, this is what motivated us to design the model using AOP.

3- The model will be able to face the dynamic changing behaviour of the system during the execution. Because of using AOP, it will be able to monitor system working, and change the working as to what the task requires at that time.

4- The nature of networks and distributed systems must be flexible and scalable to keep pace with the frequent development of these environments. Security is one of the major challenges that should be taken into account in these systems, because adding or updating security requirements will be a hard task for system developers and programmers. They need to check all the software codes to select which piece should do the security task, thus raising tangling and scattering problems of software codes. AOP solves this problem by keeping the

original without changing or just needing a tiny change, and the whole security software will be designed as a separate package. This will increase the stability and flexibility of distributed system environments.

5- The consequence model supports the sequential processing (SP), when the information is required to be processed by the intermediate node(s) before arriving at the destination node. This feature is totally controlled by AOP through doing all the required access control operations dynamically, and during the running time.

6- In addition to supporting sequential processing (SP), proposed model supports the separation of duty (SoD) concepts. In this case, the source node is able to separate its own tasks between some distributed nodes and get the result back after processing.

7- Information confidentiality is being achieved by adopting MLS concepts as access control layers in the proposed model, in which no one can have access to the data file or even a piece of information without appropriate permission. This permission is the consequence of comparing between the node clearance and security label of information, and this is done by AOP also.

8- Data integrity is being achieved by using cryptography algorithms to encrypt data during travelling between system nodes. Encryption and decryption operations are being controlled by using AOP advices. 'Before advice' will be used to encrypt the data before leaving the node, and 'after advice' will decrypt the data after satisfying all conditions of the access control layers.

9- Bidirectional data transition; by using AOP as a shield to protect the information in the context of the node by using access control and cryptography as well as in the context of the data flow by encrypting the transmitted information. The information now is free to move and travel between system

nodes, regardless of the security label of the information or security clearance of the nodes. The AOP shield will check the information, if it matches the access control requirements (as the source node wishes) then it allows the information to pass inside the node after decrypting the permitted part of the information; otherwise it uses the node as a router (bridge) to transmit the information to the next node in the distributed system.

10- To achieve high flexibility and cooperation between the distributed system nodes, as well as to satisfy the high level of security and information safety to the system, we use AOP security guard. In the proposed solution related to this side of the model, we adopted the "*domination relationship"* to restrict the allowance of the low clearance node to access unclassified information within a file which has been classified as a higher level than the node. This decision will be taken by high level nodes upon the lower ones, to allow the low level nodes to get only information classified as low, after sanitization of all sensitive and high importance information.

11- In the context of the sequential process (SP), sometimes the information needs processing accomplished by the intermediates nodes and this (these) node(s) have lower clearance than the information. In the proposal we solved this problem by sanitizing the information before it arrives at the node processer and saving the sanitized information in temporary storage. After finishing processing, the sanitized information will be added to the processing outcome to recreate the file, before the sending information process in the model. In this case we ensure the integrity and confidentiality of the sensitive information by saving it in encrypted form, to prevent anyone trying to retrieve the information by using the lower nodes in future.

Figure 4.43 shows the data flow inside the node.

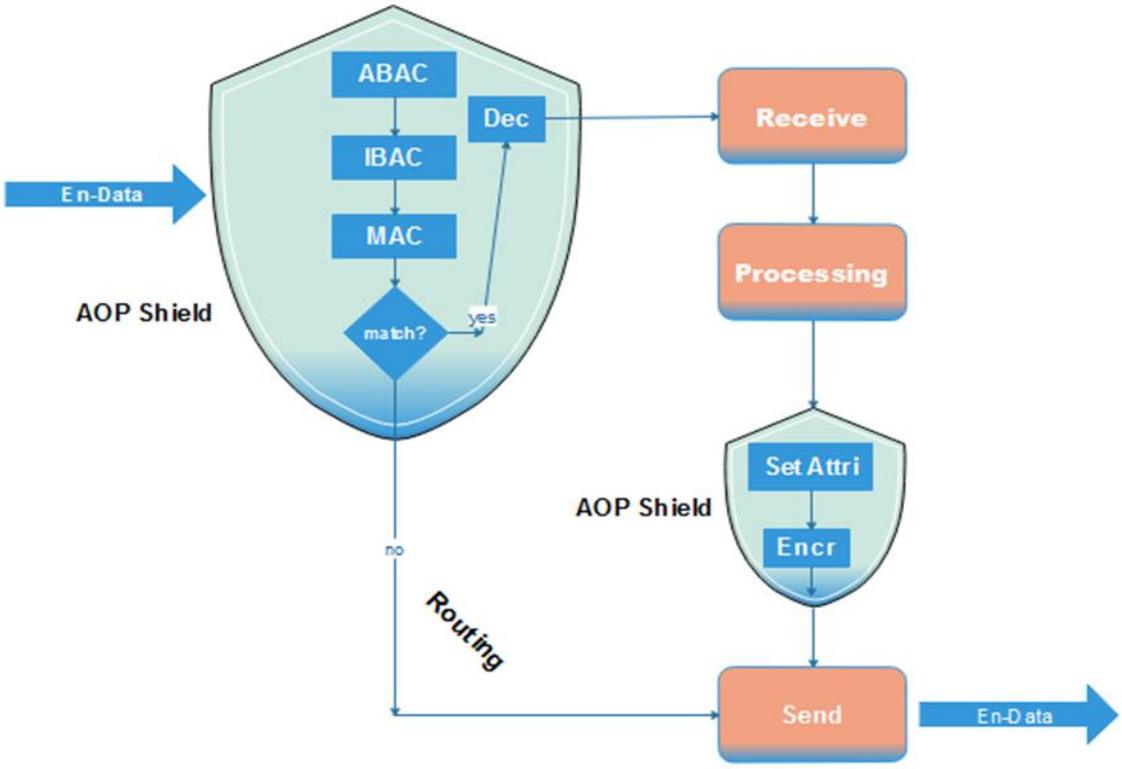


Figure 4.43. Data Flow in the Model

4.9 Summary

This chapter has provided a comprehensive description of the proposed access control model to be presented as a powerful model to prevent unauthorised access to data resources in distributed system. Beside, all fundamental topics that construct the solution have been explained in depth. The chapter started with the proposed access control model in 3AC_AOP , we have explained the elements of the model, ABAC, IBAC and MLS. Moreover, we have detailed the problems that are associated with MLS and we presented the appropriate solutions. Cryptography was also presented in this chapter as a second support solution. We have shown how we utilized AOP to create a security guard setup between system nodes to secure the communication and data flow from high to low.

Chapter 5
The Implementation of 3AC_AOP

This chapter provides an explanation of the details about the implementation of the proposed system. Firstly, we developed a general decentralized distributed system (DDS). This system comprises N number of distributed machines (for ease we will use nodes[2] rather than machines) connected by TCP/IP protocol. We have designed each node to play the role that an individual node should have in a DDS, in which each node has both status (Client and Server) simultaneously. Thus, each node has the same ability to send request, receive, respond and to do its own function.

In this chapter has focused on DDS and how to apply the proposed solution to ensure high security and privacy protection.

5.1 Methodology

In this section, the experimental results of the proposed solution have been evaluated by comparing the appropriateness of using the proposed model to develop a secure scheme and the traditional development by using OOP. Each node has been developed by OOP to perform its own function, sending, receiving information and other methods which are designed to coordinate the essential methods. We have held all these methods in the same class, in which accessing conditions have been added to the code to associate sending, receiving and handling the node function. With cryptography concerns we developed external classes to reduce the scattering and tangling caused by accessing conditions as well as the cryptography algorithm. On the other side, with AOP, the pointcuts will intercept all required methods inside the main classes without needing to recreate new classes dealing with the sending, receiving and implementation functions of the node, thus decreasing the need to modify the main code as illustrated in Figure 5.44.

[2] It's a term referring to individual server, workstation, or any network terminal. This thesis deals with node as independent code can be represented by PC. However, we include all nodes in one workstation PC but this not prevent each node can setup in individual PC.

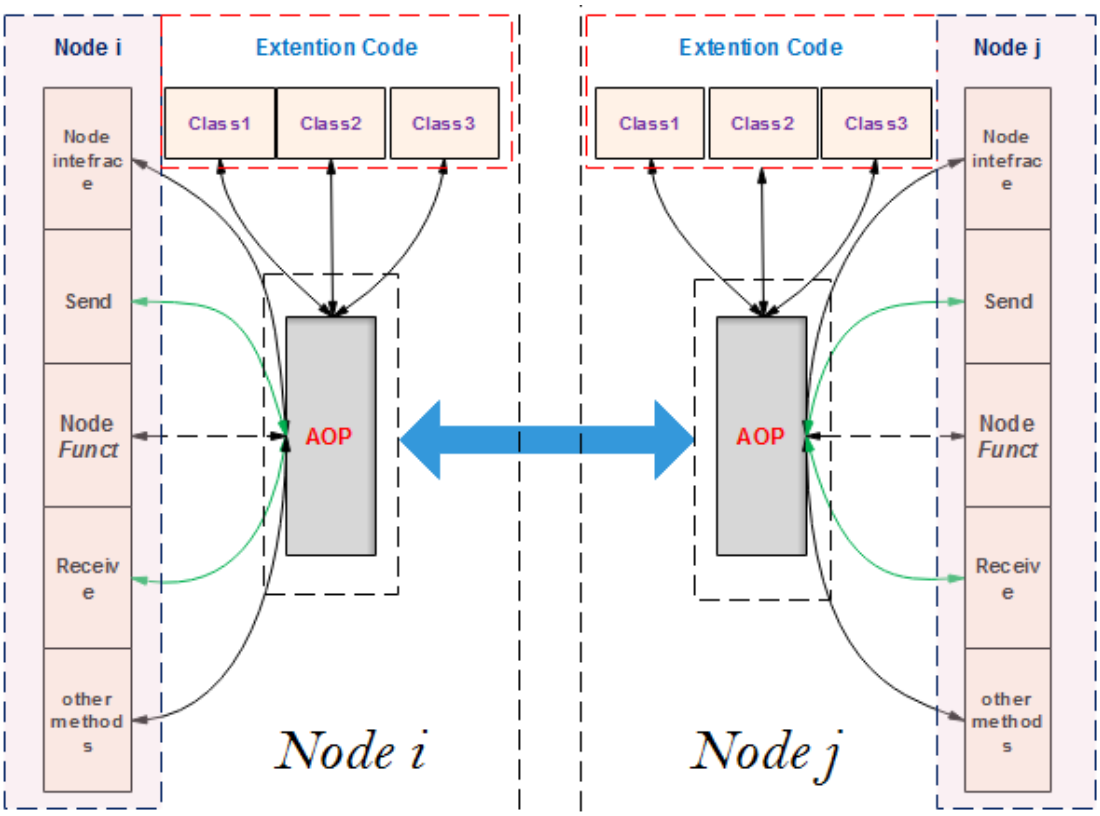


Figure 5.44 Intercept node methods using AOP

The evaluation is based on runtime performance between OOP and AOP to accomplish the required mission. This time is calculated by millisecond and different from time to time according to CPU status and the process function. The proposed solution used java eclipse Oxygen 2017 to develop the system and AspectJ to deal with AOP. The properties of the operating system are:

- Windows10
- Processor: Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz
- RAM: 16.0 GB
- System type: x64 based processor

5.2 Decentralized Distributed system

DDS has been described in the background chapter (Chapter2) however, this section re-describes it from the aspect of the coding perspective. In DDS the naturalism of the networking is slightly different from CDS. There are no central server(s), each node has both statuses (Client-Server) at the same time as shown in Figure 5.45 –b-. In comparison with the CDS this network is considered as an unstructured network and it might be that there are no direct connections between nodes, thus sending a request to the intended node(s) may require a multi-hop concept to approach the node(s) as illustrated in Figure 5.45 a. Node1 sent a request to Node4

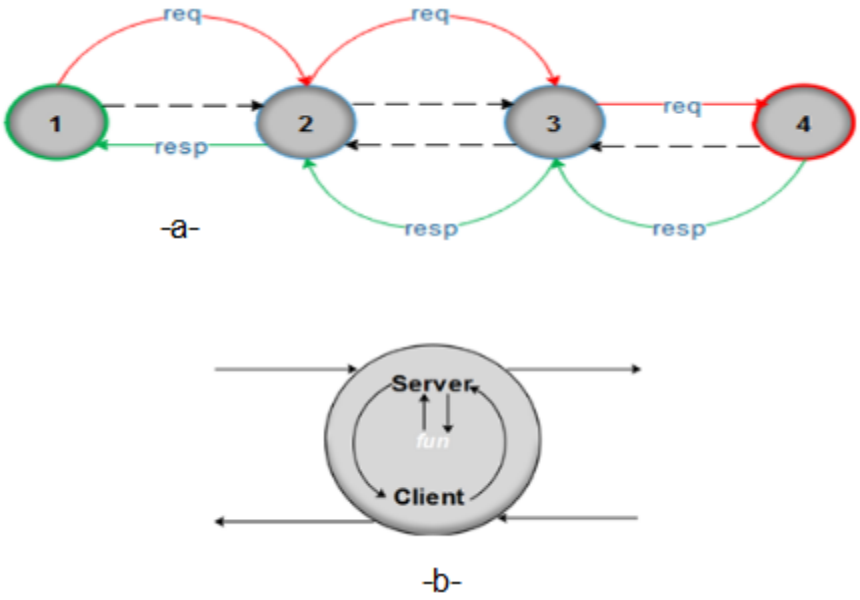


Figure 5.45. Decentralized Distributed System Adopted Scenario

there are no direct connections between these nodes. To accomplish this process, the request query would hop between Node2 and Node3 to approach Node4. This will be fine from the perspective of DDS. However, if DDS adopted security concepts, then the data transmission between nodes would be more restricted or even impossible. For example, with the system that is based on multi-level security (MLS) concepts, if the security clearance of the node and security classification of the transmitted data are not meeting MLS conditions, then the data transmission will be impossible. In the proposed solution we solve this problem by utilizing the characteristics of aspect-oriented

programming to create a shield around the node. Thus, the security processes will be implemented in the abstract level of the node's code. The proposed security scheme compromises of three main solutions: access control, cryptography and security guard. This chapter reviews the implementation details of the proposed solution, by starting with AOP and how it intercepts the code body in a specific join point. Next we are dealing in depth with the proposed solution in three phases: 1) access control 2) cryptography and 3) AOP guard.

5.3 Aspect-Oriented Programming.

More importantly, to choose carefully the relevant join points within the original code to be intercepted by the pointcuts, the objective of the proposed solution is to ensure the security and integrity of the data and the node itself. For this reason AOP pointcuts will intercept the sending and receiving operations only when a node sends a request or receives a response from other nodes. If the request however, is directed to the same node then the AOP's join point will be suspended and the code will proceed normally. Access control policies and encryption/decryption operations will be organized and applied as Aspect pointcuts moreover, access control actions will be performed by Aspect advice. We have used ***around()*** an AspectJ advice to intercept the communication because it represents the meaning of both ***before()*** and ***after()*** AspectJ advices, so we achieve a high encapsulation of the data which is transmitting between system nodes. Instead of intercepting the sending and receiving methods by method names, the pointcuts intercept the internal java reserved methods of both sending and receiving methods. The next code shows sending (creatRequest()) method in the proposed system.

Java Code : Sending method

```
public void createRequest(String server, int port) {
       try {
              socket = new Socket(server, port);
              dout = new DataOutputStream(socket.getOutputStream());

              int counter = 0;
              while ( counter < REQUEST_PROCESS.length) {
                     dout.writeUTF(REQUEST_PROCESS[counter]);
                     dout.flush();
                           if(counter == 0) {
                                  dout.writeUTF(sender);
                                  dout.flush();
```

```
                        } else if(counter == 1) {
                                dout.writeUTF(attributeList);
                                dout.flush();
                        } else if(counter == 2) {
                                dout.writeUTF(path);
                                dout.flush();
                        } else if(counter == 3) {
                                dout.writeUTF(processingPath);
                                dout.flush();
                        } else if(counter == 4) {
                                dout.writeUTF(ID);
                                dout.flush();

                                dout.writeInt(totalFile);
                                dout.flush();
        for(int i = 0; i < totalFile; i++) {
                dout.write(Files.readAllBytes(fileStack[i].toPath()));
                dout.flush();

          if(fileStack[i].getName().contains(TEMP_FILE)) {
              fileStack[i].delete();
              fileStack[i] = null;
                        }
                        }
                        }
                        counter++;
                        }
                socket.close();
            } catch (UnknownHostException e) {
                System.out.println("No neighbor found.");
            } catch (IOException e) {
                System.out.println("Response problem.");
            }
      }
```

Instead of selecting a ***createRequest(..)*** as a join point to be intercepted by Aspect pointcuts, the proposal pointcuts intercept the processing methods which are related to TCP/IP protocols in java language. The main reason to do this, is to eliminate the need to do matching between pointcut parameters and the method signature, furthermore dealing with the sending method data as a whole package, which sometimes has difficulty manipulating the code and thus, affects the code's performance, and moreover to increase the generality of applying AOP by focusing on the functionality side of the method rather than the method name. By espousing this implementation, the pointcut intercepts only (*.writeUTF ()*) method, so we achieve a high level of controlling the sending method, as well as not needing to intercept any unnecessary methods or data. The pointcut is designed based on the following pointcut expression

call (*Method-Type* writeUTF(*Type*))

And it is represented in code perspective as:

method-call(void java.io.DataOutputStream.writeUTF(java.lang.String))

The following pointcut and aspect advice will be adopted by the proposal to intercept all writeUTF methods

AspectJ code: pointcut
pointcut ReqIntercept(String message):**call**(**public final void** writeUTF(String))&&**args**(message);

Figure 5.46. shows the cross cutting of the pointcuts over (*.writeUTF ()*) method

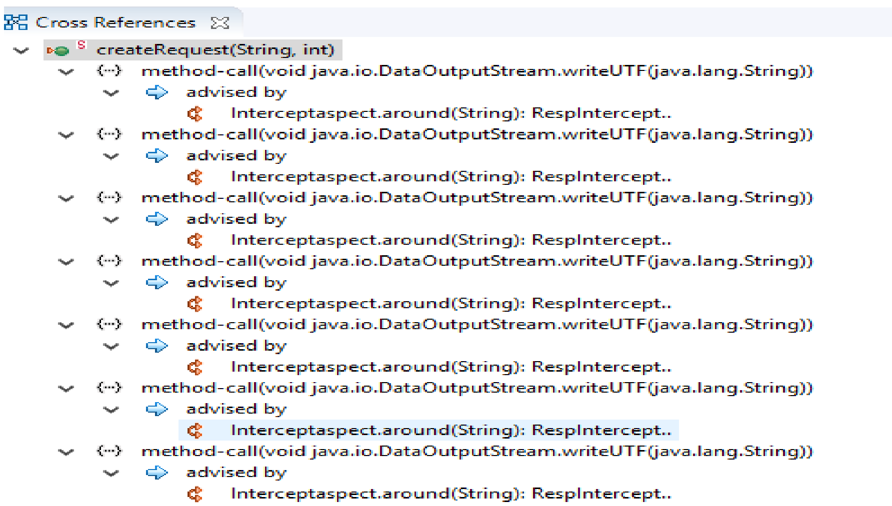


Figure 5.46. Cross Reference for a New Request

As explained on how to use pointcut to intercept the sending method, we deal with the receiving method in the same way. The pointcuts intercept (.readUTF()) method instead of method name. The following code shows what the code will receive from the source node:

Java Code: Receive method

```
public void receiveResponse(DataInputStream din) throws IOException
 {
        String serverResponse = "OK";
        while (!serverResponse.equals(“finish”)) {  // Stop Condition
                serverResponse = din.readUTF();
                  if(serverResponse.equals(“sender”)) {
                        sender = din.readUTF();
                        } else if(serverResponse.equals(“attributeList”)) {
                                attributeList = din.readUTF();
                        } else if(serverResponse.equals(“path”)) {
                                path = din.readUTF();
                        } else if(serverResponse.equals(“processingPath”)) {
                                processingPath = din.readUTF();
                        } else if(serverResponse.equals(“file”)) {
                                requester = din.readUTF();
                                totalFile = din.readInt();

                  for(int i = 0; i < totalFile; i++) {
                        byte [] byt = new byte[MAX_FILE_SIZE_MB * 1024];
                        din.read(byt);

        fileLocation = DATA_FILE_LOCATION + DATA_FILE_OUTPUT + TEMP_FILE + i;
                                Files.write(new File(fileLocation).toPath(), byt);
                                        fileStack[i] = new File(fileLocation);
                                }
                        }
                }
        }
```

The pointcut designed based on the following pointcut expression

```
call(String readUTF());
```

AspectJ code: pointcut

```
pointcut RespIntercept():call(String readUTF());
```

Figure5.47 shows the cross-cutting of the pointcuts over (`.readUTF()`) method

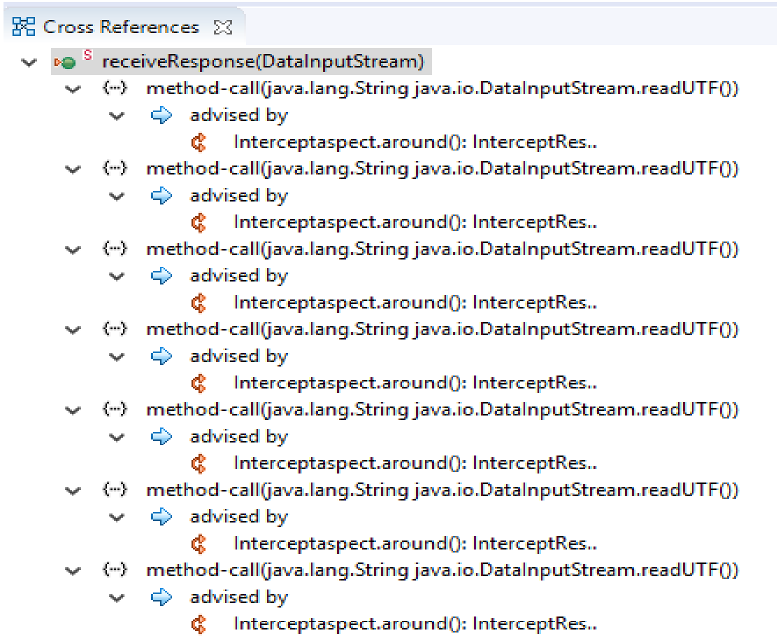


Figure 5.47. Cross Reference for Receiving a Response

5.4 3AC-AOP Solution

The proposed model deals with the security and privacy concerns as three phases:

1- Access control and data flow.
2- Cryptography.
3- Data sanitization (AOP security guard).

5.4.1 3AC_AOP Access Control Solution

This section details the development of the access control model policies in OOP and AOP. Instead of using XACML to write the access control policy which sometimes leads to complicating the coding, and may increase the tangling and scattering problem by needing to add some extra lines of codes, which in turn has a negative effect on the performance of the code, we embedded the policies within the main code in OOP and within the aspect pointcuts within AOP. As demonstrated by the previous section, the access control policy would be applied between sending and responding methods.

Within the same node, there are no access restrictions if the request is directed to the same node to do its own function as shown in Figure5.48.

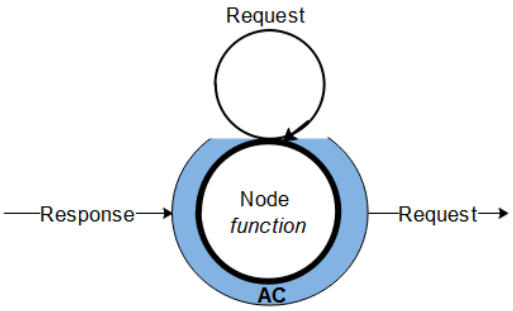


Figure 5.48. Access Control dealing with the node

However, when $Node_i$ sends or receives a request to/from any other node(s), then the access control policies will be activated. The next subsections discuss the design and enforcement of the access control model policy in DDS for both cases, OOP and AOP.

- **Access control model**

As explored in Chapter 4, the proposed access control model is combined between three access control models to produce a powerful access control model called (3AC_AOP). This combination comprises ABAC, IBAC and MLS. In the following, we deal with each model separately from a coding perspective, afterwards these models will be aggregated together in one model.

1. Attribute-Based Access Control (ABAC)

ABAC may represent the characteristics of the node, distributed system or even the environment around. This will help to add extra space to express each node as job nature, role within enterprise, node's function and so on, in addition to the environment of the distributed system which it works with, for example the period time and date of working. All these factors are aggregated to create the model.

The proposed solution focuses on the function side of the node as a restricted condition, to allow or deny accessing to the transmitted data. The system administrator will be responsible for distributing the attributes among the distributed nodes as a JSON file to represent each node separately, as shown below:

```
[
   {
       “Name”: “Node1”
        “Node_Fucntion”: “NodeFucntion”
   },
  {
       “Name”: “Node2”
        “Node_Fucntion”: “NodeFucntion”
   },
  {
       “Name”: “Node3”
        “Node_Fucntion”: “NodeFucntion”
   }
.
.
  ]
```

Algorithm 1 shows how the node will check the incoming file(s) associated with attributes with the node attributes. If the conditions are satisfied, then the file(s) will be sent to the node function to perform the mission. Afterwards, the executed attribute(s) will be deleted from the attribute list as a step of stop condition.

Algorithm1: ABAC

Input: file, attributeList

Output: processed file or denied

1. Start
2. $Node_i$ ***Send*** (file+ attributeList)
3. int k=0;
4. While (k<DDS.length) && (attributeList !=θ) *// DDS.length is node numbers*
5. If ($Node_k$.attribute ⊆ attributeList)
6. $Node_k$.*function*
7. attributeList= attributeList.Remove($Node_k$.attribute)
8. EndIf
9. k++
10. EndWhile
11. End

Hence, accessing the data file by the node will be dependent on the node's function. In the case of broadcasting without processing the file, the stop condition will convert to be (While (k<DDS.length)) only and we delete line 7. This will expand the broadcast concept through the ability to share the data between some nodes sharing with the same attributes.

ABAC
- *(k<DDS.length)* ➔ *share data between match attribute nodes (no processing)*
- *(k<DDS.length) && (attributeList !=θ)* ➔ *process data and stop when no more attrs*

AspectJ code: Attribute Access Control Coding (ABAC) (processing side)

```
String nodeAttribute;
String attribute;


pointcut InterceptResq():call(String readUTF());
String around() throws IOException:InterceptResq(){
attribute= proceed();
if (attributeList.contains(nodeAttribute))
  {
   nodeFunction();
   attributeList.replace(nodeAttribute)
   }

  return (proceed());
}
```

To conclude, the ABAC model, the source nodes set the required attributes and then sending file to be processed by different nodes in DDS according to the attributes. Figure5.49. shows the process of this model and how to distinguish between the routing path and the processing path.

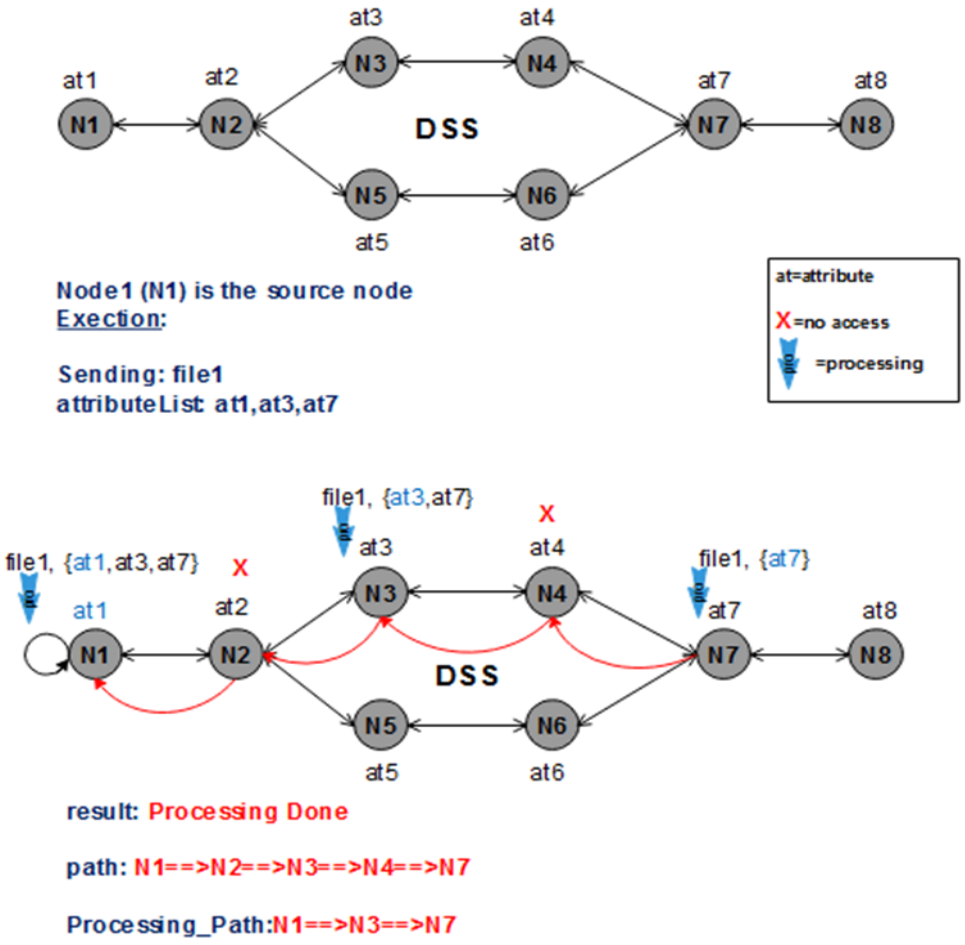


Figure 5.49. ABAC in 3AC_AOP working

2. Identity-Based Access Control (IBAC)

Complementary to ABAC, IBAC adds more restrictions to the access processing. This model forwards the data files to the intended node or nodes in case the source node sends more than one identity. Using this model is optional and would only be compulsory if the source node needed to apply a domination relationship according to the node's clearance. Similarly, with ABAC, the system administrator will be responsible for distributing the roles of this model among the distributed nodes in the form of a JSON file as seen below:

```
[
   {
      "Name": "Node1"
       "Node_ID": "NodeID"
```

```
  },
  {
      "Name": "Node2"
       "Node_ID": "NodeID"
  },
  {
      "Name": "Node3"
       "Node_ID": "NodeID"
  }
.
.
 ]
```

The proposed model adopted ABAC as the higher layer and MLS as the lower layer and both must be applied to satisfy our objective to save the data. Based on this idea, IBAC can be represented by the intermediate layer used to support the security when the source node would enforce access control policy according to the domination relationship. For this reason we cannot separate MLS from ABAC, thus the Algorithm2 will have the same orientation, adding only the IBAC restricting condition as illustrated below:

Algorithm1: ABAC+IBAC

Input: file, attributeList, IdList

Output: processed file or denied

1- Start
2- $Node_i$ ***Send*** (file+ attributeList+ IdList)
3- int k=0;
4- While (k<DDS.length) && (attributeList !=θ) *// DDS.length is node numbers*
5- If ($Node_k$.attribute ⊆ attributeList)&& ($Node_k$.Ids ⊆ IdList)
6- $Node_k$.*function*
7- attributeList= attributeList.Remove($Node_k$.attribute)
8- EndIf
9- k++
10- EndWhile
11- End

The processing will therefore include only the intended nodes in which their Id's match the receipt Id's. Figure5.50 shows the implementation of this model combining with ABAC.

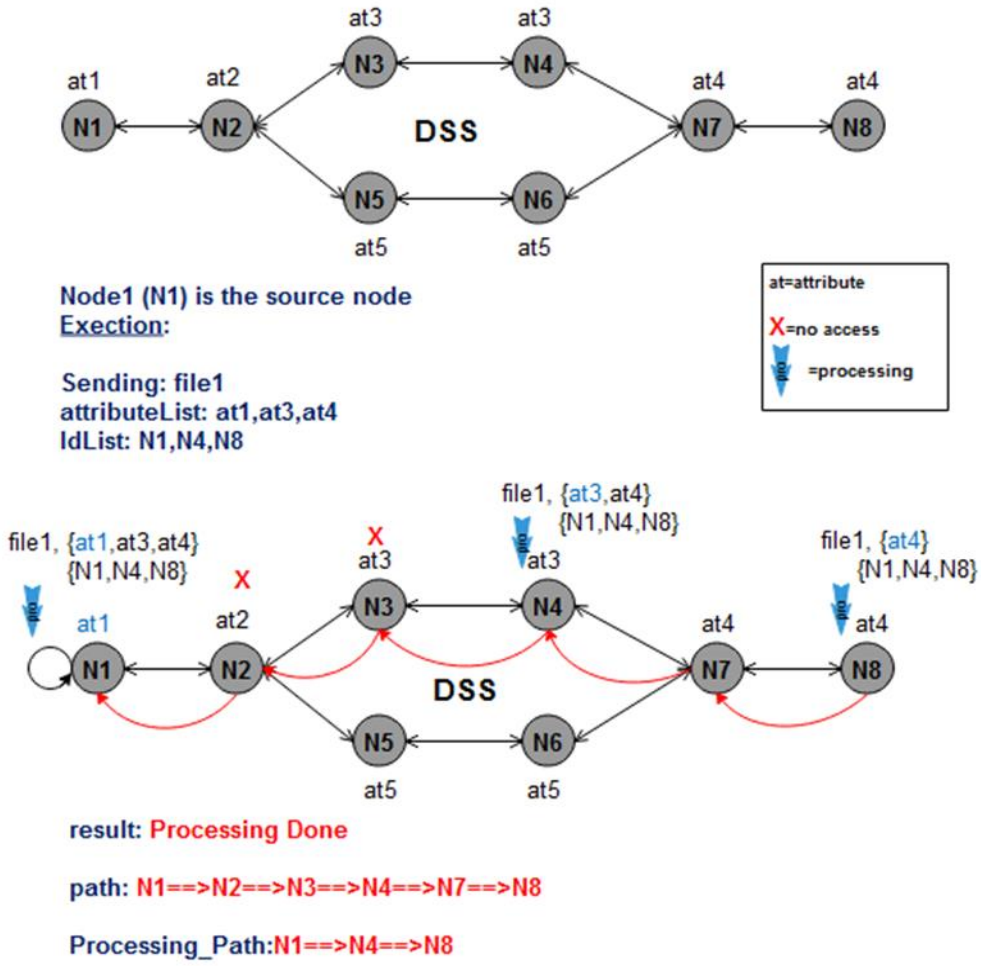


Figure 5.50. ABAC and IBAC in 3AC_AOP model

The implementation has shown, that using IBAC with ABAC will increase the reliability of the system by improving the efficiency through forwarding the file only to the intended nodes.

ABAC+IBAC

- *IdList = θ ➔ share data between match attribute nodes (no processing)*
- *IdList ≠ θ ➔ forward data only to the intended nodes.*

AspectJ code: ABAC+IBAC

```
String nodeAttribute;
String attribute;


pointcut InterceptResq():call(String readUTF());
String around() throws IOException:InterceptResq(){
attribute= proceed();
if (attributeList.contains(nodeAttribute) && IdList.contains(nodeIDs)
  {
   nodeFunction();
   attributeList.replace(nodeAttribute
   }

  return (proceed());
}
```

A question may arise here about why we need to use the ABAC policy, if IBAC can forward data only to an intended node. The answer is that ABAC provides a more comprehensive meaning by achieving a means of sharing, without the need to determine which node will do this.

3. Multiple Level Security

This model represents the infrastructure of the proposed model. The access granted depends on the security clearance of the node in addition to the security classification of the file, to prevent data spill through the access processes. In order to obtain high efficacy of this model, the node should have a security clearance, as well as the transmitted file being tagged by a security classification tag. For this reason, we have designed a DDS to give the ability to the individual node to tag its own data according to the clearance level of the node. This will give more flexibility and cooperation between distributed system works. As demonstrated in the previous models the system administrator will be responsible for distributing the security clearance of each node separately. The difference here however is that the clearance level is set up within the node interface, such that each node has a constant parameter representing the clearance level of the node. Security clearances of the DDS nodes and security classifications of

the data transmitted between distributed nodes, are represented as integer numbers between 1-4 , such that 4 is the top secret and 0 is unclassified level, 3 is secret level and 2 is classified level. This will cover both security clearance and classification. The method which is used to classify the data file will be discussed in the cryptography section.

5.4.1.1 Access Control Combination

As we mentioned before, IBAC is an option and can be used just in some special cases. This section shows the combination between ABAC and MLS, and ABAC, IBAC and MLS, and when we need to use it.

1. ABAC and MLS

This combination allows the source node (sender) to send a request to other nodes after enforcing restrictions on the file. These restrictions are different from one node to another, according to the security clearance of the node. The scenario of this combination mode will take the following forms:

1- Normal sending, if the source node sends a file to other nodes normally, then each node in the distributed system has access only if they meet the attributes, and security clearance is greater than or equal to the source node clearance level.

$Node_{(i)attr+SL} \rightarrow Node_{(j)attr+SL}$ *authorized iff attr[j]=attr[i] & SL[j]>=SL[i].*

2- Each node has the ability to divide the file into portions according to its own security clearance and restrict access to the sending file, in which each processing node has an allowance to access to part of the file not the whole, if they meet the attributes of this part and security clearance. The security clearance has one number between (1-4) when 1 is the lower level and 4 is the top secure node as shown in Figure 5.51

3- If the node has a high level of security clearance, then it will be able to divide the file between 1-4 and using the domination relationship to send the file to other nodes. In this case we need to use IBAC.

Figure 5.51 shows the ability of each node to divide the file according to the security clearance of the node.

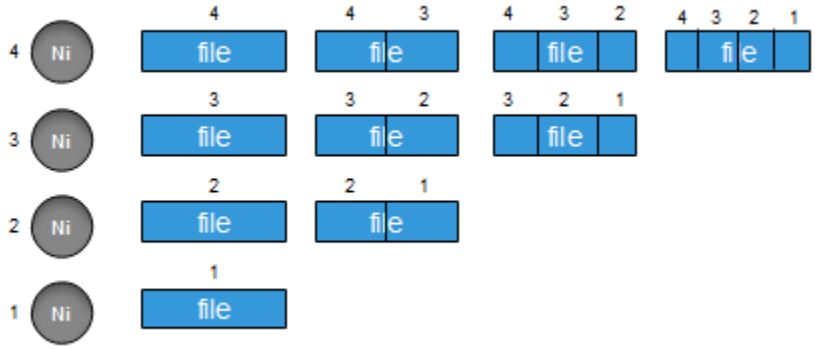


Figure 5.51. Authority to Classify the Files and Messages According to Node Security Clearance

2. ABAC, IBAC, MLS

This combination is the consequence of the need of the node with high clearance to allow lower levels to process on its transmitted file, or when the source node wishes to do the process in specific nodes. This makes the model more flexible and cooperative by achieving a high level of security and reliability. The model supports the fact that not all the information in high clearance domains is classified as highly sensitive.

The JSON file will take the form

```
[
  {
     “Name”: “Node1”
      “Node_Fucntion”: “NodeFucntion”
      “Node_ID”: “NodeID”
  },
  {
     “Name”: “Node2”
      “Node_Fucntion”: “NodeFucntion”
      “Node_ID”: “NodeID”
  },
  {
     “Name”: “Node3”
      “Node_Fucntion”: “NodeFucntion”
      “Node_ID”: “NodeID”
  }
```

.
.
.
]

We have seen that, invoking a method called AccessControl() by the pointcuts is better than if we do all the processes inside around() advice, in order to keep a high level of separation to reduce the tangling problem.

AspectJ Code: ABAC,IBAC and MLS

```
pointcut InterceptRes():call(String readUTF());
String around() throws IOException:InterceptRes(){
   return Accessing(proceed());}

public static String Accessing(String string) throws IOException {
       if (string.contains("ID")){     // To check of source node used ID
       if (string.contains("Node_ID")) {  // To check if ID is available
              Id_verify ="okay";          // public parameter
                     }
                     }

       if(string.contains("nodeAttr")&&( Id_verify =="okay")) {
         if (Sdr_SL<=Node.SL)
              {
              fileHandeling();
              string=string.replace("nodeAttr", "");
          }
              return string;
       }
```

5.4.2 3AC_AOP Cryptography

In order to ensure the integrity of data transmission among distributed system nodes, and to prevent any type of eavesdropping attacks, the proposal system has utilized AOP to encrypt and decrypt the data transmission only when it is required. We used the AES cryptography algorithm with key 128 bits. However, adding the cryptography basement to the distributed system will add burdens that may affect the effectiveness of the system in the near and long term. To minimize the encumbrance of using cryptography processing, encryption and decryption in respect of AOP will be applied on both static

and dynamic. The AOP advice will encrypt the access conditions and file separately . In the context of the file, this will be tagged previously by the source node. In the receipt node, AOP decryption advice will be applied on access condition; if there is a matching, then AOP advice will decrypt the file according to the security level, otherwise it will be re-forwarded to the next connected node(s) in an encryption form. By applying this scenario, we eliminate the need to decrypt and re-encrypt the file if this is not necessary. Moreover, the decryption process may be applied only on a specific portion of the file that matched the attribute of the source node and node security clearance as well. The 3AC_AOP model will interact with the cryptography model to support the security processes and achieve a high level of confidentiality.

One of the major objectives in this thesis is the privacy preservation. This is done by restricting the access to the whole file data by authorizing each node to arrive only to the authorized portion of the data, not all the file, while the rest of the file will still be in an encrypted form. In this case, this portion will be treated as a separate file. After the process is done then the result will be appended to the original file to recreate the whole file, before sending to the next node.

Achieving security does not necessarily mean fulfilling the privacy protection requirements. For example, if the system adopts ordinary access control and this restricts unauthorized users to modify the information but not prevent viewing the file, we achieve the security by keeping the information integrity, but the privacy side will be violated. Thus, this would lead to leaking the privacy. In the rest of this chapter cryptography will be associated with the access control model.

1. File encryption

The implementation of the cryptography side has two cases:

1. Before processing: this phase is performed when any node classifies their files and tag them to be saved in its own storage before doing any processing. For example, some system scenarios need to implement the processing in some specific time periods like payroll system (monthly), annual profit report

(annually). In this case, the source node (sender) will prepare the file in a tagged form.

2. During processing: this phase happens at run time when any source node wishes to send a file to other nodes in the system, then AOP encryption advice will intercept the send method to encrypt access conditions and file separately.

- ***Cryptography cases:***

1- Normal case: the source node sends the file(s) associated with Access conditions. We assume that the processing required the whole file to do the processing. In this case, the access authorization is only granted if the receipt node's clearance is greater than or equal to the source node level. Thus, AOP will decrypt the whole file to perform the requested process. This case includes performing the process in one node, or even in more than one node if the process needs multiple stages to complete the process, as illustrated in Algorithm3.

Algorithm3: Normal case

Inputs: encrypted file, attributes list , identities list.

Output: processed file save in source node.

```
1-  Start
2-  S.N sends Encrypted AC and File                    // S.N= source node
3-  For each R.N
4-  AOP. Decrypt (AC)
5-   If (R.N.attribute= S.N.file.attribute)        // if identities list is empty
6-     If (R.N.attribute= S.N.file.attribute)&& (R.N.id= S.N.file.id)
7-       AOP. Decrypt (file).
8-        R.N.Function(file) ➔ file*
9-       AOP. Encrypt(file*)
10-      AOP. Encrypt (AC)
11-    End if
```

12- Send to next node(s)

13- End for

14- End

2- Dominate case: this case is based on the level of security clearance of the source node, see Algortihm4.

Algorithm4: Dominate case, (Cryptography and tag)

Inputs: File, S.N.level

Output: Encrypted and tagged file

1- Start
2- Enter the file
3- Within the source node
4- Review the header of the file (file attributes) // file (.csv, .sql, .JSON….)
5- Classified the attributes in sets according to security/privacy sensitive
6- AOP Encrypt and tag each set according to classification level
7- Output: encrypted and tagged file.
8- End

The key aspect discussed is that according to the domination level, the node can divide the file in the context of a column. Let's return back to the payroll system when the source node (salary database) would send a data file including identities and other attributes like (Id, Name, Address, No.Hrs, Hrs.Price, Salary, S.C, Acc.No …). The source node has been classified as Top secret clearance (4), thus, the node has the ability to divide the file into 4 tags {Top Secret, Secret, Confidential, Unclassified} or {4, 3, 2, 1}. In our example the source node will send the file to calculate the salary,

and after that make money transactions to the employees. Salary calculations will be done in a node classified as top secret, and its function is to set the price of working hours and calculate the salary. This node needs some identities to do matching between the employees in the file and the employees in its own database, to determine the price of working hours and calculate the salary for each individual employee. Afterwards, the third node will make the money transaction to the employee. Finally, the whole file will reply back to the source node as a handled file.

Salary node will divide the file into two portions, the first one is { Id, Name, Address, No.Hrs, Hrs.Price, Salary}, the part tagged as top secret (4), the first four attributes represents the identifiers of the employees, the last two attributes represent the function of the Node2.

The second portion is { S.C, Acc.No} which are sort code and account number respectively, which will be tagged as a secret (3) in which the receipt node will use this information to make a money transaction. The function of the current node will not need to use the identifier attributes.

According to the above algorithm, these two portions will be encrypted by different keys and tags.

The processing case of the cryptography section will be discussed in the next section.

2. Message encryption

1. Before processing: this phase is performed when any node classifies and tags their messages and save them in its own storage before doing any processing. In this case, the source node (sender) will prepare the message in a tagged form.

2. During processing: this phase happens at run time when any source node wishes to send a message to other nodes in the system, then the sending method will be intercepted by AOP advice to encrypt the message(s) and access conditions separately.

- ***Cryptography cases:***

1- Normal case: the source node sends the message(s) associated with access conditions. We assume that the processing required the whole message to do the processing. In this case, the access authorization is only granted if the receipt node's clearance is greater than or equal to the source node level. Thus, AOP will decrypt the whole message to perform the requested process. This case includes performing the process in one node, or even in more than one node if the process needs multiple stages to complete the process, as illustrated in Algorithm3.

Algorithm3: Normal case

Inputs: encrypted Message, attributes list , identities list.

Output: processed file save in source node.

```
15- Start
16- S.N sends Encrypted AC and Message              // S.N= source node
17- For each R.N
18- AOP. Decrypt (AC)
19-  If (R.N.attribute= S.N.Msg.attribute)     // if identities list is empty
20-    If (R.N.attribute= S.N. Msg.attribute)&& (R.N.id= S.N.file.id)
21-      AOP. Decrypt (Msg).
22-         R.N. Receive (Msg)
23-      AOP. Encrypt(Msg)
24-      AOP. Encrypt (AC)
25-    End if
26-  Send to next node(s)
27-  End for
28- End
```

2- Dominate case: this case is based on the level of security clearance of the source node, see Algortihm4.

Algorithm4: Dominate case, (Cryptography and tag)

Inputs: encrypted Message, S.N.level

Output: Encrypted and tagged Message

9- Start
10- Enter the Msg
11- Within the source node
12- Classified the attributes in sets according to security/privacy sensitive
13- AOP Encrypt and tag each set according to classification level
14- Output: encrypted and tagged Message.
15- End

In Message level, the scenario will be slightly different from that in the file level. With the files, we have two scenarios: 1) when the source node sends a request (file) and waits for a response (Processed file), 2) when the source node sends a request (file) to destination nodes without waiting for the response. This might include intermediate processing or just sending and receiving between the source and destination nodes. With the message we will adopt the second scenario as most times the messages do not need to be processed by other nodes, and the other nodes can send the reply to the source node by different messages. This processing is called one-dimension process. In fact, in the evaluation chapter we included one-dimension processing to deal with images, to evaluate the efficiency of the model and to show the ability to deal with different types of computer files.

5.4.3 3AC_AOP Access Control with Cryptography

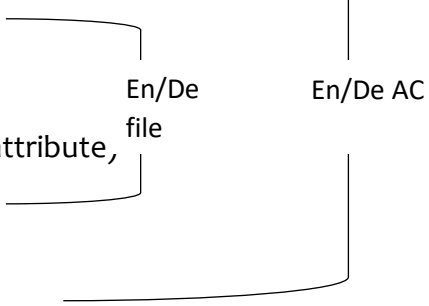
This section discusses the combination between the Access control model and cryptography model. The outcome is a powerful, fixable, high integrity and privacy protection model. Algorithm5 shows the processing in detail:

Algorithm 5: Dominate case, (processing)

Inputs: encrypted, tagged file/ attributes list, identities list

Output: processed file

1- Start
2- Enter the file
3- AOP. Encrypt (AC).
4- Send the encrypted file associated with AC
5- While(DDS.has more nodes) && attributeList!= null)
6- For each Node in DDS
7- AOP.Decrypt(AC)
8- If (attributeList) = ($Node_i$.attribute)➔// & (identityList) = ($Node_i$.id)
8- AOP. Decrypt (file)
9- $Node_i$.function(file)
10- file.attribute.remove(attribute,
11- AOP. Encrypt (file)
12- End If
13- AOP .Encrypt(AC)
14- Send the file to next $Node_i$
15- Output: reply the accomplished file to source node.
16- End



The combination starts when AOP encrypts the access conditions before sending with the associated encryptedfile(s) see step 3. In receipt node, AOP advice will decrypt these lists (see step 6) to activate the access control stage to check the willingness of the current node to deal with the file (see step 7). If the access control conditions are satisfied, then AOP decryption advice deals with the file according to the model policy

(see step 8). After, finishing the function, the file and the attribute lists will be re-encrypted sequentially (see steps 11 and 13).

Figure5.52 shows the infrastructure of the security and privacy model interacting between two nodes.

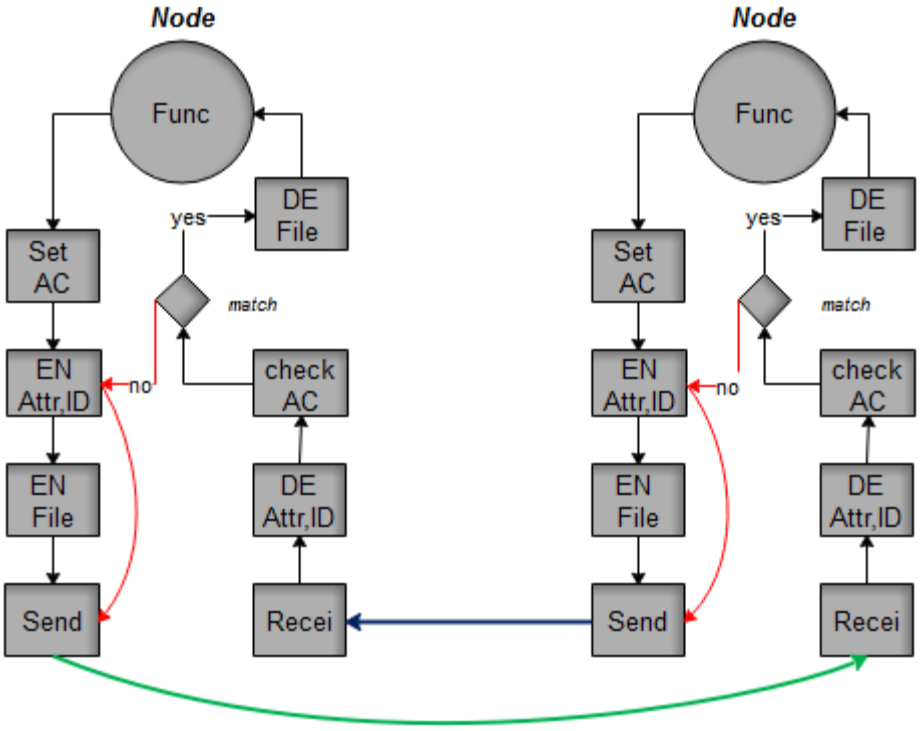


Figure 5.52. Infrastructure of Security and Privacy Concerns in the Model

5.4.4 3AC_AOP Security Guard

One of the major challenges that faces the systems which are based on MLS is to prevent data spilling during transmission between different security domains. Many solutions have been suggested to address this problem, for example clustering of distributed nodes into main clusters according to the node security domain. Data transitions between the nodes within the same cluster will work in less secure processes, whilst data transmission between different clusters will be under MLS policies (no-read up, no-write down), and may need to insert devices to run these conditions.

The problem that might arise when a node has been classified with a security clearance, is if you wish to send a file to perform a certain mission on a node that has been classified as lower, this increases the likelihood of data leakage from a high level to a lower level. One of the most widely accepted solutions is to inject a security guard (automatic, human or hybrid) between different clusters as we mentioned in chapter 4.

In the proposed system we suggested using dynamic AOP as a security guard to filter the file before sending to lower nodes. Instead of collecting the nodes in clusters and forcing the system to change the original connections between nodes to ensure high security, in the proposed solution we adopted the data sanitization method to filter the sensitive data, and get high assurance and integrity of data transmission between distributed nodes.

The data sanitization model is totally based on the domination relationship, such that only the nodes that are classified with high security clearance can authorize the nodes classified as lower clearance to perform a limited process. The consequences of applying this model are to eliminate the need for trusted computing based (TCB), and the original connection between system nodes will be same, rather than adding a heavy process over the system. This model will be invoked by aspect advice after checking the attributes and identities of the processing nodes, as explained in Algorithm6.

Algorithm6: Data Sanitization, message level

Input: message(s), attributeList, indentitiList.

Output: sanitized message(s).

1- Start.
2- Source node sends the message(s), attributeList, indentitiList.
3- For each $Node_i$ in DDS
4- If ($Node_i \subseteq$ attributeList) && ($Node_i \subseteq$ indentitiList) then // intended node
5- If ($Node_i$.SL >= message.CL) // SL=security level, CL=classification level
6- DE(message)
7- Else {

8- DE(message)

9- Sant(message) }

10- End if

11- Send the clear message to the node.

12- End if

13- End

The sanitization process at message level will take a horizontal path by sterilizing the message according to some sensitive words which have been saved in the AOP repository. In this case the source node will be responsible for how to tag the texts inside the message, and the AOP monitor will work from the other side to ensure none of sensitive words will be released. This case has a benefit, in case the source node might send a broadcast to the other nodes, and each node will see just the authorized words of the message, according to their level of security clearance. Figure5.53 shows the proceedings of the process.

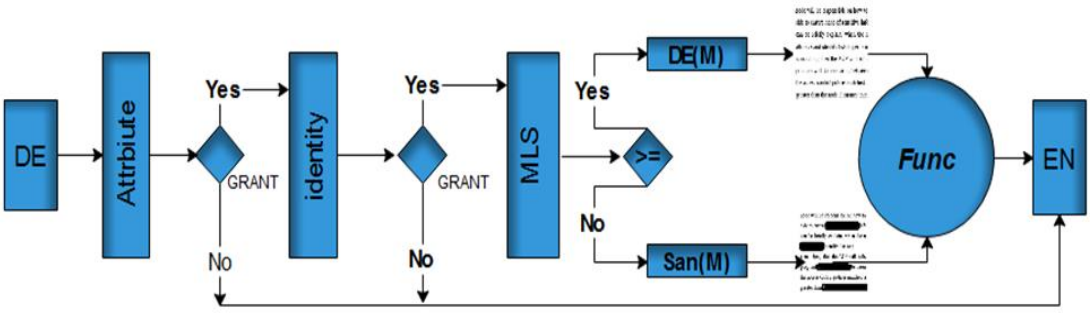


Figure 5.53. AOP Guard in Term of Messages

However, dealing with the file is different, see Algorithm7.

Algorithm7: Data Sanitization, file level

Input: file(s), attributeList, indentitiList.

Output: sanitized file(s).

1- Start.
2- Source node sends the file(s), attributeList, indentitiList.
3- For each $Node_i$ in DDS
4- If ($Node_i \subseteq$ attributeList) && ($Node_i \subseteq$ indentitiList) then // intended node
5- If ($Node_i$.SL >= file.CL) // SL=security level, CL=classification level
6- DE(file) // according to compartment or attributes
7- Else {
8- DE(file)
9- Sant(file) } // send only the matching portion of the file
10- End if
11- Send the clear file to the node.
12- Perform $Node_i$ function
13- Appended the handled portion to the original file
14- Send the result after encrypted to the next node
15- End if
16- Next for
17- End

The sanitization process at file level will take a vertical path by sterilizing the file in the context of columns, and according to security tags which associate with each column. In this case, the source node will be responsible for how to tag the file, and the AOP monitor will work from the other side to ensure none of sensitive information will be released. The processing of the second case can be briefly explained as when the node receives the file, AOP advice will do decryption to the attribute and identity lists to perform access control policy. If the attribute is sufficient and identity is matching,

then the AOP will realize that the receipt node is the intended node. Afterwards, the aspect program will do a comparison between the security classifications of the file and the node's clearance. If the access control policies are matched and the file is already divided into different tags, which may be greater than the node clearance, then the sanitization method will be invoked.

The function of the sanitization method is to generate a new sub file created from the original file, and hold only the columns that need to be handled in the receipt node. The security tags of the column(s) are equal to the node's security clearance. In this case the node will apply the function only on the authorized portion. After finishing the process, this file will be replaced with the portion in the original file to regenerate the whole file with the updated values, then re-forward it to the next node as shown in Figure 5.54.

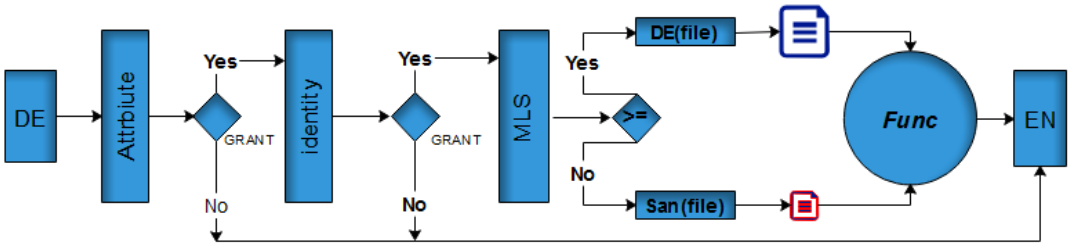


Figure 5.54. AOP Guard in Terms of Files.

This case has a benefit, in case the source node might send a broadcast to the other nodes and each node will see just the authorized word of the message, according to their level of security clearance.

5.5 Summary

This chapter has described the implementation side of the 3AC_AOP. It has detailed three processing cases: 1) access control, 2) cryptography and 3) the automatic AOP security guard. All these cases have been designed and implemented as AOP pointcuts and advices. This chapter has also shown the power of the aspect-oriented programming language and how we can utilize this tool to not only separate the non-functional process from the code, but even to control the functional side, in order to

increase the effectiveness and performance of the code. It has provided algorithms for each model to generalize the model to be applied by different programming languages, as well as to show the easiest way to apply or update the security and privacy concepts to an existing system with only minor changes .

Chapter 6
Evaluation of 3AC_AOP and Comparison with Existed Solution

This chapter provides the numerical evaluation aspect of the proposed solution. Both OOP and AOP are used to design and implement the proposed solution. This chapter shows the methodology that has been used to minimize the heaviness impact when injecting AOP into the code by optimal utilization of the main code. The results which has been evaluated in this chapter will be divided into three portions: 1) access control model 2) cryptography 3) AOP guard. The implementation sections of this chapter will deal with two types of decentralised distributed system networks, the ring network and unstructured network (internet setup) and both used the flooding algorithm to distribute and processing data.

Finally, this chapter will be concluded by comparing between 3AC_AOP model and existing solution.

6.1 3AC_AOP Time Performance

To calculate the implementation time and get accurate results we need to identify some terms that need to clarify execution time calculation as follows:

- Let γ is a set of nodes that include all nodes in the distributed system.
- Let λ is a set of nodes that include all nodes required to accomplish the task (bridges and processing nodes) such that: $\lambda \subseteq \gamma$.
- *Bridges node (hop):* is the node that is used to transfer information to the next node in the network after being denied by access control policies.

- *Processing node*: is the node that been granted by the access control model and can perform its own function on the receipted data. Let ρ is a set of the processing nodes such that: $\rho \subseteq \lambda$, and can be accounted as following:

  $$\sum_{j=0}^{\rho-1} processing$$

- *Transmitted Time (ω)*: is the time that is needed to transfer data from node A to node B and can be calculated as following:

  $$\sum_{i=0}^{\lambda-1} \omega$$

- *Access control Time* (ACT): is the time that is needed by the access control model to check the access conditions for each single node and this includes both bridges and processing nodes and can be calculated as following:

  $$\sum_{i=0}^{\lambda-1} AC$$

6.2 3AC_AOP Access Control

As been demonstrated in chapters 4 and 5, the proposed access control model consists of three access control models; Attribute-Based Access Control (ABAC), Identity-Based Access Control (IBAC) and Multi-Level Security (MLS) Access Control. The evaluations deal with these models as ABAC + MLS and ABAC+IBAC+MLS in both OOP and AOP paradigms. The proposed distributed system will be 10 nodes connected by a ring network in which each node has connection with two neighbours nodes before and after as shown in Figure 6.55.

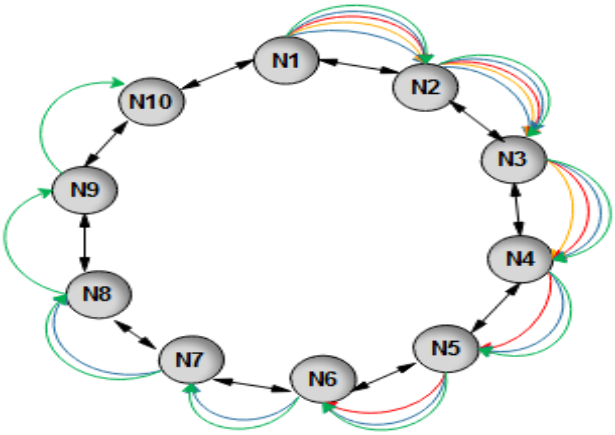


Figure 6.55 Decentralized nodes, ring distributed system

Each individual node has a unique identity (N_i) and different attributes and security level which might change according to the evaluation requirements as explained below

Node set ={ N1, N2, N3, N4, N5, N6, N7, N8, N9, N10}

Attribute set = {A_{t1}, A_{t2}, A_{t3}, A_{t4}, A_{t5}, A_{t6}, A_{t7}, A_{t8}, A_{t9}, A_{t10}}

Security.L = { 2, 4, 3, 4, 3, 1, 3, 4, 3, 2 }

6.2.1 ABAC & MLS

In this section, the access control model will be evaluated by the composed Attributed-Based and Multi-level Security Access controls models. The consequences model will be applied on the distributed system by using two case files, messages and images. In the case of files, the transmitted information has two directions which are sending requests and receiving responses. The source node sends the information to the intended nodes to perform their functions and replies with the final result to the source nodes. In another case, with messages and images, the source node will send these files

to the intended nodes without need to waiting for a response. For this reason, the transmitted information will have only one direction to complete the requested task.

1. Two Directional Experiments

This evaluation will deal with files. In this case, the source node will send the request to the intended nodes associated with the file and all required access control conditions. The file will travel between system nodes and if access conditions match on any node then the node will implement its own function according to the access control policy (i.e. match the attributes and security level of node is greater than or equal to information classification level). As mentioned before, with file level the source node will await the response of the requested task and that means the transmission will take two directions and the performance time will be as following:

$$Exexution\ time = 2\sum_{i=0}^{\lambda-1}\omega + \sum_{j=0}^{\rho-1} process + \sum_{i=0}^{\lambda-1} AC \ldots\ldots \exp(1)$$

In order to ensure accurate results, we have simulated five cases as a way to exclusive most of cases which might be used and monitoring the system in case of any unintended incidents during real time. Table 3 shows the cases and the consequences result as well as how many hops needed to complete the task for each case.

Table 3: ABAC and MLS processing cases

Case	*Request*	*Result*
A	$fun = \{At1, At3\}$ $operations = 2$	*R={N1,N2,N3}* *P={N1,N2}* ***hops=1***
B	$Fun = \{At1, At4\}$ $operations = 2$	*R={N1,N2,N3,N4}* *P={N1,N4}* ***hops=2***
C	$Fun = \{At2, At4, At6\}$ $operations = 3$	*R={N1,N2,N3,N4,N5,N6}* *P={N2,N4,N6}* ***hops=2***
D	$Fun = \{At1, At3, At5, At8\}$ $operations = 4$	*R={N1,N2,N3,N4,N5,N6,N7,N8}* *P={N1,N3,N5,N8}* ***hops=4***
E	*Fun={At2,At4, At6, At8, At10}* $operations = 5$	*R={N1,N2,N3,N4,N5,N6,N7,N8,N9,N10}* *P={N2,N4,N6,N8,N10}* ***hops=4***

According to the table , each case will be tested with different numbers of transmitted files (50, 100, 200, 400, and 800). These doubled files number ranges will help to ensure fairness and accuracy evaluation in both OOP and AOP paradigms as shown in Table 4.

Table 4. ABAC and MLS File level, OOP and AOP runtime performance.

No.Files	A		B		C		D		E	
	OOP	AOP	OOP	AOP	OOP	AOP	OOP	AOP	OOP	AOP
50	**1070**	**1114**	**1946**	**2127**	**3403**	**3580**	**3998**	**4112**	**7112**	**7654**
100	**2138**	**2188**	**3780**	**4238**	**6661**	**7245**	**8024**	**8214**	**13817**	**15326**
200	**4281**	**4384**	**7554**	**8477**	**13089**	**14673**	**16247**	**17498**	**25984**	**30398**
400	**8778**	**8841**	**14868**	**16902**	**25748**	**29227**	**32755**	**33252**	**56124**	**60862**
800	**17598**	**17606**	**29413**	**33796**	**53833**	**58961**	**65645**	**66799**	**114564**	**121941**

The following figures represent the numerical evaluations for each case.

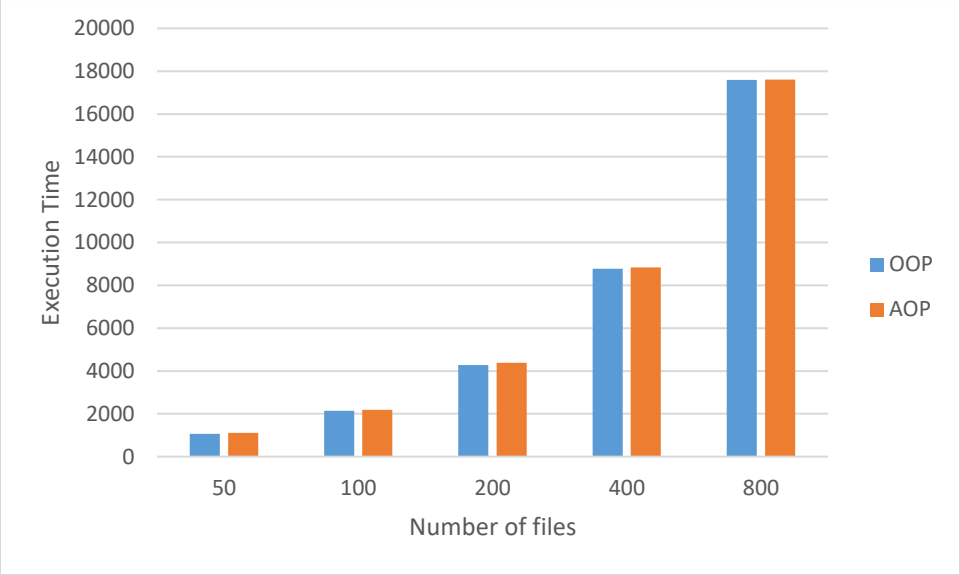


Figure 6.56. ABAC and MLS File level with OOP and AOP runtime performance CASE "A".

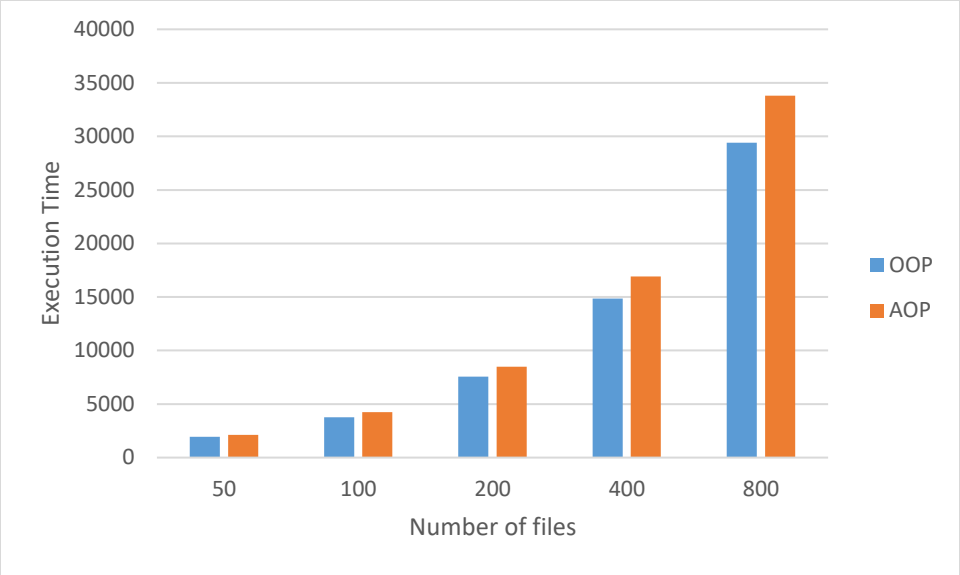


Figure 6.57. ABAC and MLS File level with OOP and AOP runtime performance CASE "B".

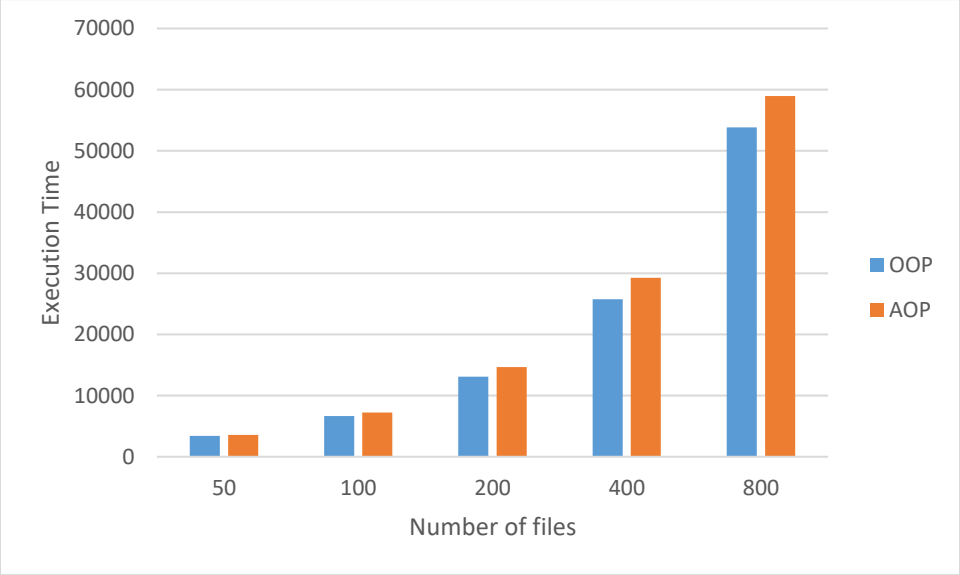


Figure 6.58. ABAC and MLS File level with OOP and AOP runtime performance CASE "C".

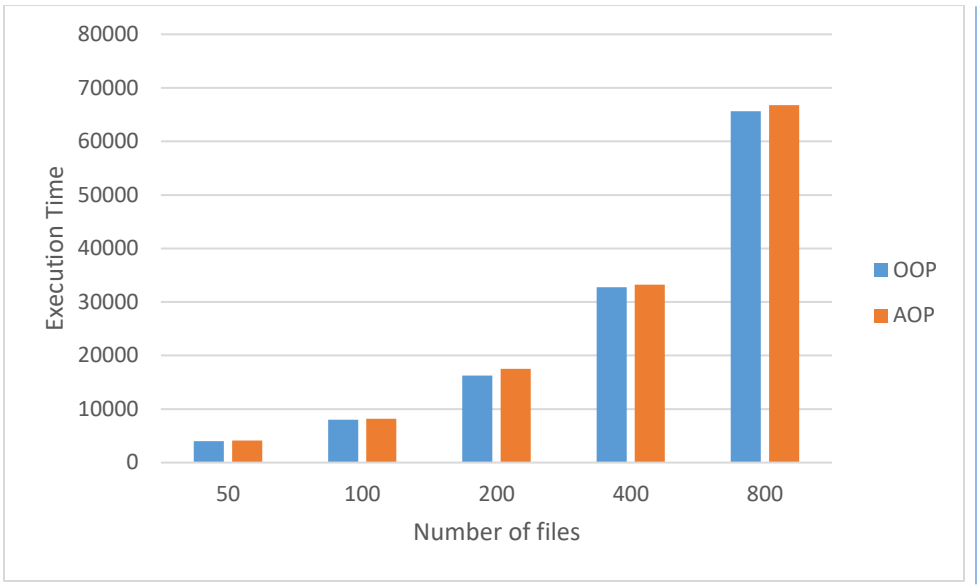


Figure 6.59. ABAC and MLS File level with OOP and AOP runtime performance CASE "D".

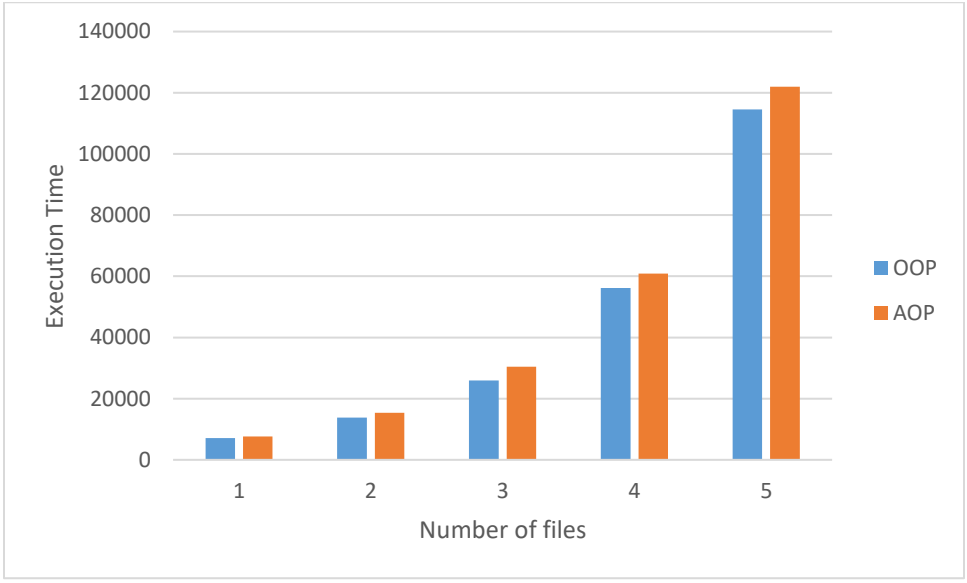


Figure 6.60. ABAC and MLS File level with OOP and AOP runtime performance CASE "E".

2. One Directional Experiments

A. Message Experiments

This evaluation will deal with messages. In this case, the source node will send the message to the intended nodes associated with the all required access control information. The message will travel between system nodes and if access conditions are granted on any node then the node will receive the message. As mentioned before, with message level the source node will not await the response of the requested task and that means the transmission will take only one direction and the performance time will be the time that is needed to finish the task in the last intended node as follows:

$$Exexution\ time = \sum_{i=0}^{\lambda-1} \omega + \sum_{i=0}^{\lambda-1} AC \ldots\ldots \exp(2)$$

Table 5 presents all cases of transmitted messages and results and the number of the required hops.

Table 5. ABAC and MLS Message level

Case	*Request*	*Result*
A	*one attribute* *SecLevel= {4, 2, 4, 2,2 , 3, 1,2, 1, 3 }*	*R={ N1,N2,N3}* *P={ N3}* ***hops=1***
B	*two attrbiute* *SecLevel ={ 4, 4, 3, 2, 4, 4, 2, 1, 1, 4}*	*R={N1,N2,N3,N4,N5,N6,N7,N8,N9,N10}* *P={ N2,N5,N6,N10}* ***hops=2***
C	*three attrbiute* *SecLevel ={ 4, 4, 3, 2, 4, 4, 2, 1, 1, 4}*	*R={N1,N2,N3,N4,N5,N6,N7,N8,N9,N10}* *P={ N2,N5,N6,N10}* ***hops=5***
D	*four attributes* *SecLevel ={4, 3, 4, 2, 3, 2, 4, 4,4,3}*	*R={N1,N2,N3,N4,N5,N6,N7,N8,N9 }* *P={ N3,N7,N8,N9}* ***hops=4***

According to the above table, each case will be tested with different numbers of transmitted messages (5000, 10000, 20000, 40000). These doubled message number ranges will help to ensure accurate evaluation in both OOP and AOP paradigms as shown in Table 6.

Table 6. ABAC and MLS Message Level, OOP and AOP performance time

No.Msg	A		B		C		D	
	OOP	AOP	OOP	AOP	OOP	AOP	OOP	AOP
5000	**11100**	**11400**	**38112**	**38932**	**39815**	**42857**	**28750**	**30420**
10000	**22050**	**22200**	**75155**	**78607**	**82153**	**84862**	**58150**	**62647**
20000	**43700**	**44800**	**150387**	**155643**	**166994**	**169983**	**124407**	**129670**
40000	**85600**	**86600**	**292425**	**312498**	**332109**	**335630**	**251050**	**259931**

In the following the evaluations figures for each case individually are shown

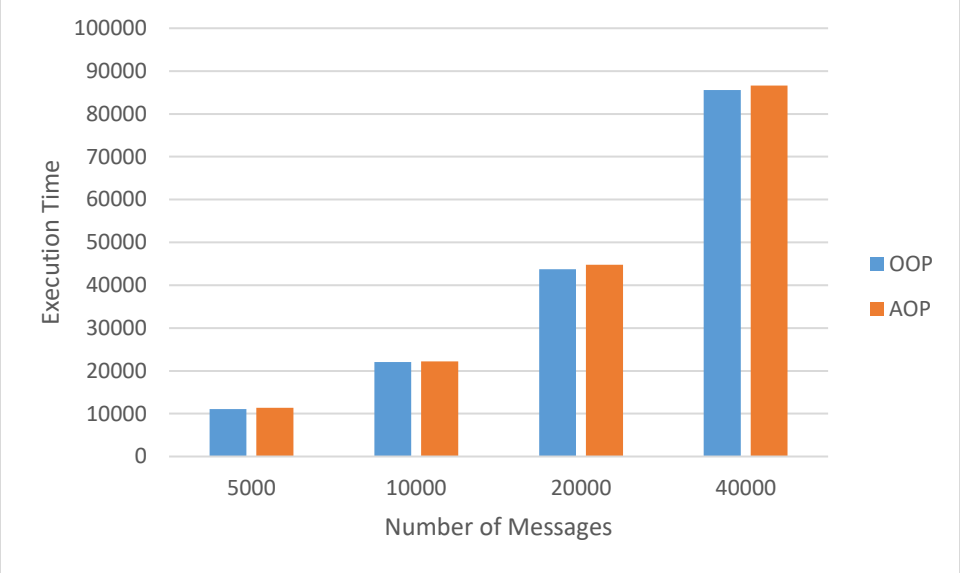


Figure 6.61.ABAC and MLS Message level with OOP and AOP runtime performance CASE "A".

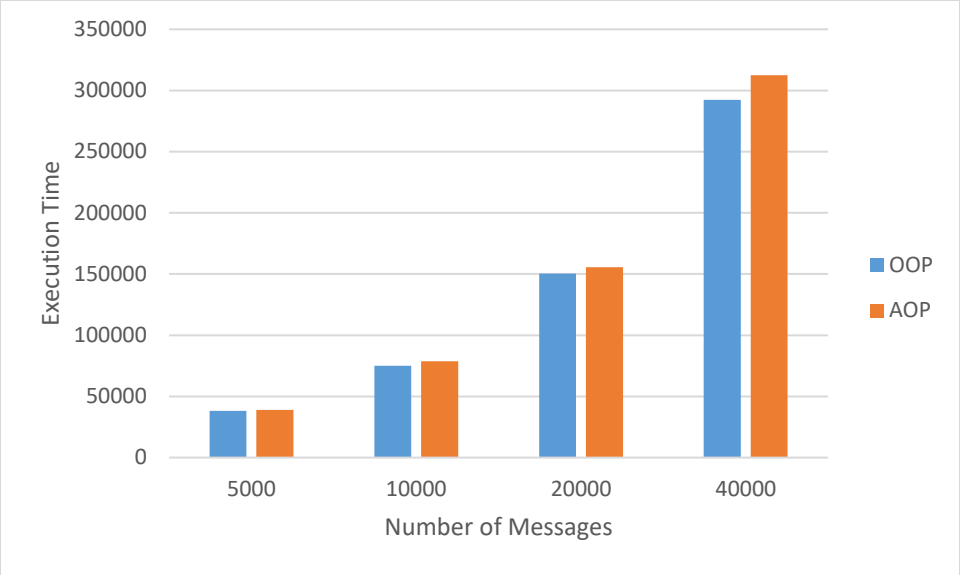


Figure 6.62. ABAC and MLS Message level with OOP and AOP runtime performance CASE "B".

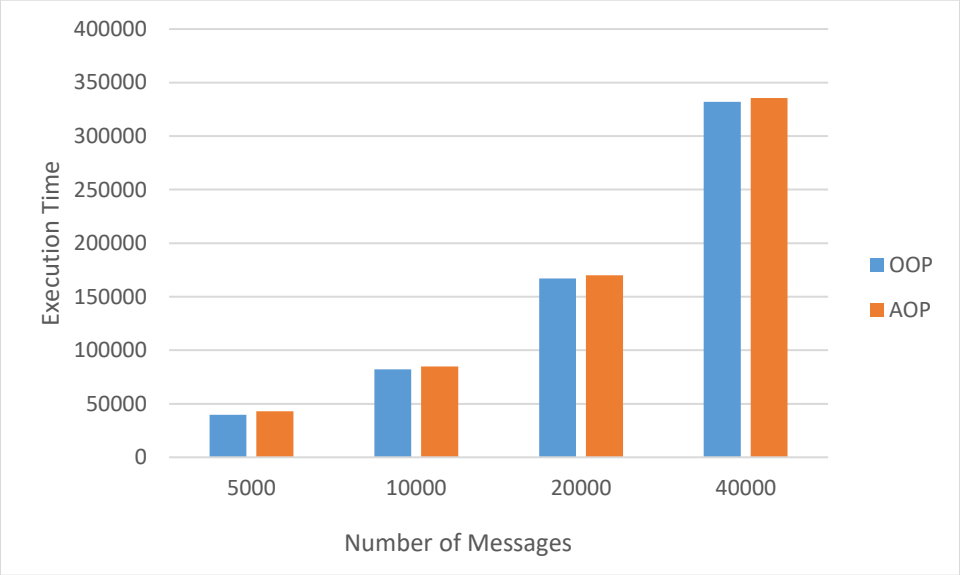


Figure 6.63. ABAC and MLS Message level with OOP and AOP runtime performance CASE "C".

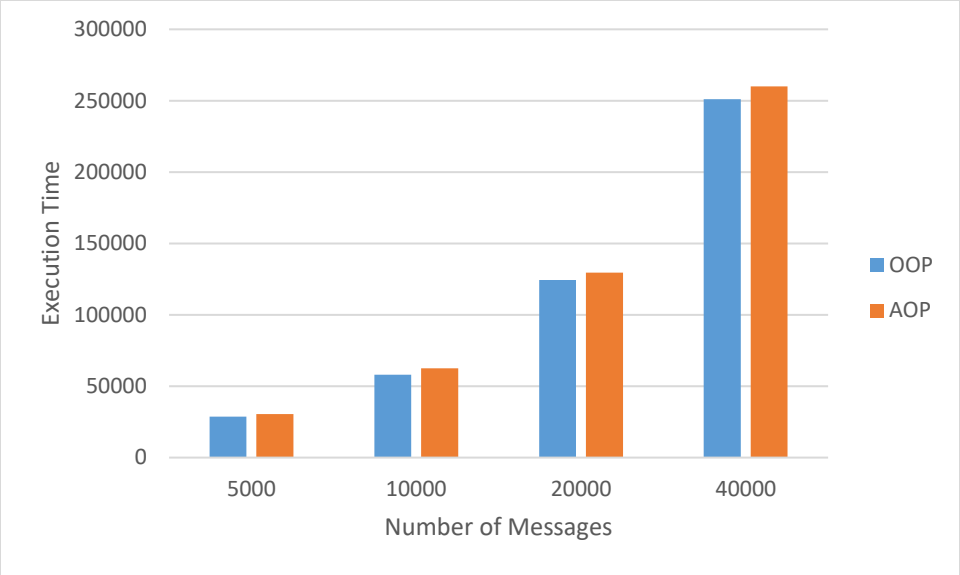


Figure 6.64. ABAC and MLS Message level with OOP and AOP runtime performance CASE "D".

B. Image Experiments

The evaluation of this section deals with the image according to message experiments, in which the image will be sent to different intended nodes without processing the contents. The process in this case will take one direction, thus the execution time will be calculated in the same way as with the message. The adopted table for this evolution is Table 5 and the number of transmitted images will be (10,20,40,80) and the image size is 100KB. Table 7 shows the results of this evaluation.

Table 7. ABAC and MLS one direction image Level, OOP and AOP performance time

No.Imgs	A		B		C		D	
	OOP	AOP	OOP	AOP	OOP	AOP	OOP	AOP
10	**8465**	**9678**	**27262**	**29342**	**68004**	**70125**	**63751**	**65874**
20	**15798**	**16012**	**57663**	**58863**	**141267**	**143874**	**129768**	**133217**
40	**30298**	**32364**	**120838**	**123895**	**280102**	**284916**	**256952**	**259812**
80	**60637**	**63157**	**239512**	**242456**	**566153**	**569784**	**507884**	**510743**

In the following the evaluations figures for each case individually are shown

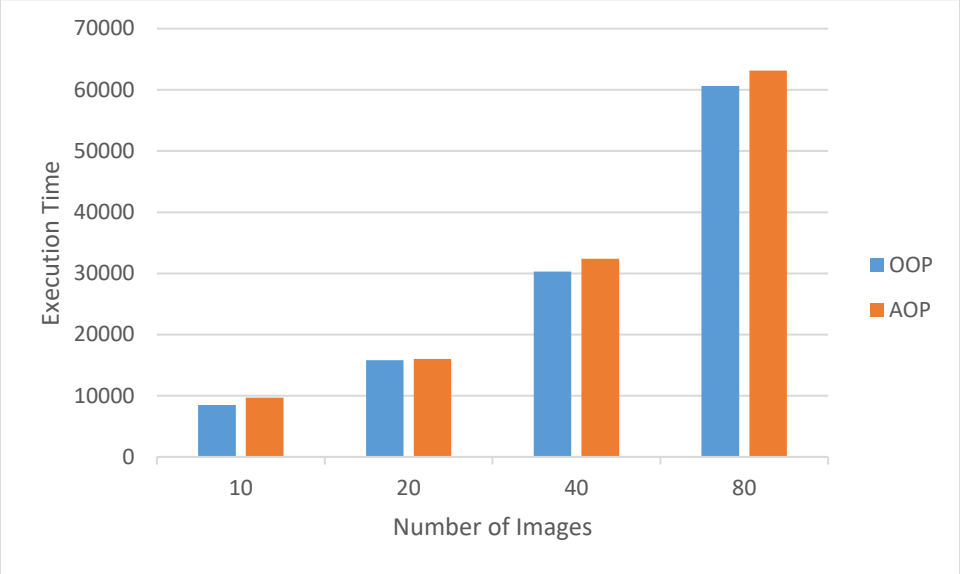


Figure 6.65. ABAC and MLS Image level with OOP and AOP runtime performance CASE "A".

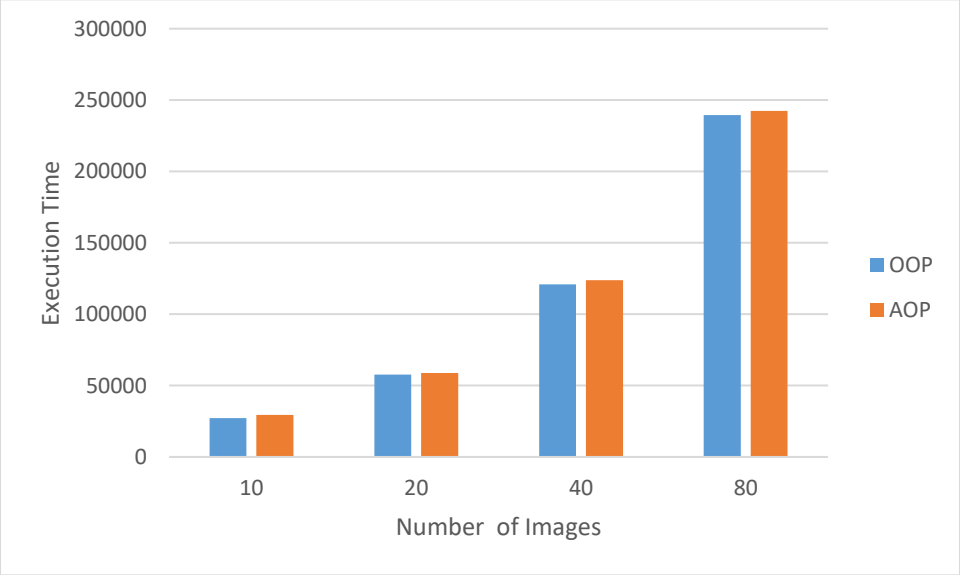


Figure 6.66. ABAC and MLS image level with OOP and AOP runtime performance CASE "B".

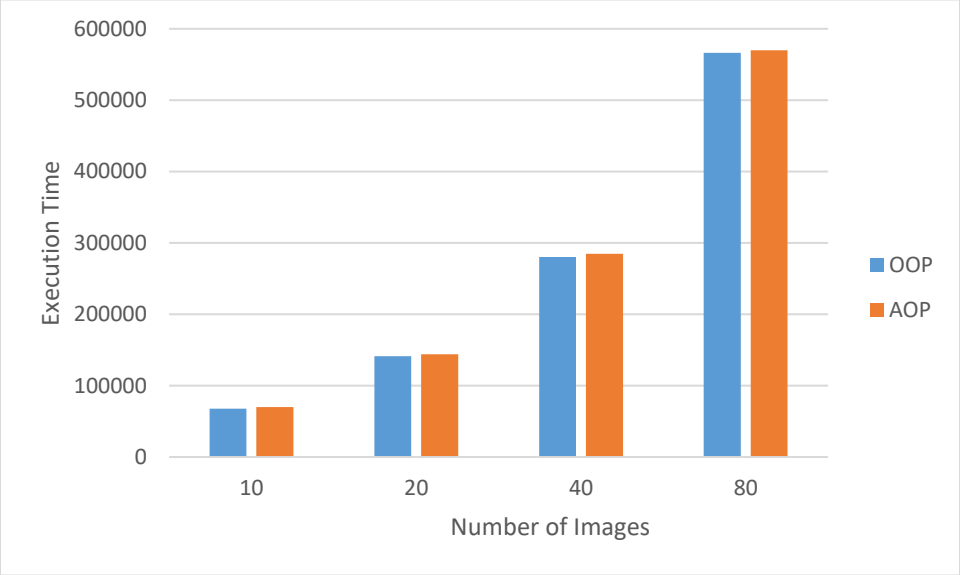


Figure 6.67. ABAC and MLS image level with OOP and AOP runtime performance CASE "C".

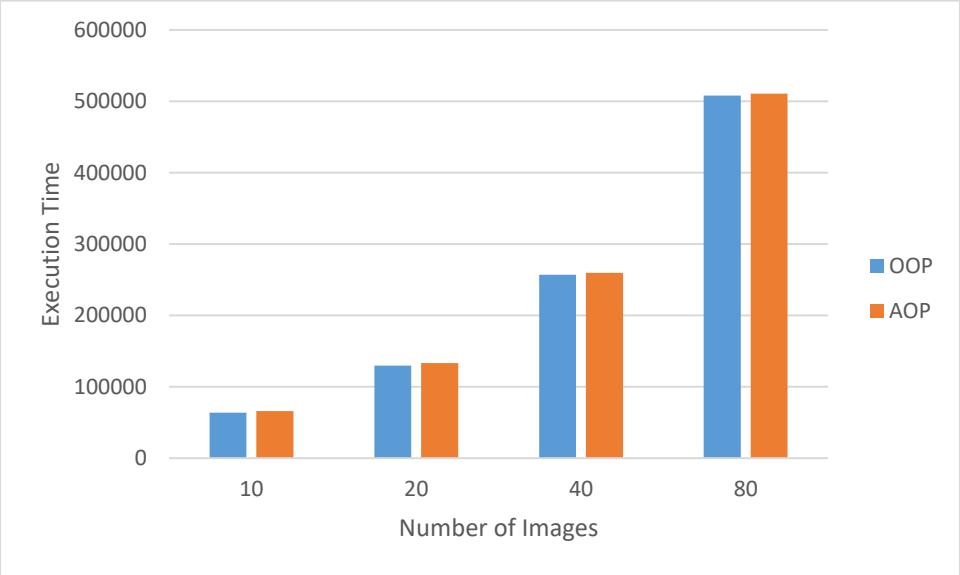


Figure 6.68. ABAC and MLS image level with OOP and AOP runtime performance CASE "D".

6.2.2 ABAC & IBAC & MLS

In this section, IBAC will be injected as an intermediate access model between ABAC and MLS. This model will help the source node to identify the intended nodes

according to the identities. As mentioned in chapters 4 &5, this model is optional and will be compulsory when adopting a domination relationship or when the source node asks specific nodes to do the process even if there are other nodes having the same attributes. The following evaluation is based on the second option when there are some nodes sharing the same attributes and security clearance.

1. Two Directional Experiments

The evaluation of this section deals with all proposed access control models. We used Identity-Based access control in addition to ABAC and MLS. In order to grant the node access to information, the associated access information which was sent by source node with the file must be filtered by the three models starting form ABAC followed by IBAC and finally MLS model. The performance of this section has been evaluated according to the Table 3 and execution time will be same as in exp (1). Table 7 shows the numerical results of the combined access control model.

Table 7. ABAC, IBAC and MLS, File level, OOP and AOP runtime performance

No.Files	A		B		C		D		E	
	OOP	AOP	OOP	AOP	OOP	AOP	OOP	AOP	OOP	AOP
50	**1181**	**1263**	**2021**	**2133**	**3577**	**3955**	**4536**	**4840**	**10958**	**11125**
100	**2316**	**2489**	**4053**	**4275**	**7156**	**7842**	**8923**	**9861**	**22426**	**24152**
200	**4679**	**4777**	**8293**	**8664**	**14523**	**15824**	**17862**	**19697**	**45241**	**47373**
400	**9578**	**9637**	**16721**	**17472**	**28873**	**31577**	**35389**	**39167**	**89782**	**93637**
800	**19140**	**19983**	**33296**	**34906**	**57558**	**63686**	**70291**	**78260**	**178845**	**182496**

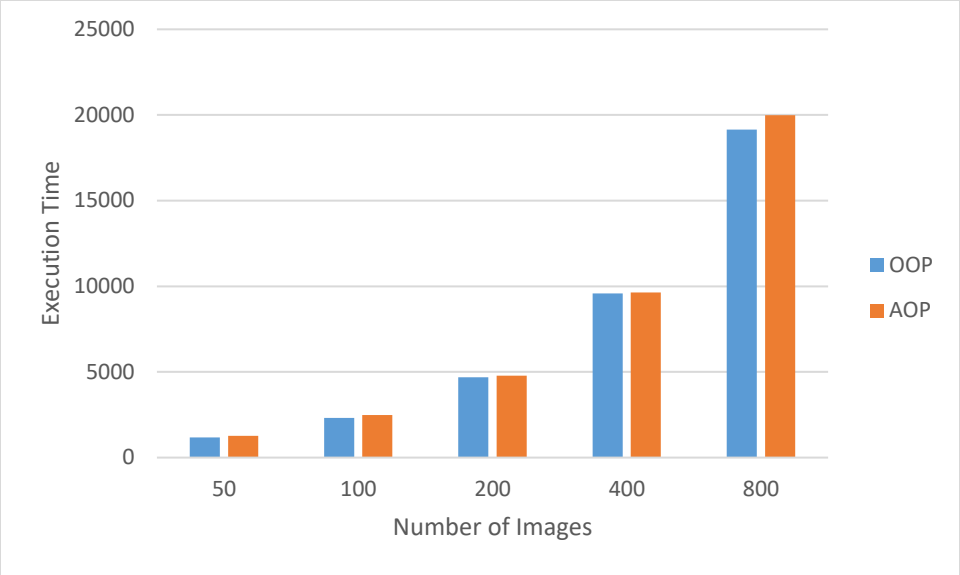


Figure 6.69 .ABAC, IDAC and MLS File level with OOP and AOP runtime performance CASE "A".

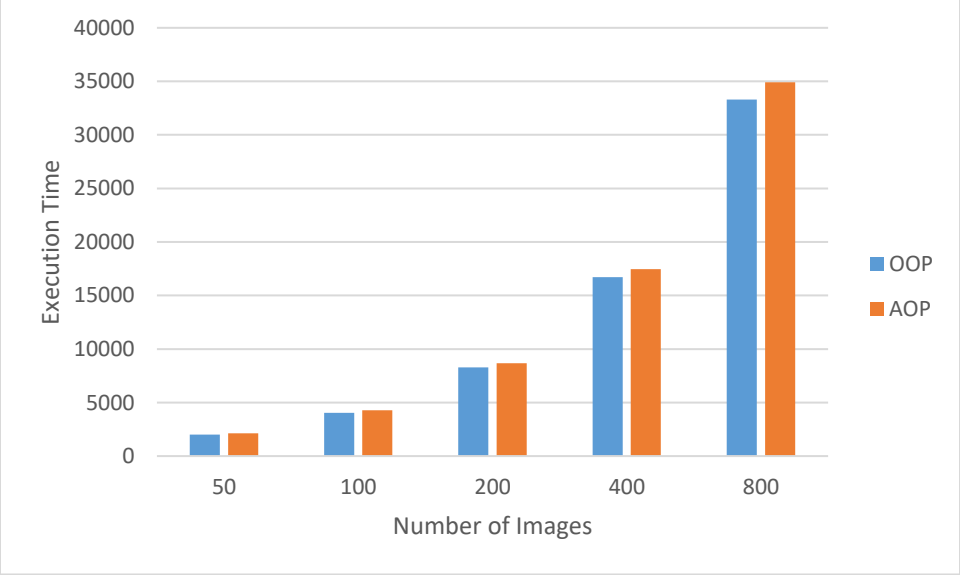


Figure 6.70.ABAC, IDAC and MLS File level with OOP and AOP runtime performance CASE "B".

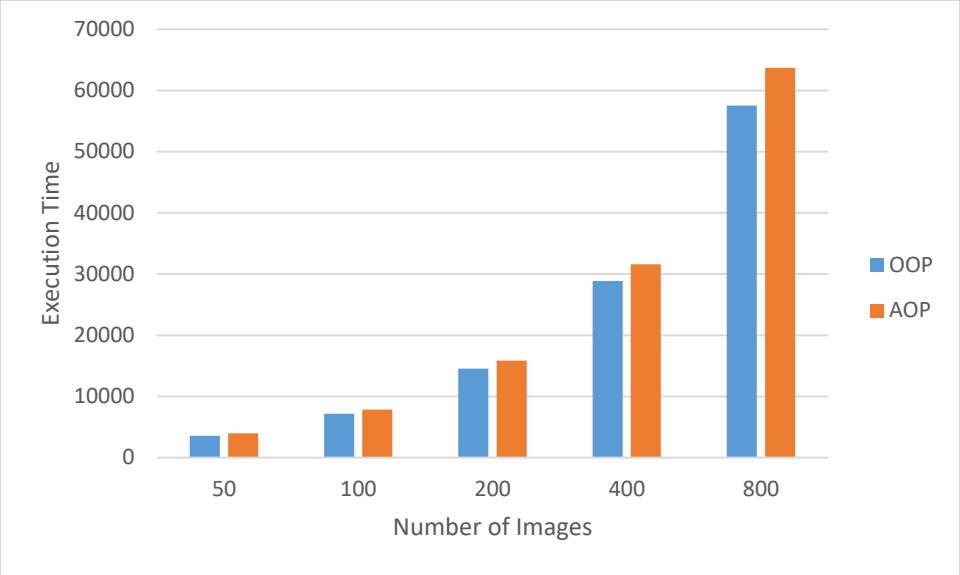


Figure 6.71.ABAC, IDAC and MLS File level with OOP and AOP runtime performance CASE "C".

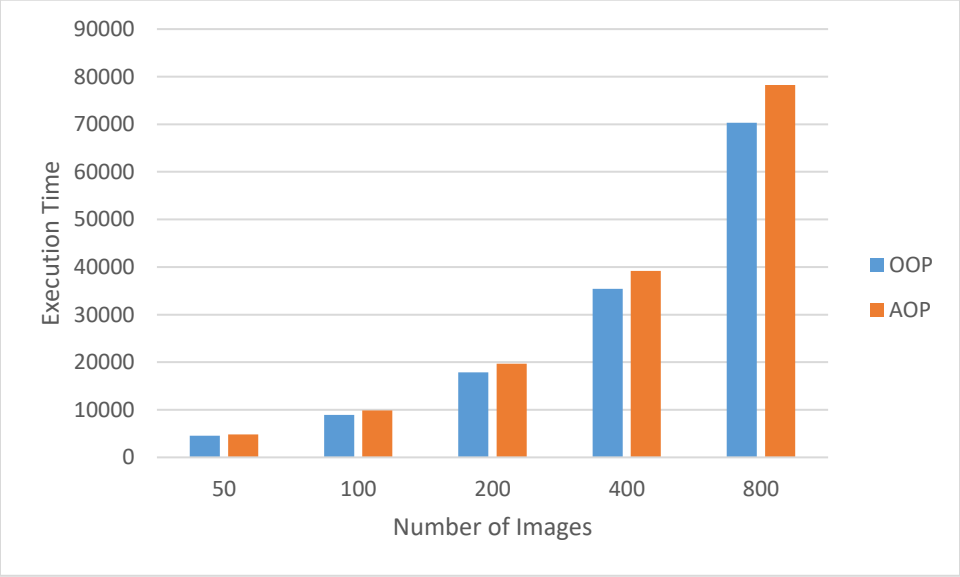


Figure 6.72.ABAC, IBAC and MLS File level with OOP and AOP runtime performance CASE "D".

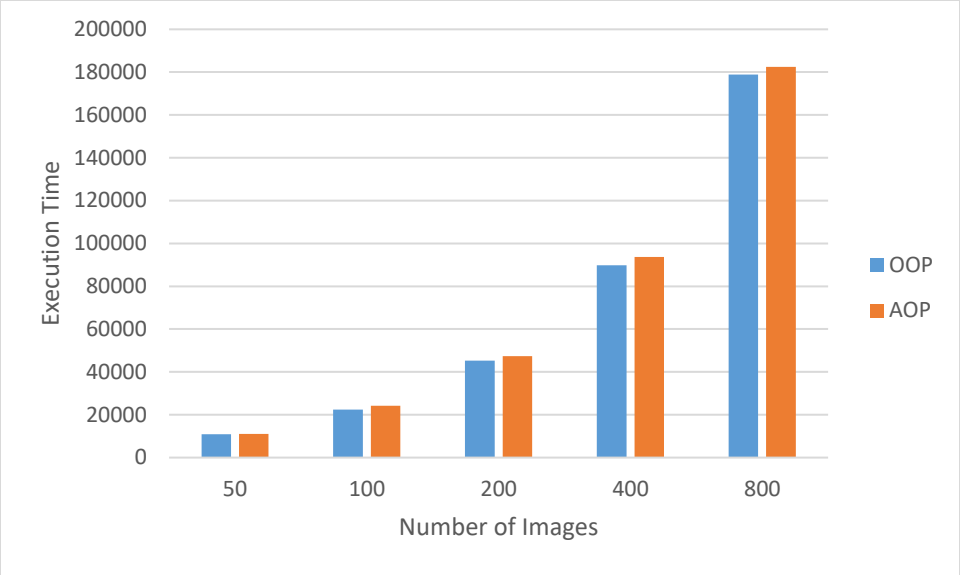


Figure 6.73. ABAC, IBAC and MLS File level with OOP and AOP runtime performance CASE "E".

2. One Directional Experiments

A. Message Experiments

The evaluation of this section deals with all proposed access control models. We used Identity-Based access control in addition to ABAC and MLS. In order to grant the node access to information, the associated access information which was sent by source node with the file must be filtered by the three models starting from ABAC followed by IBAC and finally MLS model. The runtime performance of this section has been evaluated according to the Table 5 and execution time will be calculated the same as in exp (2). Table 8 shows the numerical result of this experiments.

Table 8. ABAC, IBAC and MLS, Message level, OOP and AOP runtime performance

No.Msg	A		B		C		D	
	OOP	AOP	OOP	AOP	OOP	AOP	OOP	AOP
5000	**11225**	**11402**	**39050**	**40406**	**42500**	**43153**	**38150**	**40353**
10000	**22422**	**23613**	**78980**	**79789**	**84600**	**85892**	**77653**	**79630**
20000	**44153**	**44950**	**155380**	**159985**	**175900**	**177397**	**146273**	**149732**
40000	**86350**	**87104**	**308490**	**310126**	**336150**	**338921**	**287847**	**294560**

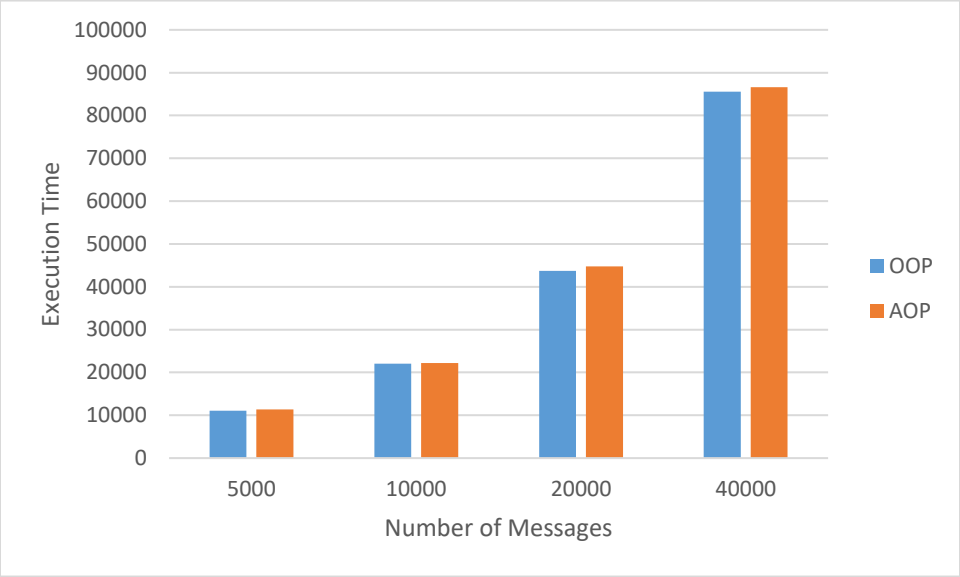


Figure 6.74. ABAC, IBAC and MLS Message level with OOP and AOP runtime performance CASE "A".

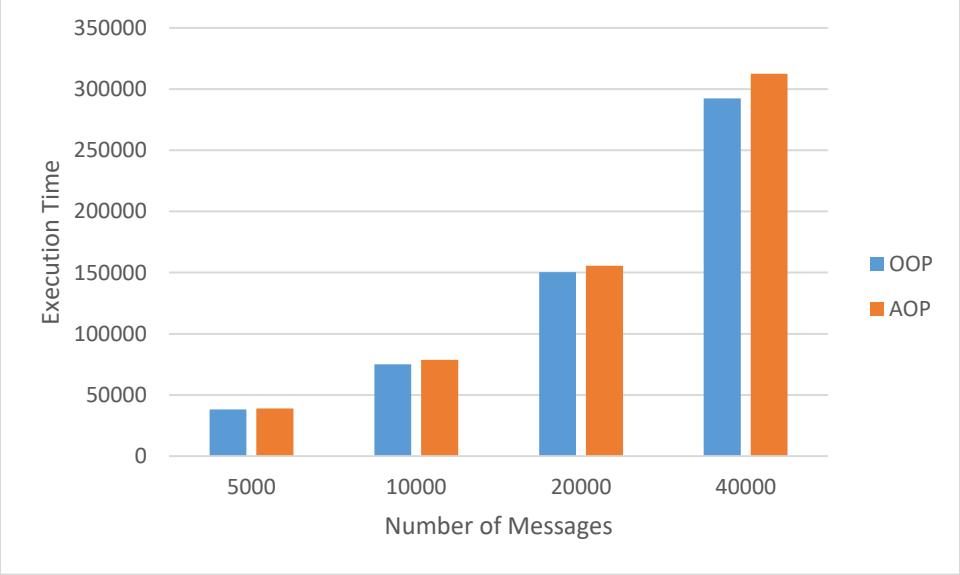


Figure 6.75. ABAC, IBAC and MLS Message level with OOP and AOP runtime performance CASE "B".

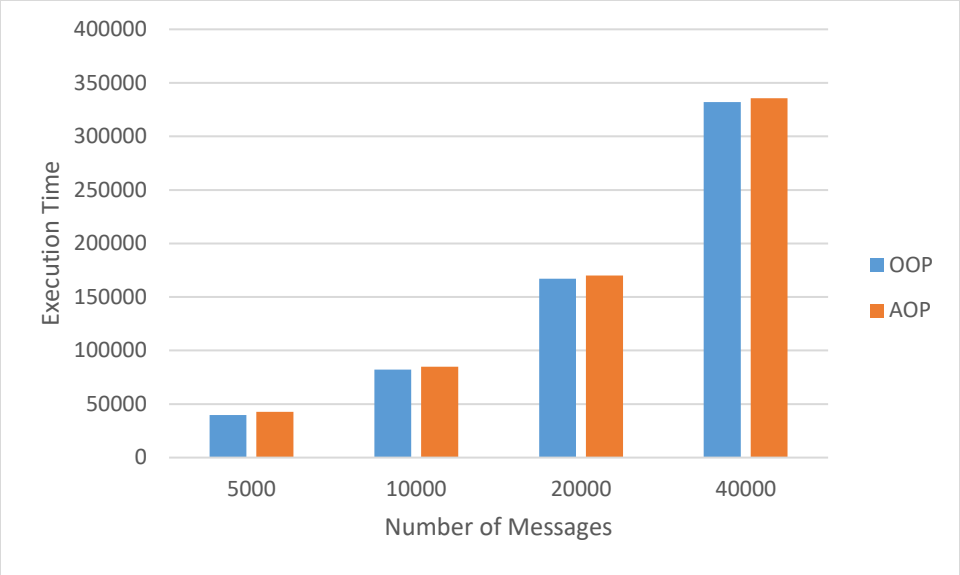


Figure 6.76. ABAC, IBAC and MLS Message level with OOP and AOP runtime performance CASE "C".

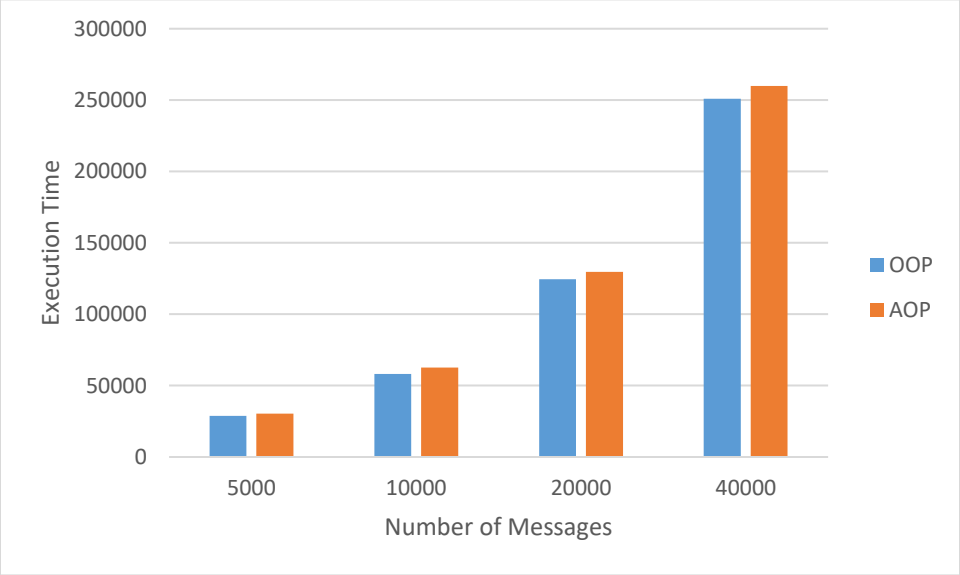


Figure 6.77. ABAC, IBAC and MLS Message level with OOP and AOP runtime performance CASE "D".

B. Image Experiments

As the same with section ***6.1.1.2.2*** the experiment deals with images by applying the combined access control. Table 9 shows the consequence result of this experiment.

Table 9. ABAC, IBAC and MLS, Image level, OOP and AOP runtime performance.

No.Imgs	A		B		C		D	
	OOP	AOP	OOP	AOP	OOP	AOP	OOP	AOP
10	**8956**	**10213**	**28345**	**30216**	**68716**	**70967**	**64997**	**67317**
20	**15824**	**16934**	**58876**	**60127**	**142569**	**144942**	**131135**	**135320**
40	**30940**	**33215**	**121938**	**124823**	**281931**	**285798**	**258236**	**261591**
80	**61345**	**63989**	**240984**	**244127**	**567836**	**571527**	**509572**	**512825**

The graphical representation of these results according to the cases are as follows:

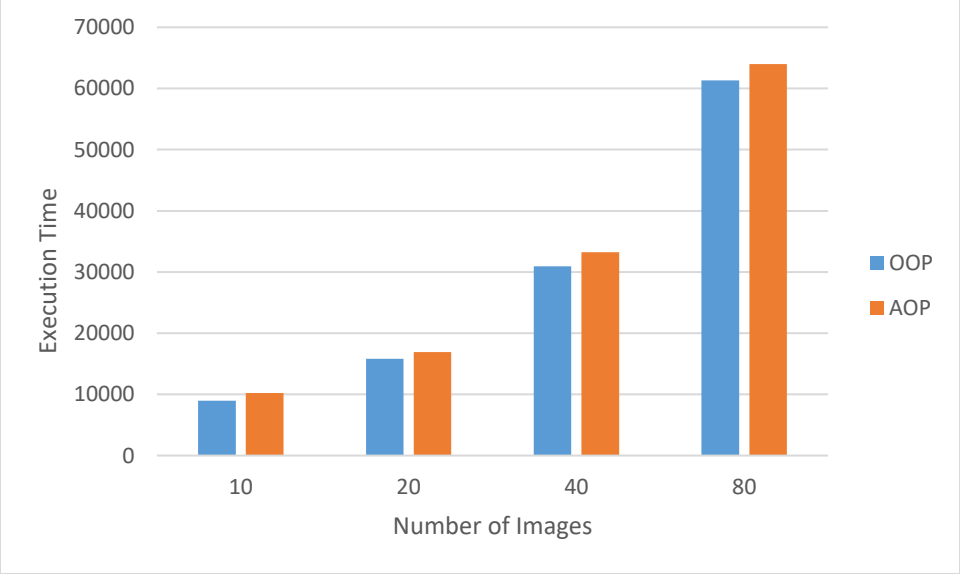


Figure 6.78. ABAC, IBAC and MLS Image level with OOP and AOP runtime performance CASE "A".

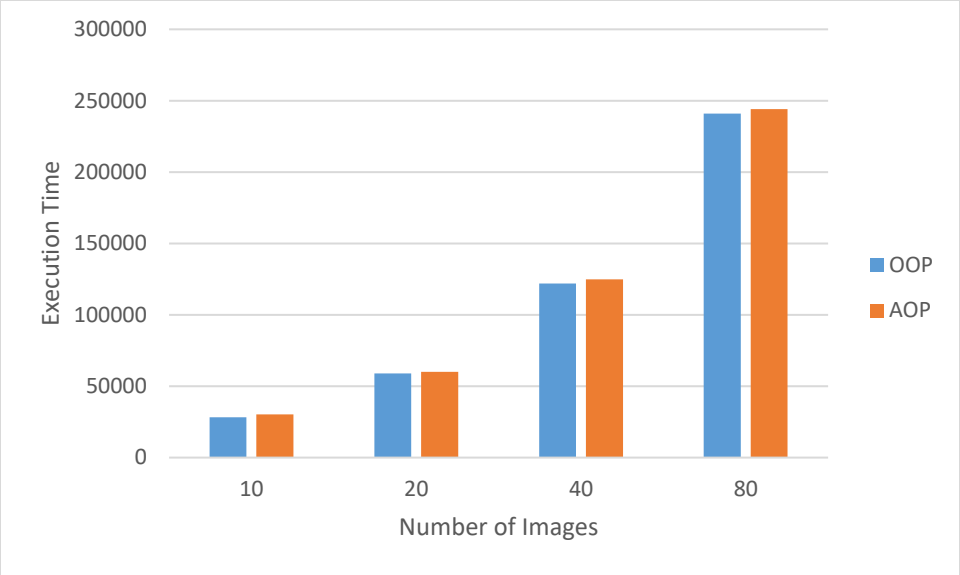


Figure 6.79. ABAC, IBAC and MLS Image level with OOP and AOP runtime performance CASE "B".

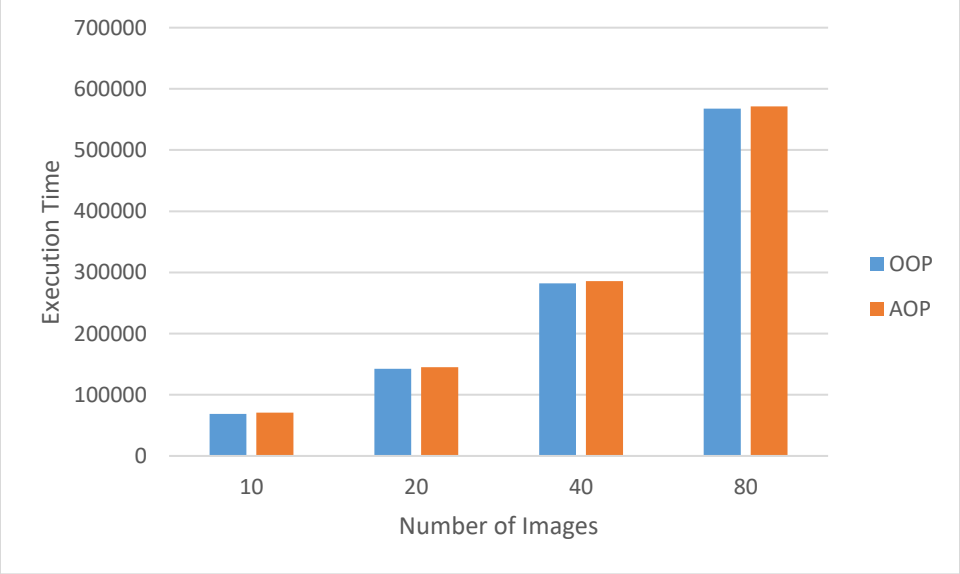


Figure 6.80. ABAC, IBAC and MLS Image level with OOP and AOP runtime performance CASE "C".

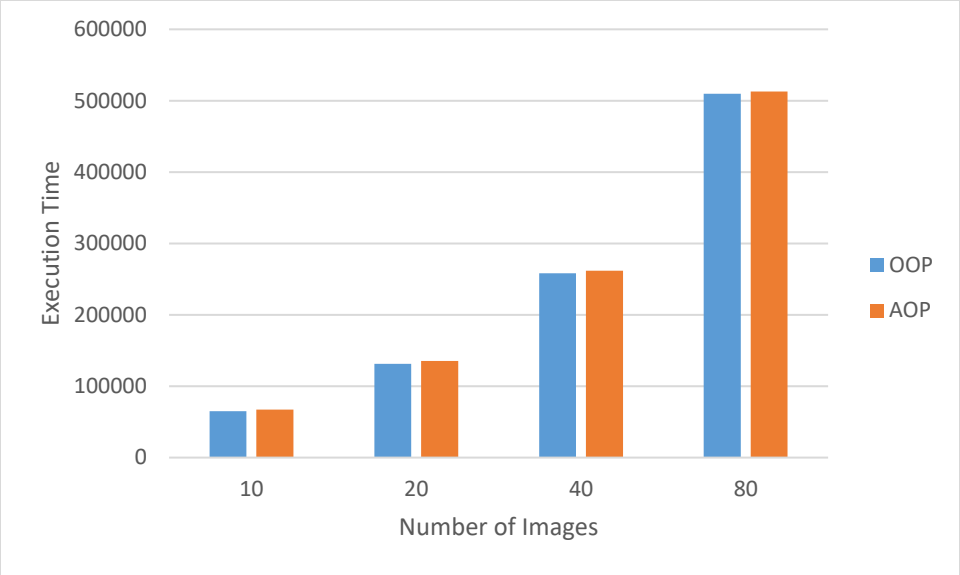


Figure 6.81. ABAC, IBAC and MLS Image level with OOP and AOP runtime performance CASE "D".

6.2.3 Comparing ABAC+MLS with ABAC+ IBAC + MLS

- **File Experiments**

  In this section, we have compared between the evaluated results in section 6.1.1 and section 6.1.2 to see the heaviness of using IBAC. The fifth case in the file level is used to evaluate this comparison as shown in Figure 6.82 with AOP, and Figure 6.83 shows the comparison between the evaluations using OOP.

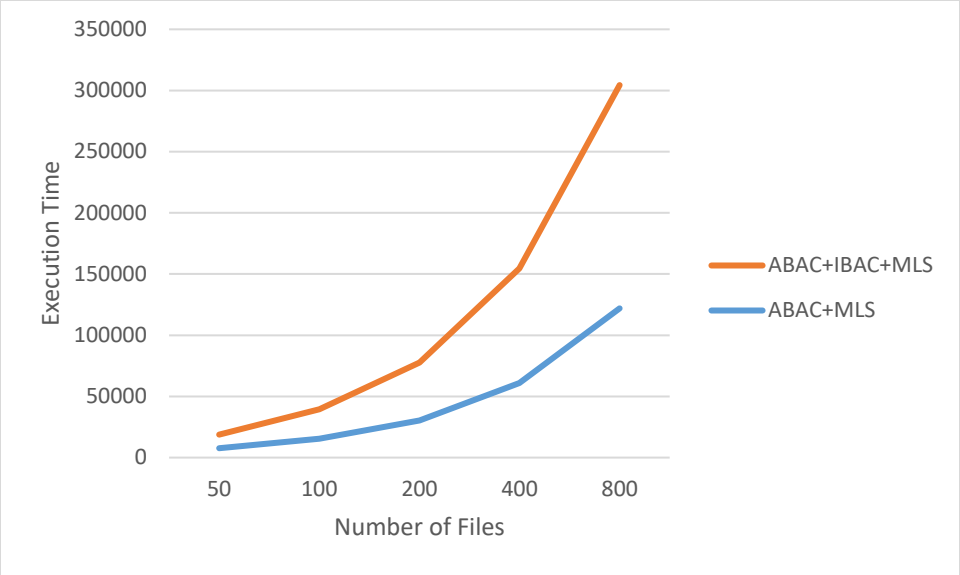


Figure 6.82. Comparing between AOP in File level with ABAC+MLS and ABAC+ IBAC and MLS.

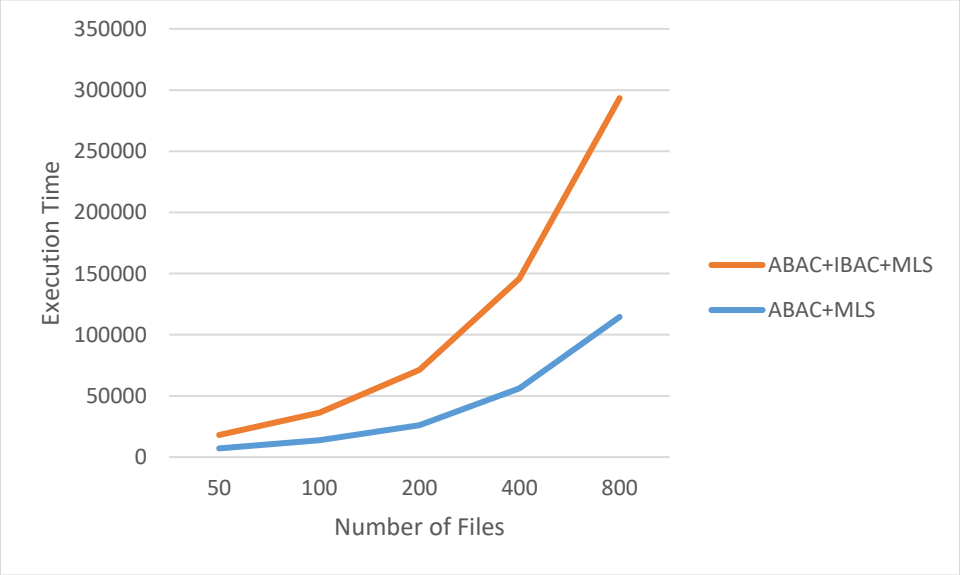


Figure 6.83. Comparing between OOP in File level with ABAC+MLS and ABAC+ IBAC and MLS.

- **Message Experiments**

  In this section we have compared between the evaluated results in section 6.1.1 and section 6.1.2 to see the heaviness of using IBAC. The fourth case in the file level is used to evaluate this comparison as shown in Figure 6.84 with AOP, and Figure 6.85 shows the comparison between the evaluations using OOP. It

is worth mentioning that in case of broadcasting the message or images, if the system nodes have the same attributes then adding IBAC will decrease the execution time because the process will be directed to a specific node, regardless of the attributes. In evaluation, the system nodes have different attributes, so using IBAC will increase the load of execution time.

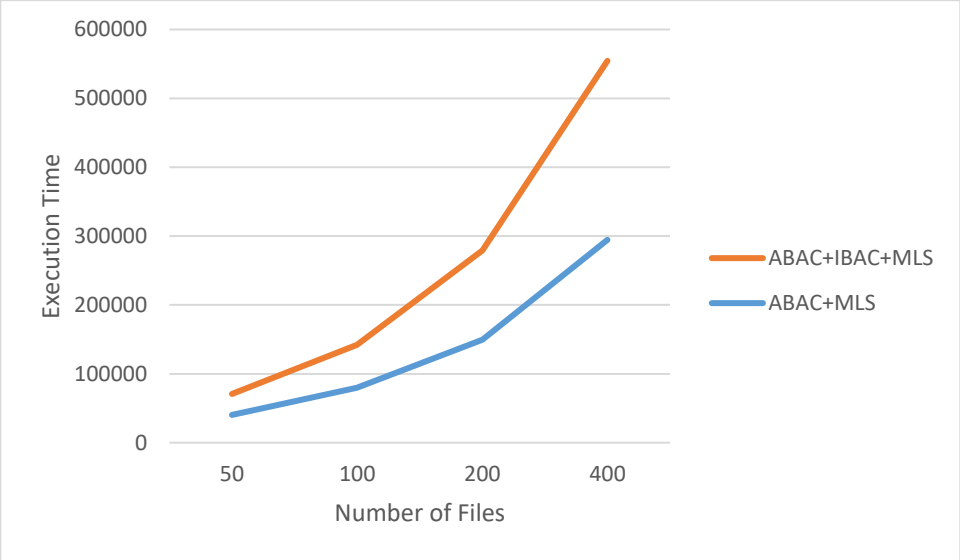


Figure 6.84. Comparing between AOP in Message level with ABAC+MLS and ABAC+ IBAC and MLS.

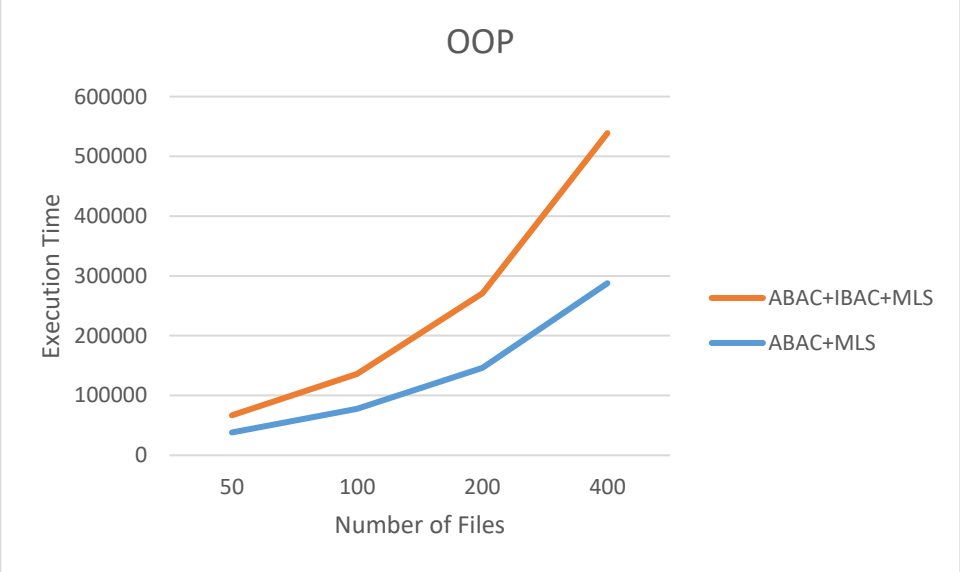


Figure 6.85. Comparing between OOP in Message level with ABAC+MLS and ABAC+ IBAC and MLS.

- **Image Experiments**

  In this section we have compared between the evaluated results in section 6.1.1 and section 6.1.2 image level to see the heaviness of using IBAC. The fourth case in the image level is used to evaluate this comparison as shown in Figure 6.86 with AOP, and Figure 6.87 shows the comparison between the evaluations using OOP.

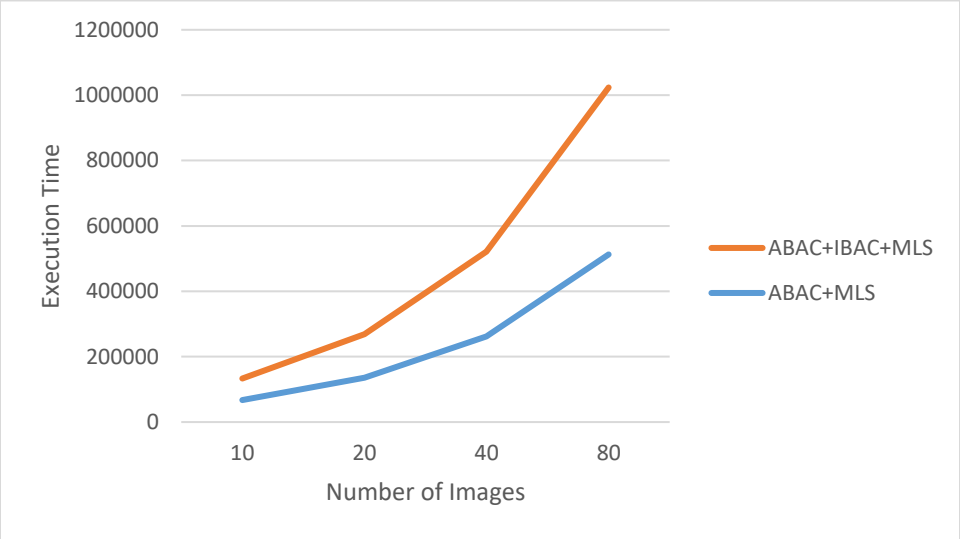


Figure 6.86.Comparing between AOP and OOP in Image level with ABAC+MLS and ABAC+ IBAC and MLS.

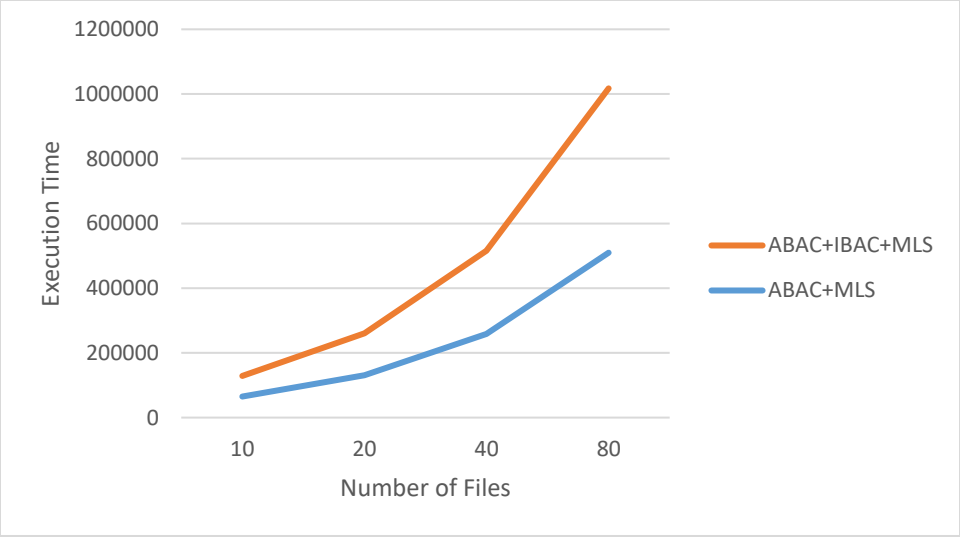


Figure 6.87. Comparing between AOP and OOP in Image level with ABAC+MLS and ABAC+ IBAC and MLS.

6.3 3AC_AOP Cryptography

In order to prevent unauthorized nodes from accessing the transmitted information as well as to keep the information in unreadable form to ensure the integrity of transmitted information between distributed system communication channels which may attacked by eavesdropping attacker, we will adopt a cryptography system. The proposed system used the AES cryptography algorithm with key size 128 to encrypt the information before sending and decrypting it after satisfying the accessing conditions. The crypto keys will be distributed by the distributed system manager in the form of a JSON file associated with access control rules or by assigning this task to a special node working as a key distribution centre (KDC). Both solutions are accepted, however the next evaluations are based on the first choice when the system manager distributes the keys between distributed nodes.

Because of inserting cryptography concerns in the system, this will lead to increasing the heaviness of the system working, thus increasing the rate of consumption of the system resources e.g batteries if we use mobile devices as nodes in the system. For this reason, we module the cryptography algorithms as dynamic aspect pointcuts in which adding, deleting or updating the crypto algorithm is carried out dynamically in real time. Hence, applying the cryptography method only when the task needs it, otherwise the system will work smoothly and this is called dynamic clustering. In another meaning, each node with its next neighbours have the same security level, this mean that they are related to the same security cluster and we supposed that the transmitted information doesn't need to be encrypted within the same cluster. Nevertheless, this cluster can change dynamically, for this reason aspect advice will be ready to deal with any incident changes in real time.

The next evaluation shows the comparison between static and dynamic cryptography aspects with both files and messages to show the ability of the model to deal with both cases.

6.3.1 Static Cryptography Aspect with Message.

Table 10 shows that the performance time of using AOP is more than the time of using OOP.

Table 10. Static cryptography aspect with message, OOP and AOP performance

No.Msg	A		B		C		D	
	OOP	AOP	OOP	AOP	OOP	AOP	OOP	AOP
5000	**12000**	**18600**	**60800**	**80800**	**55800**	**83400**	**45600**	**73000**
10000	**22500**	**36200**	**115800**	**157800**	**123800**	**162800**	**105800**	**141800**
20000	**44200**	**67600**	**220800**	**315000**	**228600**	**321400**	**216000**	**279200**
40000	**92800**	**128600**	**450800**	**618600**	**462600**	**624200**	**412000**	**548600**

The performance time will be calculated as follows:

$$Exexution\ time = \sum_{i=0}^{\lambda-1} \omega + \sum_{i=0}^{\lambda-1} AC + \sum_{i=0}^{\rho-1} Crypto \ldots\ldots \exp(3)$$

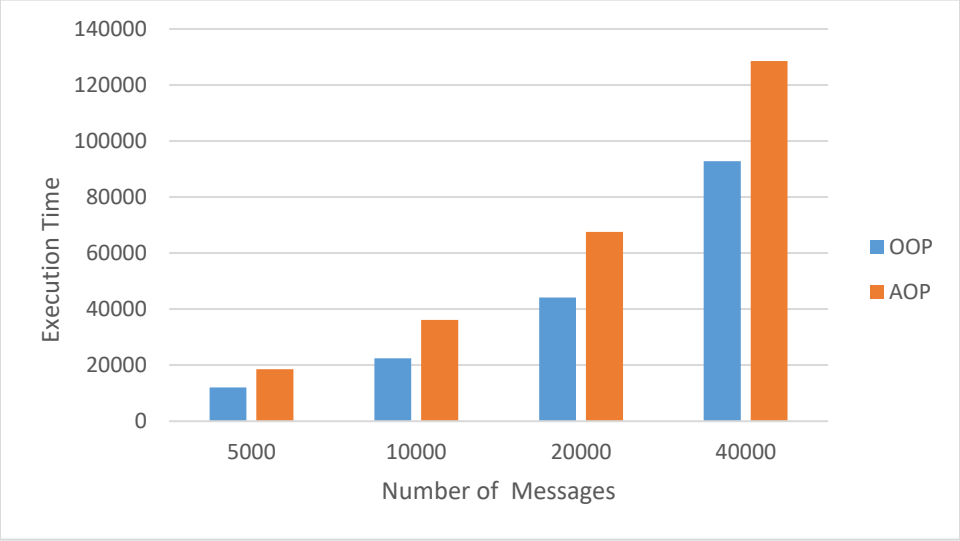


Figure 6.88. Static cryptography aspect, Message Level, CASE "A".

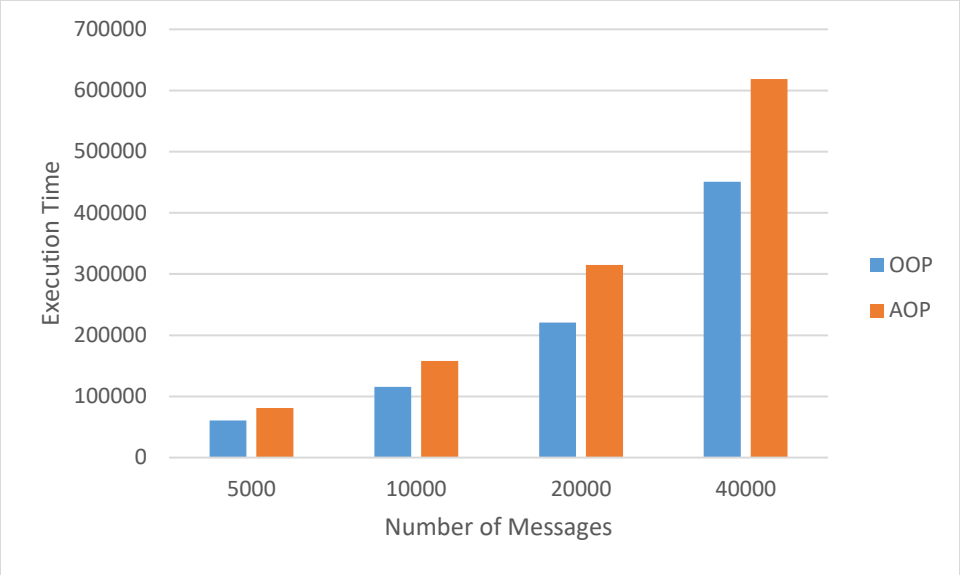


Figure 6.89. Static cryptography aspect, Message Level, CASE "B".

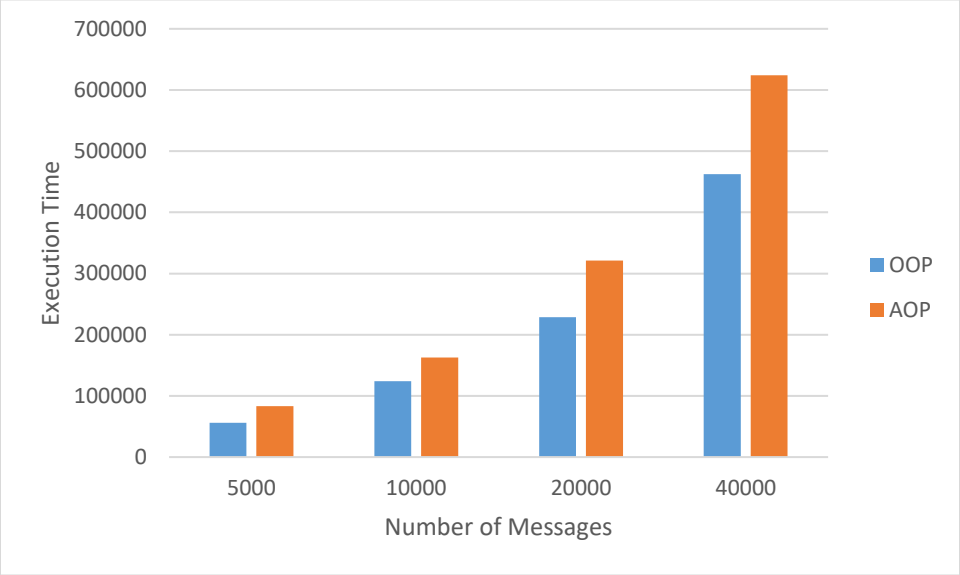


Figure 6.90. Static cryptography aspect, Message Level, CASE "C".

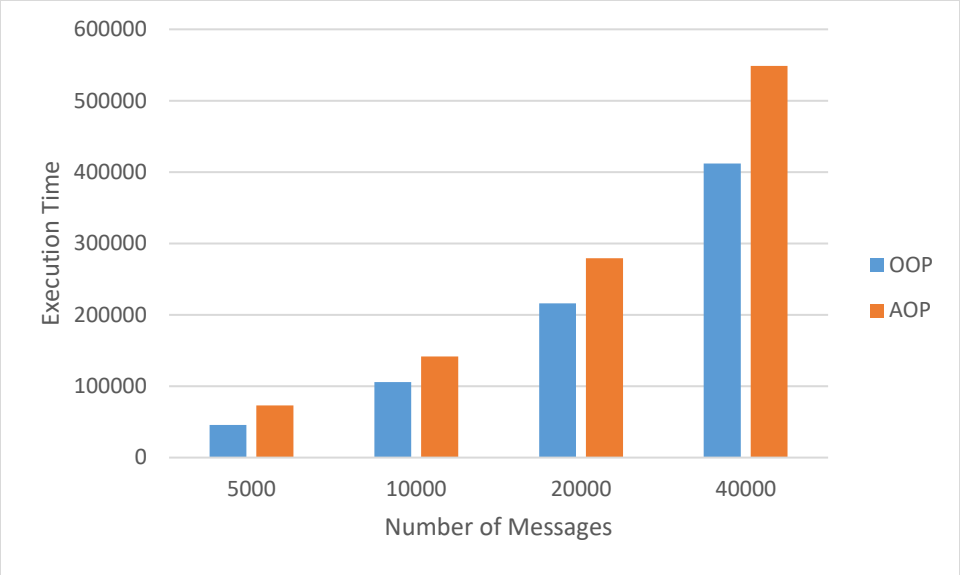


Figure 6.91. Static cryptography aspect, Message Level, CASE "D".

6.3.2 Static Cryptography Aspect with Files.

Table 11 shows the numerical results of applying static cryptography aspect with files

Table 11. Static cryptography aspect with files, OOP and AOP performance

No.Files	A		B		C		D		E	
	OOP	AOP	OOP	AOP	OOP	AOP	OOP	AOP	OOP	AOP
50	1518	1913	2453	3062	4045	4762.5	5057.5	5917.5	8261.5	9028
100	2946	3829	4910	6017	7915	9412	10038	11548	16486	17945
200	5824	7416	9624	11978	15690	18440	19896	29563	32756	35604
400	11460	14412	18636	22692	30892	36580	39512	45380	64992	71024
800	22192	28184	36840	44128	60704	72720	77808	89248	128712	139472

The performance will be calculated as follows:

$$Exexution\ time = 2\sum_{i=0}^{\lambda-1}\omega + \sum_{j=0}^{\rho-1} process + \sum_{i=0}^{\lambda-1} AC + \sum_{i=0}^{\rho-1} Crypto \ldots\ldots \exp(4)$$

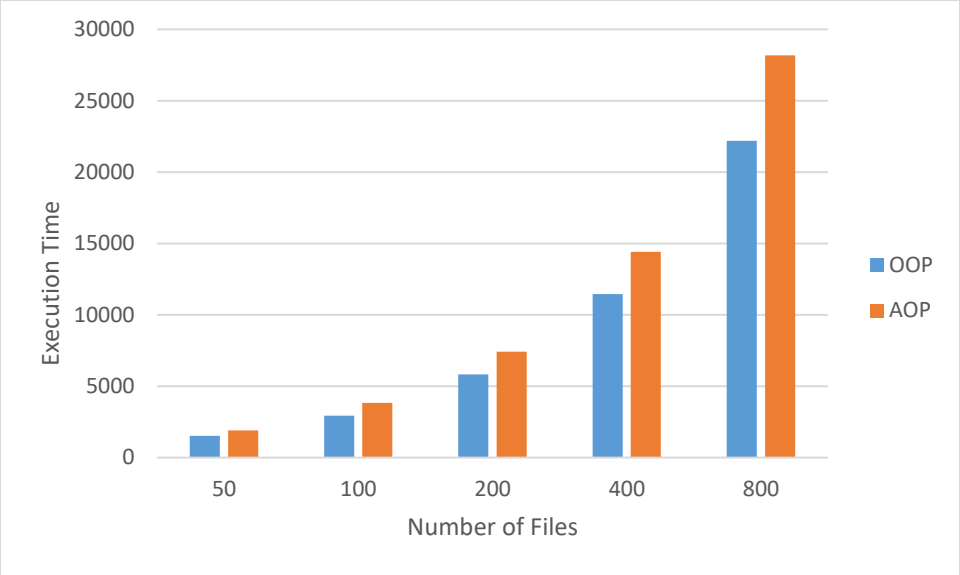


Figure 6.92. Static cryptography aspect, File Level, CASE "A".

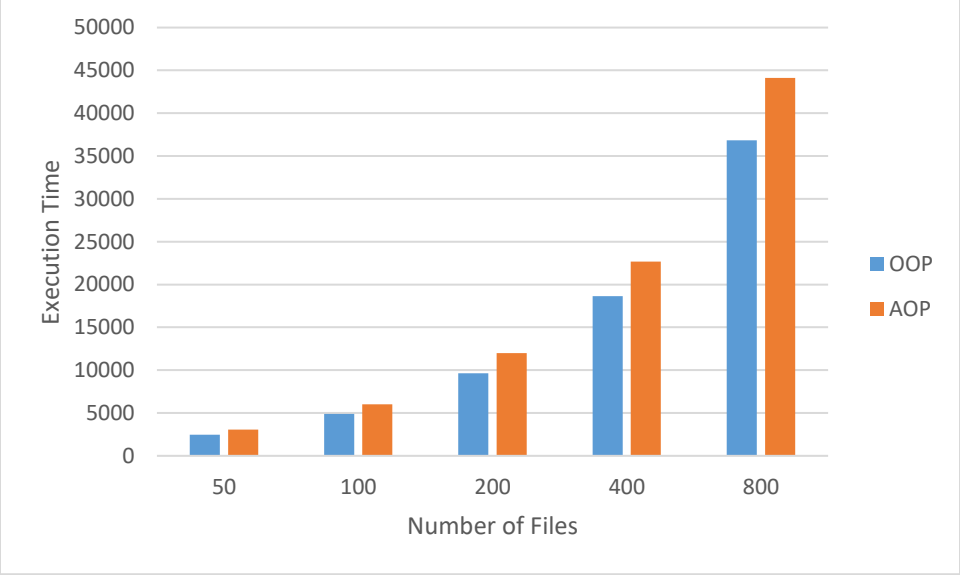


Figure 6.93. Static cryptography aspect, File Level, CASE "B".

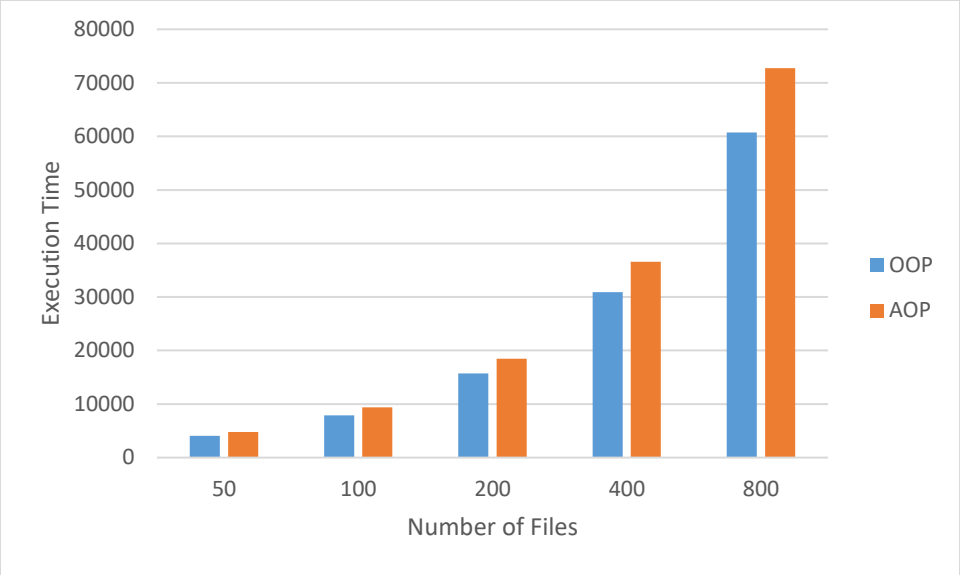


Figure 6.94. Static cryptography aspect, File Level, CASE "C".

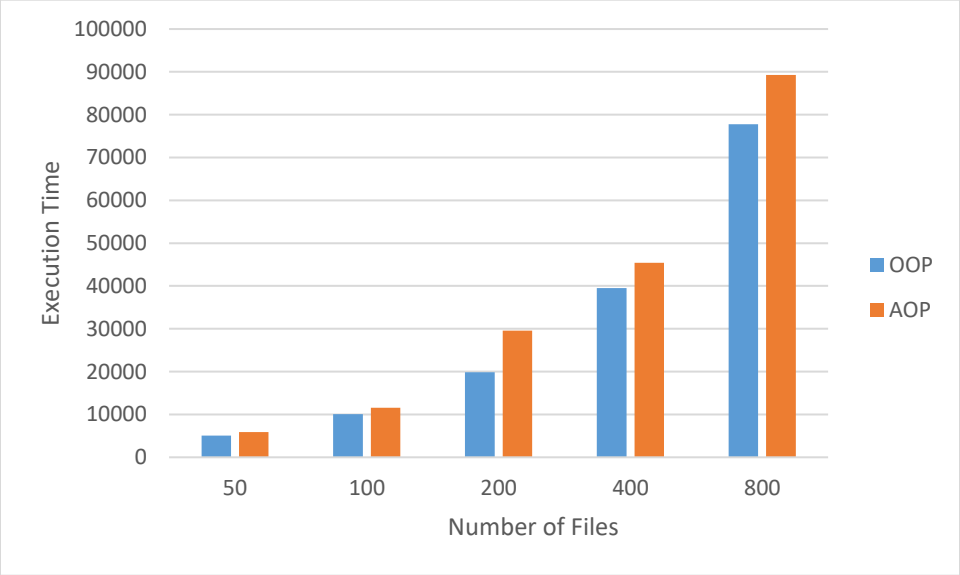


Figure 6.95. Static cryptography aspect, File Level, CASE "D".

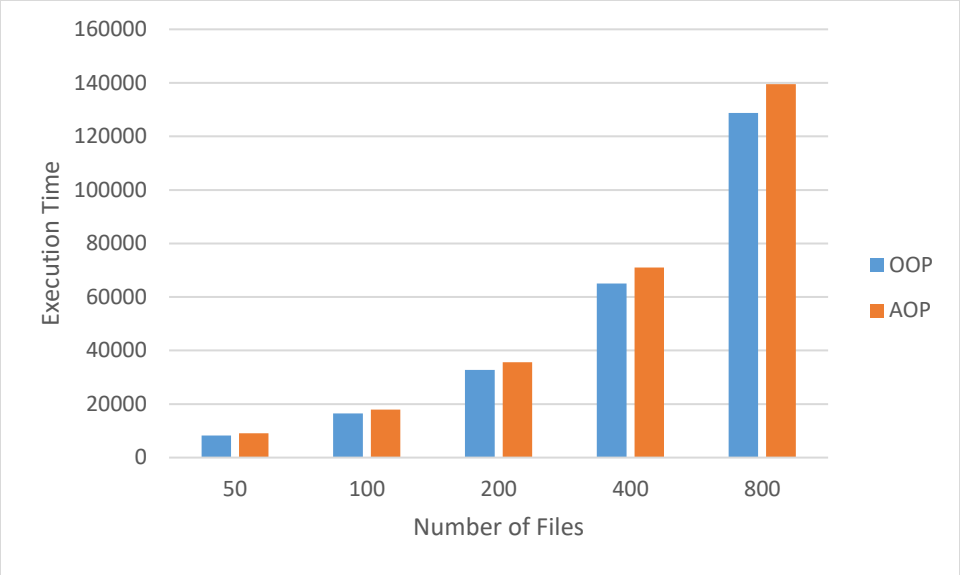


Figure 6.96. Static cryptography aspect, File Level, CASE "E".

6.4 Dynamic clustering with cryptography

The evaluation with this section adopts dynamic AOP to deal with clustering the system nodes according to the level of security clearance. This clustering will put the neighbour nodes in a cluster if they have same clearance level. In this case, there is no need to implement encryption/ decryption processing within the same cluster, thus decreasing the heavy load of the implementation which is caused by the cryptography processing, see Figure 6.97 . In fact, dynamic clustering is dependent on the clearance level of the source node in which the encryption/decryption operation will not work if the level of the cluster is greater than or equal to the source level, otherwise, the system will work normally as before. Hence, saving information from illegal access.

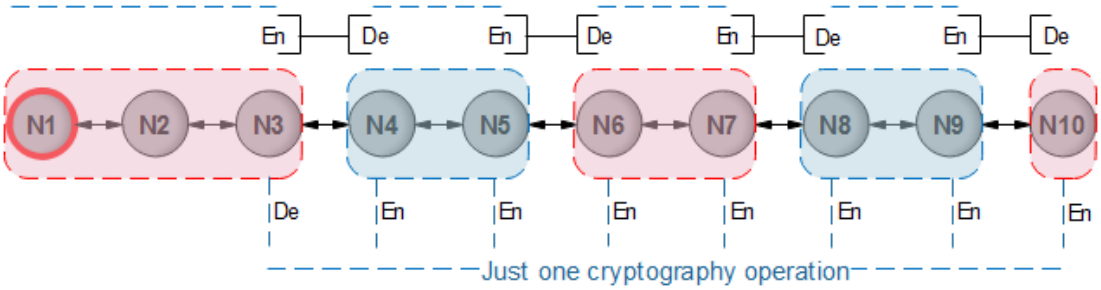


Figure 6.97. Dynamic cluster of distributed nodes, use 2 different clearance level.

Table 12 shows the numerical results of dynamic cryptography evaluation followed by the cases figures. This evaluation dealt with files only.

Table 12. Dynamic cryptography and clustering aspect with files, OOP and AOP performance

No.Files	A		B		C		D		E	
	OOP	AOP	OOP	AOP	OOP	AOP	OOP	AOP	OOP	AOP
50	1411.5	1609	2073.5	2310.5	3460.5	3919.5	4434	5005	7549	8121.5
100	2759	3153	4011	4572	6970	7748	8711	9987	14982	16030
200	5406	6208	8038	9246	13670	15258	17504	19544	29660	31740
400	10584	12100	15744	17744	26880	30052	34752	38140	58692	63516
800	20624	23408	30680	33832	53552	58552	68104	74904	116784	125112

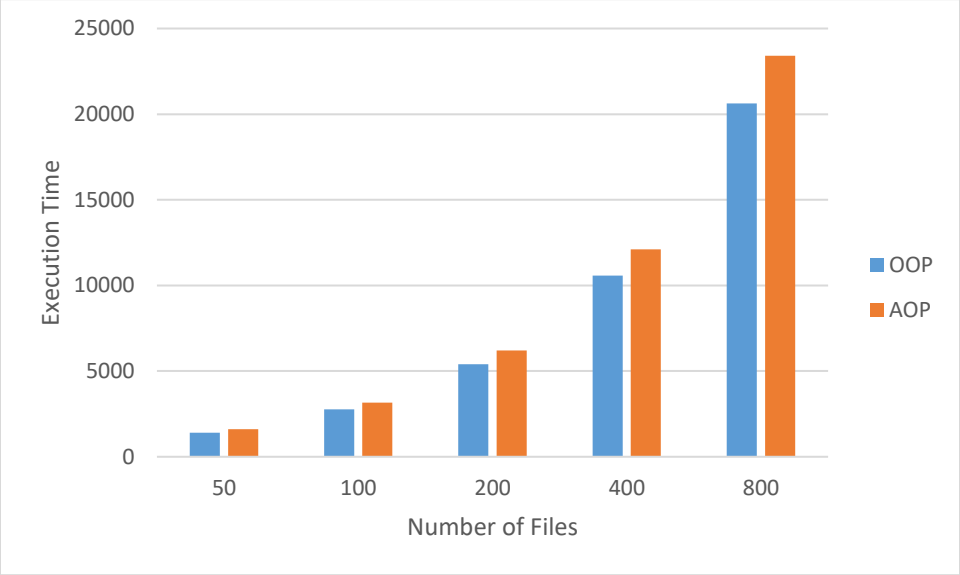


Figure 6.98.Dynamic cryptography aspect, File Level, CASE "A".

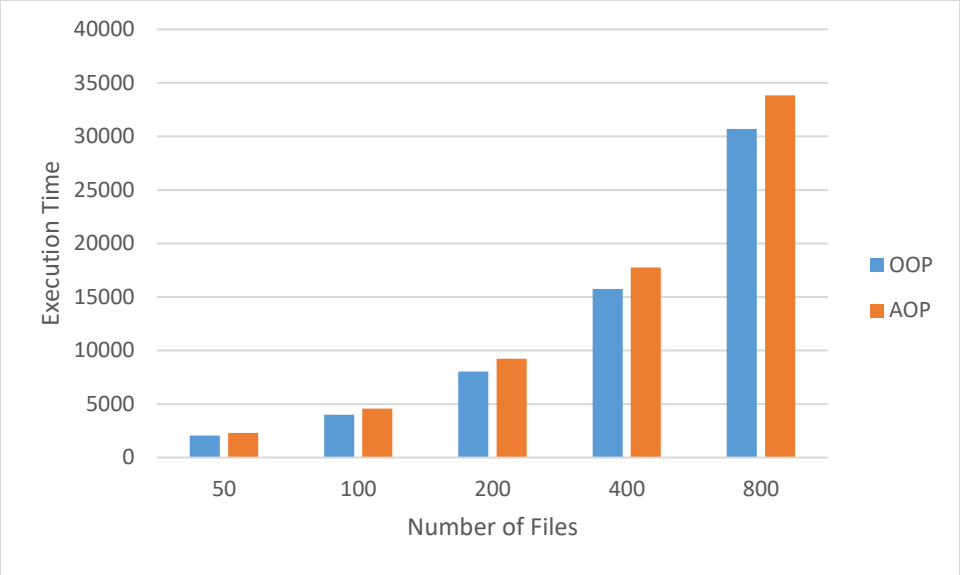


Figure 6.99. Dynamic cryptography aspect, File Level, CASE "B".

Figure 6.100. Dynamic cryptography aspect, File Level, CASE "C".

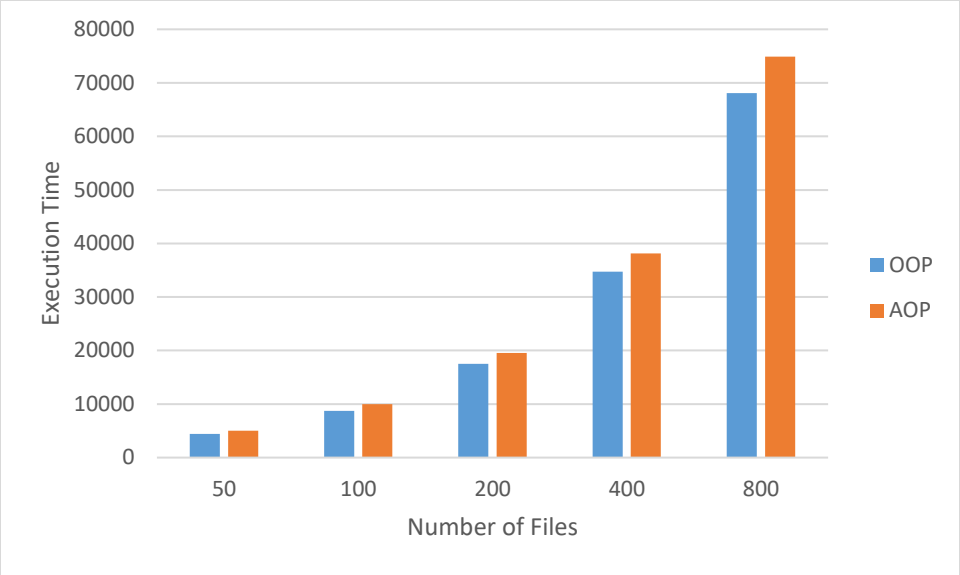


Figure 6.101. Dynamic cryptography aspect, File Level, CASE "D".

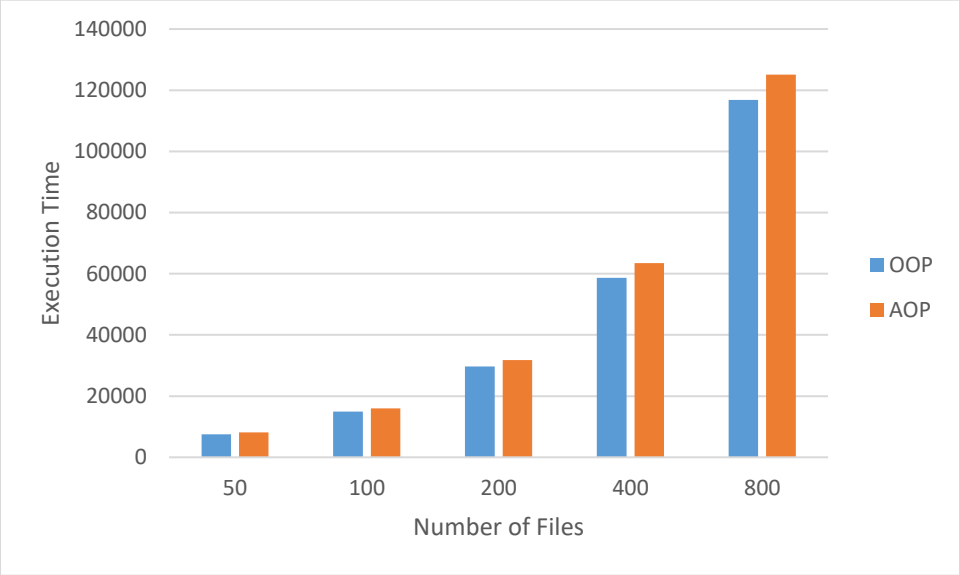


Figure 6.102. Dynamic cryptography aspect, File Level, CASE "E".

6.5 3AC_AOP Security Guard

This section deals with all the 3AC_AOP's components, access control model, cryptography and AOP security guard. In spite of the fact that the access model and cryptography have created a secure system, the cooperation between the system's

nodes are still restricted. In another meaning, all the previous evaluations have considered that the information (file and messages) are received and processed by the receipt nodes if the security clearance level of receipt nodes are greater than or equal to the source node level. Otherwise, the processing is declined and sent to the next node. This section has presented the numerical evaluations based on the domination relationship, in which the source node (sender) has the ability to ask the intended node to perform its own function even if the security clearance of the intended node is lower than that of the source node level. This is done, after allowing the receipt node to handle only the authorized portion (sanitized portion) which was classified with a level equal to the intended node's level. As we mentioned in chapters 4&5, the sanitization process with messages will be applied horizontally, while with files will be applied vertically.

In order to ensure the fairness in the proposed model's evaluation as well as trying to include various cases as much as possible, we have used the model to deal with two different decentralized distributed systems, ring connection see Figure 6.55 and random connection (internet setup) see Figure 6.103. With the latter, execution time will include the time of the sort path algorithm which been used with this setup.

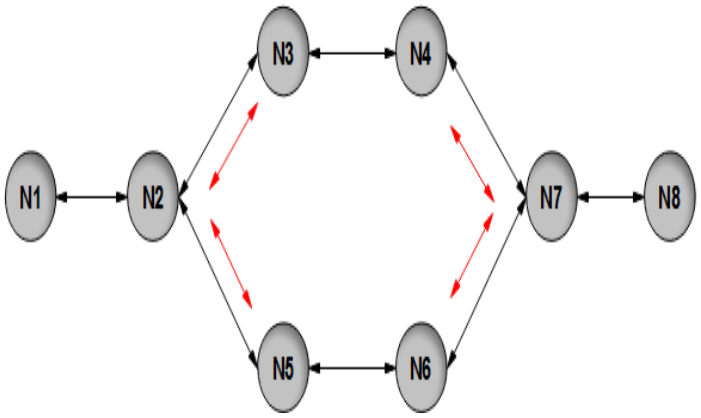


Figure 6.103. Random connection decentralized distributed system

1. Message Experiments

We discuss the case in which the source node sends messages to different nodes simultaneously. The receipt nodes might have security clearance lower than the source node. Indeed, the IBAC will play a significant role when the source node identifies the intended nodes according to the attributes and identities regardless of the security level. AOP security guards will be responsible for ensuring that the receipt information will be sanitized from any sensitive information and the intended nodes will receive only the messages that been classified with the same intended nodes levels.

In this evaluation we will use a different comparison table and we will include 4 cases only, 2 cases with ring connection and 2 for random connection as shown in Table 13.

Table 13. Ring and random connections sanitization cases, Messages level

Case	Request	Result
Ring connection		
A	*Source=N1* *Sec.Clearance= 4* *Receipt nodes=* $\{At2, At4, At6\}$ *Sec.Clearance= { 3, 4,2 }*	*R={N1,N2,N3,N4,N5,N6}* *P={N2,N4,N6}* ***hops=2*** ***En=5, Dec=5, Sanitization=2***
B	*Source=N1* *Sec.Clearance= 4* *Receipt nodes=* $\{At3, At5, At7, At9\}$ *Sec.Clearance= { 3, 4,2,1 }*	*R={N1,N2,N3,N4,N4,N5,N6,N7,N8,N9}* *P={N3,N5,N7,N9}* ***hops=4*** ***En=8, Dec=8, Sanitization=3***
$Exexution\ time = \sum_{i=0}^{\lambda-1} \omega + \sum_{i=0}^{\lambda-1} AC + \sum_{i=0}^{\rho-1} Crypto + \sum_{i=0}^{\Omega-1} santi \ldots\ldots \exp(5)$		
Random Connection		
C	*Source=N1* *Sec.Clearance= 4*	*R={N1,N2,N3,N4,N7,N8,N7,N6}*

		P={N2,N3,N6} ***hops=5*** ***En=7, De=7, Sanitization=2***
	Receipt nodes= $\{At2, At3, A6\}$ *Sec.Clearance= { 4, 2,2 }*	
D	*Source=N1* *Sec.Clearance= 4* *Receipt nodes=* $\{At2, At7, A8\}$ *Sec.Clearance= { 4, 3,2 }*	*R={N1,N2,N3,N4,N7,N8 }or { N1,N2,N5,N6,N7,N8* *P={N2,N7,N8}* ***hops=4*** ***En=5, De=5, Sanitization=2***
$Exexution\ time = \sum_{i=0}^{\lambda-1} \omega + \sum_{i=0}^{\lambda-1} AC + \sum_{i=0}^{\rho-1} Crypto + \sum_{i=0}^{\Omega-1} santi + \sum_{i=0}^{\lambda-1} ShortPath \ldots\ldots \exp(6)$		

Table 14 shows the execution time results for both cases with message level as follows:

Table 14. Execution Time Result for Ring and Random Connections with AOP Guard. Message level

No.Msg	Ring Connection				Random Connection			
	A		B		C		D	
	OOP	AOP	OOP	AOP	OOP	AOP	OOP	AOP
5000	**63500**	**87400**	**75960**	**93800**	**85600**	**104300**	**102800**	**131000**
10000	**119480**	**158300**	**133800**	**168900**	**148800**	**185400**	**183700**	**233400**
20000	**205600**	**262000**	**243600**	**276600**	**271300**	**336000**	**339700**	**425300**
40000	**374600**	**436000**	**436600**	**498000**	**525900**	**610500**	**623200**	**786100**

The next figures show the evaluation results for each case in Table 14

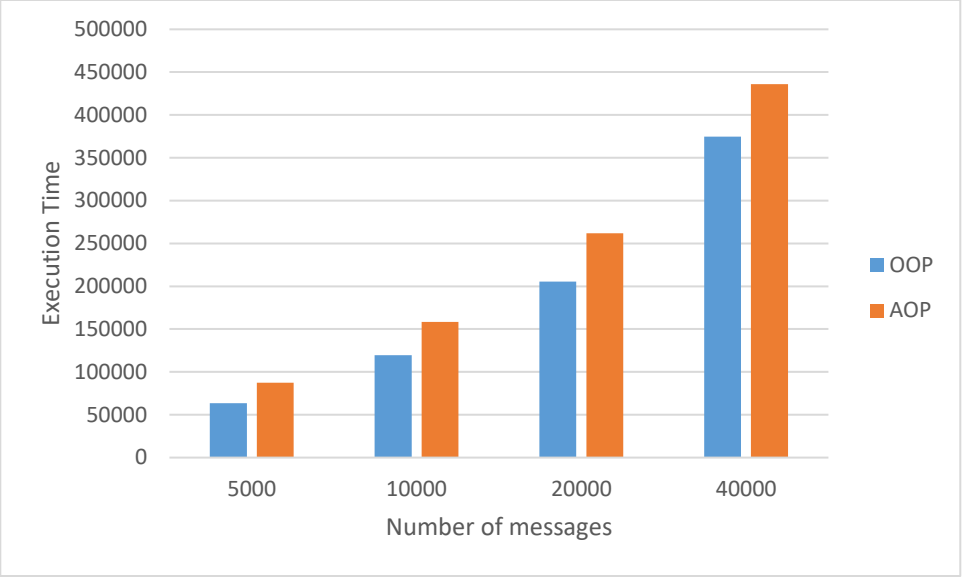


Figure 6.104. Evaluation of Sanitization aspect, Ring Connection, message level, CASE "A"

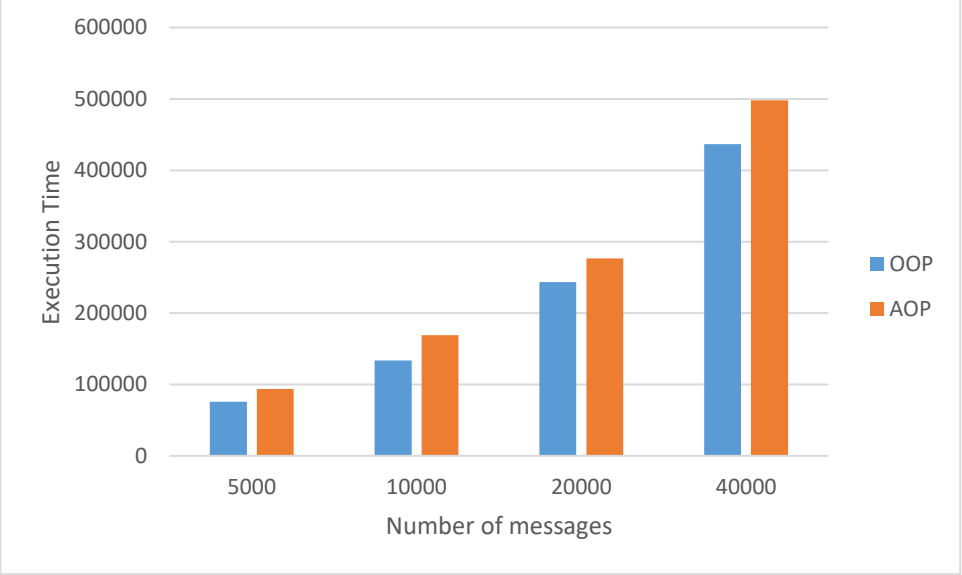


Figure 6.105. Evaluation of Sanitization aspect, Ring Connection, message level, CASE "B"

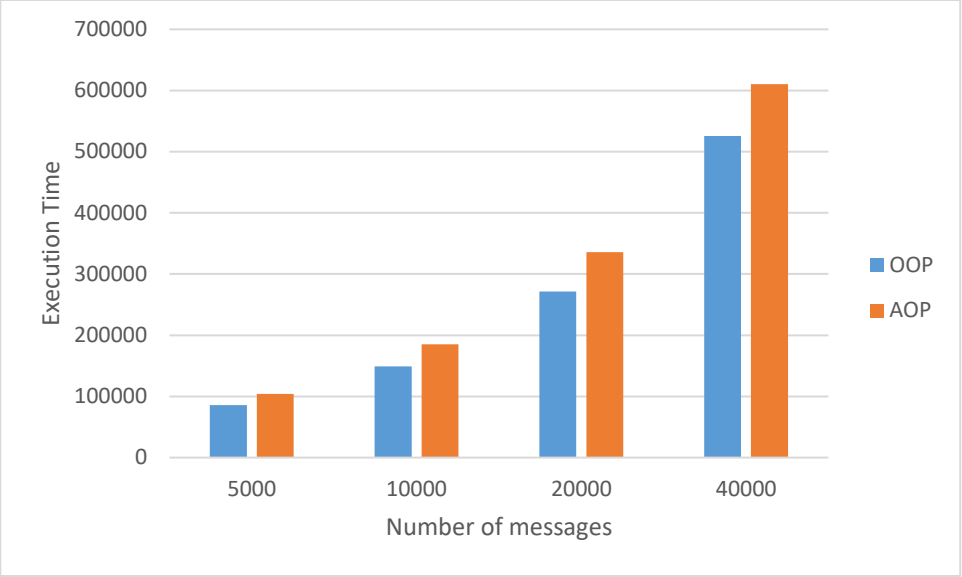


Figure 6.106. Evaluation of Sanitization aspect, Random Connection, message level, CASE "C"

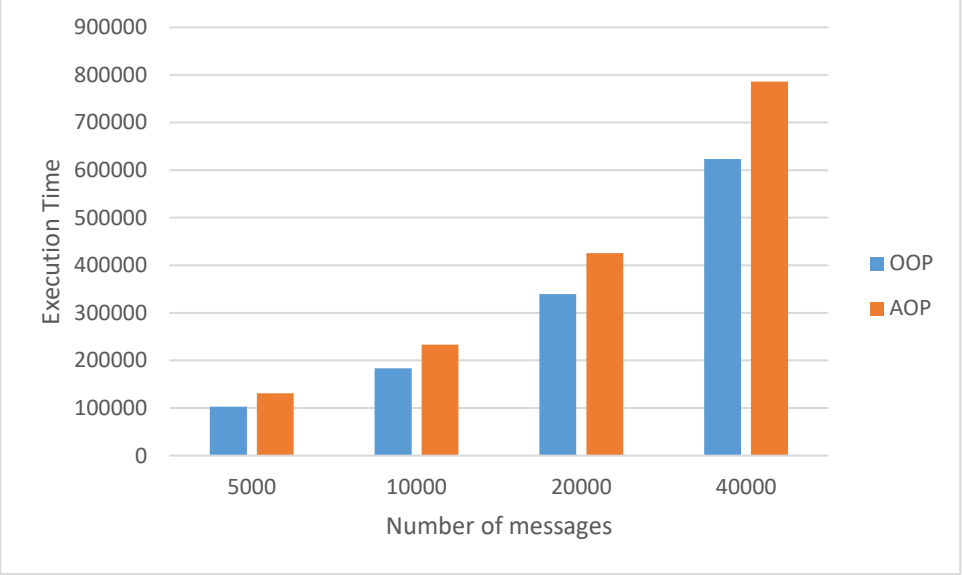


Figure 6.107. Evaluation of Sanitization aspect, Random Connection, message level, CASE "D"

2. File Experiments

With the file level, the source node will send the file(s) to the intended node according to their attributes to implement the required functions on the authorized portion only, whilst, the rest of the file will be encrypted. In this case the source node will receive

the processed response after handling the file in different security clearance nodes. The execution time here will include replay time (only hops time). Table 15 discusses four cases of evaluation scenarios, the first two deal with ring connections and the last two cases deals with random connections.

Table 15. Ring and random connections sanitization cases, Files level

Case	Request	Result
Ring connection		
A	*Source=N1* *Sec.Clearance= 4* *Receipt nodes=* $\{At2, At4, At6\}$ *Sec.Clearance= { 3, 4,2 }*	*R={N1,N2,N3,N4,N5,N6}* *P={N2,N4,N6}* ***hops=8*** ***En=6, Dec=6, Sanitization=2***
B	*Source=N1* *Sec.Clearance= 4* *Receipt nodes=* $\{At3, At5, At7, At9\}$ *Sec.Clearance= { 3, 4,2,1 }*	*R={N1,N2,N3,N4,N4,N5,N6,N7,N8,N9}* *P={N3,N5,N7,N9}* ***hops=14*** ***En=9, Dec=9, Sanitization=3***
$Exexution\ time = \sum_{i=0}^{\lambda-1} \omega + \sum_{i=0}^{\lambda-1} AC + \sum_{i=0}^{\rho-1} Process + \sum_{i=0}^{\rho-1} Crypto + \sum_{i=0}^{\Omega-1} santi \ldots\ldots \exp(7)$		
Random Connection		
C	*Source=N1* *Sec.Clearance= 4* *Receipt nodes=* $\{At2, At3, A6\}$ *Sec.Clearance= { 4, 2,2 }*	*R={N1,N2,N3,N4,N7,N8,N7,N6}* *P={N2,N3,N6}* ***hops=12*** ***En=7, De=7, Sanitization=2***
D	*Source=N1* *Sec.Clearance= 4* *Receipt nodes=* $\{At2, At7, A8\}$	*R={N1,N2,N3,N4,N7,N8 }or { N1,N2,N5,N6,N7,N8* *P={N2,N7,N8}* ***hops=9***

	Sec.Clearance= { 4, 3,2 }	***En=5, De=5, Sanitization=2***
$Exexution\ time = \sum_{i=0}^{\lambda-1} \omega + \sum_{i=0}^{\lambda-1} ShortPath + \sum_{i=0}^{\lambda-1} AC + \sum_{i=0}^{\rho-1} Process + \sum_{i=0}^{\rho-1} Crypto + \sum_{i=0}^{\Omega-1} santi\ .\exp(8)$		

Table 16 shows the execution time results for both cases as follows:

Table 16. Execution Time Result for Ring and Random Connections with AOP Guard. File Level

No.Msg	Ring Connection				Random Connection			
	A		B		C		D	
	OOP	AOP	OOP	AOP	OOP	AOP	OOP	AOP
50	**4232**	**6940**	**6782**	**9130**	**5302**	**8851**	**4680**	**6261**
100	**8410**	**11373**	**12345**	**16473**	**9780**	**14531**	**8936**	**10402**
200	**17693**	**22174**	**21463**	**27614**	**19406**	**25420**	**16730**	**19830**
400	**32466**	**36598**	**36410**	**43153**	**35572**	**44941**	**31210**	**35571**

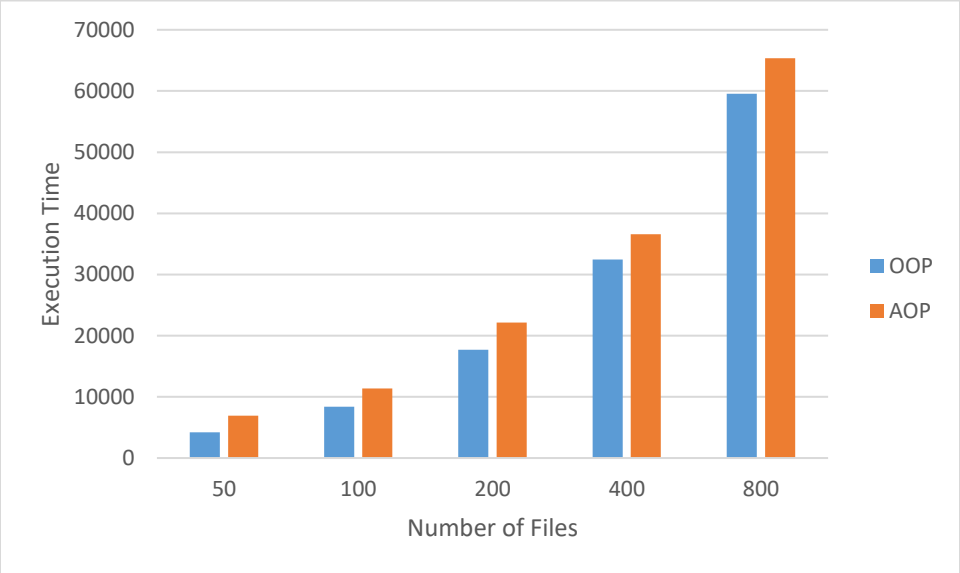


Figure 6.108. Evaluation of Sanitization aspect, Ring Connection, file level, CASE "A"

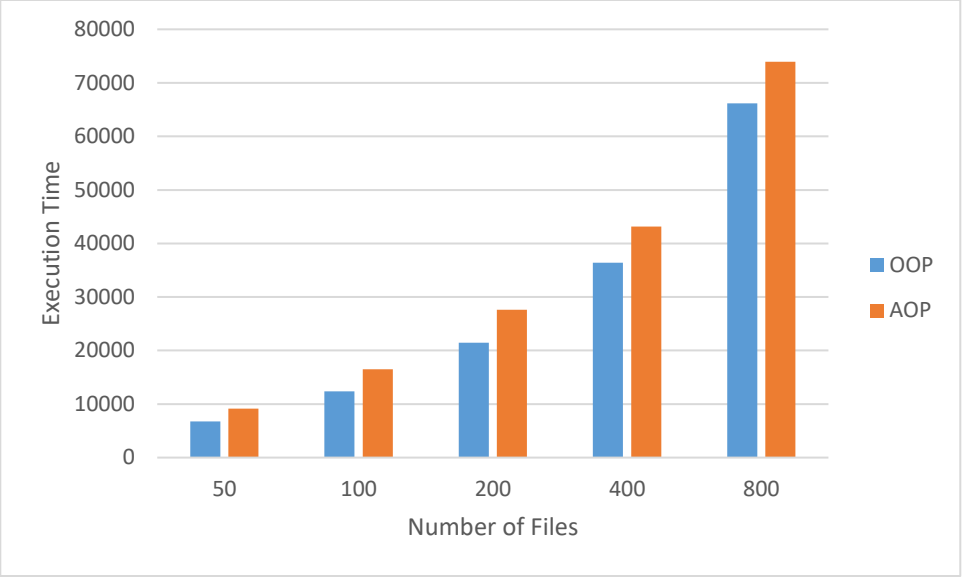


Figure 6.109. Evaluation of Sanitization aspect, Ring Connection, file level, CASE "B"

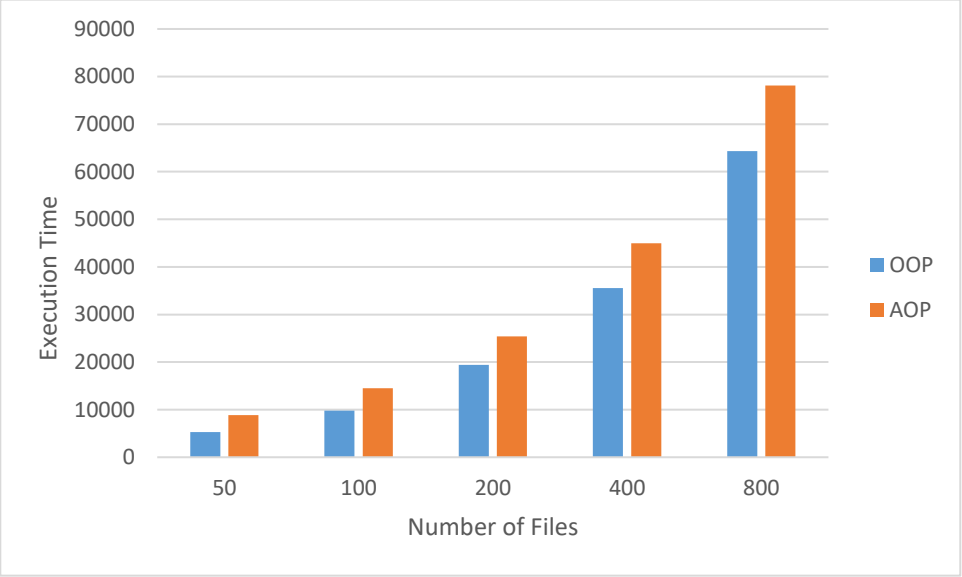


Figure 6.110. Evaluation of Sanitization aspect, Random Connection, file level, CASE "C"

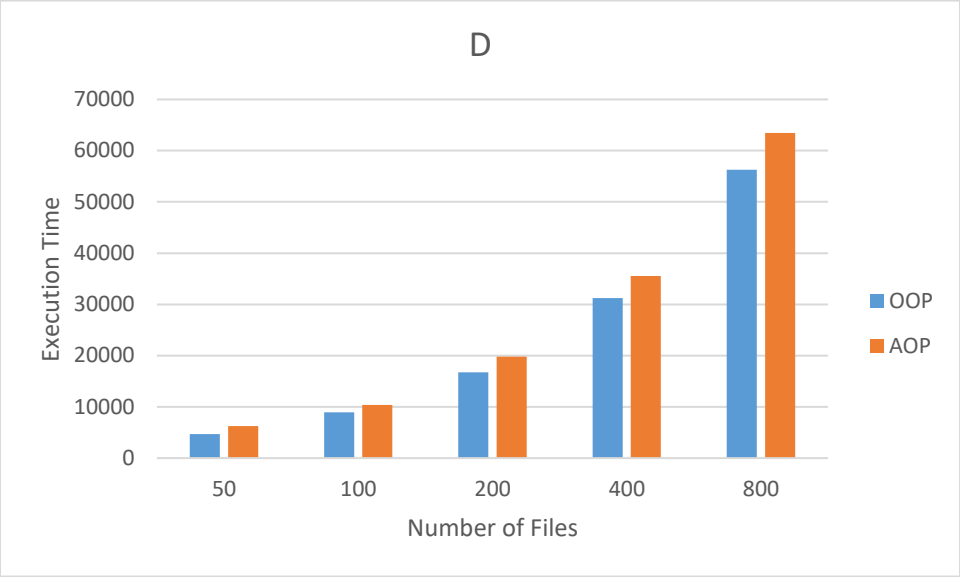


Figure 6.111. Evaluation of Sanitization aspect, Random Connection, file level, CASE "D"

6.6 Comparing with existing solution

Although many solutions have adopted AOP based on modularizing various access control models as aspect pointcuts and advices, these solutions lack idealism in taking into account the unexpected possibilities which may occur during real time and related to the nature of the environments used like distributed systems. After reviewing most of the past and current solutions, we find that DSAW [94] is the closest solution to our proposed solution, because most other solutions utilized AOP to deal with access control and cryptography between two nodes of the centralized distribution system and the solution will be coordinated and controlled by the central node (server). In spite of, reference [94] has given a problem case which used four nodes, however, their numerical evaluations used only three nodes, source, destination, and one intermediate node and dealing with FTP client and server which is connected between only two nodes. In order to get a fair comparison, we will apply their solution and ours on their problem case (4 nodes) as shown in Figure 6.112. We will start with the solution

algorithms and runtime performance calculation; to show the power of the proposed solution we will dealing both messages and files.

6.6.1 Algorithms Comparison

DSAW Solution	3AC_AOP
1- Tag and encrypt message	1- Tag and encrypt message/file
2- While (Node $_i$!=destination)	2- While (Node $_i$!=destination)
3- Decrypt message	3- Decrypt Attribute,ID,SL
4- If (SL_{Node_i} > message.Tag) then	4- If (true) Decrypt, granted, process, encrypt
5- Granted else declined	9- Else Move to $Node_{i+1}$
6- Encrypt message	5- End While
7- Move to $Node_{i+1}$	6- Mover
8- End While	

Figure 6.112. Algorithms Comparison.

As illustrated in Figure 6.112 the 3AC_AOP, firstly, decrypt the attributes, ids and security level to check the access conditions; if it matches then the next step is to decrypt the file or message, otherwise the processing does not need to decrypt the file each time. However, with DSAW solution, encrypting-decryption processes will be repeated each time before reaching and leaving a node even though it is not the authorized node. Obviously, DSAW injects information classification with the node clearance. However, in our solution we separated information security classification and node clearance.

6.6.2 Runtime Performance

Although, the numerical evaluation of DSAW adopted only messages as transmitted information between system nodes, our solution adopted both messages and files. For this reason, the time needed to implement the task will take into account both messages and files.

1. Message Experiments

With messaging, the performance time will be the time that is needed to accomplish the task starting from sending by the source node and finishing after receiving by the destination node. The relation here is between two different nodes (source and destination) as shown in the next formalization

A - DSAW

$$Exexution\ time = \sum_{i=0}^{\lambda-1} \omega + \sum_{i=0}^{\lambda-1} AC + \sum_{i=0}^{\lambda-1} Crypto$$

B- 3AC_AOP

See Exp (5)

As illustrated above, with SDAW, the processing needs to encrypt and decrypt a whole message at each node, however with our solution the access conditions only will decrypt at each node while the message is still in an encrypted form and will not decrypt only the access control granted the accessing.

2. Files Experiments

With the file level, there are two expected cases, the first, when the source node sends file(s) to the destination, in this case the execution time will be the same as with the message level. The second case is when the source node sends a request as a file and waits for the responded (processed file). In this case, the relation will be related to the same node. In both message and file levels, the solution deas with end-to-end in addition to point-to-point security. The following formula is linked with the second case

A- DSAW

$$Exexution\ time = 2 * (\sum_{i=0}^{\lambda-1} \omega + \sum_{i=0}^{\lambda-1} AC + \sum_{i=0}^{\lambda-1} Crypto\ )$$

B- 3AC_AOP

See Exp(7).

As with the message, our solution will encrypt and decrypt the accessing condition parameters while the file is still in an encrypted form until it reaches the intended node. The final result will return back to the source node as responded to the request, in this case the last processing node will encrypt the file and send it back to the source node. Throughout, the returning way the process will find there are no more attributes or functions needed to be dealt with, so all nodes will work as a bridge until reaching the source node, for this reason we use z rather than z-1. Whilst, with SDAW, as with the message, the file will be encrypted and decrypted at each hop as well as this processing being repeated after the end of the processing in the destination node. For this reason we multiply the formula by 2.

3. File Integrity.

As explained above, our solution will not grant any access to the information both in message and files level. This is done by, keeping the information in an encrypted form as well as adding AOP security guard to ensure it will not spill any sensitive information to unauthorized nodes, in which each node is granted to do a required process on a specific part of the file while the rest of the file will still be in an encrypted form. With DSAW, there is a big opportunity to release sensitive information by:

- Encrypting /decrypting operation which needs to repeat at each node even it not the intended node. This will give a chance to attack the information during this processing.
- Injecting the security tag directly to the information, thus needing to decrypt the information each time to check the security tags with the node clearance.

DSAW	3AC_AOP
Inject security tag with the message	Classified file by security tag, send access condition as a separate concerns
Encrypt/ decrypt process apply on each node between the source and destination	Encrypt/ decrypt process apply only if the access control meet the condition
Information spill is high	Information spill is too low
Need TCP	Don't Need TCP
Non dynamic information classification	Dynamic classification
No domination relationship	Domination relationship
No security guard	AOP security guard

6.6.3 Numerical Evaluation Comparison

As mentioned before, the comparison study in this section used the example case of adopting four nodes and each node has a security clearance as {Node1-Confidential, Node2-Secret, Node3-TopSecret, Node4-Confidential). The evaluation will use messages and files as transmitted information between system nodes.

1. Message Experiments

Figure 6.113 shows the processing case, -a- is DSAW processing while –b- is the 3AC_AOP solution. The task is to send message(s) from Node1 to Node4. With –a- the processing required 3 Enc/Dec with message to complete the task. However, with –b- only 1 Enc/Dec operation need with the message while 3 Enc/Dec with the attributes and security level.

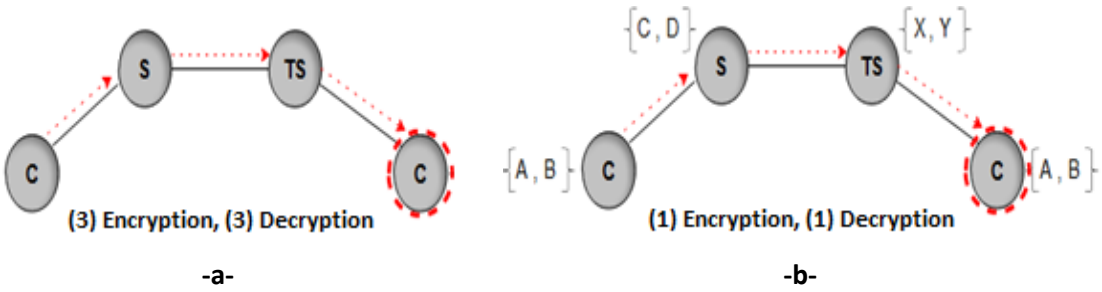


Figure 6.113. Message Comparison, -a- SDAW, -b- 3AC_AOP.

Table 17 shows the runtime performance for both cases, followed by Figure 6.114.

Table 17. Evaluation Result of comparison study, Message Level

No.Msg	3AC_AOP	DSAW
5000	19300	25200
10000	37600	48800
20000	69300	84200
400000	131800	159200

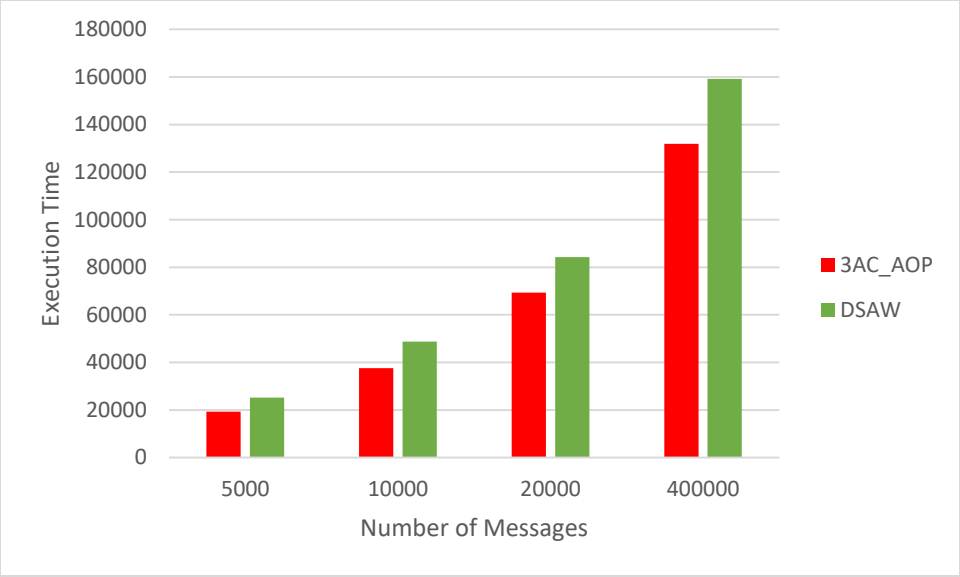


Figure 6.114. Evaluation Result, Massage Level

2. File Experiments

The evaluation result with files has two forms. The first one, when the processing sends the file to the destination without waiting for a reply as shown in Figure 6.115.-a- while –b- evaluated the case that the source node will wait for the response after processing the file in the intended nodes. Because file will travel through different security levels nodes, we will apply sanitization AOP guard to ensure the high integrity of data.

CASE 1

Figure 6.115 shows case of sending file(s) from Node1 to Node4.

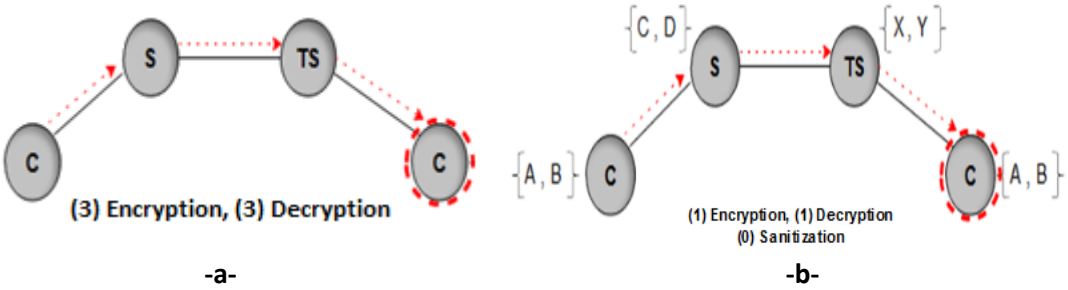


Figure 6.115. File Comparison, -a- SDAW, -b- 3AC_AOP.

With –a- the file needs the 3 Enc/Dec operations to finish the task, however with –b- the source node sends the file to the node only has {A,B} as required attributed, while with Node2 and Node3 the process will Enc and Dec the attributes, if not matched then the file will travel to the next node until reach to Node4. In this node the accessing conditions will grant the access after matching the node's attributes. In this case only the file will be decrypted to the node. Table 18 shows the evaluated results and presents the difference in runtime performance between the two solutions

Table 18. Runtime Performance, File level, Case1

No.Files	3AC_AOP	DSAW
50	1921	2782
100	3525	4760
200	6863	8218
400	12342	15542
800	20368	26587

Figure *6. 1116 shown* the evaluated result between the two solutions

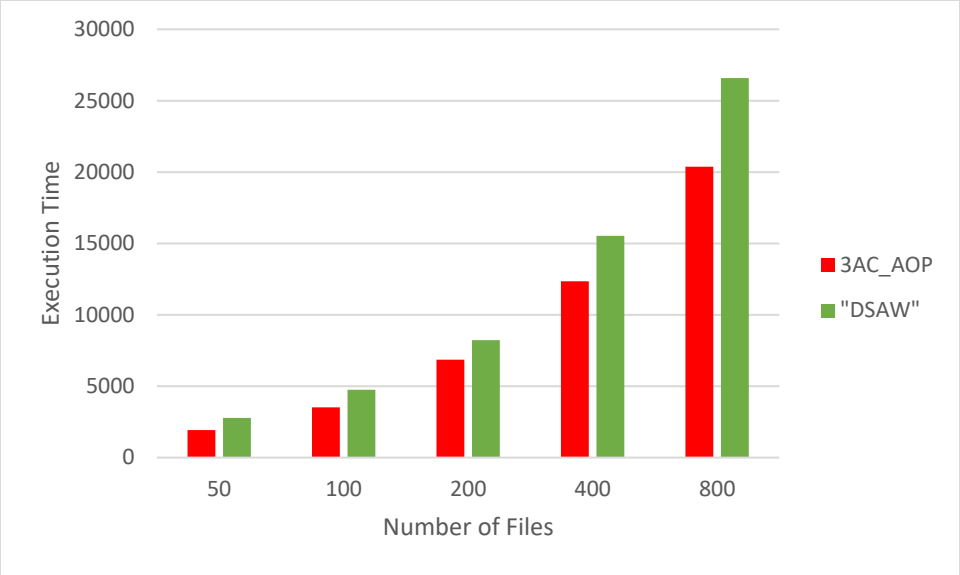
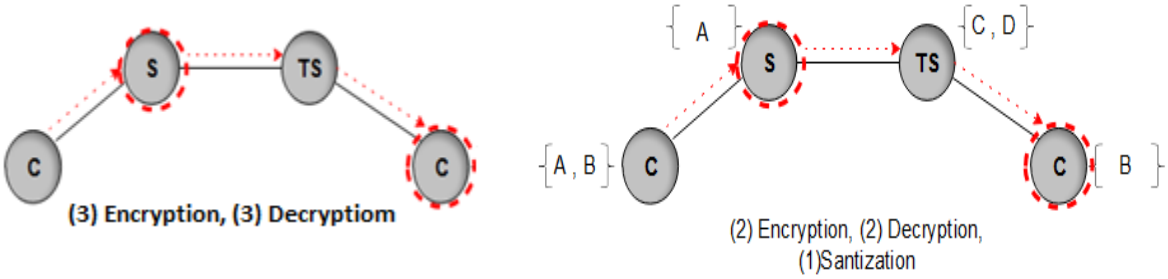


Figure 6.116. Runtime Performance, File Level, Case1

CASE 2

Figure 6.117 shows the case of sending file(s) from Node1 to Node4. The processing here needs to do an intermediate operation on the file before sending it to Node4. With –a- the file needs 3 Enc/Dec operations to finish the task, however with –b- only 2 Enc/Dec operations are needed and this is because, Node1 sends the file with {A,B} attributes. The Attribute {A} is in Node2 so the file will encrypt in Node2 to implement its own function and only to the authorized portion, the result file will re encrypt and send it to the next node. In our case the file has been processed in a node which has a security clearance greater than source node level. For this reason if the next processing or destination node has a security clearance lower than the latter node, then AOP security guard will face this case by sanitizing the receipt information to ensure not spilling any sensitive information.



-a- **-b-**

Figure 6.117. File Comparison, -a- SDAW, -b- Proposed solution, CASE 2

Table 19 shows the runtime performance result of file level case1, followed by the

Table 19. Runtime Performance, File level, Case2

No.Files	3AC_AOP	DSAW
50	2302	2782
100	4228	4760
200	7680	8218
400	13800	15542
800	23564	26587

Figure 6.118 shows the evaluated result between the two solutions

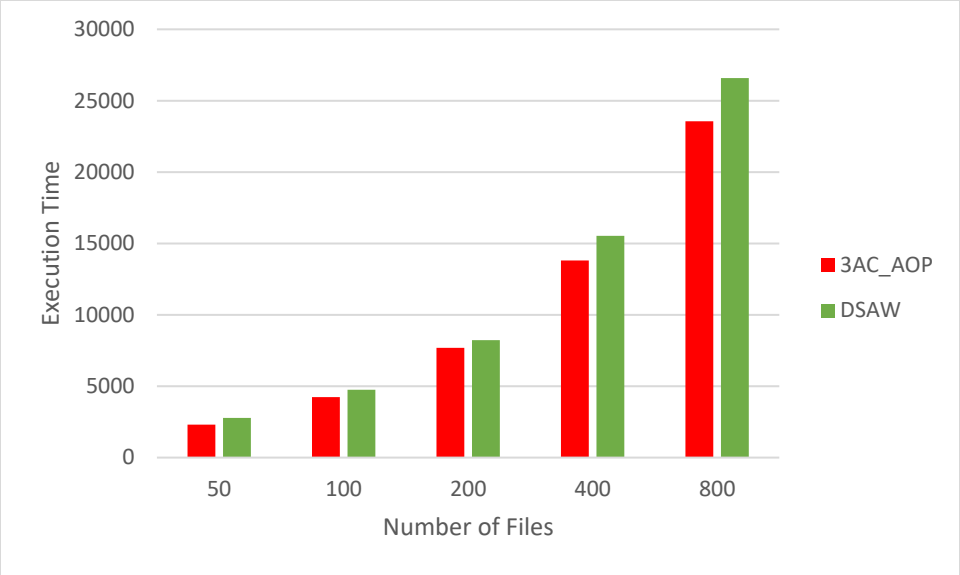


Figure 6.118. Runtime Performance, File Level, Case2.

CASE 3

This case discusses when Node1 sends a request to Node4 to handle a file and is waiting for a response. The file will travel in two directions one for sending the request and the second for receiving the response. With SDAW,-a- in Figure 6.119 Node1 sends a file to Node4 to perform its own function and send the result to Node1. This operation needs 6 Enc/Dec operations for sending and receiving. While in case –b- we just need 2 Enc/Dec operations with files and 0 sanitization because the file does not need any intermediate processing.

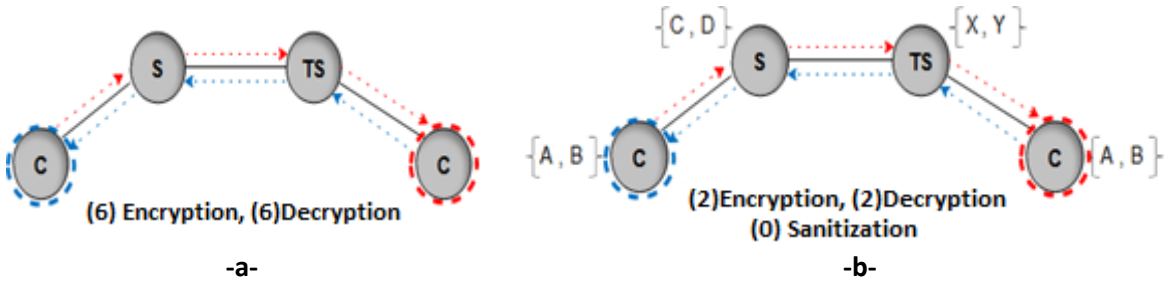


Figure 6.119. File Comparison, -a- SDAW, -b- Proposed solution, CASE 3

Table 20. Runtime Performance, File level, Case3

No.Files	3AC_AOP	DSAW
50	2011	5100
100	3860	9720
200	7020	14930
400	12980	25800
800	22150	46320

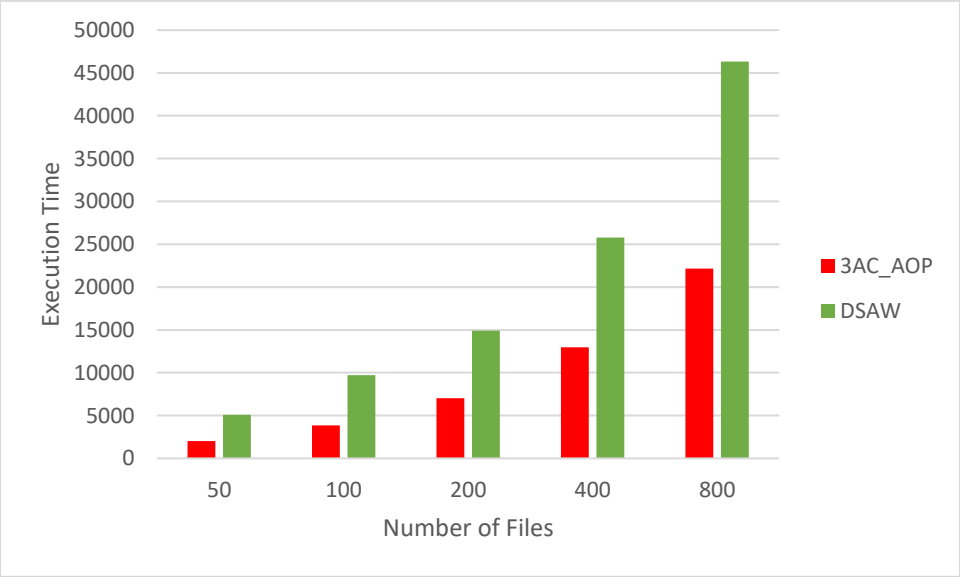


Figure 6.120. Runtime Performance, File Level, Case3.

CASE 4

With this case, Node1 send a file to destination Node4. This file needs two intermediate processes before being delivered to Node4, in Node2 and Node3 as shown in Figure 6.121. In case –a- (DSAW) the processing will be done like previous cases. However, in case –b- the processing required 3 Enc/Dec operations as well as 2 sanitization processes. This is because the file will be transformed from Node2 as TopSecret to Node3 as Secret and from Node3 to Node4 as confidential. So, the process

has to transmit from high to low. Table 21 shows the runtime performance of DSAW is better than the proposed solution due to extra sanitization processing.

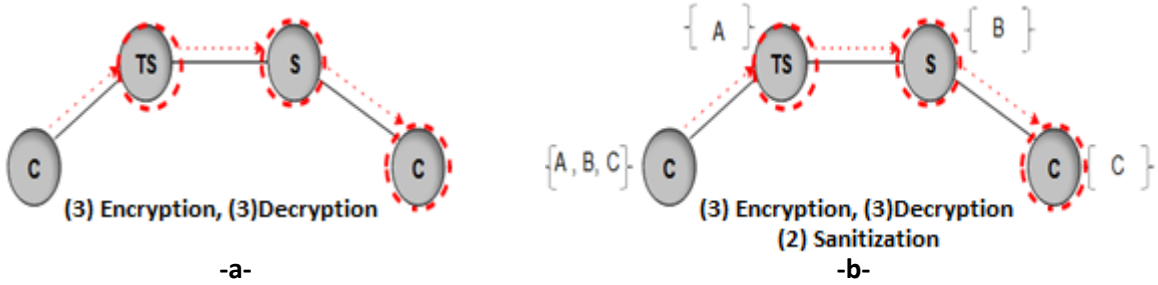


Figure 6.121. File Comparison, -a- SDAW, -b- Proposed solution, CASE 4

Table 21. Runtime Performance, File level, Case4

No.Files	3AC_AOP	DSAW
50	3010	2782
100	5220	4760
200	8995	8218
400	16240	15542
800	27980	26587

Figure 6.122.Runtime Performance, File Level, Case4.

6.6.5 Comparison Summary

According to the numerical evaluation results for four different experiments, the proposed model has shown its efficiency of runtime performance and information integrity. To ensure the fairness of the comparison, DSAW outperforms on the proposed model only if the message length is shorter than the required attributes as is also the case in experiment5. However, on the other side, information integrity is greater than DSAW because the process in the proposed solution does not need to perform Enc/Dec each time on file or message, but with DSAW these operations are repeated on each node even if the information is transmitted from high level to low. This case opens the opportunity for the lower nodes to display some sensitive information form highest nodes especially if the highest nodes are complicit with the lower level nodes. Indeed, to establish a distributed system working according to the DSAW solution, we need to use Trust Computer Based (TCB) see section 2.7.7. Whilst, AOP security guard in the proposed solution will be responsible for addressing all these issues and controlling the access to the authorized portion only.

6.7 Summary

This chapter has provided the numerical evaluation of the proposed system. The outcome of this evaluation showed the power of the solution and how it can be adopted to develop a security system in an easier and trusted way. To ensure fairness, the evaluation is based on comparing between process performance by developing a security policy in both OOP and AOP paradigms. We have tried to include all possibilities which may appear during running time or even before initiating the process. Evaluation of the proposed access control model, started with ABAC and MLS, followed by ABAC, IDAC and MLS. The consequences were highly efficient with a difference range between $\sim 0.1 - \sim 0.2$ with both message and file levels, and this is relevant to perfect usage and applying AOP pointcuts to the original codes to use the entire methods with tiny changes. However, performance difference has changed obviously, when adding cryptography as security aspect shielding so we dealt with this case by employing AOP characteristics to create a dynamic clustering of system nodes. This decreased the gaps between the different performance times for OOP and AOP by applying encryption/decryption processing only when the information travels

between different clusters or the cluster security clearance is lower than the source node clearance. The third section of this chapter presented our novel security model by utilizing AOP to work as a bidirectional automatic security guard. Multi-Level security domination relationship was adopted to allow a node with high security clearance to dominate other nodes lower than or equal to it. Thus, the highest nodes can ask the lower ones to do a requested task in a secure and private manner. Finally, this chapter provides, a comparison study between the proposed solution and other existing solutions. The outcome results show the power of the proposed solution through fulfilling a high level of security and privacy with high flexibility, manageability and scalability.

Chapter 7
Conclusion and Future Work

Achieving security and privacy concerns with all various requirements remains a major challenge which needed to be faced by software programmers and developers. This thesis has focused on the security and privacy concerns in decentralized distributed systems to ensure safe communications and processing of data transmission between system nodes. Thus, it has achieved end-to-end security requirements as well as covered point-to-point security

3AC_AOP combined access control, cryptography, security guard and was developed using OOP and AOP to achieve DDS security. This thesis proves important points as follows: applying the proposed access control model's components individually cannot fulfill the security requirements of DDS. This is because each component focuses on certain aspects of security requirements. For instance, dealing with the main AC models in the 3AC_AOP (ABAC and MLS), using ABAC only, will ensure that the distribute data over nodes have the same attributes, but will not prevent data spilling from high to low. However, using MLS only will prevent sensitive data spilling, but it will add more restrictions on the system through restricting data flow between system entities. IBAC has been inserted as an intermediate level to enhance the performance of the proposed access control model by limiting the processing only to the intended nodes. Combining these models together will eliminate the individual weaknesses and achieving a high-security solution can deal with DDS.

Modularizing the access control model by using AOP has no significant impacts in terms of runtime performance comparing with OOP. This relies on the way that the modularized model is applied to the original code by ignoring calling the main method's signature (Sending and Receiving methods) and focusing only on java reserved methods that are used with both methods. Thus, it gives the priority to convert the access control model from traditional development using OOP to advance developments using AOP. Because, with AOP, the access control model will be separated from the original code of the system, it, therefore, increases the modularity

and manageability of the code and at the same time decreases the scattering and tangling problems.

With cryptography model, nevertheless, there are obvious differences between applying this solution using OOP or AOP statistically in which AOP needs more time to execute the task. However, the dynamic AOP cryptography enhanced the efficiency of using cryptography processes in DDS. This is done by applying dynamic clustering of system nodes to be clustered according to the security clearance of the node. Thus, it decreased the number of En/De processes between clusters, not nodes and the data will travel through a cluster without a need to encrypt/decryption operations. This will give the priority to use AOP to modularize the cryptography algorithm.

Finally, applying AOP security guard (which automatic software) is more advantageous than inserting hardware security devices or using human resources to monitor the data flow between system entities. AOP security guard will ensure monitoring data flow and allow high clearance nodes to do processing in low clearance one by sanitizing the information from any sensitive data.

In addition, applying the 3AC_AOP solution on DDS by using AOP has the advantage of comparison to apply the same approach using OOP. This is because unlike OOP, the solution uses AOP can be applied dynamically that any incident occurs during runtime does not need to switch off the node, but only updates the aspect code immediately according to the situation need. Hence, it prevents any disconnecting between system nodes during the runtime.

Through the experimental and evaluation chapter, it has been shown that using sequential processing to deal with the task that needs several processes to be accomplished will be better than dealing with this process using separation of duty. This is due to several reasons such as, the intermediate processing will receive the results of the previous processes and the final processing will hold the final results.

According to the comparison between 3AC_AOP with DSAW, it has been shown that 3AC_AOP is more efficient than DSAW through separating the file from access conditions. Therefore, accessing control model will be applied first if matching then

decrypting the file for processing otherwise keeps it in an encrypted form. Hence, this ensures file integrity as well as decreases the execution time.

7.1 Future work

The presented solution in this thesis has various domains to apply on e.g service function chain (SFC) and sequential processing. Our vision in future work for the proposed solution is as follows:

1- The proposed model could be developed to be a plugin software tool, which can be applied directly to the distributed system.

2- As we have seen in the literature review, some progress has been made on how to achieve security and privacy concepts by solving a specific problem, but questions remain especially about the generality of these solutions. Thus, the next future contribution will be how to deal with aspect advices when moving from a specialized to a more generalized approach. In other words, how to apply the aspect advice to existing applications depends on the functionality of the application rather than the specific implementation of the application code. This will enhance the generality concept to achieve the security and privacy principles in general not only for a specific problem or software implementation.

3- Implement and evaluate the proposed model in a real world distributed system environment e.g. cloud computing and web service compositions.

4- The proposed system can be enhanced to deal with the concurrence problem. This problem is a significant issue of the systems that deal with MLS. When the node needs to handle different information receipt from different resource at the same time

5- The proposed model could be extended to deal with different types of transmitted information e.g. audio and video.

6- The proposed model could be enhanced to include AOP alarm to warn the system in case of any types of illegal complicity between different nodes.

7- The proposed AOP security guard could be developed to filter and detect some types of malicious code that uses steganography and watermarking to hide dirty data during the processing or transmitting.

8- Evaluate the proposed model within LAN machines to check the CPU and power consumption of the machine e.g mobile based, raspberry pi machines and sensor network.

9- Design an application based on the proposed model to work within the enterprise, taking into account maintaining the enterprise hierarchy without the need to cluster nodes according to the level of security clearance.

10- The proposed model could be enhanced to develop a reputation system by utilizing AOP to monitor the communications channels and nodes of the distributing system to make a right decision about the reputation of the channels and system nodes.

11- In evaluation chapter we used AES 128 key size as a cryptography algorithm to ensure data integrity during transmitting between system nodes. However, this not prevent that we can use different other algorithms. In future, we will reevaluate the model using key size 256 or different algorithm in addition with asymmetric algorithms.

References

[1] G. Coulouris, J. Dollimore, T. Kindberg, and G. Blair, *DISTRIBUTED SYSTEMS Concepts and Design Fifth Edition*, Publisher Addison-Wesley, 2011.

[2] M. Firdhous, "Implementation of Security in Distributed Systems - A Comparative Study," *Int. J. Comput. Inf. Syst.*, vol. 2, no. 2, p. 6, 2011.

[3] K. Nadiminti, M. D. De Assunção, and R. Buyya, "Distributed systems and recent innovations: Challenges and benefits," *InfoNet Mag.*, vol. 16, no. 3, pp. 1–5, 2006.

[4] J. P. D. Comput, G. Tuna, D. G. Kogias, V. C. Gungor, C. Gezer, and E. Taşkın, "A survey on information security threats and solutions for Machine to Machine (M2M) communications," *J. Parallel Distrib. Comput.*, vol. 109, pp. 142–154, 2017.

[5] B. Zou, M. Yang, J. Guo, J. Wang, and E. Benjamin, "Progress in Nuclear Energy Insider threats of Physical Protection Systems in nuclear power plants : Prevention and evaluation," *Prog. Nucl. Energy*, vol. 104, pp. 8–15, 2018.

[6] G. Ihor Kuz, Felix Rauch, Manuel M.T Chakarvarty, "COMP9243 — Week 8 (08s1)," vol. 8, pp. 1–24.

[7] A. Aseeri and R. Hewett, "Alleviating Eavesdropping Attacks in Sofware-Defined Networking Data Plane," CISRC '17, Oak Ridge, TN, USA, ACM, pp. 0–7, 2017.

[8] CSP, "Denial of service attacks : what you need to know," (TLP White) 2014. https://www.ncsc.gov.uk/content/files/protected_files/guidance_files/Denial-of-service-attacks-what-you-need-to-know1.pdf

[9] M. Ghobaei-arani, S. E. Dashti, A. A. Rahmanian, and M. S. Aslanpour, "CSA-WSC : cuckoo search algorithm for web service composition in cloud environments," *Soft Comput.*, Springer, 2017.

[10] J. Andress, *The Basics of Information Security Understanding the Fundamentals of InfoSec in Theory and Practice*. Elsevier Inc, 2011.

[11] A. S. Hans Baars, Kees Hintzbergen, Jule Hintzbergen, *The Basics of Information Security- A Practical Handbook*. version 18g, ISBN/EAN: 978-90-813341-1-2, 2009.

[12] K. S. Trivedi, D. S. Kim, and A. Roy, "Dependability and Security Models *," *IEEE*, in Proc. D, pp. 11–20, 2009.

[13] S. Ross Anderson, John Wiley, *Security Engineering: A Guide to Building Dependable Distributed Systems, 2nd edition, Chapter5*, 2008.

[14] R. Needham and R. Maybury, *Access Control, in Security Engineering: A guide to building dependable distributed systems , Chaspter 4, University of Cambridge*. pp 93-128, 2008.

[15] D. F. Ferraiolo and R. Sandhu, “Proposed NIST Standard for Role-Based Access Control,” *ACM Trans. Inf. Syst. Secur.*, vol. 4, no. 3, pp. 224–274, 2001.

[16] P. Kamakshi and a. V. Babu, “Preserving Privacy and Sharing the Data in Distributed Environment using Cryptographic Technique on Perturbed data,” *J. Comput.*, vol. 2, no. 4, pp. 115–119, 2010.

[17] B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu, “Privacy-preserving data publishing: A Survey of Recent Developments,” *ACM Comput. Surv.*, vol. 42, no. 4, pp. 1–53, 2010.

[18] S. T. F. Al-janabi and N. A. Ali, “Hiding Sensitive Frequent Itemsets over Privacy Preserving Distributed Data Mining,” *Fifth Sci. Conf. Inf. Technol. 2012 Dec. 19-20*, vol. 10, no. 1, pp. 91–105, 2013.

[19] J. Krüger, “Separation of Concerns : Experiences of the Crowd,” *SAC 2018 Symp. Appl. Comput. April 9–13, 2018, Pau, Fr. ACM, New York, NY, USA, 2 pages. https//doi.org/10.1145/3167132.3167458 1*, 2018.

[20] The AspectJ Team, “The AspectJTM Programming Guide.” http://www.cin.ufpe.br/~if101/aspectj/aspectj1.0/doc/progguide.pdf.

[21] L. Zheng and S. Chong, “Using replication and partitioning to build secure distributed systems,” in *SP ’03 Proceedings of the 2003 IEEE Symposium on Security and Privacy*, 2003, pp. 236–250.

[22] Finjan Blog, “A Closer Look at Multilevel Lattice Security Models,” Finjan cybersecurity, 2017, https://blog.finjan.com/multilevel-lattice-security-models/.

[23] J. Rushby, “A Trusted Computing Base for Embedded Systems,” *7th DoD/NBS Comput. Secur. Conf.*, vol. 7, pp. 294–311, 1984.

[24] J. Cui, L. Shao, H. Zhong, Y. Xu, and L. Liu, “Data aggregation with end-to-end confidentiality and integrity for large-scale wireless sensor networks,” *Peer-to-Peer Netw. Appl. 11:1022– 1037, Springer*, pp. 1022–1037, 2018.

[25] O. K. Sahingoz, “Adding Secure Communication Mechanism to Existing Distributed Applications by Means of AOP,” in *Ubiquitous Information Technologies and Applications (CUTE 2012)*, 2013, vol. 214, pp. 1–9.

[26] M. V. S. Andrew S. Tanenbaum, *DISTRIBUTED SYSTEMS Principles and Paradigms*, 2nd ed. Prentice-Hall, Inc., Upper Saddle River, 2006.

[27] R. Wattenhofer, *Principles of Distributed Computing*. ETH Swiss Federal Institute of Technonlogy Zurich, 2016.

[28] D. T. Bourgeois, *Information Systems for Business and Beyond*. Washington: The Saylor Academy, 2014.

[29] J. P. L. Kenneth C. Laudon, *Management Information systems: Managing The Digital Firm*, Twelfth. 2012.

[30] R. J. Paul, “What an Information System Is , and Why Is It Important to Know This,” *J. Comput. Inf. Technol. 18(2)*, pp. 95–99, 2010.

[31] M. Hugoson, “Centralized versus decentralized information systems: A historical flashback,” *Impagliazzo, J., Järvi, T., Paju, P. (Eds.), Hist. Nord. Comput. (pp. 106-115). Ger. Springer.*, 2009.

[32] E. Greene, P. Proctor, and D. Kotz, “Smart Health Secure sharing of mHealth data streams through cryptographically- enforced access control,” *Smart Heal.*, no. December 2017, pp. 1–17.

[33] A. Harrington and C. D. Jensen, “Cryptographic Access Control in a Distributed File System,” *Proc. 8th ACM Symp. Access Control Model. Technol. (SACMAT 2003) . ACM, Villa Gall. Como, Italy.*, 2003.

[34] P. P. Hadke and I. Technology, “Use of Neural Networks in Cryptography : A Review,” *n Futur. Trends Res. Innov. Soc. Welf. (Startup Conclave), World Conf. IEEE*, pp. 1–4, 2016.

[35] H. C. A. Van Tilborg, “FUNDAMENTALS OF CRYPTOLOGY,” *New York, Kluwer Academic Publishers*, pp. 1–172, 2000.

[36] I. Journal, C. Applications, and E. H. No, “A Symmetric Key Cryptographic Algorithm,” *Int. J. Comput. Appl.*, vol. 1, no. 15, pp. 1–4, 2010.

[37] M. S. A. Arya, Prashant Kumar, “Comparative Study of Asymmetric Key Cryptographic Algorithms,” International Journal of Computer Science & Communication Networks, Volume 5, Issue 1, 2015 pp. 17–21.

[38] K. Krishnan, “Symmetric Key cryptosystem,” pp. 1–19, 2004, http://www4.ncsu.edu/~kksivara/sfwr4c03/lectures/lecture9.pdf

[39] E. Whitfield D Hellman, “New Directions in Cryptography,” *IEEE Trans. Inf. Theory*, vol. IT-22, no. 6, pp. 644–654, 1976.

[40] Hellman M E ,The mathematics of public-key cryptography, Scientific American 241 130-9,1979

[41] R. Kissel and M. Scholl, “Guidelines for Media Sanitization Guidelines for Media Sanitization,” *NIST Spec. Publ. 800-88, Sept. 2006, http//csrc.nist.gov/publications/nistpubs/800- 88/NISTSP800-88_rev1.pdf.*

[42] Z. Cai, S. Member, Z. He, X. Guan, Y. Li, and S. Member, “Collective Data-Sanitization for Preventing Sensitive Information Inference Attacks in Social Networks,” *EEE Trans. DEPENDABLE Secur. Comput.*, vol. 5971, no. c, pp. 1–14, 2016.

[43] B. NEWS, “BBC NEWS _ World _ Europe _ Readers ‘declassify’ US document.”2005, http://news.bbc.co.uk/1/hi/world/europe/4506517.stm .

[44] A. Division and I. A. Directorate, “Redacting with Confidence : How to Safely Publish Sanitized Reports Converted From Word to PDF CLASSIFICATION // X1,” vol. 6704, no. 410, 2005.

[45] V. o. Safonov, Using aspect-oriented programming for Trustworthy software development , Wiley & Sons (2008) .

[46] A. H. Rashid, N. Binti, and M. Yasin, “Privacy preserving data publishing : Review,” *Int. J. Phys. Sci.*, vol. 10, no. 7, pp. 239–247, 2015.

[47] C. Obimbo, “Privacy Preserving Data Publishing : A Classification Perspective,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 5, no. 9, pp. 29–34, 2014.

[48] B. C. M. Fung, M. Prefectural, W. Utilization, R. Chen, and P. S. Yu, “Privacy-Preserving Data Publishing : A Survey of Recent Developments,” *ACM Comput. Surv.. 42. 10.1145/1749603.1749605.*, no. May 2014, 2010.

[49] M. P. Khaled El Emam, Sam Jabbouri, Scott Sams, Youenn Drouet, “Evaluating Common De-Identification Heuristics for Personal Health Information,” *J Med Internet Res. 2006;8(4)e28.*, 2006.

[50] A. Anjum, N. Ahmad, S. U. R. Malik, S. Zubair, and B. Shahzad, “An efficient approach for publishing microdata for multiple sensitive attributes,” *J. Supercomput.*, Springer Science+Business Media, LLC, 2018.

[51] S. Mukherjee, “Should Non-Sensitive Attributes be Masked? Data Quality Implications of Data Perturbation in Regression Analysis,” *IEEE*, pp. 223–229, 1998.

[52] T. S. Gal, T. C. Tucker, A. Gangopadhyay, and Z. Chen, “A data recipient centered de-identification method to retain statistical attributes,” *J. Biomed. Inform.*, vol. 50, pp. 32–45, 2014.

[53] R. S. Sandhu and P. Samarati, “Access Control : Principles and Practice,” *IEEE Commun. Mag. (Sept.)*, pp. 40–48, 1994.

[54] L. Bauer and E. W. Felten, “A General and Flexible Access-Control System for the Web,” *11th USENIX Secur. Symp.*, August, 2002.

[55] L. Chen, “Analyzing and Developing by Role-Based Access Control Models,” Doctoral thesis, Royal Holloway, University of London, United Kingdom, 2011.

[56] A. Majumder, S. Namasudra, and S. Nath, “Taxonomy and Classification of Access Control Models for Cloud Environments,” *Contin. Rise Cloud, Mahmood Z (ed.). Springer, London*, pp. 23–33, 2014.

[57] R. Ausanka-crues, “Methods for Access Control : Advances and Limitations,” *arvey Mudd Coll. 2004. Retrieved December 07, 2012, from, http//www.cs.hmc.edu/wmike/public_html/ courses/security/s06/projects/ryan.pdf*.

[58] R. Thion, *Access control models*, Cyber Warfare and Cyber Terrorism , IGI-Global, pp.318–326, DOI: 10.4018/978-1-59140-991-5.ch037, 2008.

[59] V. C. Hu and R. Kuhn, "Guide to Attribute Based Access Control (ABAC) Definition and Considerations NIST Special Publication 800-162 Guide to Attribute Based Access Control (ABAC) Definition and Considerations."

[60] Y. A. Younis, "Securing Access to Cloud Computing for Critical Infrastructure," Doctroal thesis, Liverpool John Moores University., 2015.

[61] L. Lapadula and L. J. Lapadula, "Secure Computer Systems : Mathematical Foundations," *TR-73-278, vol. 1, ESD/AFSC, Hansom AFB, Bedford, Mass*, p. 29, 1973.

[62] M. S. XIN JIN, "ATTRIBUTE-BASED ACCESS CONTROL MODELS AND IMPLEMENTATION IN CLOUD INFRASTRUCTURE AS A SERVICE," Doctroal thesis, The University of Texas at San Antonio, 2014.

[63] D. Servos, S. L. Osborn, and W. Ontario, "Current Research and Open Problems in Attribute-Based Access Control," *ACM Comput. Surv.*, vol. 49, no. 4, pp. 1–45, 2017.

[64] R. Kissel, "Glossary of Key Information Security Terms," NISTIR 7298 Revision 2, 2013. https://www.nist.gov/publications/glossary-key-information-security-terms-1

[65] W. Rjaibi, "An Introduction to Multilevel Secure Relational Database Management Systems," *Proc. 2004 Conf. Cent. Adv. Stud. Collab. Res. Markham, Ontario, Canada*, pp. 232 – 241, 2004.

[66] T. G. of C. GOC, "The Goverment of Canada GOC Levels of Security," 2017. [Online]. Available: https://www.tpsgc-pwgsc.gc.ca/esc-src/protection-safeguarding/niveaux-levels-eng.html.

[67] R. C. Scott D Smith, "Shedding Light on Cross Domain Solutions," 2015. https://www.sans.org/reading-room/whitepapers/dlp/shedding-light-cross-domain-solutions-36492

[68] H. Bidgoli, *Handbook of Information Security, Volume 3, Threats, Vulnerabilities, Prevention, Detection and Management*, ISBN 0-471. 2006.

[69] J. McLEAN, "A COMMENT ON THE 'BASIC SECURITY THEOREM' OF BELL AND LaPADULA," vol. 20, no. February, pp. 67–70, 1985.

[70] Z. Tang, X. Ding, Y. Zhong, L. Yang, and K. Li, "A Self-Adaptive Bell – LaPadula Model Based on Model Training With Historical Access Logs," *IEEE Trans. Inf. FORENSICS Secur.*, vol. 13, no. 8, pp. 2047–2061, 2018.

[71] E. E. O. R. Lindgreen and I. S. Herschberg, "On the validity of the Bell-LaPadula model," *Elsevier Sci. Ltd*, vol. 13, pp. 317–333, 1994.

[72] C. Lee *et al.*, “A Cluster-Based Multilevel Security Model for Wireless Sensor Networks,” *7th Int. Conf. Intell. Inf. Process. (IIP), Oct 2012, Guilin, China. Springer, IFIP Adv. Inf. Commun. Technol. AICT-385, pp.320-330, 2012, Intell. Inf. Process. VI. <10.1007/978-3-642-3*, pp. 0–11.

[73] M. Bishop, “Position : ‘ Insider ’ is Relative,” *Proceedings of the New Security Paradigms Workshop*, 2005.

[74] Alfred Tong, “CISSP Domain 3 Security Engineering – Part 1 – Security Architecture. https://www.alfredtong.com/security-2/cissp-domain-3-security-engineering-part-1-security-architecture-cheat-sheet/.” 2016.

[75] T. Xin and I. R. Ã, “A lattice-based approach for updating access control policies in real-time,” *Inf. Syst.*, vol. 32, pp. 755–772, 2007.

[76] M. Kiviharju, *Enforcing Role-Based Access Control with Attribute-Based Cryptography in MLS Environments*, 2nd ed. 2017.

[77] Hossein Bidgoli, Handbook of Information Security, Volume 3, Threats, Vulnerabilities, Prevention, Detection and Management, Chpater 205, ISBN 0-471-64832-9, John Wiley, 2006 .

[78] M. R. Heckman and R. R. Schell, “Using Proven Reference Monitor Patterns for Security Evaluation,” *Information*, vol. 7, p. 23, 2016.

[79] T. Jaeger, “Reference Monitor,” *Encycl. Cryptogr. Secur.* , vol. 2, pp. 1038–1040, 2011.

[80] M. Gregg, “CISSP Exam Cram Security Architecture and Models.” 2013, http://www.pearsonitcertification.com/articles/article.aspx?p=1998558 .

[81] D. C. Latham, “DEPARTMENT OF DEFENSE STANDARD, DEPARTMENT OF DEFENSE, TRUSTED COMPUTER SYSTEM EVALUATION CRITERIA,” *Dep. Def.*, p. P80, 1986.

[82] E. T. B. Earl, “Multilevel Security, CHAPTER 8, Cambridge Computer Laboratory, ” pp. 239–274, https://www.cl.cam.ac.uk/~rja14/Papers/SEv2-c08.pdf

[83] A. G. and B. P. Z. Sebastian, “A SURVEY OF C OVERT C HANNELS AND COUNTERMEASURES IN COMPUTER NETWORK PROTOCOLS,” *IEEE Commun. Mag. (Sept.)*, vol. 9, no. 3, pp. 44–57, 2007.

[84] S. J. Murdoch, “Covert channel vulnerabilities in anonymity systems,” *Univ. CAMBRIDGE*, no. 706, p. p15, 2007, https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-706.pdf .

[85] C. Maney, “Security issues when Data transfers information domains: Do guards effectively address the problem?,” GIAC Security Essentials Certification (GSEC), Practical Assignment Version 1.4b–Option 1, 2004.

[86] M. Hicks, S. Tsang, and N. Swamy, “Toward Specifying and Validation Cross-Domain Policies,” *Tech. Rep. CS-TR-4870, Univ. Maryl.*, pp. 1–8, 2007.

[87] N. Security, “National Information Assurance (IA) Glossary,” Committee on National Security Systems ., no. 4009, p. p20, 2010. https://www.hsdl.org/?view&did=7447

[88] A. Zambrano, S. Gordillo, and J. Fabry, “a Fine Grained Aspect Coordination Mechanism,” *Int. J. Softw. Eng. Knowl. Eng.*, vol. 20, no. 7, pp. 1025–1042, 2010.

[89] Y. EL-Manzalawy, “Aspect Oriented Programming.” 2004 [online]. https://www.developer.com/lang/article.php/3308941/Aspe

[90] S. Kotrappa and P. J. Kulkarni, “Multilevel Security Using Aspect Oriented Programming AspectJ,” *2010 Int. Conf. Adv. Recent Technol. Commun. Comput.*, pp. 369–373, Oct. 2010.

[91] R. Laddad, “I want my AOP! ”, Part1,” *Jan 18 2002*. [Online]. Available: https://www.javaworld.com/article/2073918/core-java/i-want-my-aop---part-1.html.

[92] D. Fletcher and F. Akkawi, “From research to operations: integrating components with an aspect-oriented framework and ontology,” *Aerosp. …*, pp. 3064–3078, 2004.

[93] S. Koirala, “Aspect Oriented Programming in .NET.,” 2014. [online] https://www.codeproject.com/KB/architecture/AOP.aspx .

[94] M. García, D. Llewellyn-Jones, F. Ortin, and M. Merabti, “Applying dynamic separation of aspects to distributed systems security: a case study,” *IET Softw.*, vol. 6, no. 3, p. 231, 2012.

[95] F. Ortin, L. Vinuesa, and J. M. Felix, “the Dsaw Aspect-Oriented Software Development Platform,” *Int. J. Softw. Eng. Knowl. Eng.*, vol. 21, no. 7, pp. 891–929, 2011.

[96] M. Chibani, A. Bourouis, C. Science, and O. El Bouaghi, “THE USE OF THE ASPECT ORIENTED PROGRAMMING (AOP) PARADIGM IN DISCRETE EVENT SIMULATION DOMAIN : OVERVIEW AND PERSPECTIVES,” *SDIWC*, no. 1, pp. 653–660, 2013.

[97] S. R. Raheman, A. K. Rath, and H. B. M, “Dynamic Slice of Aspect Oriented Program : A Comparative Study,” *Int. J. Recent Innov. Trends Comput. Commun.*, vol. 2, no. 2, pp. 249–259, 2008.

[98] T.-T. Nguyen, N.-T. Truong, and V.-H. Nguyen, “Verifying Java Object Invariants At Runtime,” *Int. J. Softw. Eng. Knowl. Eng.*, vol. 21, no. 4, pp. 605–619, 2011.

[99] M. Pinto and J. M. Horcas, “How to develop secure applications with Aspect-Oriented Programming,” *Int. Conf. Risks Secur. Internet Syst.*, pp. 4–6, 2013.

[100] H. Hawkins, "Develop aspect-oriented Java applications with Eclipse and AJDT Updated tools make AOP easier for beginners and veterans alike," no. September, pp. 1–19, 2004.

[101] AspectJ Team, "AspectJ Tutorial ineclipse." [online] https://www.eclipse.org/aspectj/doc/next/progguide/starting.html

[102] E. Hilsdale and J. Hugunin, "Advice weaving in {AspectJ}," *Proc. 3rd Int' Conf. Asp. Softw. Dev. {({AOSD}-2004)}*, pp. 26–35, 2004.

[103] K. Chen and C.-M. Huang, "A Practical Aspect Framework for Enforcing Fine-Grained Access Control in Web Applications," *Inf. Secur. Pract. Exp.*, pp. 156–167, 2005.

[104] K. Chen and D. Wang, "AN ASPECT-ORIENTED APPROACH TO PRIVACY-AWARE ACCESS CONTROL," *Mach. Learn. Cybern. 2007 Int. Conf.*, vol. 5, no. August, pp. 3016–3021, 2007.

[105] F. Yang, C. Hankin, F. Nielson, and H. R. Nielson, "Predictive access control for distributed computation," *Sci. Comput. Program.*, vol. 78, no. 9, pp. 1264–1277, 2013.

[106] K. Chen and C.-M. Huang, "On Designing Access Control Aspects for Web Applications," *Softw. Eng. Prop. Lang. Asp. Technol.*, pp. 1–8, 2005.

[107] R. Toledo and É. Tanter, "Secure and modular access control with aspects," *Proc. 12th Annu. Int. Conf. Asp. Softw. Dev. - AOSD '13*, p. 157, 2013.

[108] T. Scheffler, S. Schindler, and B. Schnor, "Enforcing Location Privacy Policies through an AOP-based Reference Monitor," *Internet Secur. (WorldCIS), 2012 World Congr.*, pp. 51–56, 2012.

[109] X. Li, N. A. Naeem, and B. Kemme, "Fine-granularity access control in 3-tier laboratory information systems," *IEEE Proc. 9th Int. Database Eng. Appl. Symp.*, pp. 391–397, 2005.

[110] I. Ray, R. France, N. Li, and G. Georg, "An aspect-based approach to modeling access control concerns," *Inf. Softw. Technol.*, vol. 46, no. 9, pp. 575–587, 2004.

[111] J. Pavlich-mariscal, L. Michel, and S. Demurjian, "A Formal Enforcement Framework for Role-Based Access Control Using Aspect-Oriented Programming," pp. 537–552, 2005.

[112] C. Braga, "A transformation contract to generate aspects from access control policies," *Softw. Syst. Model.*, vol. 10, no. 3, pp. 395–409, 2011.

[113] B. F. Truyen, "The Fast Guide to Model Driven Architecture The Basics of Model Driven Architecture," (White paper) 2006, https://www.omg.org/mda/mda_files/Cephas_MDA_Fast_Guide.pdf

[114] M. Hazaa and A. Ghani, "Secure role based access control systems using aspect-orientation designing," *GCC Conf. ...*, no. 1, pp. 1–5, 2009.

[115] S. Kallel, M. Mezini, A. Charfi, and M. Jmaiel, “Aspect-based enforcement of formal delegation policies,” *Proc. 2008 3rd Int. Conf. Risks Secur. Internet Syst. Cris. 2008*, pp. 9–17, 2008.

[116] P. Colombo and E. Ferrari, “Enforcement of Purpose Based Access Control within Relational Database Management Systems,” *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 11, pp. 2703–2716, 2014.

[117] J. Pavlich-Mariscal, T. Doan, L. Michel, S. A. Demurjian, and T. C. Ting, “Role Slices: A Notation for RBAC Permission Assignment and Enforcement,” *Data Appl. Secur. XIX, 19th Annu. IFIP WG 11.3 Work. Conf. Data Appl. Secur.*, vol. 3654, no. 3654, pp. 40–53, 2005.

[118] A. Mourad and S. Ayoubi, “New Approach for the Dynamic Enforcement of Web Services Security,” 2010 Eighth Annual International Conference on Privacy, Security and Trust , September, 2010.

[119] M. S. Idrees, S. Ayed, N. Cuppens-Boulahia, and F. Cuppens, “Dynamic security policies enforcement and adaptation using aspects,” *Proc. - 14th IEEE Int. Conf. Trust. Secur. Priv. Comput. Commun. Trust. 2015*, vol. 1, pp. 1374–1379, 2015.

[120] L. F. Samiha Ayed, Muhammad Sabir Idrees, Monica Pinto, “Security Aspects A Framework for Enforcement of Security Policies using AOP.” Signal-Image Technology & Internet-Based Systems, SITIS, pp. 301–308, 2013.

[121] S. Ayed, M. Sabir, I. Nora, and C. Frederic, “Achieving dynamicity in security policies enforcement using aspects,” *Int. J. Inf. Secur.*, vol. 17, no. 1, pp. 83–103, 2018.

[122] R. Ramachandran and D. J. Pearce, “AspectJ for Multilevel Security,” *CP4IS ’06*, vol. 20, pp. 13–17, 2006.

[123] H. Ulusoy, M. Kantarcioglu, E. Pattuk, and K. Hamlen, “Vigiles : Fine-grained Access Control for MapReduce Systems,” *In:IEEE BigData*, 2014.

[124] K. Padayachee, “An Aspect-Oriented Approach to Enhancing Multilevel Security with Usage Control : An Experience Report,” In International Multi Conference of Engineers and Computer Scientists , pages 1060–1065, 2007 .

[125] A. Hernandez and F. Nielson, “History-sensitive versus future-sensitive approaches to security in distributed systems,” *ICE2010 - 3rd Interact. Concurr. Exp. - EPTCS*, vol. 38, no. Ice, pp. 29–43, 2010.

[126] S. M. Khan, K. W. Hamlen, and M. Kantarcioglu, “Silver Lining : Enforcing Secure Information Flow at the Cloud Edge,” *IC2E ’14 Proc. 2014 IEEE Int. Conf. Cloud Eng.*, pp. 37–46, 2014.

[127] E. Kajo-mece, L. Kodra, and E. Vrenozaj, “Protection of Web Applications Using Aspect Oriented Programming and Performance Evaluation,” *BCI’12, Sept. 16–20, 2012, Novi Sad, Serbia.*, vol. i, pp. 46–50, 2012.

[128] G. Hermosillo, R. Gomez, L. Seinturier, and L. Duchien, “Using Aspect Programming to Secure Web Applications,” vol. 2, no. 6, pp. 53–63, 2007.

[129] Z. J. Zhu and M. Zulkernine, “A model-based aspect-oriented framework for building intrusion-aware software systems,” *Inf. Softw. Technol.*, vol. 51, no. 5, pp. 865–875, 2009.

[130] M. Coates, “DETECT AND RESPOND TO ATTACKS FROM WITHIN THE APPLICATION,” , V1.1, 2009.

[131] V. Schiavoni and V. Qu, “A Posteriori Defensive Programming : an Annotation Toolkit for DoS-resistant Component-Based Architectures,” *ACM 1-59593-108-2/06/0004*, 2006.

[132] K. Padayachee, “An Aspect-Oriented Model to Monitor Misuse,” *Springer Netherlands*, pp. 273–278, 2007.

[133] S. H. Georg, G., Ray, I., Anastasakis, K., Bordbar, B., Toahchoodee, M., & Houmb, “An aspect-oriented methodology for designing secure applications,” *Inf. Softw. Technol.*, vol. 50, pp. 846–864, 2009.

[134] J. Horcas, M. Pinto, L. Fuentes, W. Mallouli, and E. M. De Oca, “An approach for deploying and monitoring dynamic security policies,” *Comput. Secur.*, vol. 58, pp. 20–38, 2016.

[135] G. Hermosillo, R. Gomez, L. Seinturier, and L. Duchien, “AProSec: An aspect for programming secure web applications,” *Proc. - Second Int. Conf. Availability, Reliab. Secur. ARES 2007*, pp. 1026–1033, 2007.

[136] L. K. E. Mece, “Towards full protection of web application based on Aspect Oriented Programming,” *Double Blind Peer Rev. Int. Res. J.*, vol. 12, no. 1, 2012.

[137] K. Kawauchi and H. Masuhara, “Dataflow Pointcut for Integrity Concerns,” *roceedings AOSD 2004 Work. AOSD Technol. Appl. Secur.*, no. C, 2004.

[138] G. Fan, H. Yu, L. Chen, and D. Liu, “Aspect Oriented Approach to Building Secure Service Composition,” *Proc. Asia Pacific Softw. Eng. Conf. APSEC 2010. IEEE Comput. Soc. Press. Sydney Press. Sydney, Aust.*, pp. 176–185, 2010.

[139] A. Falcarin, C. Falcarin, D. S. Tampering, P. Falcarin, M. Baldi, and D. Mazzocchi, “Software Tampering Detection using AOP and mobile code,” *Present. Int. Conf. Asp. Softw. Dev. (AOSD’04), Lancaster, UK,* 2004.

[140] H. Ulusoy, M. Kantarcioglu, and E. Pattuk, “TrustMR : Computation Integrity Assurance system for MapReduce,” *Proc. IEEE Conf. Big Data (Big Data),* 2015.

[141] P. Falcarin, R. Scandariato, and M. Baldi, “Remote Trust with Aspect-Oriented Programming,” *n IEEE Adv. Inf. Netw. Appl. (AINA-06). IEEE,* pp. 2–7, 2006.

[142] D. Wang and K. Chen, “Supporting Patients â€TM Privacy Preferences Using Aspects,” *pn. J. Med. Inf.*, vol. 29, pp. 117–128, 2010.

[143] and H. R. N. Hankin, Chris F. Nielson, “Advice from Belnap Policies,” *Comput. Secur. Found. Symp. ,IEEE*, pp. 234–247, 2009.

[144] P. Yu, J. Sendor, G. Serme, and A. S. De Oliveira, “Automating privacy enforcement in cloud platforms,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7731 LNCS, pp. 160–173, 2013.

[145] C. Vanden Berghe and M. Schunter, “Privacy injector - Automated privacy enforcement through aspects,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 4258 LNCS, pp. 99–117, 2006.

[146] R. T. Hassen Mestiri, Younes Lahbib, Mohsen Machhout, “An AOP-Based Fault Injection Environment for Cryptographic SystemC Designs,” *J. Circuits Syst. Comput*, vol. 24, no. 1, pp. 1–22, 2015.

[147] A. A. Thulnoon, B. Lempereur, and Q. Shi, “Using Aspect Oriented Programming to Enforce Privacy Preserving Communication in Distributed Systems,” *ICC ’17 Proc. Second Int. Conf. Internet things, Data Cloud Comput. , Cambridge, United Kingdom — March 22 - 23, 2017*, vol. 2, 2017.

[148] Axiomatics Team, “Axiomatics.” [online] https://www.axiomatics.com/.

[149] TrustedRUBIX Team, “TrustedRUBIX.” [online] http://rubix.com/cms/abac_arch.

[150] SimShield Team , “SimShield.” [online]

https://www.raytheon.com/capabilities/rtnwcm/groups/gallery/documents/digitalasset/rtn_216073.pdf

[151] Forcepoint Team, “Forcepoint.” [online] https://www.forcepoint.com/ .

[152] H. D. Rolvwlf, R. Ri, and D. D. H. Q. U. I. D. D, “Developing a Novel Holistic Taxonomy of Security Requirments,” *Procedia Comput. Sci. 62 (2015)*, vol. 62, no. Scse, pp. 213–220, 2015.