Journal of Software Engineering

ISSN 1819-4311

Journal of Software Engineering 5 (3): 78-90, 2011
ISSN 1819-4311 / DOI: 10.3923/jse.2011.78.90
© 2011 Academic Journals Inc.

Multi Agent System Architecture Oriented Prometheus Methodology Design to Facilitate Security of Cloud Data Storage

Amir Mohamed Talib, Rodziah Atan, Rusli Abdullah and Masrah Azrifah Azmi Murad
Department of IT, Information System, Faculty of Computer Science, University Putra Malaysia, 43400 UPM, Serdang, Selangor, Malaysia

Corresponding Author: Amir Mohamed Talib, Department of IT, Information System, Faculty of Computer Science, University Putra Malaysia, 43400 UPM, Serdang, Selangor, Malaysia

ABSTRACT

Security plays an important role in the development of Multi Agent Systems (MAS). However, a careful analysis of software development processes shows that the definition of security requirements is, usually, considered after the design of the system. This is, mainly, due to the fact that agent oriented software engineering methodologies have not integrated security concerns throughout their developing stages. Designing a team of agents that can work together toward a common goal is one of the challenges in the research area of agent-oriented software engineering. Prometheus is an agent-oriented software engineering methodology. The Prometheus Design Tool (PDT) is a graphical editor which supports the design tasks specified within the Prometheus methodology for designing agent systems. The tool propagates information where possible and ensures consistency between various parts of the design. The main purpose of this paper is to design MAS architecture that can be used to facilitate confidentiality, correctness assurance, availability and integrity of Cloud Data Storage (CDS) or cloud data center. The proposed MAS architecture includes five types of agents: Cloud Service Provider Agent (CSPA), Cloud Data Confidentiality Agent (CDConA), Cloud Data Correctness Agent (CDCorA), Cloud Data Availability Agent (CDAA) and Cloud Data Integrity Agent (CDIA).

Key words: Cloud computing, agent-oriented methodology, cloud data storage, security, multi agent system, prometheus methodology, prometheus design tool

INTRODUCTION

Security plays an important role in the development of MAS and is considered as one of the main issues to be dealt for agent technology to be widely used outside the research community. As a result, research on security for MAS is an important area within the agent research community. However, the research has been mainly focused on the solution of individual security problems of CDS.

Due to the complexity of CDS security tasks, the proposed system is based on the integration of different types of intelligent agents and hybrid architecture. The design and programming of agents should be focused on maximizing their performance measure which embodies the criterion for success of an agent's behavior (Russell *et al.*, 1995). Most important issues that are required include security of the agents and system (Bradshaw *et al.*, 2010; Hamidi and Mohammadi, 2006). The interface should exhibit intelligent features that assist the cloud user in taking actions to control the security process.

The Prometheus Design Tool (PDT) is a freely available 1 (Padgham and Winikoff, 2002) tool, running under Java 1.5, which supports the software designer who is using the Prometheus methodology. PDT provides graphical support for the design phases of the methodology, allowing the designer to enter and edit diagrams and descriptors for entities. PDT also enforces certain constraints (e.g., that an action performed by an agent in the system overview diagram must also appear in the relevant agent overview diagram) and also checks the design for various consistency conditions. PDT also supports the design and development of intelligent multi-agent systems and complements the Prometheus methodology (Padgham and Winikoff, 2002), although it can also be used with other approaches. It is based on specific agent entities such as goals, plans, percepts, actions and protocols and provides both direction and support to software engineers. It also includes automated support for testing and for partial code production.

There are several researchers' attempts to develop security related ontology (Wang and Guo, 2009; Stepanova *et al.*, 2009; Denker *et al.*, 2005; Blanco *et al.*, 2008; Schumacher, 2003; Kim *et al.*, 2005; Jutla and Bodorik, 2005; Stallings, 2010; Venter and Eloff, 2003) ontology driven and based MAS (Falasconi *et al.*, 1996; Tran and Low, 2008; Beydoun *et al.*, 2009; Sharp *et al.*, 2011), ontology based approach (Noy and McGuiness, 2001) and ontology based development (Tsoumas and Gritzalis, 2006). In order to build ontology for security services, it is beneficial to understand the need for ontology and some security ontology works. However, linkages to domain CDS security are not emphasized. This is where this study shall close the gap. However, the main contribution of this paper is the MAS architecture design shall be conceptualized using Prometheus Design Tool to identify the types of agents and agent architecture that include the gap identified earlier.

None of the existing agent oriented methodologies, to our knowledge, have been demonstrated enough evidence to support claims of adequately integrate security modeling during the whole software development stages. Only recently, some initial steps have been taken towards this direction. Liu *et al.* (2002) has initiated work that provides ways of modeling and reasoning about non-functional requirements (with emphasis on security). Liu *et al.* (2002) is also using the concept of a soft goal to assess different design alternatives and how each of these alternatives would contribute positively or negatively in achieving the soft goal.

In recent years, Multi-Agent System (MAS) has been an active research topic. Due to the difficulties in solving process planning and production scheduling problems using traditional centralized problem solving methodology, MAS approach-a distributed problem-solving paradigm is used as another attempt to solve the planning and scheduling problems. As a distributed problem-solving paradigm, MAS breaks complex problems into small and manageable sub-problems to be solved by individual agents co-operatively.

Prometheus have been chosen compared to other four methodologies (Tropos, MaSE, Use-Case BDI and Gaia) because (a) it covers start-to-end development stages, (b) simple method, (c) easy to understand, (d) mature or have been described in sufficient detail to be of real use, (e) provide tool support and (f) comprehensive documentation for reference.

However, it is not advisable to follow the Prometheus methodology strictly to allow users to choose relevant steps and parts to help solving their problem (Padgham and Winikoff, 2002). All the methodologies discussed above (Gaia, Tropos, MaSE, Prometheus and Use-Case BDI) primarily take an implementation point of view and focus heavily on developing a system rapidly. These methodologies also fall short in non-functional capability considerations of system development. A major decision to be made during the architectural design is what agents should be included.

Several types of agents can be designed to support information security management (Russell *et al.*, 1995). In the proposed system, key agents should be the decision maker agent and controller agent.

Some of the main methods and techniques for analyzing and designing an agent based system have been described. Their framework focuses on four major aspects of a methodology: con-cepts and properties, notations and modeling techniques, process and pragmatics Sturm and Shehory (2004). This technique of evaluation can be viewed as a feature-based technique.

Yu and Cysneiros (2002), compared three methodologies based upon an attribute-based framework which addresses the same four major areas used by Sturm and Shehory (2004). In their research, a case study based on some specific criteria has been selected and the evaluation of some agent oriented methodologies is provided in the context of the proposed case study. Their evaluation framework provides a set of questions to be discussed.

The study presented by Cernuzzi and Rossi (2002) checks the level to which some specific qualitative features exist in a methodology using attributes tree, it calculates the level to which the evaluated method meets the requirements of its users by assigning a weight to each feature of the method, grading each feature and manipulating a weighted average grade.

CLOUD DATA SECURITY POLICIES IN CLOUD COMPUTING

A security policy is a set of rules for determining the maximum permissible access rights for a particular process to a particular segment, given the attributes of both the process and the segment. The following subsections describe several specific security policies that fall under these four goals in our MAS architecture.

Confidentiality policy: In Cloud computing, confidentiality plays a major part especially in maintaining control over organizations' data situated across multiple distributed cloud servers. It is a must when employing a Public cloud due to public clouds accessibility nature. Asserting confidentiality of users' profiles and protecting their data, that is virtually accessed, allows for cloud data security protocols to be enforced at various different layers of cloud applications.

Cloud Data access control issue is mainly related to security policies provided to the users while accessing the data. In a typical scenario, a small business organization can use a cloud provided by some other provider for carrying out its business processes. This organization will have its own security policies based on which each employee can have access to a particular set of data. The security policies may entitle some considerations wherein some of the employees are not given access to certain amount of data. These security policies must be adhered by the cloud to avoid intrusion of data by unauthorized users (Bowers *et al.*, 2009; Kormann and Rubin, 2000).

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 our proposed Formula-Based Cloud Data Access Control (FCDAC), principals are called cloud users. Cloud users get permissions to access resources via membership in roles. Our approach focuses not only of providing a cloud user by a user name and password but by define and enforce expressive and flexible access structure for each cloud user as a logic formula over cloud data file attributes.

Correctness assurance policy: Goal of correctness assurance to ensure cloud users that their cloud data are indeed stored appropriately and kept intact all the time in the cloud to improve and maintain the same level of storage correctness assurance even if cloud users modify, delete or append their cloud data files in the cloud.

In CDS, there are many potential scenarios where data stored in the cloud is dynamic, like electronic documents, photos, or log files etc. Therefore, it is crucial to consider the dynamic case, where a user may wish to perform various block-level operations of update, delete and append to modify the data file. Our MAS architecture uses to secure and organize the CDS blocks as well as proposes to facilitate the correctness of user' data cloud security.

Availability policy: Availability is one of the most critical information security requirements in Cloud computing because it is a key decision factor when deciding among cloud vendors as well as in the delivery models. The service level agreement is the most important document which highlights the trepidation of availability in cloud services and resources between the CSP and client. Therefore by exploring the information security requirements at each of the various cloud deployment and delivery models, vendors and organizations can become confident in promoting a highly protected safe and sound cloud framework.

Integrity policy: Integrity of the cloud data has to deal with how secure and reliable the data cloud computing. This could mean that you have provided secure backups, addressed security concerns and increased the likelihood that your data will be there when you need it. In a cloud environment a certification authority is required to certify entities involved in interactions, these include certifying physical infrastructure servers, virtual servers, environments users and the networks devices. Data integrity in the cloud system means to preserve information integrity (i.e., not lost or modified by unauthorized users). As data is the base for providing cloud computing services, such as Data as a Service (DaaS), Software as a Service (SaaS), Platform as a Service (PaaS), keeping data integrity is a fundamental task. The integrity requirement lies in applying the due diligence within the cloud domain mainly when accessing data. Therefore ACID (atomicity, consistency, isolation and durability) properties of the cloud's data should without a doubt be robustly imposed across all cloud computing deliver models.

PROMETHEUS METHODOLOGY OVERVIEW

Prometheus (Padgham and Winikoff, 2002), consists of three phases, as shown in Fig. 1; system specification, architectural design and detailed design. The first phase, system specification, where the system is specified using goals and scenarios; the system's interface to its environment is described in terms of actions, percepts and external data and functionalities are defined. The second phase, architecture design, where agent types are identified; the system's overall structure is captured in a system overview diagram and scenarios are developed into interaction protocols. The third phase, detailed design, where the details of each agent's internals are developed and defined in terms of capabilities, data, events and plans; process diagrams are used as a stepping stone between interaction protocols and plans. But the use of the JACK development environment is strongly recommended because both Prometheus and JACK are oriented toward BDI agent. Prometheus development tool PDT provided an option to the developer to automatically generate JACK skeleton java code.

System specification phase: The system specification phase focus on the analysis techniques. For the purpose of clarity, some low level scenario details are ignored in this description to instead concentrate on the big picture. The system specification phase focuses on system requirement

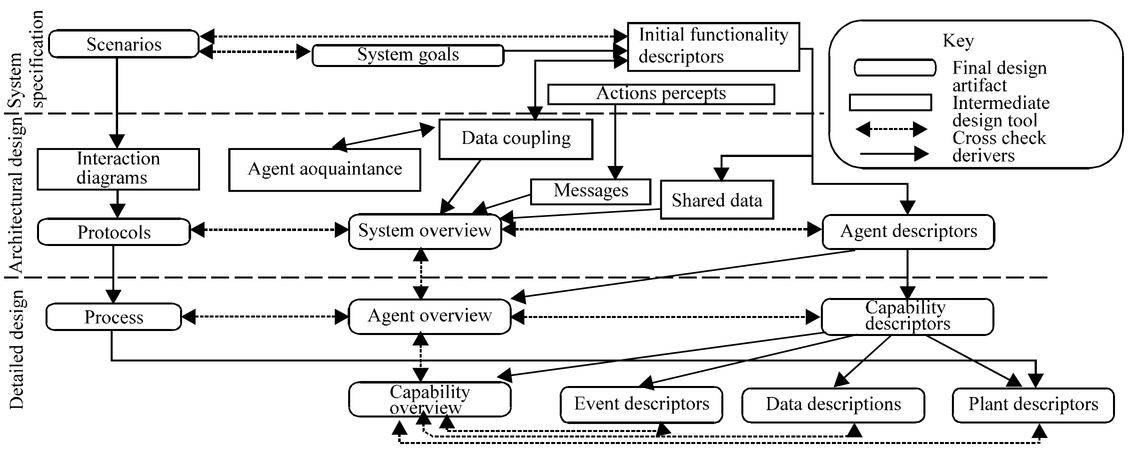

Fig. 1: The phases of the prometheus methodology

definition to capture the system goals and sub-goals. The system goals are the centre construct of the system specification and are fundamental to AOS. The goals are systematically captured by searching for intentional words in the initial system documents. Once the main goals are identified, the other goals and sub-goals can emerge by using, refinement techniques (asking how? and why) (Van Lamsweerde, 2001).

Architecture design phase: The architecture design phase will use the system specification artefacts to build the system architecture. The system architecture will be developed in three main steps. In the first step, the application agent types are specified; in the second step, the system interactions are specified, in the third step, the system overviews are designed.

Specification of agent types: The objective of this step is to identify the types of agents embraced by the required application. During step one; agent descriptors will also be developed. This objective will be reached through the implementation of low coupling and highly cohesion principals in the diagram of system functionalities. The techniques are centre around the relationships between the functionalities and data related to these functionalities.

Developing system interactions: Specifying the system interaction (interactions between agents) is the second step of the architecture design phase. The purpose of this step is to capture the dynamic aspects of the system, by developing interaction diagrams from a scenario, generalizing interaction diagrams to interaction protocols then developing protocol and messages descriptors. Prometheus Development Tool (PDT) has the capability to compile the protocol scripts allocated in the system overview design and then generate the protocol diagram automatically.

Detailed design phase: Prometheus phases are integrated with each other. The detailed design phase uses the system architecture artefacts and develops two main aspects: first the internal individual agent capabilities and process; second the capabilities, plan and events analysis.

Capabilities: The first step in the detailed design is to compile the agent descriptor and agent capabilities to develop the agent overview diagram.

Process specification: The next step in building the individual agent is to identify the internal process specification of a single agent and specify its activities structure. Prometheus does borrow agent UML (AUML) activity diagram notation to present the process specifications. This means PDT does not provide any support to the process diagram. However, the best method to start building this task is to look into the protocols involved in the agent structure, the scenarios developed and the goals of the agent.

Capabilities overview diagram: The last milestone in the detailed design phase is to turn the analysis artefacts into the implementation platform and achieve Prometheus agent architecture, which is based on the concept Belief-Desire-Intention (BDI). This will enforce the analysis process to focus on the internal structure of the system capabilities. Each capability is then decomposed into its lower level and the set of plans that provide the details of how to achieve the goals retrieved.

DESIGN BASED PROMETHEUS METHODOLOGY

PROPOSED MAS ARCHITECTURE

In MAS architecture, we proposed five types of agents: Cloud Service Provider Agent (CSPA), Cloud Data Confidentiality Agent (CDConA), Cloud Data Correctness Agent (CDCorA), Cloud Data Availability Agent (CDAA) and Cloud Data Integrity Agent (CDIA). The proposed MAS overview diagram illustrated in Fig. 2.

Figure 3 presents the agent overview diagram developed as an example using PDT detailed design process. PDT has the ability to validate each design entity dynamically while the development process is running.

The rest of agents are described as follows:

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

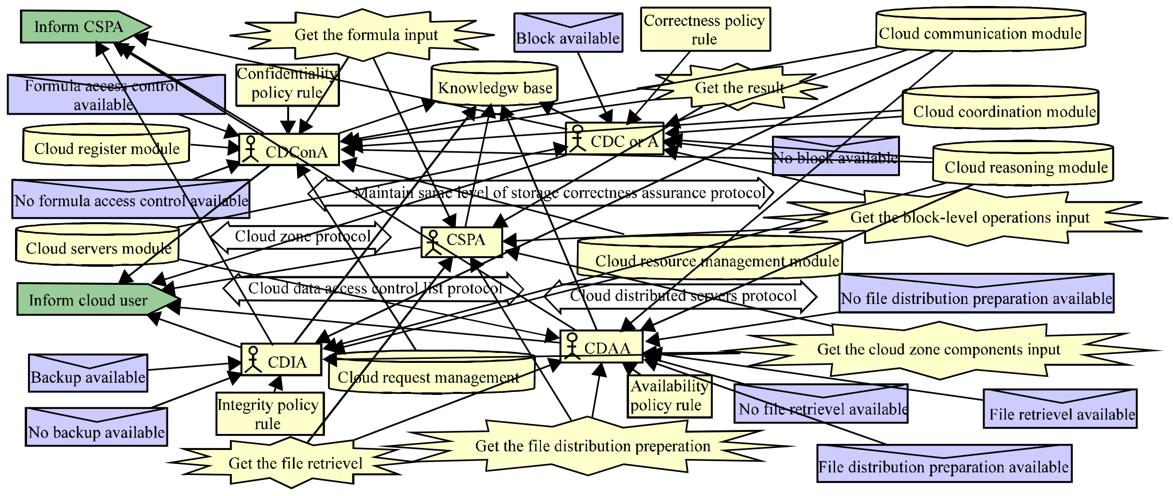
Inform CSPA
Get the formula input
Block available
Correctness policy rule
Cloud communication module
Formula access control available
Confidentiality policy rule
Knowledgw base
Get the result
Cloud coordination module
Cloud register module
CDConA
CDC or A
No block available
Cloud reasoning module
No formula access control available
Maintain same level of storage correctness assurance protocol
Get the block-level operations input
Cloud servers module
Cloud zone protocol
CSPA
Cloud resource management module
Inform cloud user
Cloud data access control list protocol
Cloud distributed servers protocol
No file distribution preparation available
CDAA
Get the cloud zone components input
Backup available
CDIA
Cloud request management
Availability policy rule
No file retrievel available
File retrievel available
No backup available
Integrity policy rule
Get the file distribution preperation
Get the file retrievel
File distribution preparation available

Fig. 2: Proposed MAS overview diagram

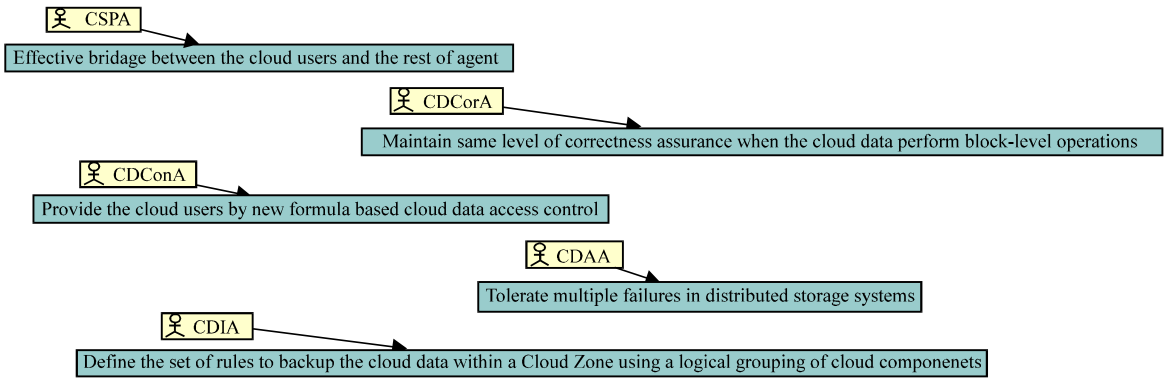
CSPA
Effective bridage between the cloud users and the rest of agent
CDCorA
Maintain same level of correctness assurance when the cloud data perform block-level operations
CDConA
Provide the cloud users by new formula based cloud data access control
CDAA
Tolerate multiple failures in distributed storage systems
CDIA
Define the set of rules to backup the cloud data within a Cloud Zone using a logical grouping of cloud componenets

Fig. 3: Agents overview diagram

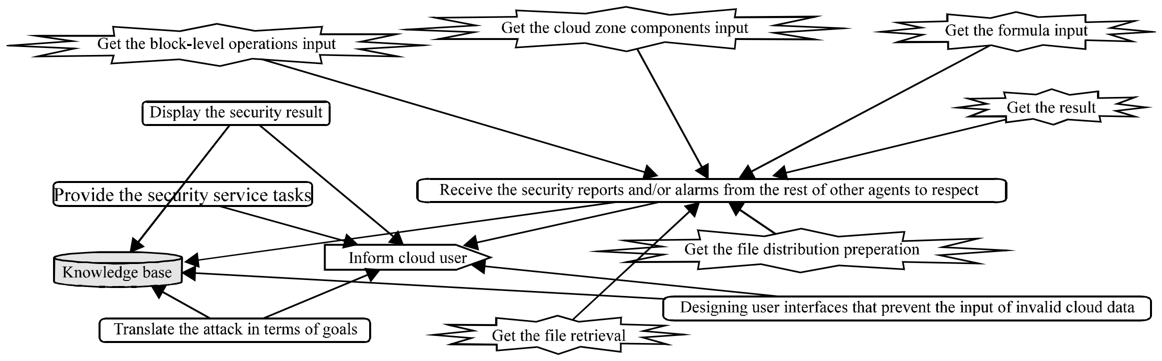
Get the block-level operations input
Get the cloud zone components input
Get the formula input
Display the security result
Get the result
Provide the security service tasks
Receive the security reports and/or alarms from the rest of other agents to respect
Get the file distribution preperation
Knowledge base
Inform cloud user
Designing user interfaces that prevent the input of invalid cloud data
Translate the attack in terms of goals
Get the file retrieval

Fig. 4: Detail design of CSPA architecture

system under the behavior of CSP. CSPA has the following actions as illustrated in CSPA architecture (Fig. 4):

- Provide the security service task according to the authorized service level agreements (SLAs) and the original message content sent by the CDCorA, CDConA, CDAA and CDIA
- Display the security policies specified by 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

Cloud data confidentiality agent (CDConA): This agent facilitates the security policy of confidentially for CDS. Main responsibility of this agent is to provide 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 (Talib *et al*., 2011a). 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 new access control is called as:

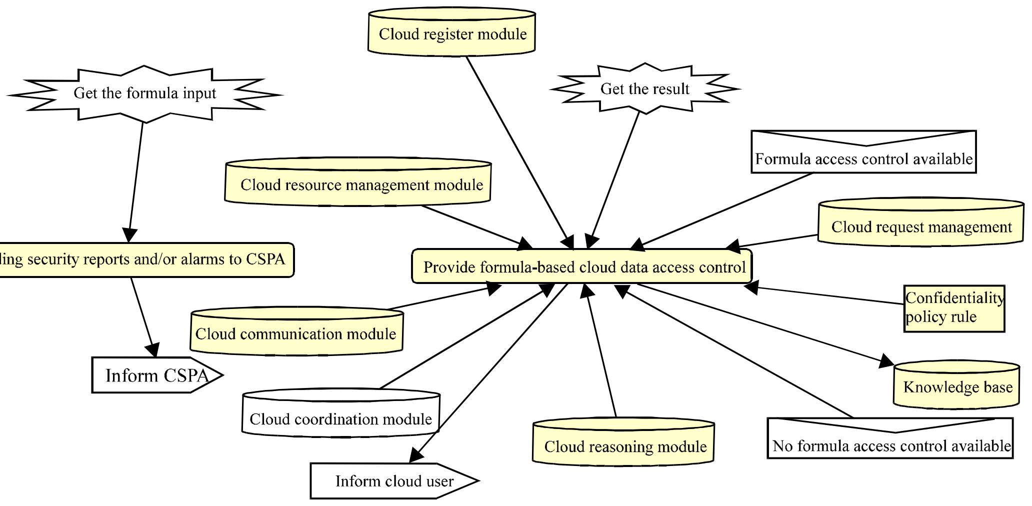
Cloud register module
Get the formula input
Get the result
Formula access control available
Cloud resource management module
Cloud request management
Sending security reports and/or alarms to CSPA
Provide formula-based cloud data access control
Confidentiality policy rule
Cloud communication module
Inform CSPA
Knowledge base
Cloud coordination module
Cloud reasoning module
No formula access control available
Inform cloud user

Fig. 5: Detail design of CDConA architecture

- Formula-based cloud data access control (FCDAC)

This agent is also notifies CSPA in case of any fail caused of the techniques above by sending security reports and/or alarms.

Formula-Based Cloud Data Access Control (FCDAC) and also named as a SecureFormula 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. SecureFormula is an additional confidentiality layer used by our system to verify that the cloud users' login page is a genuine.

If you are a cloud user, you are required to register first to the system and write your valid email and enter your SecureFormula during your first login. Your SecureFormula will be sent to your email. Be ensured that, your SecureFormula is not your password. Do not set your SecureFormula to be the same as your password.

Sign in from your computer:

- Enter your Cloud User ID
- Verify that your SecureFormula image is correct
- Confirm by entering your password

Our confidentiality layer guaranteed that, even if your password is correct and your SecureFormula is incorrect, then you will not be able to login.

The architecture of CDConA consists of five modules, as shown in Fig. 5. Cloud Communication Module provides the agent with the capability to exchange information with other agents, including the CDConA, CDCorA, CDAA, CDIA 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

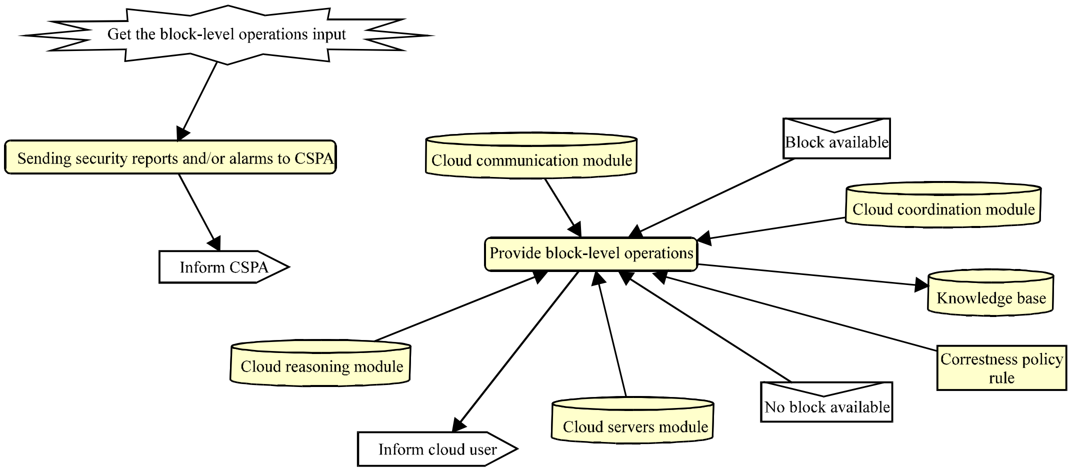
Get the block-level operations input
Sending security reports and/or alarms to CSPA
Inform CSPA
Cloud communication module
Block available
Cloud coordination module
Provide block-level operations
Knowledge base
Correstness policy rule
Cloud reasoning module
Cloud servers module
No block available
Inform cloud user

Fig. 6: Detail design of CDCorA architecture

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 (Talib *et al*., 2011a).

Cloud data correctness agent (CDCorA): This agent facilitates the security policy of correctness assurance for CDS. Main responsibility of this agent is to perform various block-level operations and generate a correctness assurance when the cloud user performs update operation, delete operation, append to modify operation or insert operation. This agent notifies CSPA in case of any fail caused of the techniques above by sending security reports and/or alarms.

The architecture of the CDCorA consists of four modules, as shown in Fig. 6. Cloud Communication Module provides the agent with the capability to exchange information with CSPA. Cloud Coordination Module provides the agent with the following mechanisms. If the data is updated then the data encryption is performed. If the data is deleted then the data encryption is performed. If the data is Append then the data encryption is performed. If the data is inserted then the data encryption is performed. Cloud Reasoning Module calculates the necessary amount of cloud resources to complete the service based on the required Service Level Agreements (SLA) by utilizing the information obtained from the knowledge base and the correctness assurance policy rule. Cloud Servers Module performs the block-level operations of encryption and decryption when the cloud user update, delete, append and insert his/her data.

In CDS, there are many potential scenarios where data stored in the cloud is dynamic, like electronic documents, photos, or log files etc. Therefore, it is crucial to consider the dynamic case, where a cloud user may wish to perform various block-level operations of update, delete and append to modify the data. Our proposed correctness assurance protocol is not going to be genuine if there is absent of SecureFormula. So in case of: Update operation: The cloud user needs to enter his/her SecureFormula plus 00, Delete operation: The cloud user needs to enter his/her SecureFormula plus 01, Append operation: The cloud user needs to enter his/her SecureFormula plus 10 and Modify operation: The cloud user needs to enter his/her SecureFormula plus 11.

Cloud data availability agent (CDAA): This agent facilitates the security policy of availability for CDS. Main responsibility of this agent is to receive and display the security issues that offer by

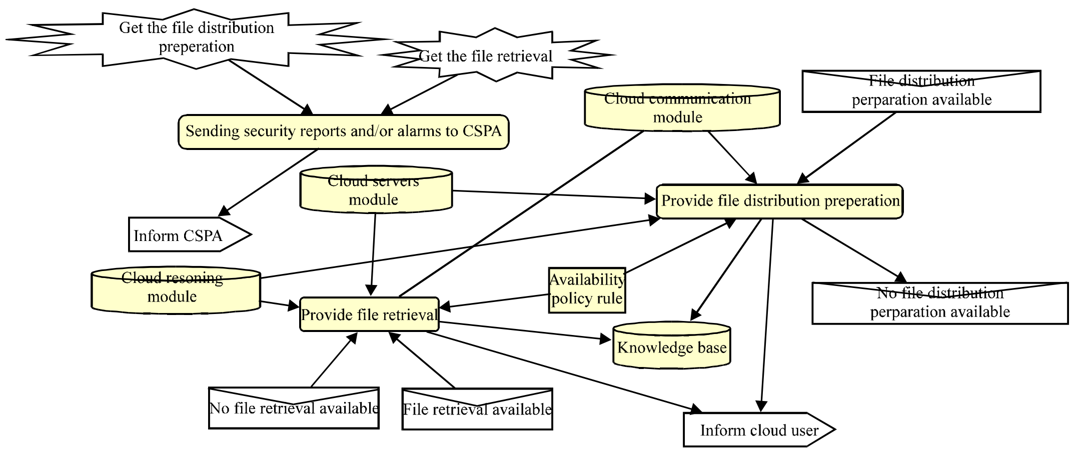
Get the file distribution preperation
Get the file retrieval
Sending security reports and/or alarms to CSPA
~~Cloud communication~~ module
~~File distribution~~ perparation available
~~Cloud servers~~ module
Provide file distribution preperation
Inform CSPA
~~Cloud resoning~~ module
Availability policy rule
Provide file retrieval
~~No file distribution~~ perparation available
Knowledge base
No file retrieval available
File retrieval available
Inform cloud user

Fig. 7: Detail design of CDAA architecture

its sub-agents of CDDPA and CDRA. CDAA facilitate two new techniques of file distribution preparation and file retrieval (Talib *et al.*, 2011b).

This agent is also notifies CSPA in case of any fail caused of the techniques above by sending security reports and/or alarms.

Cloud data availability is to ensure that the cloud data processing resources are not made unavailable by malicious action. Our MAS architecture is able to tolerate multiple failures in cloud distributed storage systems.

To ensure the availability, we explain the notions of global and local cloud attack blueprints. To detect intrusions, the CDAA receives a set of goals representing the global cloud attack blueprints. To recognize this global cloud attack blueprint, it must be decomposed in local cloud sub-blueprints used locally by the different agents distributed in the CDS. In general agents can detect only local cloud attacks because they have a restricted view of the CDS. So, we make a distinction between a global cloud attack blueprint and local cloud sub-blueprints. A global cloud blueprint is an attack blueprint, derived from the security policies specified at a high level by the CSPs, that the MAS must detect and the detection of this blueprint will be notified only to CDAA. A local cloud blueprint is a blueprint derived from the global cloud blueprint but that must be detected by local agents. For a CDAA over-viewing the global cloud attack blueprint the probability of an attack is equal to 1, while for the local agent it is below 1 (Talib *et al.*, 2011b).

The architecture of the CDAA consists of three modules, as shown in Fig. 7. Cloud Communication Module provides the agent with the capability to exchange information with CDAA and CSPA. Cloud Servers Modules provides the agent with the following mechanisms: (1) disperse the data file redundantly across a set of distributed servers and (2) enable the cloud user to reconstruct the original data by downloading the data vectors from the servers. Cloud Reasoning Module provides the CDAA with the specific misbehaving server(s) and server colluding attacks by utilizing the information obtained from the knowledge base and the availability policy rule.

Cloud data integrity agent (CDIA): This agent facilitates the security policy of integrity for CDS. It is used to enable the cloud user to reconstruct the original cloud data by downloading the cloud data vectors from the cloud servers. Main responsibility of this agent is backing up the cloud data regularly from "CloudZone" and sending security reports and/or alarms to CPSA when:

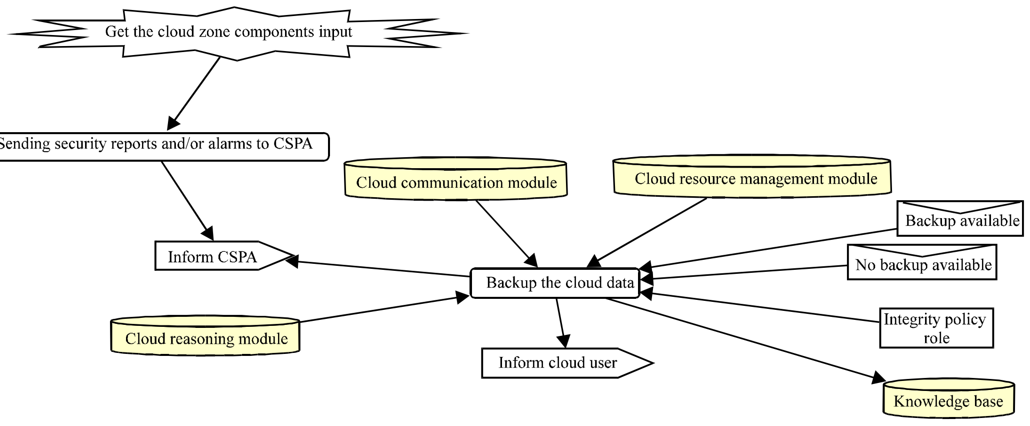

Fig. 8: Detail design of CDIA architecture

- Human errors when cloud data is entered
- Errors that occur when cloud data is transmitted from one computer to another
- Software bugs or viruses
- Hardware malfunctions, such as disk crashes

Our proposed integrity layer named as CloudZone. In CloudZone, we introduce the first provably-secure and practical backup cloud data regularly that provide reconstruct the original cloud data by downloading the cloud data vectors from the cloud servers.

CloudZone requirements

- CloudZone only backs up the MS SQL databases. It does not back up other MS SQL files such as program installation files, etc
- CloudZone does not support component-based backup
- CloudZone does not use Visual SourceSafe (VSS) for backup and restore
- CloudZone supports backup and recovery of Windows Oracle 10 g

With CloudZone cloud backup, you can select any of the following as backup objects:

- Oracle Server 10 g running on Windows
- Microsoft SQL Server 2000, 2005 and 2008
- Microsoft Exchange Server 2003 and 2007

The architecture of the CDIA consists of three modules, as shown in Fig. 8. Cloud Communication Module provides the agent with the capability to exchange information with CDIA, CDConA, CDCorA, CDAA and CSPA. Cloud Resources Management Modules provides the agent with the following mechanisms. If the CDIA registered as CDIA-VIP then back-up of the data is performed successfully. If the CDIA did not register as CDIA-VIP, it asks the cloud user to back-up the data manually. Cloud Reasoning Module shows the reasons of in case the result of the back-up the data is failed by utilizing the information obtained from the knowledge base and the integrity policy rule.

CONCLUSION

When choosing a methodology for a problem, considering the complexity of methodologies is necessary. Methodologies which propose large and complex models in their development phases or ones which have lots of dependencies between their models may be unsuitable for analyzing and designing a system.

It can be concluded that based on the characteristics of CDS security, security based multi-agent system architecture supports faster security of cloud resources irrespective of its location and resolve a task based on planning algorithms using intelligence agents.

The system is based on multi-agent system architecture, consisting of CDConA, CDCorA, CDAA, CDAA and one effective bridge agent CSPA (Cloud Service Provider Agent). The agent is designed using Prometheus methodology.

REFERENCES

Beydoun, G., G. Low, H. Mouratidis and B. Henderson-Sellers, 2009. A security-aware metamodel for multi-agent systems (MAS). Inform. Software Technol., 51: 832-845.

Blanco, C., J. Lasheras, R. Valencia-Garc, E. Fernandez-Medina, A. Toval and M. Piattini, 2008. A systematic review and comparison of security ontologies. Comput. Inform. Sci., 1: 813-820.

Bowers, K.D., A. Juels and A. Oprea, 2009. HAIL: A high-availability and integrity layer for cloud storage. http://eprint.iacr.org/2008/489.pdf

Bradshaw, J.M., N. Suri, M. Kahn, P. Sage, D. Weishar and R. Jeffers, 2010. Terraforming Cyberspace: Toward a policy-based grid infrastructure for secure, scalable and robust execution of Java-based multi-agent systems. http://www.neotake.com/ebook/terraforming-cyberspace-toward-a-policy-based-gri/xzwbowj.html

Cernuzzi, L. and G. Rossi, 2002. On the evaluation of agent oriented modeling methods. Proceedings of the Workshop on Agent Oriented Methodology, November, 2002, Seattle, WA, USA., pp: 21-30.

Denker, G., L. Kagal and T. Finin, 2005. Security in the semantic web using owl. Inform. Security Technical Report, 10: 51-58.

Falasconi, S., G. Lanzola and M. Stefanelli, 1996. Using ontologies in multi-agent systems. Proceedings of the 10th Workshop on Knowledge Acquisition for Knowledge-Based Systems, November 9-14, 1996, Banff, Canada.

Hamidi, H. and K. Mohammadi, 2006. Modeling fault tolerant and secure mobile agent execution in distributed systems. Int. J. Intell. Inform. Technol., 2: 21-36.

Jutla, D.N. and P. Bodorik, 2005. Sociotechnical architecture for online privacy. Security Privacy, 3: 29-39.

Kim, A., J. Luo and M. Kang, 2005. Security ontology for annotating resources. Comp. Sci., 3761: 1483-1499.

Kormann, D.P. and A.D. Rubin, 2000. Risks of the passport single signon protocol. Comput. Networks, 33: 51-58.

Liu, L., E. Yu and J. Mylopoulos, 2002. Analyzing security requirements as relationships among strategic actors. Proceedings of the 2nd Symposium on Requirements Engineering for Information Security, October 16, 2002, Raleigh, North Carolina, pp: 1-14.

Noy, N.F. and L.D. McGuiness, 2001. Ontology development 101: A guide to creating your 1st ontology. Standard Knowledge Systems Laboratory Technical Report KSL-01-05 and standard Medical Informatics Technical Report SMI-2001-0880.

Padgham, L. and M. Winikoff, 2002. Prometheus: A methodology for developing intelligent agents. Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems: Part 1, July 15-19, 2002, Bologna, Italy, pp: 37-38.

Russell, S.J., P. Norvig, J.F. Canny, J.M. Malik and D.D. Edwards, 1995. Artificial Intelligence: A Modern Approach. Prentice Hall, Englewood Cliffs, NJ., USA.

Schumacher, M., 2003. Security Engineering with Patterns: Origins, Theoretical Model and new Applications. Springer-Verlag, New York, USA., ISBN-13: 9783540407317, Pages: 208.

Sharp, B., A.S. Atkins and H. Kothari, 2011. An ontology based multi-agent system to support HABIO outsourcing framework. Expert Syst. Appl. Int. J., 38: 6949-6956.

Stallings, W., 2010. Cryptography and Network Security: Principles and Practices. 5th Edn., Prentice-Hall, Upper Saddle River, New Jersey, ISBN-10: 0136097049, pp: 744.

Stepanova, D., S. Parkin and A. van Moorsel, 2009. A knowledge base for justified information security decision-making. Newcastle University, http://www.techrepublic.com/whitepapers/a-knowledge-base-for-justified-information-security-decision-making/1174737.

Sturm, A. and O. Shehory, 2004. A framework for evaluating agent-oriented methodologies. J. Autonomous Agents Multi-Agent Syst., 8: 131-164.

Talib, A.M., R. Atan, R. Abdullah and M.A.A. Murad, 2011a. Ensuring security and availability of cloud data storage using multi agent system architecture. Proceedings of the 3rd Malaysia Joint Conference on Artificial Intelligence, July 20?22, 2011, UNITEN Putrajaya Campus, Malaysia.

Talib, A.M., R. Atan, R. Abdullah and M.A.A. Murad, 2011b. Towards new data access control technique based on multi agent system architecture for cloud computing. Proceedings of the International Conference on Digital Information Processing and Communications, July 7-9, 2011, Czech Republic, Ostrava.

Tran, Q.N.N. and G. Low, 2008. MOBMAS: A methodology for ontology-based multi-agent systems development. Inform. Software Technol., 50: 697-722.

Tsoumas, B. and D. Gritzalis, 2006. Towards an ontology-based security management. Proceedings of the 20th International Conference on Advanced Information Networking and Applications, April 18-20, 2006, Vienna, pp: 985-992.

Van Lamsweerde, A., 2001. Goal-oriented requirements engineering: A guided tour. Proceedings of the 5th IEEE International Symposium on Requirements Engineering, August 27-31, 2001, Toronto, Ont., Canada, pp: 249-262.

Venter, H.S. and J.H.P. Eloff, 2003. A taxonomy for information security technologies. Comput. Security, 22: 299-307.

Wang, J.A. and M. Guo, 2009. OVM: An ontology for vulnerability management. Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies, April 13-15, 2009, Knoxville, TN, USA.

Yu, E. and L.M. Cysneiros, 2002. Agent-oriented methodologies-Towards a challenge exemplar. Proceedings of the 4th International Workshop on Agent-Oriented Information Systems, May 27-28, 2002, Toronto.