Journal of Information Security, 2015, 6, 118-130
Published Online April 2015 in SciRes. http://www.scirp.org/journal/jis
http://dx.doi.org/10.4236/jis.2015.62013

Scientific Research Publishing

Ensuring Security, Confidentiality and Fine-Grained Data Access Control of Cloud Data Storage Implementation Environment

Amir Mohamed Talib

College of Computer and Information Sciences, Al-Imam Muhammad Ibn Saud Islamic University, Riyadh, KSA
Email: ganawa53@yahoo.com

Received 15 February 2015; accepted 13 April 2015; published 16 April 2015

Abstract

With the development of cloud computing, the mutual understandability among distributed data access control has become an important issue in the security field of cloud computing. To ensure security, confidentiality and fine-grained data access control of Cloud Data Storage (CDS) environment, we proposed Multi-Agent System (MAS) architecture. This architecture consists of two agents: Cloud Service Provider Agent (CSPA) and Cloud Data Confidentiality Agent (CDConA). CSPA provides a graphical interface to the cloud user that facilitates the access to the services offered by the system. CDConA provides each cloud user by definition and enforcement expressive and flexible access structure as a logic formula over cloud data file attributes. This new access control is named as Formula-Based Cloud Data Access Control (FCDAC). Our proposed FCDAC based on MAS architecture consists of four layers: interface layer, existing access control layer, proposed FCDAC layer and CDS layer as well as four types of entities of Cloud Service Provider (CSP), cloud users, knowledge base and confidentiality policy roles. FCDAC, it's an access policy determined by our MAS architecture, not by the CSPs. A prototype of our proposed FCDAC scheme is implemented using the Java Agent Development Framework Security (JADE-S). Our results in the practical scenario defined formally in this paper, show the Round Trip Time (RTT) for an agent to travel in our system and measured by the times required for an agent to travel around different number of cloud users before and after implementing FCDAC.

Keywords

Cloud Computing, Cloud Data Storage, Cloud Service Provider, Formula-Based Cloud Data Access Control, Multi-Agent System and Secure Java Agent Development Framework

How to cite this paper: Talib, A.M. (2015) Ensuring Security, Confidentiality and Fine-Grained Data Access Control of Cloud Data Storage Implementation Environment. *Journal of Information Security*, **6**, 118-130.
http://dx.doi.org/10.4236/jis.2015.62013

1. Introduction

Cloud computing describes applications that are extended to be accessible through the Internet. These cloud applications use large data centers or cloud data storage CDS and powerful servers that host Web applications and Web services. Anyone with a suitable Internet connection and a standard browser can access a cloud application. Cloud computing consists of multiple cloud computing service providers (CSPs). In terms of software and hardware, a cloud system is composed of many types of computers, storage devices, communications equipment, and software systems running on such devices [1].

Heterogeneity and diversity of cloud services, and the domains' diverse access requirements in cloud computing environments would require fine-grained access control policies. In particular, access control services should be flexible enough to capture dynamic, context or attribute/credential based access requirements, and facilitate enforcement of the principle of least privilege. Such access control services may need to integrate privacy protection requirements derived from complex rules. It is important that the access control system employed in clouds is easily managed and its privilege distribution is administered efficiently. It is also important to ensure that the cloud delivery models provide generic access control interfaces for proper interoperability, which demands for a policy neutral access control specification and enforcement framework that can be used to address cross-domain access control issues [2]. Also, the access control models should be able to capture relevant aspects of service level agreements (SLAs).

Cloud Data Storage (CDS) is composed of thousands of cloud storage devices clustered by network, distributed file systems and other storage middleware to provide cloud storage service for users. The typical structure of CDS includes storage resource pool, distributed file system, service level agreements (SLAs), and service interfaces, etc. Globally, they can be divided by physical and logical functions boundaries and relationships to provide more compatibilities and interactions. CDS is tending to combine with CDS security, which will provide more robust security [1].

Also, CDS refers to any type of data storage that resides in the cloud, including: services that provide database-like functionality, unstructured data services (file storage of digital media, for example), data synchronization services, or Network Attached Storage (NAS) services. Data services are often consumed in a pay-as-you-go model or, in this case, a pay-per-GB model (including both stored and transferred data). CDS offers a number of benefits, such as the ability to store and retrieve large amounts of data in any location at any time. CDS services are fast, inexpensive, and almost infinitely scalable; however, reliability can be an issue, as even the best services do sometimes fail. Transaction support is also an issue with cloud-based storage systems, a significant problem that needs to be addressed for storage services to be widely used in the enterprise [2].

Confidentiality defined as ensuring that user data which resides in the cloud cannot be accessed by unauthorized party. This can be achieved through proper encryption techniques taking into consideration the type of encryption: symmetric or asymmetric encryption algorithms, also key length and key management in case of the symmetric cipher. Actually, it is all based on the CSP. For instance in [3], Mozy Enterprise uses encryption techniques to protect customer data whereas Amazon S3 does not. It also depends on the customer awareness where they can encrypt their information prior to uploading it. Also, The CSP should ensure proper deployment of encryption standards using NIST standards in [4].

Access control regulates accesses to resources by principals. It is one of the most important aspects of the security of a system. A protection state or policy contains all information needed by a reference monitor to enforce access control. The syntax used to represent a policy is called an access control model. In FCDAC, principals are called cloud users. Cloud users get permissions to access resources via membership in roles. We present a new way to realize FCDAC in JADE-S. Our approach focuses not only on providing a cloud user by a user name and password but defining and enforcing expressive and flexible access structure for each cloud user as a logic formula over cloud data file attributes. Each data file is stored on the cloud in the format as is shown in **Figure 1**.

A multi-agent system (MAS) consists of a number of agents interacting with each other, usually through exchanging messages across a network. The agents in such a system must be able to interact in order to achieve their design objectives, through cooperating, negotiating and coordinating with other agents. The agents may exhibit selfish or benevolent behavior. Selfish agents ask for help from other agents if they are overloaded and never offer help. For example, agents serving VIP (Very Important Person) cloud users for CSP service never help other agents for the same service. Benevolent agents always provide help to other agents because they consider system benefit is the priority. For example, agents serving normal cloud users for CSP service are always

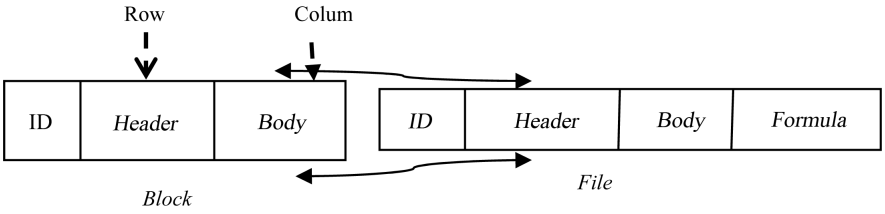

Figure 1. Format of cloud data file stored in the CDS.

ready to help other agents to complete their tasks.

Our contributions: Theoretically, this paper contributes the description of the first provably-secure and practical FCDAC which provide access structure of each cloud user by defining it as a logic formula over cloud data file attribute.

In our FCDAC scheme, we assume that the system is composed of the following parties: the CSP, many cloud users and many cloud servers or CDSs. To access cloud data files shared by the CSP or cloud users, download cloud data files of their interest from cloud servers and then decrypt. Neither the CSPs nor cloud users should be always online and they might only get online just on necessity basis. Cloud servers are always online and operated by the CSP. For simplicity, we assume that the only access privilege for cloud users is cloud data file reading. Extending our proposed FCDAC scheme to support cloud data file writing is trivial by asking the cloud user to sign the new cloud data file. Cloud servers will follow our proposed FCDAC scheme in general, but try to find out as much secret information as possible based on their inputs. More specifically, we assume cloud servers are more interested in cloud file contents and cloud user access privilege information than other secret information. Cloud servers might collude with a small number of malicious users for the purpose of harvesting file contents when it is highly beneficial. Communication channel between the CSP/cloud users and cloud servers are assumed to be secured under existing security protocols. Cloud users would try to access files either within or outside the scope of their access privileges. To achieve this goal, unauthorized users may work independently or cooperatively.

The CSP stores a large amount of information on the cloud server. Since the cloud computing environment is untrusted environment, the CSP will encrypt the cloud data before putting them on the cloud server. Here we assume that the smallest information access unit is called a “cloud file” stored in the cloud server. This is an abstract concept and it may have different meanings in different systems. To provide FCDAC, the encryption will be conducted at the file level based on our proposed MAS techniques which provide access structure of each cloud user by defining it as a logic formula over cloud data file attribute. Only the CSP can make updates to the cloud data. We also assume that there exist pre-distributed secrets between CSP and cloud server, and between CSP and cloud users. The illustration of FCDAC application scenario described as: 1) cloud user will send a data access request to the CSP; 2) CSP will provide the MAS architecture by service level agreement (SLA); 3) CSP certified the access control; 4) cloud user insert its encryption key; and 5) then only authorized cloud user can access the cloud data file, as illustrated in **Figure 2**.

In this paper, in Section 2 we present a discussion of the related work. Section 3 provides a brief introduction of JADE security model (JADE-S) platform. Section 4 describes our proposed FCDAC based MAS architecture. Section 5 highlights the design goals of our proposed FCDAC scheme. The implementation steps have been discussed in Section 6. Results and discussions have been described in detail in Section 7. In, Section 8 presents some concluding remarks and future work.

2. Related Work

Many existing works close to our work can be found in the areas of “access control of outsourced data” Kallahalla *et al*. proposed Plutus as a cryptographic file system to secure file storage on untrusted servers [5], Goh *et al*. proposed SiRiUS which is layered over existing file systems such as NFS but provides end-to-end security [6], Ateniese *et al*. [7] proposed a secure distributed storage scheme based on proxy re-encryption. Specifically, the data owner encrypts blocks of content with symmetric content keys. In [8], Vimercati *et al*. proposed a solution for securing data storage on untrusted servers based on key derivation methods [9]. In this proposed scheme, each file is encrypted with a symmetric key and each user is assigned a secret key.

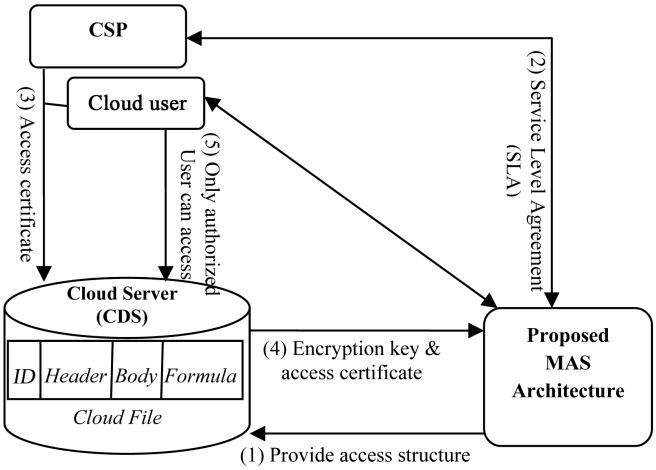

Figure 2. Illustration of the FCDAC application scenario.

KAoS uses DARPA Agent Markup Language (DAML) as the basis for representing and reasoning about policies within Web Services, Grid Computing, and multi-agent system platforms [10]. KAoS also exploits ontology for representing and reasoning about domains describing organizations of human, agent, and other computational actors [11]. Rei is a deontic logic based policy language that is grounded in a semantic representation of policies in RDF-S [12]. However, developers of KAoS augured that the pure Web Ontology Language (OWL) based method has difficulty in definition of some types of policy [11].

An access control system is typically described in three ways: access control policies, access control models and access control mechanisms [13]. Policy defines the high level rules according to which access control must be regulated. Model provides a formal representation of the Algebra of Communicating Processes (ACPs). Mechanism defines the low level functions that implement the controls imposed by the policy and formally stated in the model.

Access Control Policies can be generally divided into three main policy categories: Discretionary Access Control (DAC), Mandatory Access Control (MAC), and Role-Based Access Control (RBAC). Early DAC models, such as the access control matrix model [14] and the HRU (Harrison-Ruzzo-Ullman) model [15], provide a basic framework for describing DAC policy. In these models, it is the users' discretion to pass their privileges on to other users, leaving DAC policies vulnerable to Trojan Horse attacks [13]. The lattice-based multilevel security policy [16], policies represented by the Bell-LaPadula model [17].

3. JADE Security Model (JADE-S) Platform

In order to understand the role of security in the transmission of private and critical information (e.g. cloud data file) through an open environment (e.g. cloud computing and CDS), it is necessary to define the concepts that determine the diverse security levels:

✓ *Confidentiality*: is the property that ensures that only those that are properly authorized (*users*) may access the information.

In the case of MAS these properties are especially important, due to the autonomy and mobility of agents. MAS without security support could not be used in an open environment such as CDS if it deals with critical cloud data, because communications could be spied or the identities of the agents could be easily faked.

JADE-S [18] is a plug-in of JADE that allows adding some security characteristics in the development of MAS, so that they can start to be used in real environments. It is based on the Java security model [19] and it provides the advantages of the following technologies:

- JAAS (*Java Authentication and Authorization Service*) [20]: it allows establishing access permissions to perform certain operations on a set of predetermined classes, libraries or objects.
- JCE (*Java Cryptography Extension*) [19]: it implements a set of cryptographic functions that allow the developer to deal with the creation and management of keys and to use encryption algorithms.
- JSSE (*Java Secure Socket Extension*) [21]: it allows exchanging critical information through a network using

a secure data transmission (SSL).

3.1. Basic Concept

JADE platform may be located in different hosts and have different containers. In order to introduce security in cloud computing, JADE-S structures the agent platform as a multi-user environment in which all components (agents, containers, etc.) belong to authenticated (through a login and a password) users, who are authorized by the administrator of the system to perform certain privileged critical actions.

Thus, in each platform there is a permissions file that contains the set of actions that each user is authorized to perform [22]. Internally, an agent proves its identity by showing an Identity Certificate signed by the Certification Authority (provided in a transparent way to the agent when it registers in the system and provides the login and the password of its owner). Using these digitally signed certificates the platform may allow or deny certain actions to each agent.

3.2. Secure Communication

In order to provide a secure communication between agents located in different hosts or containers, JADE-S uses the SSL protocol (*Secure Socket Layer*) [23] that provides privacy and integrity for all the connections established in the platform. This is a way of being protected against network sniffers.

3.3. Permissions and Access Restrictions

In JADE-S platform the permissions to access resources are given to the different entities by following the mechanism defined by the new system provided by Java (JAAS [20]) for user-based authentication. Thus, it is possible to assign permissions to parts of the code and to its executers, restricting the access to certain methods, classes or libraries depending on who wants to use them. An entity can only perform an action (send a message, move to another container) if the Java security manager allows it. The set of permissions associated to each identity is stored in the access rights file of the platform (which is also unique and is loaded when the platform is booted).

Java provides a set of permissions (apart from those that may be defined by the user) on the basic elements of the language: AWT Permission, File Permission, Socket Permission, etc. Moreover, JADE-S provides other permissions related to the behavior of the agents: Agent Permission, Container Permission, etc. Any and each permission has a list of related actions that may be allowed or denied.

3.4. Authentication

As explained above, each component of the platform belongs to an authenticated user. The user that boots the platform also owns the AMS1 and DF2 agents and the main container. When a user wants to join the platform (through one of his/her agents), it has to provide his/her login and password. These data are checked with the passwords file contained in the platform, which is stored in a ciphered way, like UNIX passwords. The passwords file is unique and is loaded with the main container. Each agent owned by this user will have an Identity Certificate that contains its name, its owner and the signature of the Certification Authority.

3.5. Certification Authority and Certificates

The Certification Authority is the entity that signs the certificates of all the elements of the platform. To do that, it owns a couple of public/private keys so that, for each certificate, it creates an associated signature by ciphering it with its private key (which is secret). Then, when the identity of an entity has to be checked, the signature may be unencrypted with the public key of the Authority (which is publicly known) and we can check that the identity of an entity wanted to prove matches the one provided by the Authority. The secure platform JADE-S provides a Certification Authority within the main container. Each signed certificate is only valid within the platform in which it has been signed.

4. Our Proposed Fcdac Based on Mas Architecture

The MAS architecture consists of two agents: Cloud Service Provider Agent (CSPA) and Cloud Data Confiden-

tiality Agent (CDConA). CSPA interact with the cloud users to fulfill their interests. CDConA provides a CSP by definition and enforcement expressive and flexible access structure for each cloud user, in which each agent is autonomous and able to cooperate, coordinate and communicate with other agents intelligently to satisfy the cloud users' needs. CDConA has two types of sub-agents of Cloud Data Flexible Confidentiality Agent (CDFConA) and Cloud Data Expressive Confidentiality Agent (CDEConA).

Our proposed FCDAC based on MAS architecture consists of four layers: interface layer, existing access control layer, proposed FCDAC layer and CDS layer, as shown in **Figure 3**. At the interface layer, the CSPA receive the security report from CDConA and handle the interaction with the cloud users in terms of graphical user interface. Existing access control layer is to differentiate between our proposed FCDAC access control and other existing access control lists of identification, authorization and authentication (Access Control Polices (ACPs), Role-Based Access Control (RBAC), Access Control List (ACL), Discretionary Access Control (DAC), Mandatory Access Control (MAC)… etc.), details of these existing access control techniques highlighted in Section 2 (Related Work). At the proposed FCDAC layer, the agents are responsible for giving permissions to both VIP and NOT-VIP cloud users' requests. CDS layer is considered as the environment layer in which our proposed FCDAC scheme provided an access control.

Also, there are four types of entities: CSPs, cloud user, knowledge base and confidentiality policy rules in which our proposed FCDAC scheme obtained and utilized the information from these entities.

Description of a single agent as follow:

4.1. Cloud Service Provider Agent (CSPA)

Is the users' intelligent interface to the system and allow the cloud users to interact with the security service environment. The CSPA provides graphical interfaces to the cloud user for interactions between the system and the cloud user. CSPA act in the system under the behaviour of CSP. It is the only agent that can execute outside the main container of the platform and make remote requests through cloud server. CSPA has the following responsibilities as stated in **Table 1**.

4.2. Cloud Data Confidentiality Agent (CDConA)

This agent facilitates the security policy of confidentiality for CDS. Main responsibility of this agent is to pro-

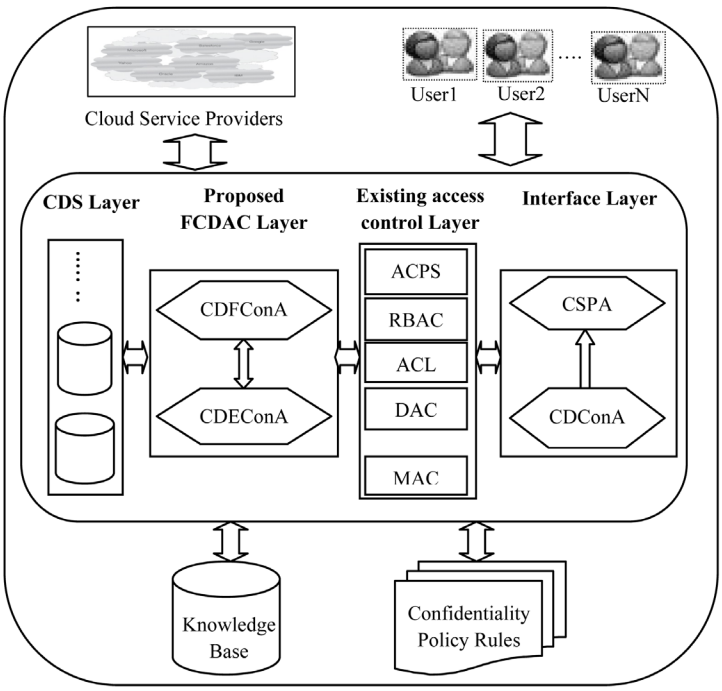

Figure 3. Proposed FCDAC based on MAS architecture.

Table 1. Responsibilities of CSPA.

Agent Name	Responsibilities
Cloud Service Provider Agent (CSPA)	Provide the security service task according to the authorized service level agreements (SLAs) and the original message content sent by the CDConA. Display the security policies specified by the CSP and the rest of the agents. Designing user interfaces that prevent the input of invalid cloud data. Receive the security reports and/or alarms from the rest of other agents to respect. Translate the attack in terms of goals. Monitor specific activities concerning a part of the CDS or a particular cloud user. Creating security reports/alarm systems.

vide a CDS by new access control rather than the existing access control lists of identification, authorization and authentication. This agent provides a CSP to define and enforce expressive and flexible access structure for each cloud user. Specifically, the access structure of each cloud user is defined as a logic formula over cloud data file attributes, and is able to represent any desired cloud data file set. This agent is also notifies CSPA in case of any fail caused of the techniques above by sending security reports and/or alarms.

There are two different types of CDConA, namely: Cloud Data Flexible Confidentiality Agent (CDFConA) and Cloud Data Expressive Confidentiality Agent (CDEConA). CDFConA acts as a dispatching center, with the following advantages. Firstly, the CDFConA is the unique access point where authorization agents can contact the CDConA. Secondly, CDFConA acts as the exclusive resource access entry. This makes the cloud users are only able to access the cloud data file via CDFConA.

CDFConA is divided into two types of agent based on the exhibited behavior. The first is CDFConA-VIP, which has selfish behavior and only serves for VIP (Very Important Person) cloud users. The second is NOT-VIP CDFConA, which has cooperative behavior and serves for both VIP cloud users and NOT-VIP (normal cloud users). For example, if CDFConA-VIP is overloaded, it sends help request to the first available NOT-VIP CDFConA. If this NOT-VIP CDFConA is unable to provide the help to CDFConA-VIP, the CDFConA-VIP sends the same help request to the next available NOT-VIP CDFConA. CDFConA-VIP keep sending the same request for help until one of the NOT-VIP CDFConA accepts or it is not overloaded. NOT-VIP CDFConA also takes the same action as CDFConA-VIP when they are overloaded. However, NOT-VIP CDFConA only has knowledge about other NOT-VIP CDFConA, but not CDFConA-VIP.

4.2.1. Cloud Data Flexible Confidentiality Agent (CDFConA)

The architecture of CDFConA consists of five modules, as shown in **Figure 4**. Cloud Communication Module provides the agent with the capability to exchange information with other agents, including the CDEConA and CSPA. Cloud Register Module facilitates the registration function for CDConA. Cloud Request Management Module allows the agent to act as the request-dispatching center. Cloud Resource Management Module manages the usage of the cloud resources. Cloud Reasoning Module is the brain of the CDConA. When the request management module and resource management module receive requests, they pass those requests to reasoning module by utilizing the information obtained from the knowledge base and the confidentiality policy rule.

4.2.2. Cloud Data Expressive Confidentiality Agent (CDEConA)

The architecture of the CDEConA consists of two modules, as shown in **Figure 5**. *Cloud Communication Module* provides the agent with the capability to exchange information with CSPA and CDFConA. *Cloud Cooperation Module* provides the agent with the following mechanisms. If the CDEConA is registered as CDFConA-VIP it asks for help when it is overloaded. If the CDEConA is registered as NOT-VIP CDFConA, it always offers help to other CDEConAs when it is not overloaded.

5. Design Goals

To ensure the security for CDS under the aforementioned secure MAS architecture, we aim to design efficient mechanisms for confidentiality of cloud user's data and achieve the following goal: to define and enforce expressive and flexible access structure for each cloud user by defines a logic formula over cloud data file attri-

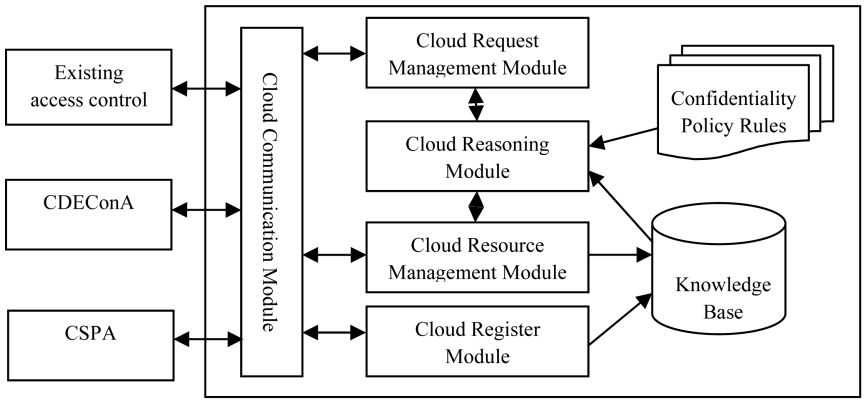

Figure 4. CDFConA architecture.

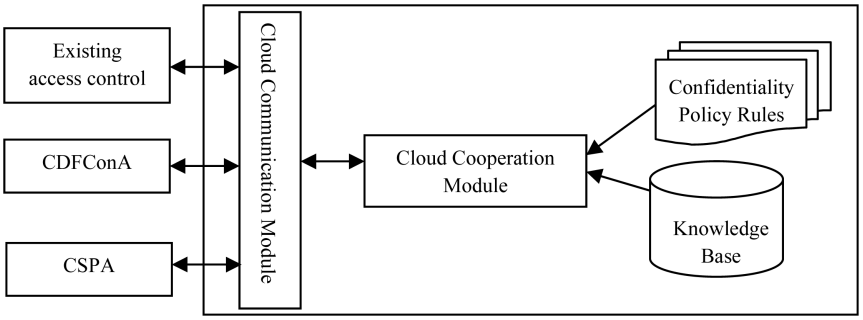

Figure 5. CDEConA architecture.

butes, this logic formula named as FCDAC.

6. Implementation

Before using any security mechanism provided by JADE-S on an agent platform, first it is necessary to define a set of default Java permissions that allow executing the basic code of JADE (including network access, graphical interface, etc.). Recall that the only allowed actions are those explicitly included in the permissions file. These permissions have to appear in the "basic.policy" file, which is read by default when the platform is booted.

Formula-Based Cloud Data Access Control (FCDAC) it's an access policy determined by our MAS architecture, not by the CSPs. It's also define as access is granted not based on the rights of the subject associated with a cloud user after authentication, but based on attributes of the cloud user. In our system, CDConA provide access structure of each cloud user by defining it as a logic formula over cloud data file attribute.

Having done that, we have built, on top of the security services provided by JADE-S and data access control that is based on two levels of permissions:

- Root: the cloud user has access to all the cloud data files that may be performed on the CDS (indicated explicitly one by one in the permissions cloud data file).
- Guest: the permissions of this cloud user are quite limited. CDConA can only perform the basic actions to request services from the system (send and receive messages).

The content of the passwords cloud data file containing these two users has the same format that the UNIX passwords cloud data file. This file can be created directly with the UNIX cloud user management capabilities or through the options provided by JADE. The set of permissions of each cloud user is stored in the access policy cloud data file associated to the platform. Thus, we associate one of these cloud users to each of the types of agents:

✓ Internal agents: the CDConA agent that executed within the main container of the platform (cloud servers, cloud data files and cloud users and Broker). They belong to the root cloud user; thus, they do not have any

constraint on the actions they can perform. We have taken this decision to facilitate the interaction among them (they can send and receive messages from any agent, or register in the DF or the AMS. As they are internal to the platform (they execute in the main container) and they have been programmed by us, they will not perform any malicious action (kill other agents, deregister other agents, fake the identity of another agent, etc.).

✓ External agents: the Personal agents that execute in an external container in another host belong to this category. They have been associated to the guest cloud user, so that they can neither access the main container (to join the other agents) nor access the DF (to modify the information of other agents). They can only communicate with the internal agents CDConA through the Broker; therefore, they cannot pretend to have the identity of other agents or kill other agents. All these constraints are necessary because we do not have any control over these agents, and they may have been programmed to perform dangerous actions.

Figure 6 and **Figure 7** illustrated and show diverse windows of the interface, used to insert the parameter and start execution of our agents.

Determining Enforcement Expressive and Flexible Keys for Cloud Data File Encryption

To provide fine-grained access control, we propose to encrypt every data file with a different secret. Here we do not assume the adoption of any specific symmetric encryption algorithm and leave the choice to our proposed MAS architecture when it is deployed. However, an efficient mechanism must be designed to allow MAS, CSPs and cloud users to manage the encryption keys. In the worst case, if the CDS contain n files f1, f2,…,fn and each file is encrypted with a randomly generated secret key, the CDS overhead on the cloud user will be linear to n. At the same time, when a cloud user needs to access its data files, the communication overhead between the agents and the cloud user for key distribution will also be linear. This overhead can be overwhelming for many cloud users when we consider that the CDS can easily contain millions of files. Therefore, a more encryption key scheme must be designed.

To solve this problem, we propose the basic idea of FCDAC scheme to generate the data file encryption keys through the two types of cloud users (VIP and NOT-VIP). Every key can be derived by combining its parent key

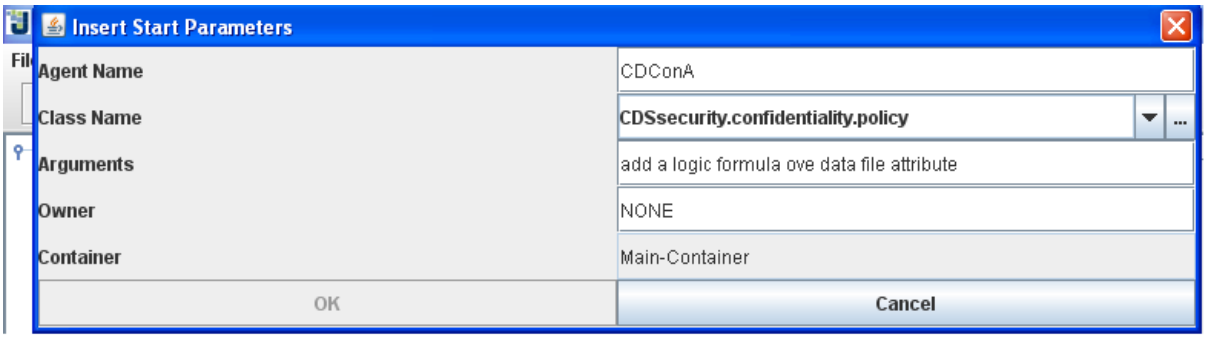

Figure 6. Define the arguments of our CDConA.

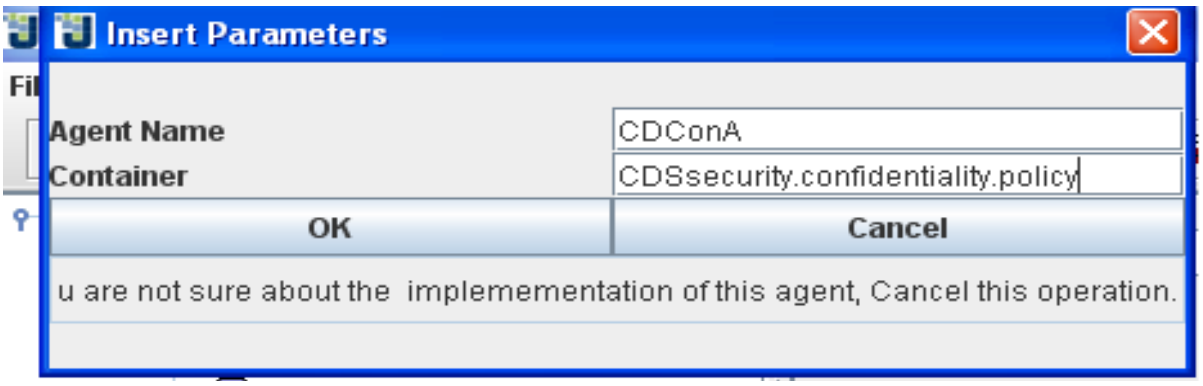

Figure 7. Insert the parameter and press ok to start execution.

that provided by CDConA agent and some public information from confidentiality policy roles and knowledge base entities. Since the derivation procedure uses a two-way function, we can calculate the secret keys of the parent key. In this way, the CSP will need to maintain only the root keys (full information described in the first four paragraphs in the implementation section). During the key distribution procedure, the CSP receive the secret key from CSPA and send the secret key to cloud users based on their access rights. The cloud user will derive the secret key to decrypt the data files.

The communication workflow between agents as shown in **Figure 8** decreases the level of complexity in agents' platform communication. Classically, MAS architecture needs to deal with middle server such as a message brokers. By decreasing such complexity of workflow, it will increase the autonomous level that doesn't need any responds from cloud user. In agent-based communication workflow, all components will be two ways interaction and the massaging platform will use the standard messaging proposed by FIPA which is ACL (Agent Communication Language), each agent will analyze each message priorities and in which action should be taken. For example, CDConA need to receive some service from CDFConA where the receiver agent will analyze the message that received by agent-receiver. Then, it will translate it to ACL. The sender agent also will identify which platform offer, which type of service and which message type they are using. The output of these messages will be registered in cloud register module and will be stored in cloud communication module.

VIP cloud user scenario:

- 1A. VIP cloud user will request the allowance from CSP for access the cloud application.
- 2A. CSP will ask CDEConA to provide the VIP cloud user by new access control.
- 3A. CDEConA will inform CDFConA to register the VIP cloud user request as communication ACL in cloud register module.
- 4A. CDFConA will response to CDConA that the request have been registered. Then CDConA will provide VIP cloud user by the encryption key.
- 5A. CDConA will inform CSP that the request of VIP cloud user has been provided.

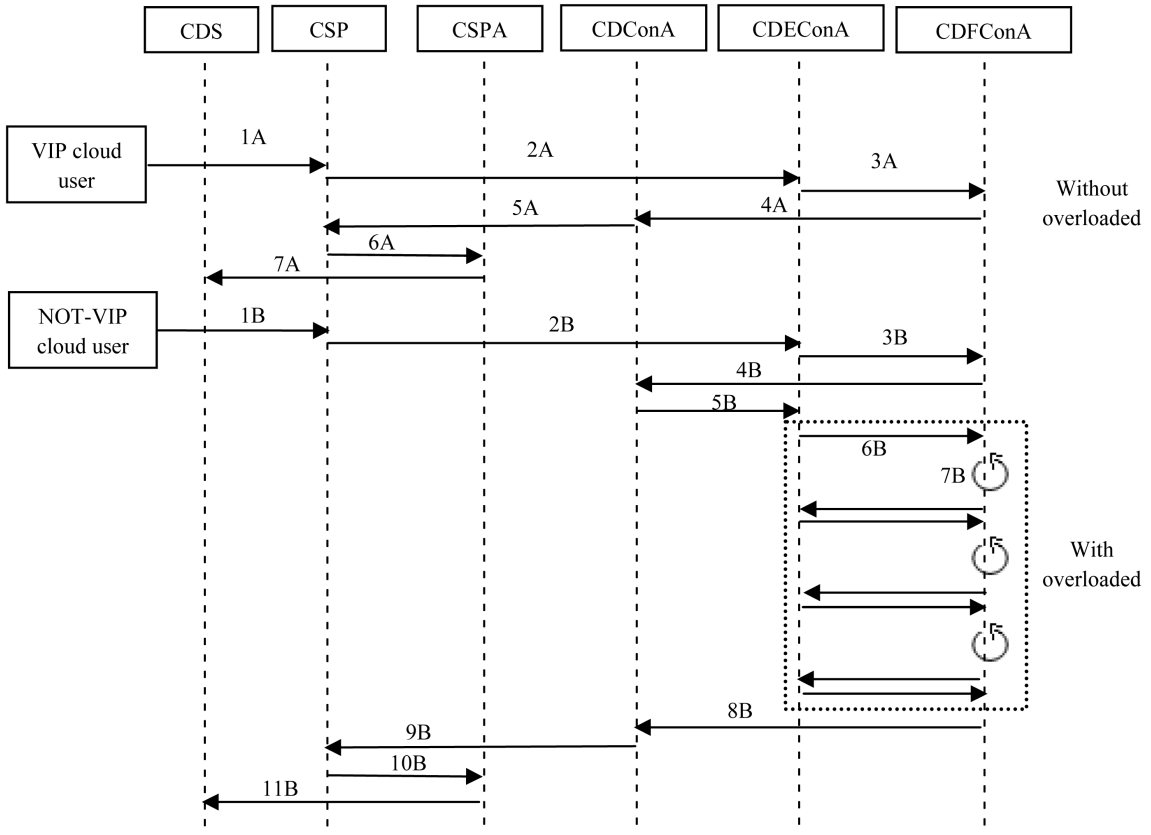

Figure 8. Agent-based communication workflow.

- 6A. CSP will provide CSPA by the SLA.
- 7A. VIP cloud user will be able to login in the CDS and access the cloud application.

NOT-VIP cloud user scenario:

- 1A. NOT-VIP cloud user will request the allowance from CSP for access the cloud application.
- 2A. CSP will ask CDEConA to provide NOT-VIP cloud user by new access control.
- 3A. CDEConA will inform CDFConA to register NOT-VIP cloud user request as communication ACL in cloud register module.
- 4A. CDFConA will response to CDConA that the request have been registered. Then CDConA will provide the NOT-VIP cloud user by the encryption key.
- 5B CDConA will ask CDEConA to register the ACL message of NOT-VIP cloud user in cloud cooperation module.
- 6B Due to, CDFConA has a selfish behavior with a NOT-VIP cloud user then, there is no message will be registered by CDEConA. So CDEConA will send the request back to CDFConA.
- 7B CDFConA will keep sending the same request for help until one of the CDFConA accepts. This step will repeat many times unless one of CDFConA accept and serve NOT-VIP cloud user.
- 8B CDFConA will response to CDConA that the request have been registered. Then CDConA will provide NOT-VIP cloud user by the encryption key.
- 9B. CDConA will inform CSP that the request of NOT-VIP cloud user has been provided.
- 10B. CSP will provide CSPA by the SLA.
- 11B. NOT-VIP cloud user will be able to login in the CDS and access the cloud application.

7. Evaluation of Result Analysis

To evaluate the performance of our system against the scale of the system, we measure the times required for an agent to travel around different number of cloud users before and after implementing FCDAC. The results are presented in **Table 2** and are plotted **Figure 9**. The data show that the Round Trip Time (RTT) for an agent to travel in our system changes more or less linearly over the number of cloud users in the system, both the cases. This is due to the additional time to travel an additional cloud user. The overhead for each additional cloud user is more or less the same. The overhead introduced is due to the extensive use of the MAS architecture to encrypt and decrypt each file attribute of cloud user, which is time consuming especially when the FCDAC's key is long. But a longer FCDAC's key gives a stronger protection to the system. Hence, a trade-off between performance and security for our MAS architecture is identified.

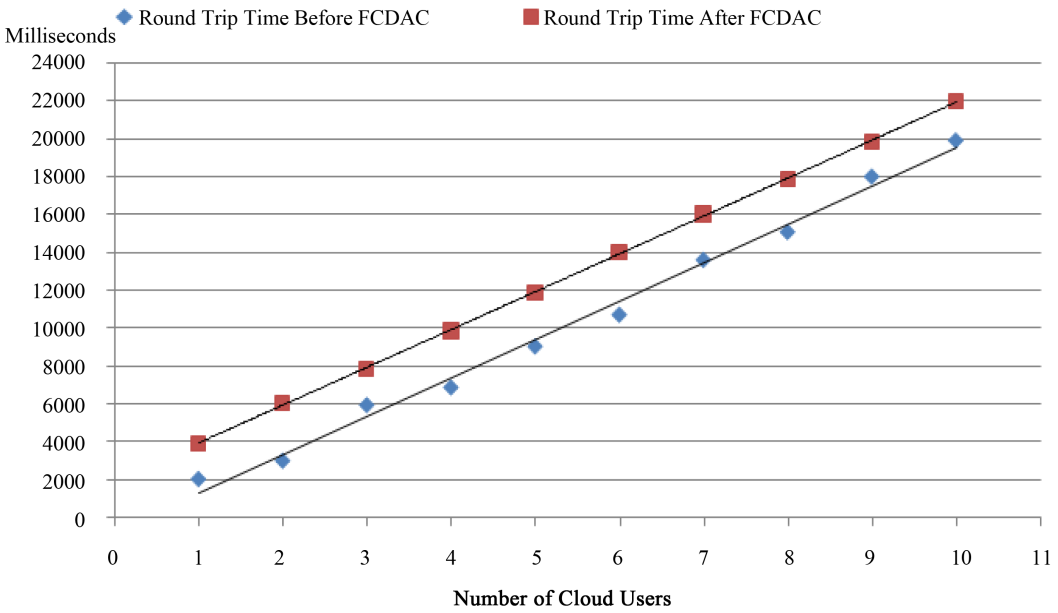

Figure 9. Average of round trip time before and after FCDAC.

Table 2. Average of RTT of our MAS in CDS.

Number of Cloud Users	RTT Before FCDAC	RTT After FCDAC
1	2043.7	3988.74
2	3013.46	6098.19
3	5945.7	7908.77
4	6880.197	9921.02
5	9040.71	11900.18
6	10713.9	14066.79
7	13605.4	16080.5
8	15076.7	17913.16
9	18002.43	19893.63
10	19901.623	22000

8. Conclusions and Future Work

This paper aims at formula-based cloud data access control (FCDAC) in cloud computing. One challenge in this context is to achieve fine-grained and data confidentiality, which is not provided by current work. FCDAC, it's an access policy determined by our MAS architecture, not by the CSPs. It's also defined as access is granted not based on the rights of the subject associated with a cloud user after authentication, but based on attributes of the cloud user. In this paper we propose secure MAS architecture to achieve this goal. We want to argue in this paper that secure MAS architecture is good alternative to fully implement complex programs in distributed environments such as cloud computing. Our proposed FCDAC based on MAS architecture consists of four layers: interface layer, existing access control layer, proposed FCDAC layer and CDS layer as well as four types of entities of Cloud Service Provider (CSP), cloud users, knowledge base and confidentiality policy roles. In fact, a complete prototype of the application is permanently running on a server provided by AgentCities in a worldwide agent-based network of services. Even though there is still a lot of work to be done in the security field in MAS, this paper tries to show that it is feasible to apply concepts of information security in these systems. Our results in the practical scenario defined formally in this paper, show the Round Trip Time (RTT) for an agent to travel in our system changes more or less linearly over the number of cloud users in the system measured by the times required for an agent to travel around different number of cloud users before and after implementing FCDAC.

In future work, we would focus on how we can utilize our agents system to encrypt the CDS and deploy it to real use, and we will compare the algorithm with some existing algorithms related to our work.

Acknowledgements

The author would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper.

References

[1] Talib, A.M., Atan, R., Abdullah, R. and Murad, M.A.A. (2010) Security Framework of Cloud Data Storage Based on Multi Agent System Architecture: Semantic Literature Review. *Computer and Information Science*, **3**, 175. http://dx.doi.org/10.5539/cis.v3n4p175

[2] Talib, A.M., Atan, R., Abdullah, R. and Murad, M.A.A. (2012) Towards a Comprehensive Security Framework of Cloud Data Storage Based on Multi Agent System Architecture. *Journal of Information Security*, **3**, 295-306. http://dx.doi.org/10.4236/jis.2012.34036

[3] Joshi, J.B.D. (2004) Access-Control Language for Multi-Domain Environments. *IEEE Internet Computing*, **8**, 40-50. http://dx.doi.org/10.1109/MIC.2004.53

[4] Mather, T., Kumaraswamy, S. and Latif, S. (2009) Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance. O'Reilly Media, Inc.

[5] Lori, M. (2009) Data Security in the World of Cloud Computing. Co-Published by the IEEE Computer and Reliability Societies, 61-64.

[6] Kallahalla, M., Riedel, E., Swaminathan, R., Wang, Q. and Fu, K. (2003) Plutus: Scalable Secure File Sharing on Untrusted Storage. in Fast, 29-42.

[7] Goh, E.-J., Shacham, H., Modadugu, N. and Boneh, D. (2003) SiRiUS: Securing Remote Untrusted Storage. in NDSS, 131-145.

[8] Ateniese, G., Fu, K., Green, M. and Hohenberger, S. (2006) Improved Proxy Re-Encryption Schemes with Applications to Secure Distributed Storage. *ACM Transactions on Information and System Security* (*TISSEC*), **9**, 1-30. http://dx.doi.org/10.1145/1127345.1127346

[9] Di Vimercati, S.D.C., Foresti, S., Jajodia, S., Paraboschi, S. and Samarati, P. (2007) Over-Encryption: Management of Access Control Evolution on Outsourced Data. *Proceedings of the* 33*rd International Conference on Very Large Data Bases*, 123-134.

[10] Atallah, M.J., Blanton, M., Fazio, N. and Frikken, K.B. (2009) Dynamic and Efficient Key Management for Access Hierarchies. *ACM Transactions on Information and System Security* (*TISSEC*), **12**, 18. http://dx.doi.org/10.1145/1455526.1455531

[11] Johnson, M., Chang, P., Jeffers, R., Bradshaw, J.M., Soo, V.W., Breedy, M.R., Bunch, L., Kulkarni, S., Lott, J. and Suri, N. (2003) KAoS Semantic Policy and Domain Services: An Application of DAML to Web Services-Based Grid Architectures. *Proceedings of the AAMAS*, Melbourne, 14-18 July 2003.

[12] Tonti, G., Bradshaw, J.M., Jeffers, R., Montanari, R., Suri, N. and Uszok, A. (2003) Semantic Web Languages for Policy Representation and Reasoning: A Comparison of KAoS, Rei, and Ponder. In: Fensel, D., Sycara, K. and Mylopoulos, J., Eds., *The Semantic Web-ISWC* 2003, Springer, Berlin, 419-437. http://dx.doi.org/10.1007/978-3-540-39718-2_27

[13] Kagal, L. (2002) Rei: A Policy Language for the Me-Centric Project.

[14] Samarati, P. and de Vimercati, S.C. (2001) Access Control: Policies, Models, and Mechanisms. In: Focardi, R. and Gorrieri, R., Eds., *Foundations of Security Analysis and Design*, Springer, Berlin, 137-196. http://dx.doi.org/10.1007/3-540-45608-2_3

[15] Lampson, B.W. (1974) Protection. *ACM SIGOPS Operating Systems Review*, **8**, 18-24. http://dx.doi.org/10.1145/775265.775268

[16] Harrison, M.A., Ruzzo, W.L. and Ullman, J.D. (1976) Protection in Operating Systems. *Communications of the ACM*, **19**, 461-471. http://dx.doi.org/10.1145/360303.360333

[17] Denning, D.E. (1976) A Lattice Model of Secure Information Flow. *Communications of the ACM*, **19**, 236-243. http://dx.doi.org/10.1145/360051.360056

[18] Bell, D.E. and LaPadula, L.J. (1973) Secure Computer Systems: Mathematical Foundations. DTIC Document.

[19] Fabio, B., Giovanni, C., Tiziana, T., Giovanni, R. and Roland, M. (2007) JADE Administrator's Guide. Last Update, 2007.

[20] Sun Microsystems (2010) Java Security. http://java.sun.com/security

[21] Sun Microsystems (2014) Authentication and Authorization Service (JAAS). http://java.sun.com/products/archive/jaas/

[22] Sun Microsystems (2014) Java Secure Socket Extension (JSSE) Reference Guide. http://download.oracle.com/javase/6/d ocs/technot es/gui-des/security/jsse/JSSERefGuide.html

[23] Sun Microsystems (2014) Java Default Policy Implementation and Policy File Syntax. http://java.sun.com/j2se/1.4/docs/gui de/security/Poli cyFile s.html

[24] Netscape Communication Corporation (2014) Introduction to SSL. http://developer.netscape.com/docs/manuals/security/ss-lin/contents.htm