A Policy-Based Management Framework For Cloud Computing Security

By
Olubisi Atinuke Runsewe

Thesis submitted to the Faculty of Graduate and Postdoctoral Studies
in partial fulfilment of the requirements for the degree of

Master of Science
in
Electronic Business Technologies

uOttawa

L'Université canadienne
Canada's university

University of Ottawa
Ottawa, Ontario

Abstract

Cloud Computing has changed how computing is done as applications and services are being consumed from the cloud. It has attracted a lot of attention in recent times due to the opportunities it offers. While Cloud Computing is economical, the security challenges it poses are quite significant and this has affected the adoption rate of the technology. With the potential vulnerabilities being introduced by moving data to the cloud, it has become imperative for cloud service providers to guarantee the security of information, leaving cloud service consumers (e.g., enterprises) with the task of negotiating the terms and conditions of services provided by the cloud service providers as well as trusting them with their data. Although various security solutions used for addressing the security of data within the enterprises are now being applied to the cloud, these security solutions are challenged due to the dynamic, distributed and complex nature of the cloud technology.

This thesis proposes a novel Policy-Based Management (PBM) framework capable of achieving cross-tenant authorization, handling dynamic and anonymous users while reducing the security management task to address cloud security. The framework includes an access control model adapted to the cloud environment that adopts features from role-based, task-based and attribute-based access control frameworks for a fine-grained access control. We demonstrate how this framework can be applied to develop an access control system for an enterprise using cloud services. The framework verifies the correctness of access control policies for cloud security through reasoning technique.

Acknowledgements

Above all, I would like to thank God for the strength and perseverance he bestowed upon me during this research thesis.

My deepest appreciation goes to my family especially my husband, Olupaseayo for his love, encouragement, patience and unconditional support, both financially and emotionally throughout my program. My daughters (Kanyinsola, Sophia and Audrey), for being the best, as always, for which my mere expression of thanks does not suffice.

This thesis would not have been possible without the help, support and patience of my supervisor, Dr. Samaan, who has been invaluable on an academic level for which I am grateful. The experience has been an interesting and rewarding one.

Last but not least, my deepest gratitude goes to my parents (late father, who passed during the completion of my thesis) and in-laws who are my greatest teachers in life lessons. Thanks for their continued love, support, efforts and their constant encouragement throughout my studies.

Table of Contents

List of Figures

List of Tables

List of Acronyms

AA	Attribute Authority
ABAC	Attribute Based Access Control
ACL	Access Control List
ACM	Access Control Mechanism
ASP	Answer Set Programming
CIA	Confidentiality, Integrity and Availability
COPS	Common Open Policy Service
CSA	Cloud Security Alliance
DAC	Discretionary Access Control
DLV	DataLog with disjunction (V)
DSR	Design Science Research
DMTF	Distributed Management Task Force
EIS	Enterprise Information Systems
EMR	Electronic Medical Record
FHE	Fully Homomorphic Encryption
HIPAA	Health Insurance Portability and Accountability Act
IAAS	Infrastructures as a Service
IBAC	Identity Based Access Control
IDP	Identity Provider
IETF	Internet Engineering Task Force
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
MAC	Mandatory Access Control
MTAC	Multi-tenancy Access Control
NIST	National Institute of Standards and Technology
PAAS	Platform as a Service
PAP	Policy Administration Point
PBM	Policy Based Management
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PIP	Policy Information Point
PIPEDA	Personal Information Protection and Electronic Documents Act
PKI	Public Key Infrastructure
QOS	Quality of Service
RBAC	Role Based Access Control
REA	Role Enablement Authority
RUBAC	Rule Based Access Control
SAAS	Software as a Service

SAML	Security Assertion Markup Language
SEC2	Secure Elastic Cloud Computing
SOA	Service Oriented Architecture
SP	Service Provider
TBAC	Task Based Access Control
TPM	Trusted Platform Module
UCON	Usage Control Model
UMU-XACML	University of Murcia (UMU) - XACML
VLAN	Virtual Local Area Network
VM	Virtual Machine
WSPL	Web Services Policy Language
X-GTRBAC	XML-based Generalized Temporal Role Based Access Control
XAAS	Anything as a Service
XACML	eXtensible Access Control Markup Language
XML	eXtensible Markup Language

Chapter 1: Introduction

Cloud Computing has changed how computing is done as applications and services are consumed from the cloud. It is a business model based on the concept of multi-tenancy, virtualization and shared infrastructure (Mell & Grance, 2011). According to the National Institute of Standards and Technology (NIST), Cloud Computing is defined as ”a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes availability and it is composed of five essential characteristics, three delivery models, and four deployment models” (Mell & Grance, 2011).

Cloud Computing has attracted a lot of interest in recent times from individuals, groups, enterprises and government due to the opportunities offered, which includes reduced cost, application portability (i.e., users can work from home, or at client locations), enhanced collaboration, agility, scaling and availability through optimized and efficient computing (Min, Young-Gi, Hyo-Jin Shin & Young-Hwan Bang, 2012). Cloud services such as Software as a Service (SaaS), Platform as a Service (PaaS) or Infrastructure as a Service (IaaS) are made available to cloud service consumers in a pay-as-you-go model through a web browser, or through a mobile or desktop application from cloud service providers such as Google, Amazon Web Services and Microsoft, which often consist of various applications, databases, and typically operate with multiple users, multiple tenants and sometimes span multiple geographical regions.

Despite all the benefits gained from using cloud services, security has been found to be the biggest concern hindering the adoption of the cloud technology by enterprises and individuals. Within the enterprises, various approaches have been used to address security and these have been based on control (i.e., control of information, devices, and infrastructure) inside the enterprise perimeter. However, with Cloud Computing, the enterprise information is stored outside the enterprise perimeter beyond the firewall, rendering the traditional view of the perimeter obsolete to address the security challenges enterprises are exposed.

Securing information in the cloud to meet the needs of enterprises has become one of the most challenging ongoing researches for Cloud Computing. More recently, Policy-Based Management (PBM), which has been implicitly used to manage security in industries such as telecommunication and distributed systems, are now being adopted within the cloud environment to enforce security (Waller et al, 2011). Policies are a well-known approach to protecting security of users in dynamic, distributed environments.

Within the enterprise, access control is considered as a fundamental aspect of the overall security solution to prevent sensitive data from unauthorized access of malicious users and access control models are usually seen as policy frameworks for implementing and ensuring the integrity of security policies that mandate how information can be accessed and shared on a system (Reeja, 2012). Various access control frameworks and models have been proposed defining a collection of standards and procedures to grant the very basic level of protection according to security requirements.

Traditional access control models such as the discretionary, the mandatory and the role-

based access control are used within the enterprise to address security, however, these models are not sufficiently expressive for a highly flexible and dynamic environments as the cloud as they were designed to support the enterprise. Access control and controlled disclosure of certain fragments or versions of information is still a research challenge and an active research field for Cloud Computing security (Paladi, Gehrmann & Morenius, 2013).

This paper proposes a novel policy-based framework for access control in the cloud because traditional models are challenged. We employ an attribute-driven, role and task based access control model in our framework that allows for the evaluation and enforcement of policies at run-time. Our framework is capable of achieving cross-tenant authorization and supporting fast revocation of permissions, constraints such as dynamic separation of duties, and applications such as task management. Contrary to existing static models for access control (e.g., access control lists (ACL) in which the relationship between users and permissions is largely unchanging during operation of the security system), the use of attributes in our framework allows for dynamic permission validity. Since roles are not sufficiently expressive to meet the needs of cloud applications, the framework also leverages attributes of users with roles to achieve fine-grained access control on cloud resources. Additionally, the introduction of a trust mechanism allows the exchange of authorization information among cloud tenants.

1.1 Research Motivation

Over the past few years, Cloud Computing has gained increased interest from individual, groups, enterprises and government looking to reduce their costs of IT based on the offering

of services such as Software as a Service (SaaS), Platform as a Service (PaaS) or Infrastructure as a Service (IaaS) to cloud consumers. Figure 1.1 shows the search interest in Cloud Computing over time. From Figure 1.1, the numbers on the graph reflect how many searches have been done for a particular term, relative to the total number of searches done on Google over time. They do not represent absolute search volume numbers, because the data is normalized and presented on a scale from 0-100.

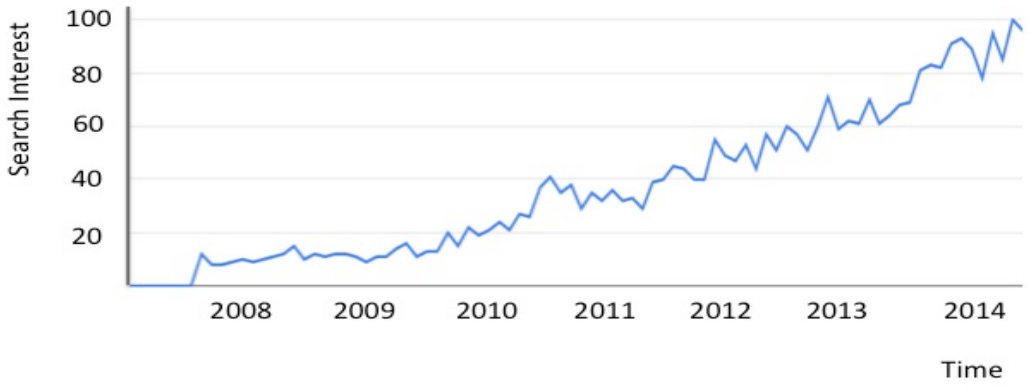


Figure 1.1: Search Interest for Cloud Computing as at April 2014 (Google, 2014)

With this increased interest, security has become a major concern among several other issues such as availability, performance, resiliency, interoperability, data migration and transition from legacy systems as indicated in a survey carried out by IDC in 2009. Figure 1.2 shows that 87.5% of responders cited security as a major concern to cloud services consumption. (IDC, 2009).

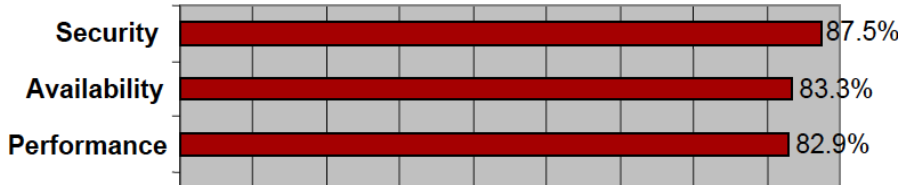
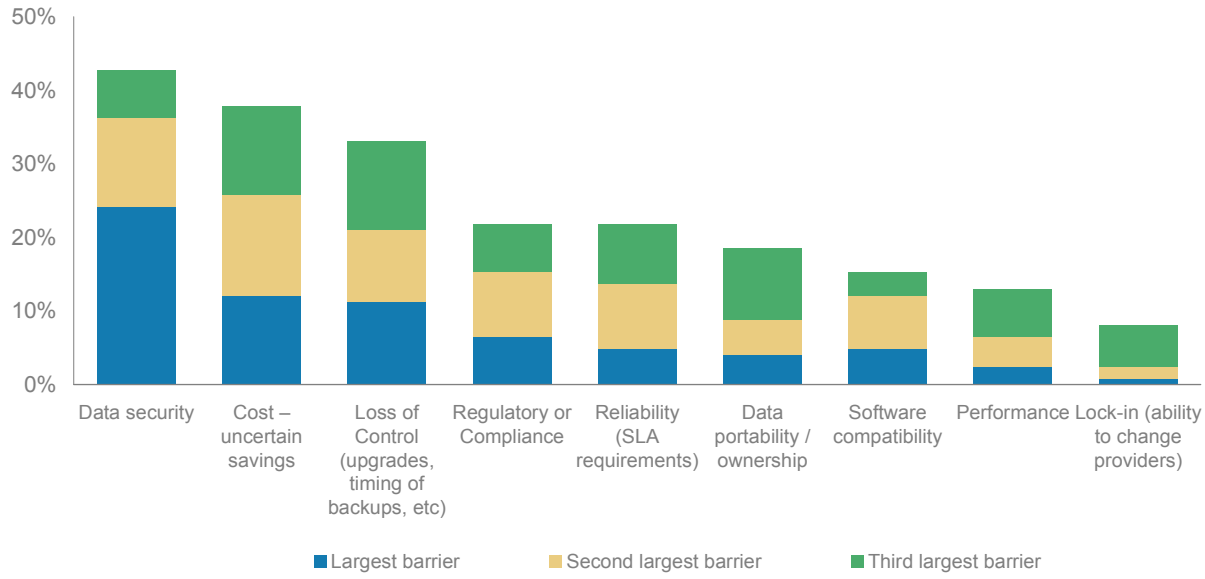


Figure 1.2: IDC Survey of Cloud Computing Security (IDC, 2009)

In another survey carried out by Morgan Stanley in 2011, security was found to be the largest barrier to cloud adoption, though it has slowly dissipated over the past few years, but it is still the largest barrier. Figure 1.3 shows the various forms of concerns with data security topping the list.



Source: AlphaWise[SM], AlphaWise[SM], Morgan Stanley Research

Figure 1.3: Security still the Largest Barrier to Cloud Adoption (Morgan Stanley, 2011)

Security consciousness and concerns arise within the enterprise as soon as applications are run in the cloud beyond the firewall. In an enterprise, sensitive information continues to reside within the enterprise boundary and it is subject to physical, logical and personnel security, and access control policies. However, in the cloud, the enterprise information is stored outside the enterprise boundary therefore additional security checks to ensure information security are needed. In many cases, securing information involves limiting the types of operations that can be performed as well as the efficiency of information replication and synchronization. Even though cloud security has improved a lot during the past few

years, it is still not up to the standards of most large enterprises.

1.2 Problem Statement

Enterprises and individuals have been hesitant to adopt Cloud Computing because of current cloud systems' inability to provide varying levels of security to various types of data. The cloud provides different services like SaaS, PaaS, and IaaS and this places different levels of security requirements in the cloud environment. Well-known security issues such as data loss, malicious users, unauthorized disclosure, pose serious threats to enterprise data. Moreover, the multi-tenancy model and pooled computing resources in Cloud Computing have introduced new security challenges that require novel techniques to tackle them. Multi-tenancy refers to a principle where the same service instance of the software runs on a server, serving multiple consumers (tenants).

Security issues identified by Gartner in 2008 that need to be addressed before enterprises consider switching to the Cloud Computing model are access control, compliance, data location, data segregation, availability, recovery and, long-term viability (Gartner, 2008). To address access control within enterprises, various levels of security monitoring or rules for different devices are used. Whereas, in the cloud, the security layers of the cloud platform are all merged into a single platform with fewer controls in place since most cloud services are commercial, multi-tenant facilities, protected using passwords, and once users login, they have access to all the resources. Without the tools to provide access controls measures that only let people with certain attributes access certain objects or resources, the cloud remains an insecure platform.

This research addresses how access can be given to resources based in the cloud because the traditional model of application-centric access control where each application keeps track of its collection of users and manages them is not feasible in the cloud (Min, Min, Shin & Bang, 2012). In the cloud, big players such as Amazon and Windows Azure use access control lists (ACLs) to control access to resources. This approach has limitations both in its ability to easily scale and it fails to enforce the principle of least privilege. ACLs are maintained within subtenants with no option to grant access permission to users outside the subtenant (Harnik et. al., 2011). Some recent efforts (Singh & Singh, 2013), (Khan, 2012), (Zhu, Liu & Song, 2011), (Gitanjali, Sukhjit & Jaitley, 2013), (Sirisha & Kumari, 2010) that use policy-based approaches such as role-based access control and attribute-based access control mechanisms to ensure that authorized users access the information and system usually result in the adjustment of the security requirements to fit the mechanism at hand, leading to limitations in policy specification. Nevertheless, the lack of generality, both in modeling access requirements and accessed resource, hinders the reuse of these policy frameworks and their adaptation to new cloud scenarios.

Other approaches to mitigate access control issues in the cloud apply cryptographic techniques. These techniques are complex and data owner suffer heavy computational overheads for key distribution and data management. Hence, the performance impact of cryptographic operations due to the time required is questioned (Wood et al., 2009). Moreover, the use of encryption limits data search and use, and to achieve the right tradeoff between security, functionality and efficiency can be difficult. By this, a system that controls access to objects/resources based on authorization attributes of subjects, attributes of objects/resources as well as system attributes, which conform to policies, is required.

1.3 Research Objective

This study proposes an expressive and flexible Policy-based Management (PBM) framework to address access control issues for Cloud Computing security. With this framework, an access structure can be enforced on each user, which precisely designates the set of resources that the user is allowed to access. To this end, the thesis objectives are to:

1. Investigate the characteristics of Cloud Computing and specify the requirements for designing an access control solution that will support those characteristics.
2. Analyze the current access control models for Cloud Computing and evaluate their suitability in line with the identified Cloud Computing access control requirements.
3. Derive a framework capable of handling dynamic and anonymous users, and reducing security management tasks.

Table 1.1 summarizes the research framework.

Table 1.1: Research Framework

Steps	Description
Observation	As Cloud Computing is gaining popularity, the technology comes with several issues such as performance, resiliency, interoperability, data migration and transition from legacy systems, in which security is a major concern hindering its adoption.
Problem Statement	Traditional methods for securing information are challenged by cloud-based architectures because Cloud Computing encompasses many systems and technologies such as networks, databases, operating systems, virtualization, SOA and web services, which makes security issues for many of these systems applicable to the cloud. Existing approaches that apply cryptographic techniques are complex and suffer computational overhead. Recent efforts that use mechanisms such as role-based access control usually result in the adjustment of the security requirements to fit the mechanism at hand, leading to limitations in policy specification.

Enthymeme	An access control solution specific for the cloud is required for such a dynamic, distributed and complex environment as the cloud.
Thesis	Develop a Policy-Based Management (PBM) framework to control access to data thereby ebbing the fear of relinquishing control of information to the cloud.
Objective	1. Investigate the characteristics of Cloud Computing and specify the requirements for designing an access control solution that will support those characteristics. 2. Analyze the current access control models for Cloud Computing and evaluate their suitability in line with the identified Cloud Computing access control requirements. 3. Derive a model capable of handling dynamic and anonymous users and reducing security management tasks.
Research questions	How to control access to cloud data based on an authorization system? How to support different roles and cloud user attributes using the proposed framework due to the complexity of the cloud environment? How authorization decisions can be provided dynamically, i.e., change rules & policy combination based on contextual information? How to provide a more flexible, more efficient, and more secure access control solution for Cloud Computing?

1.4 Research Methodology

This section discusses the methodology used for carrying out this research. It introduces the research methodology – Design Science Research (DSR) – applied to design the framework. A research model was also defined to indicate activities involved in each phase of the research and finally, the research strategy used for setting performance target, measuring security compliance and tracking security metrics is described.

1.4.1 Design Science Research Model

Design Science Research (DSR) involves the design of novel or innovative artifacts and the analysis of the use and/or performance of such artifacts to improve and understand the behavior of aspects of Information Systems. According to Aken (2005), the main goal of

DSR is to develop knowledge that can be used to develop solutions to problems. This mission can be compared to the one of 'explanatory sciences' like the natural sciences and sociology, which is to develop knowledge to describe, explain and predict. In design science research, as opposed to explanatory science research, academic research objectives are more of a logical nature. Henver et al. (2004), established seven guidelines to assist researchers, reviewers, editors, and readers understand the requirements for effective design-science research as follows

- *Design as an Artifact:* Design-science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation.
- *Problem Relevance:* The objective of design-science research is to develop technology-based solutions to important and relevant business problems.
- *Design Evaluation:* The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.
- *Research Contributions:* Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies.
- *Research Rigor:* Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.
- *Design as a Search Process:* The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.
- *Communication of Research:* Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.

Design science is active with respect to technology, engaging in the creation of technological artifacts that impact people and organizations. It focuses on problem solving but often takes a simplistic view of the people and the organizational contexts in which designed artifacts must function. The design-science paradigm seeks to create "what is effective" (Henver et al., 2004).

The DSR research model has 5 steps in its lifecycle. Step 1 is for the awareness of the problem, which is usually achieved by a new development or a reference in the discipline. The output of this phase is an initial research proposal. Step 2 is the suggestion of a solution with the output of a tentative design based on the information acquired in phase1. Step 3 focuses on developing the solution and, it has an IT artifact as its output while step 4 is used to evaluate the resulting artifact against the initial problem and the implicit and explicit criteria extracted in steps 1 & 2. The output of this phase is performance measures. The final phase concludes with the output of overall research results (Takeda et al., 1990).

Given the artificial nature of organizations and the information systems that support them, the design-science paradigm can play a significant role in resolving the fundamental dilemmas that have plagued IS research: rigor, relevance, discipline boundaries, behavior, and technology (Henver et al., 2004).

Within this research setting, the design-science research paradigm is suitable with respect to technology as it focuses on creating and evaluating innovative IT artifacts that enable organizations to address important information-related tasks.

1.4.2 Research Model

A research prototype was developed based on the Design Science Research. The research model consists of a number of closely related activities. Activities in this model continuously overlap instead of following a sequence and the first step determines how the last step will be taken. Figure 1.4 below shows the research model. A software development lifecycle (SDLC) is used to develop the solution in the 3rd step of the DSR model.

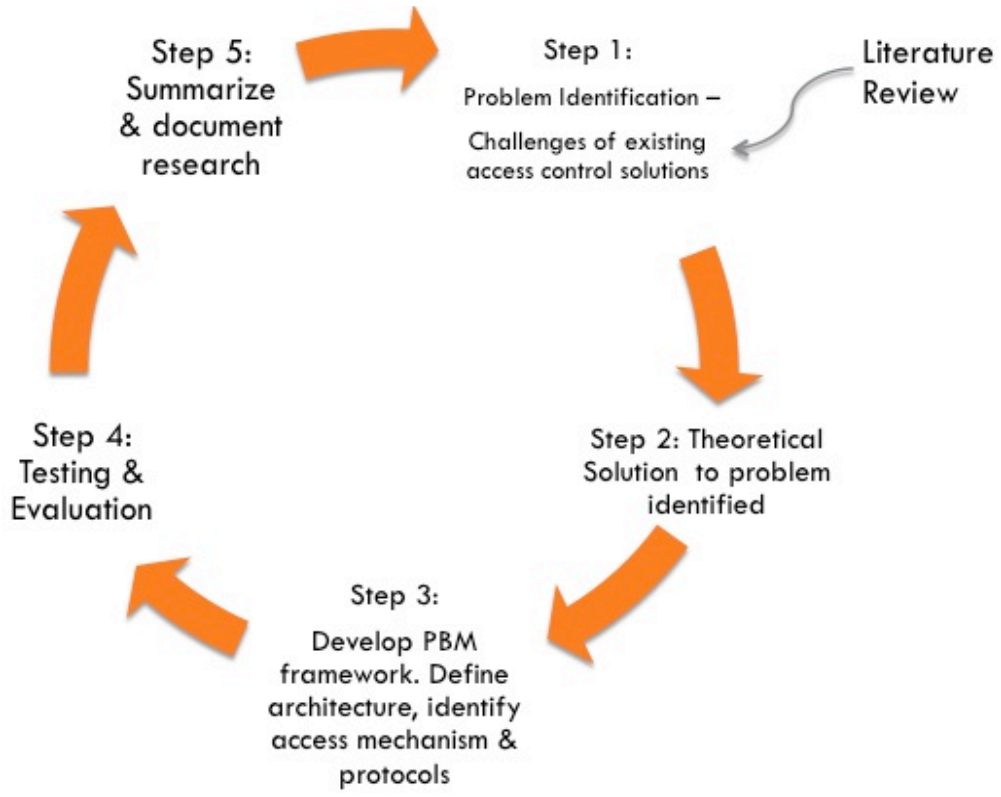


Figure 1.4: Research Model

DSR Step 1 & 2

- *Literature Review:* An extensive literature review is conducted in order to gain in-depth knowledge of the subject matter and to make inferences from the work of experts in the

field of Cloud Computing Security, Enterprise Information Security Management, Policy Based Management and Access Control. The sources of information stem from journals, articles, conference proceedings, government reports, academic papers, books, case studies, specifications, international standards and best practices.

- *Problem Identification & Theoretical Solution (DSR step 1 & 2):* After the problem was identified and the theoretical solution arrived at, the PBM framework is developed.

DSR Step 3

- *SDLC:* In this phase, the Software Development Lifecycle (SDLC) is used as a guide to develop the framework with focus only on the requirements specification, design and implementation phases.
    - *Specification Phase:* Here, the idea of formalizing the distinction between what the framework must do and how it does what it must do develops to ensure a problem-centered approach for understanding the problem before implementation. This involves the requirements specification and constraints identification. Therefore, the specification phase serves as a statement of the problem to be solved and the constraints limiting the implementation options.
    - *Design Phase:* In the design phase, the solution that satisfies the specifications is developed.
        - *Definition of the Security Architecture:* At this stage, the overall security architecture of the framework is defined.
        - *Definition of Service Primitives:* This stage defines the service primitives required to implement the specified services. The primitives determine the interface presented to the applications and the parameters that must be

passed between architectural layers.

- *Selection of Underlying Access Mechanisms:* At this stage, underlying mechanisms are selected to implement the services. A mechanism is a basic technology or algorithm. The mechanisms are selected based on the required services, constraints, and performance factors.
- *Determine Service Protocols:* At this stage, the service protocols that tie service mechanisms together to provide the required services are determined. A protocol is an end-to-end operation that uses one or more mechanisms to implement a service. Protocols are selected based on the required services, constraints, and performance factors. Great care was taken to ensure that the chosen protocol does not undermine the security of the underlying mechanisms. SAML was chosen.

- *Implementation Phase:* The implementation phase translates the design into reality. This phase consists of developing the required framework, testing and verifying the implementation and gathering performance data.

DSR Step 4

- *Testing:* The framework was evaluated using an Answer Set Programming (ASP) solver for different scenarios and compliance was checked against target queries.

DSR Step 5

- *Summarizing & Writing of Report*: The results are summarized and documented.

1.4.3 Research Design Strategy

This section is concerned with the overall approach taken to gather information as well as the style taken to address the research questions. The research design strategy used is qualitative and it relies on using records, journals, books and articles to describe, analyze, and explain past events and beliefs on the topic. The documents are reviewed to cover all the relevant literature of top quality and most important contributions are identified and analyzed to get an overview of the current state of knowledge. See Table 1.2 for the research design classification.

Table 1.2: Research Design Classification

Category	Option
Degree of problem crystallization	Exploratory
Method of data collection	Document review
Control of variables	Experimental
Purpose of study	Descriptive
Time Dimension	Cross-sectional
Topical scope	Scenarios
Research environment	Simulation

1.4.4 Data Collection Procedures

The data collection method used involved collecting literatures on the topic from various sources, which included previous studies. These documents (e.g., books, journals, reports and articles), which were hard copy or electronic, were chronologically examined and core ideas, concepts and facts were pulled together to make sense in the context of the topic under study.

1.5 Measurement Criteria

The criteria used to evaluate the framework are presented in Table 1.3. It identifies the main construct of this research and the operational concepts. It lists the functions to be considered for the enforcement properties.

Table 1.3: Measurement Criteria

Construct	Operational concepts	Properties	Criteria	Scale
Policy Based Management for Cloud Computing Security	Efficiency & Effectiveness	Level of confidence Level of availability Level of Integrity	Authentication Confidentiality impact Integrity impact Availability impact	Number of successful exploits Number of vulnerabilities exploitable Number of service denials
	Policy Flexibility	Level of policy expressiveness	Scenarios Operational or situational awareness	Number of scenarios addressed Number of situations adapted
	Safety	Correctly stored policies Tamper-proof	Safety constraints Operational steps required	Number of permissions leaked Number of operational steps involved
	Policy Specification & Implementation	Policy coverage Policy combination Conflict resolution	Potential errors in policy specification Policy elements Conflict resolution	No of faults detected No of rules permitted Ease of policy elements combination Number of conflicts involved/resolved
	Granularity of control	Level of granularity (Fine-grained)	Constraints supported	Support for least privilege Support separation of duties

1.6 Validation Strategy

A logic-based policy management approach, eXtensible Markup Language (XACML), which is a standard for specifying and enforcing access control policies, is used for specifying policies in this framework. To validate our framework, checking the correctness of policy specification and implementation is carried out. The correctness of policy specification is critical to ensure the correctness of the implementation and enforcement of policies. Therefore, to identify inconsistencies and differences between our policy specifications and their expected functions, we use XACML 2.0 mutants generator (XACMUT), a framework that performs the complete process of mutation analysis, i.e., apply a set of mutation operators to a given XACML policy to detect fault.

To ensure correctness of policy implementation, we use a systematic method to represent XACML policies in Answer Set Programming (ASP). The expressivity of ASP, such as its ability to handle default reasoning, allows us to represent XACML policies in a way that cannot be handled in the other logic-based approaches.

1.7 Research Contributions

Policy-Based Management is a very promising solution in cloud scenarios. While the use of policies for access control may be well established in the enterprise, it is not as evident in the Cloud Computing field. Hence this thesis contributes to knowledge in the field of Cloud Computing security research by focusing on access control challenges that limit the adoption of cloud technology. The features of our framework are as follows:

- The use of dynamic role administration, which provides a flexible way of managing user permissions and access control. The access control framework is able to support

fast revocation of credentials, constraints and applications such as task management.

- Support for dynamic permission validity, which contrast with older static models for access control. In this framework, attributes of users allow for dynamic changing of permissions based on conditions outside the access control framework.
- Grouping of permissions for ease of management by leveraging roles. Since roles are not in themselves sufficiently expressive to meet the needs of cloud applications, to achieve a fine-grained access control on resources, the framework leverage attributes of users with roles.
- Two-fold authorization where permissions are given to roles and tasks. Roles are used to activate different instances preventing subjects from accessing resources when not executing corresponding tasks.
- The use of a trust mechanism that allows permissions to be given to unknown subjects.

The key contributions of this thesis are:

- Design of a new architecture and model scalable, flexible and decentralized, capable of enforcing fine-grained access control policies on resources based in the cloud, the introduction of a trust mechanism allows decentralization to be achieved and granting permissions to unknown entities. In addition, the framework is capable of evaluating and enforcing access control at run-time.
- A comparison of existing access control models used in the cloud environment is made to determine their applicability in cloud scenarios.

1.8 Thesis Organization

The rest of this thesis is organized as follows:

Chapter 2: Reviews the risks and challenges of enterprise information security systems. It discusses measures that enterprises take to minimize their threats, secure intellectual property and maintain the security of information. It also discusses Policy-Based Management approaches as a viable solution for managing the security of enterprise information systems.

Chapter 3: Reviews the background of Cloud Computing, its security challenges and benefits. It presents access control as a solution to ensure Cloud Computing security, and some access control models proposed for the cloud environment are discussed.

Chapter 4: Introduces design-ready, reusable standards and protocols used in the completion of this research.

Chapter 5: Presents our proposed architecture and model for Cloud Computing security to ensure cloud users are only allowed access to specific virtual resources.

Chapter 6: In this chapter, the implementation and validation of our proposed access control model is discussed.

Finally, in *Chapter 7*, our contributions and future work plans are presented.

Chapter 2: Enterprise Information Systems Security

Enterprise information systems (EIS) are systems implemented by enterprises to manage their business processes. For enterprises to balance possible threats to their information systems, security measures are implemented. Information security is an area that deals with the protection of the confidentiality, integrity and availability of information and its critical elements, including the software and hardware that use, store, process and transmit that information through the application of policy, technology, education and awareness (Khoo, Harris & Hartman, 2010). The aim of information security is to ensure the continuity of business in a structured manner and mitigate damage caused by security events (Michelberger & Labodi, 2012).

Addressing information security is a core necessity for most, if not all, enterprises. Customers, business partners, vendors, suppliers are demanding it as concerns about privacy and identity theft rise. A number of best practice frameworks exist to help organizations assess their security risks, implement appropriate security controls, and comply with governance requirements as well as privacy and information security regulations.

This chapter provides background information on how enterprises ensure the security of their information systems. It discusses the measures taken to minimize threats and vulnerabilities and finally, introduces Policy-Based Management as a viable solution to address the security of enterprise information systems, outlining the challenges and benefits.

2.1 Requirements for EIS Security

The need for information security can be formulated from the following major requirements (Benson et al., 1999):

- *Confidentiality:* Controlling who gets to read information in order to keep sensitive information from being disclosed to unauthorized recipients.
- *Integrity:* Assuring that information and programs are changed, altered, or modified only in a specified and authorized manner.
- *Availability:* Assuring that authorized users have continued and timely access to information and resources.
- *Configuration:* Assuring that only authorized users change the configuration of a system or a network and only in accordance with established security guidelines.

Satisfying these security requirements requires a range of security services, which includes:

- *Authentication:* Ascertaining that the identity claimed by a party is indeed the identity of that party. Authentication is generally based on what a party knows (e.g., a password), what a party has (e.g., a hardware computer-readable token), or who a party is (e.g., a fingerprint).
- *Authorization:* Granting of permission to a party to perform a given action (or set of actions).
- *Auditing:* Recording each operation that is invoked along with the identity of the subject performing it and the object acted upon (as well as later examining these records).

- *Non-repudiation:* The use of a digital signature procedure affirming both the integrity of a given message and the identity of its creator to protect against a subsequent attempt to deny authenticity.

A balance must be maintained to provide a secure environment and to ensure confidentiality, integrity and availability (CIA), while allowing the flexibility needed for an enterprise to profitably operate (see Figure 2.1, CIA Triangle) as too much security will result in unusable systems and too little will expose risk and hamper integrity (Owen, 2009).

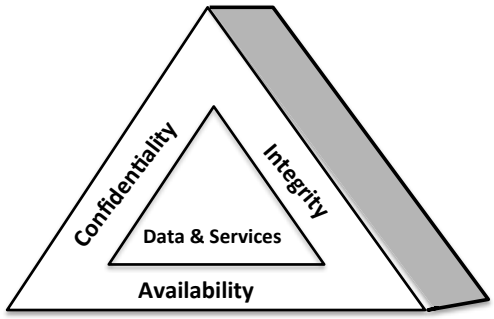


Figure 2.1: CIA Triad (Owen, 2009)

2.2 EIS Security Risks

Security challenges can be classified as threats and vulnerabilities. Vulnerabilities are weaknesses in the system while threats are events that may result from the weaknesses (Saleh et al., 2011). Examples of vulnerabilities are insufficient testing, lack of audit trail, unprotected communication lines, insecure network architecture, inadequate security awareness and lack of continuity plans while threatening risks includes human error, natural disasters (e.g., floods, earthquakes, storms), hardware and software problems, cybercriminals, disgruntled employees, terrorists attacks and intentional attacks (e.g.,

sniffers, password cracking, scanning, denial of service, malicious code) (Liu & Wu, 2010).

2.3 EIS Security Solutions

Controlling access and usage of shared resources are the most important goals of security systems in a shared network. With the expansion of computer networks (especially Internet), the attitude towards information security and other shared resources has entered a new phase (Shahram et al., 2012). Information security technology solutions can be classified as either proactive to prevent actions before the occurrence of the problem (e.g., cryptography, digital signature, anti-virus) or reactive enough to respond after the occurrence of security problem (e.g., firewalls, passwords). They are implemented at the following system levels: network, host and application levels (Shahram et al., 2012).

Even though securing information systems is critical for enterprises, it is not a complete solution as not all information breaches involve information systems Therefore, a holistic approach taking into consideration areas shown in Figure 2.2 as well as a system perspective must be taken to ensure security (Sharma D., 2011).

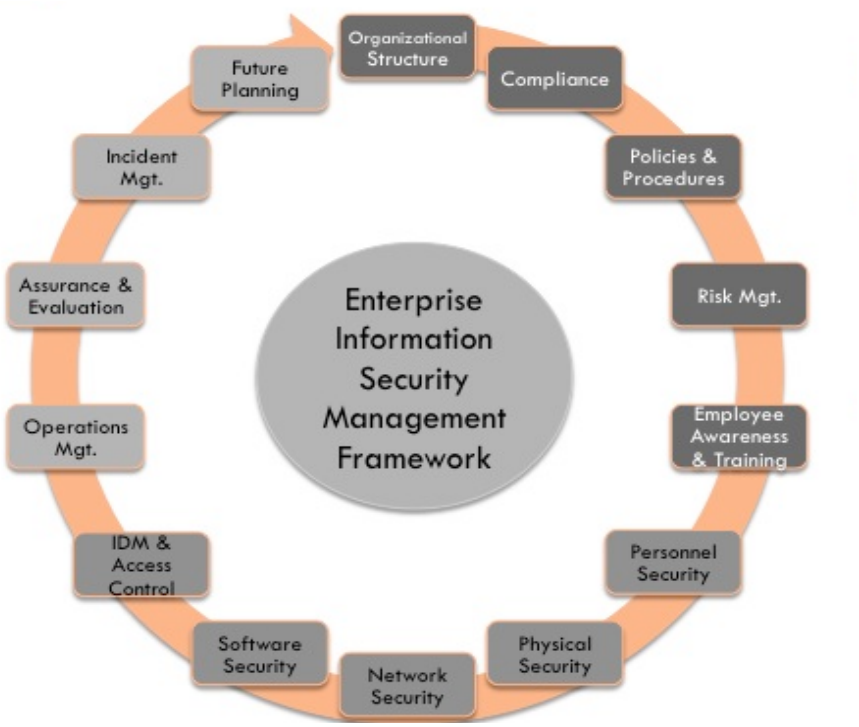


Figure 2.2: Enterprise Security Management Framework (Sharma D. , 2011)

The following are the common security solutions used (Liu and Wu, 2010):

- *Data Encryption:* Data encryption is conducting a change on the information content through certain rules (such as: algorithm), so that the attacker cannot acquire the encrypted information and cannot obtain the correct information content.
- *Virtual Private Network (VPN):* VPN is a kind of utilization of tunnel and encryption method in the open Internet to establish a private and secure network.
- *Firewall:* Firewall is a kind of device to assist in ensuring information security. According to specific rule, it will permit or limit the information passing through. A firewall may be a set of dedicated hardware components or a set of software components installed on the hardware.

- *Intrusion Detection System (IDS):* IDS has the ability to make up for the insufficiencies of firewalls. In general, a firewall can only conduct limitation on the access of certain services but it cannot detect whether the entered packet is abnormal or not. The intrusion detection system can analyze the entered packet and compare it with the established invasion trait database so as to examine abnormality and provide warning (Liu & Wu, 2010).
- *Policies:* Policy is a critical component of an overall enterprise information protection strategy. It is defined according to the National Institute of Standards and Technology (2006) as *"Aggregate of directives, regulations, rules, and practices that prescribes how an organization manages, protects, and distributes information"* (Karadsheh, 2012). They specify who is allowed to perform which action, on which object depending on properties of the requester and, of the object as well as parameters of the action and environmental factors (e.g., time) (Coi & Olmedilla, 2005). Policies can be used to define management strategies for network devices, access control systems or Internet services. Any quality security program begins and ends with policies. They are the least expensive control to execute, but the most difficult to implement properly (Whittman & Mattford, 2009). The purpose of a security policy is to protect people and information, set rules for expected behavior by users, define and authorize the consequences of violation, minimize risks and help to track compliances with regulation (Diver, 2007). There are three different types of security policies as follows:
    - *Enterprise Information Security Policy (EISP): is a* general security policy and supports the mission, vision and direction of an enterprise and sets the

strategic direction, scope for all security efforts (Whittman & Mattford, 2009).

- *Issue-Specific Security Policy (ISSP)*: addresses specific areas of technology e.g., e-mail usage, Internet usage, privacy and network usage.
- *System Specific Policy (SysSP):* focuses on decisions taken by management to protect a particular system, such as defining the extent to which individuals will be held accountable for their actions on the system (Karadsheh, 2012).

According to Bishop (2002), the goal of a security policy is to maintain the integrity, confidentiality, and availability of information resources. To enforce the above requirements, authentication, access control and auditing, three mutually supportive technologies are used (Siewe, 2005). The main benefits from using policy are improved scalability and flexibility for the management system. Scalability is improved by uniformly applying the same policy to large sets of devices and objects, while flexibility is achieved by separating the policy from the implementation of the managed system (Damianou et al., 2002). Policies are written in different languages such as XACML (OASIS, 2013), Ponder (Damianou, Dulay, Lupu, & Sloman, 2001), Rei (Kagal & Lalana, 2002), WSPL (OASIS, 2003) defined in Policy Servers. The most common policies are access control (MAC, DAC, RBAC, ABAC), obligation policies, privacy policies, and history-based / time-constraint policies (Ribeiro et al., 2000).

2.4 Policy-Based Management for EIS Security

This section examines in details policies as a solution for enterprise information systems security and, the security services that can be employed to enforce policies. The

administrative approach within the enterprise for specifying the behavior of a system under different circumstances is referred to as Policy-Based Management (PBM). It allows the response of the system to a given situation to be changed quite simply, by changing the policy, without the need to modify the underlying software (Waller et al., 2011).

In the context of this research, Policy-Based Management involves the specification of policies 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.

Policy-Based Management was originally developed for reducing the administrative complexity of reconfiguring the network whenever the behavior of the network needed to be changed (Davy and Barrett, 2005). Solutions are based on security policies that must be defined and implemented by manipulating protocols (e.g., LDAP, COPS, and Diffserv) inherent in managed devices (Jude, 2001). PBM is widely adopted by organizations such as the Internet Engineering Task Force (IETF) and the Distributed Management Task Force (DMTF).

2.4.1 PBM Architecture

Policy management architecture is required to transfer, store and enforce policies in a domain once policies have been defined for that domain. A typical PBM architecture consists of a management console, a policy decision point (PDP), a policy repository and a policy enforcement point (PEP). The management console is used for creating, editing, validating and translating policies. The PDP helps with the policy decision-making, translation and configuration of policies. The policy repository is used for storing, searching and retrieval of policies. While at the PEP, policies are enforced (Waller, 2004). See Figure 2.3.

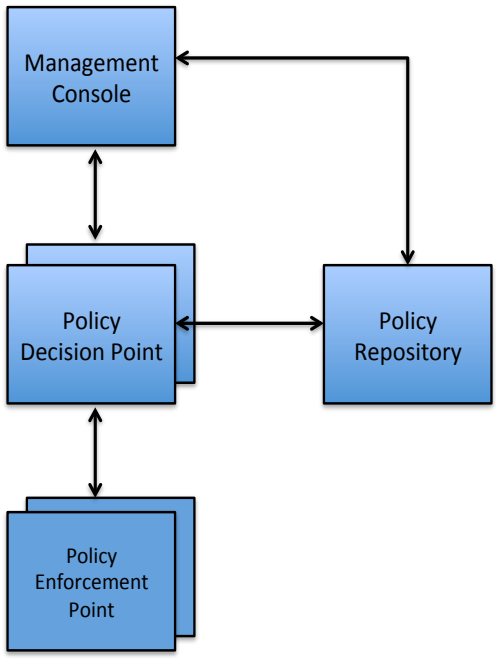


Figure 2.3: A Typical Policy-Based Management Architecture (Waller, 2004)

From Figure 2.3, when an access request is made for a resource, the PEP requests for a decision from the PDP point. The PDP then generates appropriate instructions for the PEP after evaluating the policies in the policy repository. Policies are added to the policy repository from the management console. The main requirements for policy management architecture are (Kagal, Finin & Hendler, 2005):

- *Well-defined Interface:* Policy architectures need to have a well-defined interface independent of the particular implementation in use that is clear and unambiguous.
- *Flexibility and Level of Abstraction:* The system architecture should be flexible enough to allow addition of new types of devices with minimal updates and recoding of existing management components.
- *Interoperability:* The system should be able to work with other architectures that may

exist in other administrative domains.

- *Conflict Detection.* It should be able to check for conflicts with existing policies.
- *Scalability.* It should maintain quality performance under increased system load.

2.4.2 PBM Challenges

Policy-Based Management presents the following challenges: (Waller et al., 2011)

- The PBM system has to take in policies covering a variety of topics in addition to security (e.g. resource allocation), and from a variety of sources. These policies may be expressed in multiple languages at different levels of abstraction, and must be translated into a common language for use at the point decisions are made.
- The decision-making process using the defined policies must be correct, and the implementation of the policy actually has to happen.
- Conflict as a result of multiple policies from multiple sources will need to be resolved. Policy conflicts can occur when the conditions of two or more policy rules that apply to set of managed entities are simultaneously satisfied, but the actions of two or more of these policy rules conflict with each other (Davy & Barret, 2005).
- Policy-Based Management system activities will need to be performed without impacting performance or cost (Waller et al., 2011).

To achieve high assurance, policies need to be precisely and unambiguously specified, and accurately implemented. An example of certification schemes is the Certification Common Criteria, which contains an assurance class on flaw remediation, but it is rarely used and does not provide methodological support for analyzing the security impact of

system changes. Currently, certification schemes do not provide a reliable way of assessing the trustworthiness of a system (Waller et al., 2011). More modern approaches e.g. Enterprise Privacy Authorization Language (EPAL), Web Services Policy Language (WSPL) and eXtensible Access Control Markup Language (XACML) directly exploit the properties of the requester in order to make an authorization decision and do not use credentials in order to certify the properties of the requester.

2.4.3 PBM Benefits

The use of Policy-Based Management has the following benefits: (Martin, 1999)

- *Central Management:* By using the directory services, the configuration of multiple instances of the same entity, or the configuration of different entities can be done from a single tool.
- *Well-defined Interfaces:* Given that the policies are stored in the directory services, any application could add new policies or look at existing policies.
- *Interoperability:* Policy-Based Management is a way to define standardized element configurations by abstracting vendor-specific parameters. This allows a different Policy Manager to define policies used by a different system, without knowing all the implementation details of the policies.
- *Added Ease of Use:* Since the details of the configuration are hidden to the administrator, configuration is easy and identical across multiple system versions or vendor implementations.

2.5 Concluding Remark

Recently, information security within the enterprise has undergone dramatic change from focus on primarily securing the enterprise perimeter. With Cloud Computing, the picture has become far more complex. The variety of ways to secure enterprise information systems has rendered the traditional view of the perimeter obsolete to address the security challenges enterprises are exposed to. However, the central requirement for security still remains the same, i.e., policies are required to implement security of information.

Chapter 3: Cloud Computing Security

Cloud Computing security is an evolving sub-domain of computer security, network security, and, more broadly, information security. It refers to a broad set of policies, technologies, and controls deployed to protect data, applications, and the associated infrastructure of Cloud Computing" (Reeja, 2012). In the context of this research it involves identifying how access policies can be implemented in the cloud to address unique threats and challenges thereby providing information security assurance. This chapter introduces Cloud Computing, its security challenges and benefits. It presents access control as a solution for Cloud Computing security and introduces access control models that have been identified for use within the cloud environment.

Cloud Computing promotes anything as a service (XaaS). It allows enterprises to scale resources up and down, as they require (i.e., the "pay-as-you-go" model of computing), making security of information a major requirement for services offered from the cloud. The multitenant nature of the cloud is vulnerable to information leaks, threats and attacks as well as problems inherited from virtualization and SOA technologies, (Grundy & Miller, 2010) and therefore, it is important to have strong access control policies in place to maintain the confidentiality, integrity and availability of data.

The complexity of security in a cloud environment is illustrated in Figure 3.1 (Subashini & Kavitha, 2011). The lower layer represents the different deployment models of cloud namely private, community, public and hybrid cloud. The next layer represents the delivery models namely SaaS, PaaS, and IaaS. The delivery models form the core of cloud, which exhibit certain characteristics such as rapid elasticity, measured service, on-demand self-

service, multi-tenancy and resource pooling as shown in the upper and, each layer has different security requirements (Subashini & Kavitha, 2011). According to Asma, Chaurasia & Mokhtar (2012), the cloud system can be secured by solving the SaaS, PaaS and IaaS security issues indirectly and, adequate security can be achieved by solving the information, virtualized environment and communication security issues.

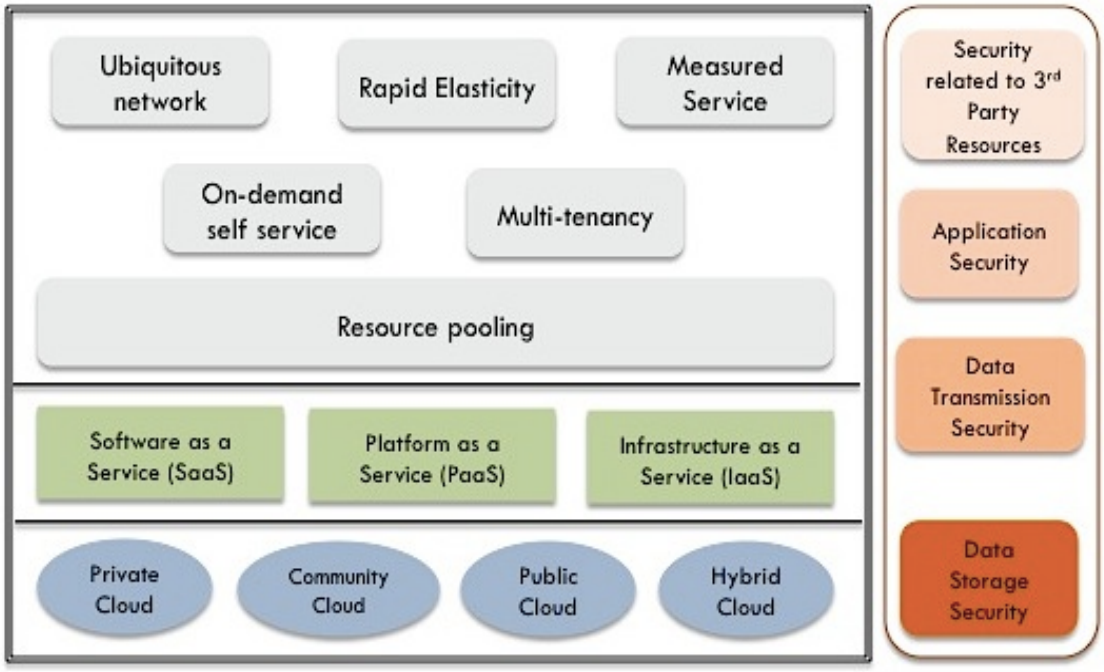


Figure 3.1: Complexity of Security in a Cloud Environment (Subashini and Kavitha, 2011)

3.1 Cloud Computing Overview

Cloud Computing is a new business model based on the concepts of virtualization, multi-tenancy and shared infrastructure (Mell & Grance, 2011). It is a shift in how computing is done because it encompasses activities such as Web 2.0, Web services, the Grid, and Software as a Service (SaaS), which enable users to use data and software residing on the Internet rather than on a personal computer or local server. With Cloud Computing, new

resources can be quickly and dynamically added to a consumer's resource pool within minutes, allowing cloud consumers to obtain as much computation and storage resources as they require, while only paying for the precise amount that they use (Wood et al., 2009). The benefits of Cloud Computing are numerous based on its pay as you go model, which includes reduced cost, application portability (i.e., users can work from home, or at client locations), enhanced collaboration, agility, scaling and availability through optimized and efficient computing (Min, Shin, & Bang, 2012; Asma, Chaurasia, & Mokhtar, 2012).

Many important companies such as Amazon, Google, IBM, Microsoft, and Yahoo are the forerunners that provide Cloud Computing services and recently, more companies such as Salesforce, Facebook, YouTube and MySpace have started providing all kinds of Cloud Computing services for Internet users (Pearson, 2012).

3.1.1 Characteristics of Cloud Computing

Key characteristics identified for cloud Computing are: (Zissis & Lekkas, 2012)

- *Flexibility/Elasticity:* Users can rapidly provision computing resources, as needed, without human interaction. Capabilities can be rapidly and elastically provisioned, in some cases automatically, to quickly scale in or out.
- *Scalability:* Cloud architecture can scale according to demand. New nodes can be added or dropped from the network with limited modifications to infrastructure set up and software.
- *Broad Network Access:* Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous platforms (e.g.,

mobile phones, laptops, and PDAs).

- *Location Independence:* There is a sense of location independence, in that the customer generally has no control or knowledge over the exact location of the provided resources, but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter).
- *Reliability:* This improves through the use of multiple redundant sites, which makes Cloud Computing suitable for business continuity and disaster recovery.
- *Economies of Scale and Cost Effectiveness:* Cloud implementations, regardless of the deployment model, tend to be as large as possible in order to take advantage of economies of scale.

3.1.2 Types of Cloud Models

There are two types of cloud models – delivery and deployment models.

Delivery Models: They are Infrastructure as a Service (IaaS), Software as a Service (SaaS) and as a Service (PaaS). See Figure 3.2.

- *Infrastructure as a service (IaaS):* is the foundation of all cloud services (bottom layer). It supplies a set of virtualized infrastructure component such as virtual machines, e.g., Amazon S3.
- *Platform as a Service (PaaS)*: is the middle layer in cloud services. It enables programming environment to access and utilize additional application building block, e.g., Google App engine.
- *Software as a Service (SaaS):* operates on the virtualized and pay-per-use costing model whereby software applications are leased out to contracted organization by

specialized SaaS vendor, e.g., Sales force (Zargar, Takabi, & Joshi, 2011).

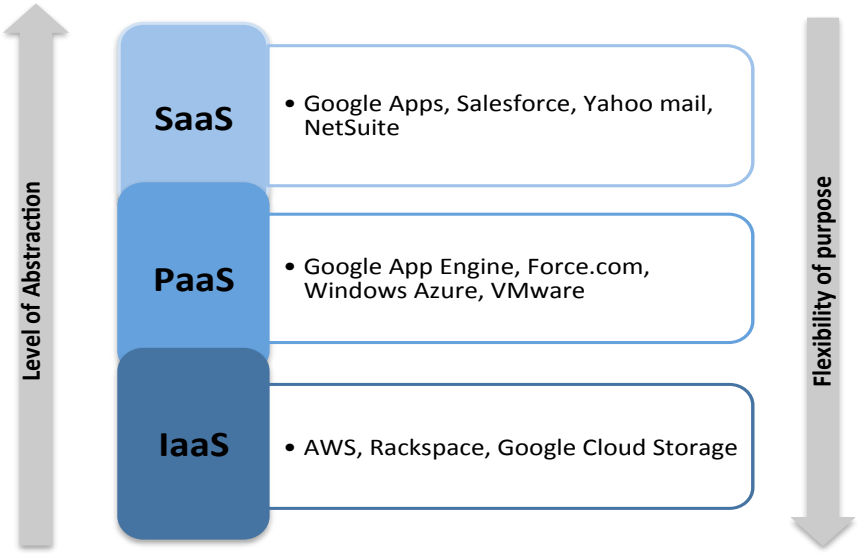


Figure 3.2: Cloud Delivery Models

Deployment Models: Four deployment models have been identified for cloud architecture solutions: (Zissis & Lekkas, 2012). See Figure 3.3.

- *Private Cloud:* Cloud infrastructure is operated for a private organization. It may be managed by the organization or a third party, and may exist on premise or off premise.
- *Community Cloud:* Cloud infrastructure is shared by several organizations and supports a specific community that has communal concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be managed by the organizations or a third party, and may exist on premise or off premise.
- *Public Cloud:* Cloud infrastructure is made available to the general public or a large industry group and is owned by an organization selling cloud services.
- *Hybrid Cloud:* Cloud infrastructure is a composition of two or more clouds (private,

community, or public) that remain unique entities, but are bound together by standardized or proprietary technology, which enables data and application portability (e.g., cloud bursting for load-balancing between clouds).

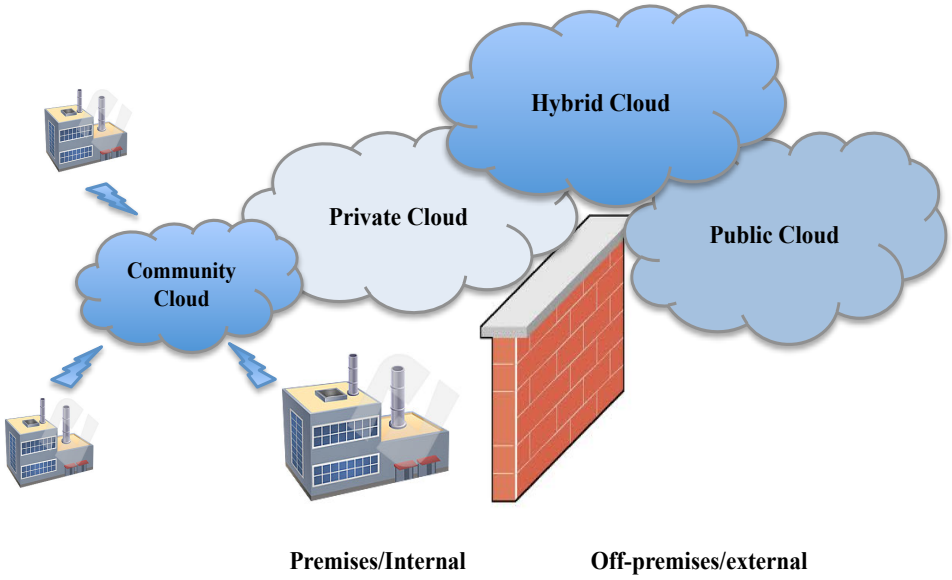


Figure 3.3: Cloud Deployment Model

3.2 Security Considerations for the Cloud

Security measures assumed in the cloud must be made available to the consumers to gain their trust. In 2011, the Cloud Security Alliance (CSA) published security guidance for critical areas to focus in Cloud Computing, addressing 14 domains. These are Cloud Computing Architecture, Governance & Enterprise Risk Management, Compliance and Audit, Legal issues, Information Management and Data Security, Portability and Interoperability, Traditional Security, Business Continuity and Disaster Recovery, Data Center Operations, Incident Response, Notification and Remediation, Application Security, Encryption and Key Management, Identity and Access Management, Security as a Service,

Virtualization. (CSA, 2011) In order to have a secured Cloud Computing deployment, all these areas must be considered.

3.3 Security Architecture for Cloud Computing

The practice of applying a comprehensive and rigorous method for describing current or future structure of enterprise information security systems is a common practice for enterprises to align security with goals. However, in the cloud, the role of enterprise architect has shifted and it is now distributed among cloud service providers. The enterprise, cloud provider and/or 3rd party are now involved in delivering appropriate security controls to mitigate cloud security threats. Figure 3.4 represents the logical representation of Cloud Computing security architecture, the cloud actors and the security architectural components defined for each actor.

The NIST Cloud Computing reference architecture defines five major actors in the cloud: *cloud consumer, cloud provider, cloud carrier, cloud auditor* and *cloud broker*. Each actor is an entity (a person or an organization) that participates in a transaction or process and/or performs tasks in Cloud Computing. A cloud provider is a person, organization or entity responsible for making an infrastructure, platform or software available to cloud consumers as a service. The person or organization that maintains a business relationship with, and uses one or more of these services from cloud providers, is a cloud consumer. The cloud auditor is a party that can conduct independent assessment of cloud services, performance and security of cloud implementation, the cloud broker is an entity that manages the use, performance and delivery of cloud services while the cloud carrier is the intermediary that provides

connectivity and transport of cloud services from cloud providers to cloud consumers (NIST, 2011).

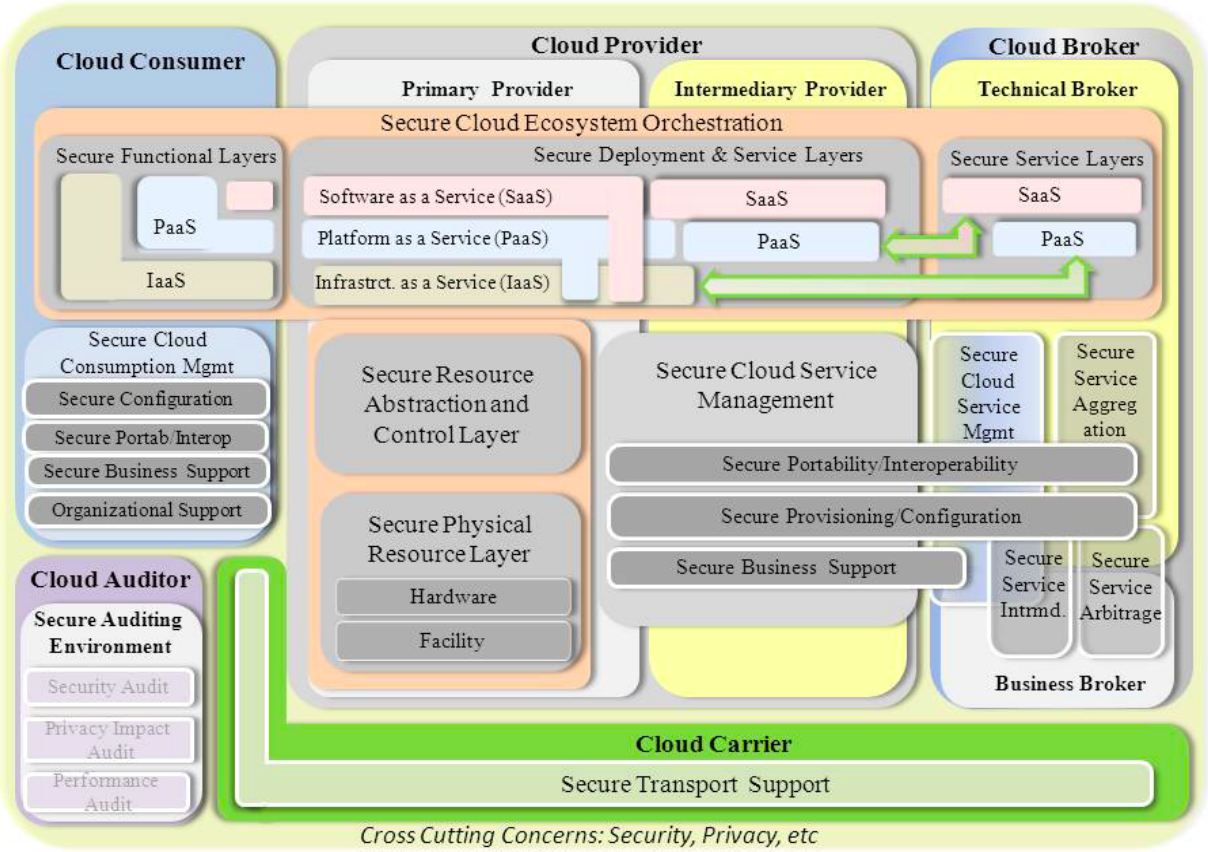


Figure 3.4: Computing Security Reference Architecture (NIST, 2013b)

3.4 Security Benefits of Cloud Computing

Cloud service providers argue that the cloud enhances security rather than degrade it. The following types of arguments have been posed in support of this position: (Karn, 2011)

- Cloud providers use built-in backups and safeguards that are not found in many desktop and enterprise servers.
- In a cloud application, security upgrades and patches can be deployed immediately to

all users (unlike in a desktop model where installation depends on the user and, as a result, often does not happen).

- All users receive the same high level of security regardless of their size.
- Most organizations operating systems in-house cannot match the technological expertise of a cloud provider.
- Because data are fragmented and dispersed throughout the cloud, the data are not readable by humans.
- Compliance analysis may be simplified since the user has only one cloud system to deal with versus many in house systems.
- The cloud provider is an uninterested third party and therefore, has no motivation to see the user's data.
- Disaster recovery and data storage are cheaper in the cloud due to volume.

3.5 Security Challenges of Cloud Computing

Security concerns are the biggest in the cloud according to various surveys (IDC, 2009; Fujitsu, 2010; Morgan Stanley, 2011). There are a number of security concerns associated with Cloud Computing which fall under two broad categories: Security issues faced by cloud providers (i.e., organizations providing SaaS, IaaS and PaaS via the cloud) and security concerns faced by cloud consumers. Specific security concerns include comprising the virtualization software. Perlin et al., (2010) noted that most security problems stem from loss of control, lack of trust and multi-tenancy.

- *Loss of Control:* This is due to the fact that data applications and resources are

located with the cloud provider and consumers need to rely on the service providers for data security & privacy, resource availability and monitoring or repairing of services or resources.

- *Lack of Trust:* Trust requires taking risks. Basically trust and risk are opposite sides of the same coin and some monitoring or auditing capabilities would be required to increase the level of trust.
- *Multi-tenancy:* Cloud tenants share a pool of resources and have opposing goals, which can bring about conflicts of interests. Also, sharing the same physical infrastructure adds new threats to the Cloud Computing environment, which makes it an even more attractive target for intruders due to the distributed nature of the cloud model. Furthermore, there may be various incentives for competing tenants to initiate attacks against each other due to the commercialized nature of the cloud environment (Zargar, Takabi, & Joshi, 2011; Perlin et al, 2010).

3.5.1 New Security Problems for the Cloud

In addition to the abovementioned issues, there are also new areas of security concerns that delay Cloud Computing adoption: (Chow, 2009)

- *Cheap data and data analysis:* The rise of Cloud Computing has created enormous data sets that can be monetized by applications such as advertising. Google, for instance, leverages its cloud infrastructure to collect and analyze consumer data for its advertising network. Because of privacy concerns, enterprises running clouds collecting data have felt increasing pressure to anonymize their data.

- *Cost-effective Defense of Availability:* Availability also needs to be considered in the context of an adversary whose goals are simply to sabotage activities. The damages are not only related to the loss of productivity, but extend to losses due to the degraded trust in the infrastructure, and potentially costly backup measures.
- *Increased Authentication Demands:* The movement towards increased hosting of data and applications in the cloud and lesser reliance on specific user machines is likely to increase the threat of phishing and other abusive technologies aimed at stealing access credentials, or otherwise derive them, e.g., by brute force methods.
- *Mash-up Authorization:* Mash-up of data will become widespread with the adoption of Cloud Computing and this has potential security implications, both in terms of data leaks, and in terms of the number of sources of data a user may have to pull data from – this, in turn, places requirements on how access is authorized for reasons of usability. While centralized access control may solve many of these problems, it may not be possible – or even desirable.

Security is one major hurdle that Cloud Computing needs to overcome before its mass adoption. How then can cloud providers and consumers respond to these security-related problems?

3.6 Cloud Computing Security Solutions

Various groups and organizations have developed security solutions and standards for the cloud. The Cloud Security Alliance (CSA) developed a security guide that identifies many areas for security concerns in Cloud Computing. By the end of 2011, the CSA released its third version of "*Security Guidance for Critical Areas of Focus in Cloud Computing"* in

which one architecture domain, five governance domains and eight operational domains are identified and discussed, providing a set of guidelines for the cloud providers, consumers to follow (CSA, 2011).

Cryptography is also an often touted remedy for cloud security based on its *fully homomorphic encryption* (FHE), dubbed by some as the field's "Holy Grail" and recently realized as a fully functional construct with promise for cloud privacy. However, Dijk & Juels (2010) argued that cryptography alone would not enforce the privacy demanded by common Cloud Computing services, even with such powerful tools as FHE and that other forms of privacy enforcement, such as tamper-proof hardware, distributed computing, and complex trust ecosystems are required.

Kailash et al. (2012) mentioned that concerns about the confidentiality and integrity of data and computation are a major deterrent for enterprises looking to embrace Cloud Computing and they proposed a method to build a trusted computing environment for Cloud Computing system by integrating the trusted computing platform.

In addition, a number of researches (Juniper Networks, 2013; Fang Hao, 2010) are focused on improving data center networks, location independence of virtual machines (VMs), scalability, and private networks in data centers to ensure secured data. Virtual machines of different customers may reside on the same physical machine, and their data packets may share the same local area network (LAN). Such lack of isolation brings security risks to users e.g. it is possible for a hacker to conduct attacks towards another Amazon elastic computing (EC2) user who shares hardware resources with the hacker in the cloud. Secure elastic Cloud Computing (SEC2) was proposed as a scalable data center network

architecture to support secure Cloud Computing for both enterprise and individual users. SEC2 eliminates the scalability limitation caused by VLANs offering effective isolation between different customers while allowing physical resource sharing. Users can specify and manage their individual security and QoS policy settings the same way they manage the conventional on-site networks. This architecture can also enable users to combine cloud-based resources seamlessly with their existing network infrastructure through VPN.

On the other end of the spectrum, Google has proposed a "government cloud", which creates entirely separate hardware, software, and administrators (with appropriate background checks) for special customers. While such cloud service can be very secure, it is also very expensive — almost like building a separate data center for each customer (Fang Hao, 2010).

In addition, access control mechanisms such as RBAC, ABAC are applied to the cloud to ensure that authorized users access the data and system (Khan, 2012).

3.7 Access Control for Cloud Computing Security

This section reviews access control as a solution for ensuring Cloud Computing security. Enterprises use access control mechanisms to mitigate the risks of unauthorized access to their data, resources, and systems. Their corresponding access control mechanisms and access control models can take several forms – they can make use of different technologies and underlying infrastructure components with varying degrees of complexity. However, with Cloud Computing, the current access control techniques are not well suited to meet the challenges of Cloud Computing environment as they were designed to support the enterprise

environment. Due to the dynamic and diverse nature of the cloud platform, having a strong access control in place is important to ensure security of data or information in the cloud.

3.7.1 Access Control Basics

Access control is concerned with determining the allowed activities of legitimate users, mediating every attempt by a user to access a resource in the system by implementing security policies (NIST, 2006). A typical architecture of an access control system is as described in the architecture of a PBM system in Section 2.4.1. Abstractions to be considered when planning to implement an access control system are as described in Figure 3.5 (NIST, 2006).

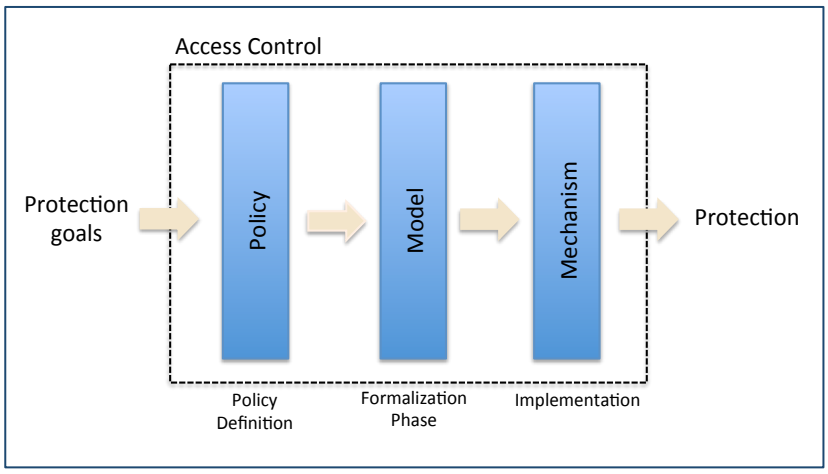


Figure 3.5: Design of an Access Control System (Samarati & Vimercati, 2001)

- *Policies:* High-level requirements that specify how access is managed and who may access information under what circumstances.
- *Mechanisms:* These translate users' access requests and enforce access control policies at a high level e.g. access control lists (ACL).
- *Model:* This is a formal presentation of security policy enforced by the system and it

is useful for proving theoretical limitations of a system. Access control *models* bridge the gap in abstraction between policy and mechanism. Security models are usually written to describe the security properties of an access control system e.g. discretionary access control (DAC). On the one hand, a model may be rigid in enforcing a single policy while on the other enable enforcement and expression of a wide variety of policies.

From Figure 3.5, the formalization phase between the policy definition and its implementation as a mechanism allows the definition of a formal model representing the policy and its working, making it possible to define and prove security properties that systems enforcing the model will enjoy. Therefore, proving that the model is "secure" and that the mechanism correctly implements the model implies that the system is "secure" (Samarati & Vimercati, 2001). The separation between policies and mechanisms introduces independence between protection requirements to be enforced on the one side, and mechanisms enforcing them on the other, making it possible to: (Samarati & Vimercati, 2001)

- Discuss protection requirements independently of their implementation,
- Compare different access control policies as well as different mechanisms that enforce the same policy, and
- Design mechanisms able to enforce multiple policies.

Mechanisms able to enforce multiple policies avoid the drawback of tying a mechanism to a specific policy because a change in the policy would require changing the whole system. Upon deciding on which access control model (e.g., DAC, MAC, RBAC) to implement, then different possible access control mechanisms that are available to work within these models

can be supplemented. To implement security policies, access request between an authorized user and protected resources are captured by the access control system, which determines whether the access request is authorized, or not.

3.7.1.1 Access Control Policies

There are several well-known access control policies, which can be categorized as either discretionary or non-discretionary.

- *Discretionary Access Control Policies (DAC):* Discretionary access control grants access based on the identity of requesters and provides flexibility of assigning access rights by owner of resources. It is also called Identity-Based Access Control (IBAC) or Authorization-based access control. The access control matrix provides a basic framework for describing DAC. Typical mechanisms for enforcing DAC policies include Authorization Table, Access Control List (ACL) and Capability list. Each of them interprets and implements the access matrix in a different ways. DAC policy tends to be very flexible and is widely used in the commercial and government sectors. However, it has the following drawbacks: (NIST, 2006), (NIST, 2013a)
    - Information can be copied from one object to another, which makes it difficult to maintain safety policies and verify that safety policies are not compromised while opening up the system up to Trojan horse susceptibility.
    - No restrictions apply to the usage of information when a user receives it.
    - Privileges for accessing objects are decided by the owner of the object rather than through a system wide policy that reflects the organization's security requirements.
    - DAC incurs scalability and management problems as the numbers of users

and resources increases. Additionally, users do not necessarily understand their assigned rights and responsibilities and system security can be seriously undermined by the inappropriate use of root or administrator access capabilities.

- *Non-Discretionary Access Control Policies:* All access control policies other than DAC are grouped in the category of non-discretionary access control (NDAC). Policies in this category have rules that are not established at the discretion of the user. Controls are established only through administrative actions and cannot be changed by users. Popular non-discretionary access control policies include mandatory access control (MAC), role-based access control (RBAC) and temporal constraints (NIST, 2006)

3.7.1.2 Access Control Mechanisms

Access control mechanism (ACM) is a mechanism to control access (operations including reading, writing, and deleting) to resources (files and directories) of a system. Access control mechanism defines the low level (software and hardware) functions that implement the controls imposed by the policy and formally stated in the model (Samarati & Vimercati, 2001). In order to determine a user's ability to perform an operation, the access control mechanism compares the security attributes of the users (such as identifiers, roles, groups) to the resources (such as types, access control lists, sensitivity labels) based on attribute-matching algorithm or pre-determined set of rules (NIST, 2006). ACMs can either be centralized or decentralized.

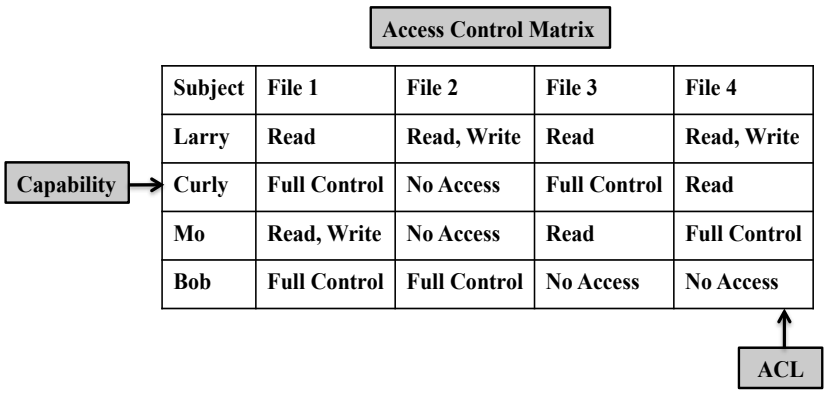
- *Centralized:* Access control information is stored on the side of the mechanism. Centralized ACM are highly flexible and easy to implement. However, they are not

as scalable and easy to maintain as decentralized ACM due to increase in the volume of information required to be managed as the number of users increases.

- *Decentralized:* Here, a small portion of the information on access control is stored on the side of the mechanism, which makes this mechanism easier to maintain since information required by a user is presented to the side of the mechanism at the time of access control, making it more scalable than centralized ACM (Arakawa & Sasada, 2011).

Examples of access control mechanisms include access control lists (ACL) & capability lists, role-based access control (RBAC) mechanism, rule-based access control (RuBAC), and XML-based mechanisms.

- *Access Control Lists (ACLs) & Capability lists:* ACLs is an old security mechanism that provides a straightforward way of granting or denying access for a specified user or groups of users. The ACL is a column within an access control matrix showing different subjects that can access an object while the capability list is the row containing the actual permissions assigned to a subject (Meyers, 2002). See Figure 3.6. The ACL has performance hits, storage inefficiencies, lacks fine granularity and lacks support for least privilege. In environments where there is significant user turnover, ACLs have serious difficulties and are platform-dependent. With a capability or access list, it is difficult to review the subjects that can access a particular object (NIST, 2006).



Subject	File 1	File 2	File 3	File 4
Larry	Read	Read, Write	Read	Read, Write
Curly	Full Control	No Access	Full Control	Read
Mo	Read, Write	No Access	Read	Full Control
Bob	Full Control	Full Control	No Access	No Access

Figure 3.6: Capability List bound to Subjects and ACL bound to Objects (Meyers, 2002)

- *Role-based Access Control (RBAC) Mechanism:* RBAC is being increasingly recognized as an efficient access control mechanism that facilitates security administration by assigning users to roles and thus adds a layer of abstraction between users and permissions. RBAC has been proposed as an alternative approach to traditional access control mechanisms both to simplify the task of access control administration and to directly support function-based access control (Kumar & Kumar, 2013).
- *Rule-based Access Control Mechanism (RuBAC):* RuBAC allows users to access systems and information based on predetermined and configured rules. RuBAC encompasses a broad range of systems and may be combined with other models such as RBAC and DAC. RuBAC intercepts every access request and compares the rules with the rights of the user to make access decisions.
- *XML-approach Access Control Mechanism:* XML-approach and other access control languages provide capabilities for composing policies from scratch, allowing users to specify a policy, together with the authorizations through the programming of the

language. However, as with the RuBAC, the limitation is in the expressive power of higher-order logic such as the expressions of historical-based constraints and domain constraints. Efforts have led to the development of XML-approach frameworks for specification of access control. XACML by OASIS and XACL by IBM are policy specification frameworks geared towards securing XML documents that can be applied to other resources as well. Currently, these languages do not provide support for MAC and DAC but a recent extension of XACML incorporates RBAC (NIST, 2006).

3.7.1.3 Access Control Models

Access control models are generally concerned with mediating the actions of a subject (e.g., user, system) to access an object (e.g., directory, file, screen, keyboard, memory, storage, printer), and how this access can occur. Access control models are usually seen as frameworks for implementing and ensuring the integrity of security policies that mandate how information can be accessed and shared on a system (Garg & Mishra, 2012).

There are three classic access control models: MAC, DAC and RBAC. These models make a clear separation between authentication and authorization. In open and dynamic scenarios, where clients and servers may not be known to each other in advance, credential-based access control has been proposed. Traditional separation between authentication and authorization cannot be applied anymore and trust management was proposed as a solution. These make access decisions based on policies containing credentials that effectively combine authentication and authorization (Zhao H. , 2012).

Recently, work on access control models focus on business process access control models and their relative effectiveness (Garg & Mishra, 2012). The following are examples of some

popular access control models include MAC/DAC, RBAC (Ferraiolo & Kuhn, 1992), ABAC (Burmester, 2012) and TBAC (Thomas & Sandhu, 1997).

3.7.2 Access Control Models for Cloud Computing Security

Within the cloud, simpler access control models often cannot adequately meet the complex access control requirements and so more granular, dynamic models and mechanisms are needed. Some access control models that have been identified for use within the cloud environment are hereby presented in three broad categories as follows:

- *Attribute-based Models:* Danwei, Xiuli & Xunyi (2009) proposed an access control architecture based on the usage control (UCON) authorization model. The UCON model not only includes DAC, MAC and RBAC, but also contains the digital rights management (DRM), and trust management, covering the two important problems of security and privacy in the modern business and information system demands. This model manages concepts such as subject, object, right, obligation, condition, and attribute. The main contribution of this proposal is the inclusion of a negotiation model in the authorization architecture to enhance the flexibility of access control for cloud services. Then, when the access requested does not match access rules, it provides users with a second access choice through negotiation in certain circumstances, rather than refusing access directly. However, Gouglidis & Marvridis (2009) found that the UCON model lacks improved administrative capabilities on the level of a domain and that policy resolution can be cumbersome.
- *RBAC Models:* Different variants of RBAC have been proposed to enable access control in the Cloud Computing environment. Li, Liu, Wei, Liu & Liu (2010)

proposed S-RBAC (an extension of RBAC and Administrative RBAC 97 (ARBAC97)), which uses layered structures to achieve system-level and tenant-level access control to solve the SaaS system access control problems but they did not consider the temporal constraints on permission license. At the API level, Sirisha & Kumari (2010) proposed a two-stage access control mechanism using the RBAC model. However, only registered users from white-listed domain can access the cloud service. Task and Role-Based Access Control (TRBAC) was considered a viable model for cloud environment and was found to dynamically validate access permissions for users based on the assigned roles and tasks performed by users with an assigned role. However, Ma, Wu, Zhang & Li (2012) noted that TRBAC is centralized and does not consider local and global access control integration and their communication in a distributed environment. Also, with TRBAC, it is hard to make classification of tasks and, TRBAC could not deal with some business rules such as delegation and closing account, which are common and important to support efficient execution of business activities.

X-GTRBAC model, an enhanced hybrid version of the X-RBAC and GTRBAC models, was also found suitable. X-GTRBAC relies on the certification provided by trusted third parties (e.g., PKI Certification Authority) to assign roles to users and also considers the context (e.g., time, location) to directly affect the level of trust associated with a user (as part of user profile) to make access decisions (Meghanathan, 2013).

- *Multi-tenancy Models:* To facilitate communication between the various tenants of a cloud, especially if they offer services to each other and intend to collaborate over the cloud by taking advantage of the close coupling between the users' machine, there

should be access control policies in place. In order to achieve multi-tenant access control, Calero, Edwards, Kirschnick, Wilcock & Wray (2010) proposed a multi-tenancy authorization system (MTAS) by extending RBAC with a coarse-grained trust relation among collaborating tenants. Multitenant collaborations are enabled in MTAS by bridging two tenants with a cross-tenant trust relation. In addition, Tang & Sandhu (2013) proposed a family of MT-RBAC models by extending RBAC model with the components of tenants and issuers to address multi-tenant authorization for collaborative cloud services. MT-RBAC aims to enable fine-grained cross-tenant resource access by building tenant-level granularity of trust relations. Yang, Lai & Lin (2013) used identity management and RBAC model to design a Role-Based Multi-Tenancy Access Control (RB-MTAC) with consideration of multi-tenant and multi-user cloud environment. The RB-MTAC method can easily assign the functions or resource with access privilege to users, in order to enhance processing performance, quality of service (QoS), and security as well as privacy.

3.8 Concluding Remark

Access control policies, which have been implicitly used to manage security in industries (e.g., telecommunications) and in distributed systems, are now being applied to Cloud Computing due to the complexity of the environment and the ineffectiveness of the traditional security management techniques.

Currently, users use diverse access control solutions available to secure their data and control its dissemination within the enterprise. Each of these access control model approaches has its advantages, disadvantages and feasibility scope. Some researchers have

even tried to combine different access control mechanisms to build more powerful models. However, with Cloud Computing, the suitability of these access control models needs to be revisited to ensure the security of data in the cloud.

Chapter 4: Reference Frameworks

This chapter discusses the standard frameworks (RBAC, ABAC, TBAC) and protocols (SAML and trust model) for the exchange of authentication, authorization and attribute data used in this thesis. It also presents the policy language (XACML), and the conceptual categorization framework for identifying cloud access control requirements. XACML is used for the specification of policies because of its expressiveness and flexibility in specifying access control policies.

4.1 XACML

The eXtensible Access Control Markup Language (XACML), an XML-based fine-grained policy language, is a well-established standard to define and enforce policies. It is consistent with the policy framework laid down by IETF and DMTF and, widely used. (OASIS, 2013).

The main components of the XACML architecture include components as described in the standard IETF and access control architecture – Policy Enforcement Point (PEP), Policy Decision Point (PDP), Policy Information Point (PIP), Policy Administration Point (PAP), and obligations service. The PEP performs access control by receiving decision requests, consulting the PDP for authorization decision, and enforcing the decisions. The PDP evaluates applicable policies and yields authorization decisions, together with obligations and advice, if any. The PIP acts as a source of attribute values, such as subject, resource, action and environment attributes. The PAP administrates policies and policy sets and makes them available to the PDP. The obligations service handles obligations forwarded by the PEP. However, XACML does not specify how PIP, PAP and obligations service should

behave and how they should be implemented. Figure 4.1 below depicts the high-level flow of XACML including various interfaces and actors in making authorization decisions (OASIS, 2013).

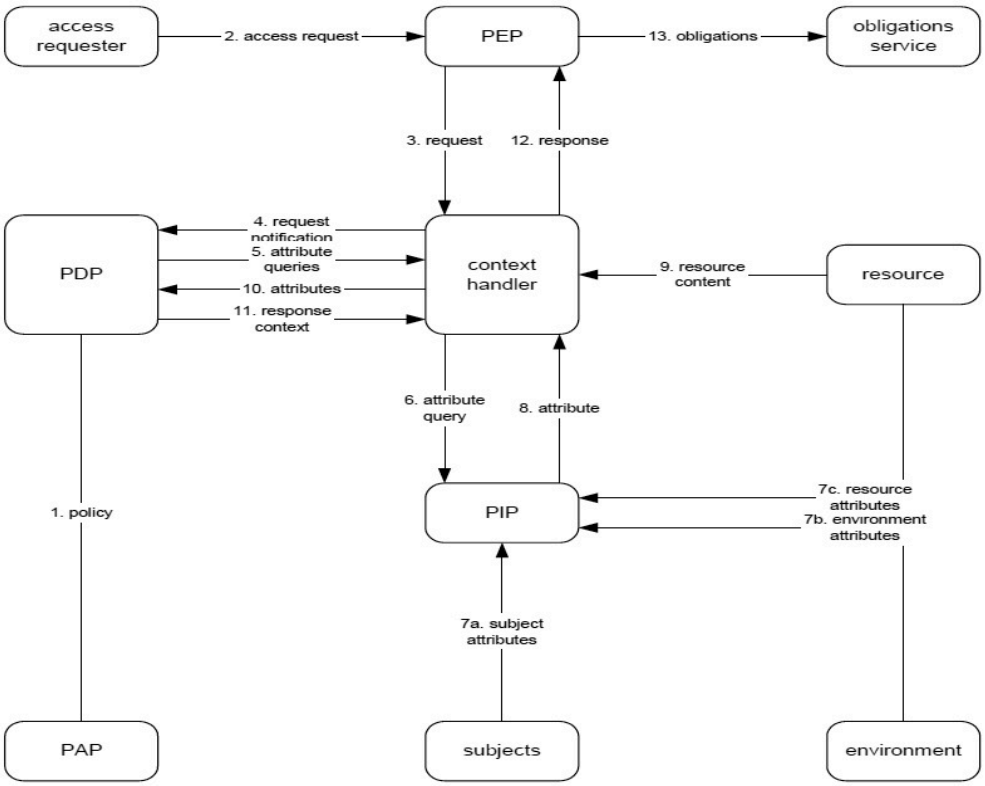


Figure 4.1: XACML Authorization (OASIS, 2013)

XACML specifies three types of XML documents: requests, responses, and policies. Standard XACML uses three basic elements in constructing access control policies: *rule*, *policy* and *policy set* and, allows hierarchical nesting of them. See Figure 4.2.

- An XACML *rule* is the most basic policy element. It has three main components: a target, a condition and an effect.
    - The *target* defines a set of subjects (S), resources (R), and actions (A) that a rule, policy or policy set applies to.
    - The *effect* (E*)* is the return value of the evaluation, which can be either permit,

deny, non-applicable or indeterminate. If a *request* originates from a subject to perform an action on a resource within an environment. The attribute values in the *request* are compared with those included in the *target* and if all the attributes match then the *request* is applicable. If the *request* and the *target* attributes do not match, then the *request* is non-applicable, and if the evaluation results in an error, then the *request* is indeterminate. If a request satisfies the *target* of a policy, then the *request* is further checked against the rule set of the policy; otherwise, the policy is skipped without further examination.

  - The *condition* specifies restrictions on the attributes in the target and refines the applicability of the rule. The intuitive reading of an XACML *rule* is that, if the condition of the *rule* evaluates to be true, then the access control decision to perform action by a subject on a resource are given by the *effect* attribute.

- A *policy* can consist of a set of *rules*.
- A *policy set* holds policies and other *policy sets*.

The XACML policy evaluation algorithm uses policy-combining algorithms (PCA) to recursively compute the decision of a nested rule/policy. The OASIS specification identifies four standard combining algorithms:

- *Deny-overrides:* If any of the policies or rules denies, the entire response says denies.
- *Permit-overrides:* If any of the policies or rules permits, the entire response says permits.
- *First-applicable:* Result is evaluated based on first rule or policy from the list of

applicable rules.

- *Only-one-applicable:* Result ensures that only one policy is applicable to the incoming request.

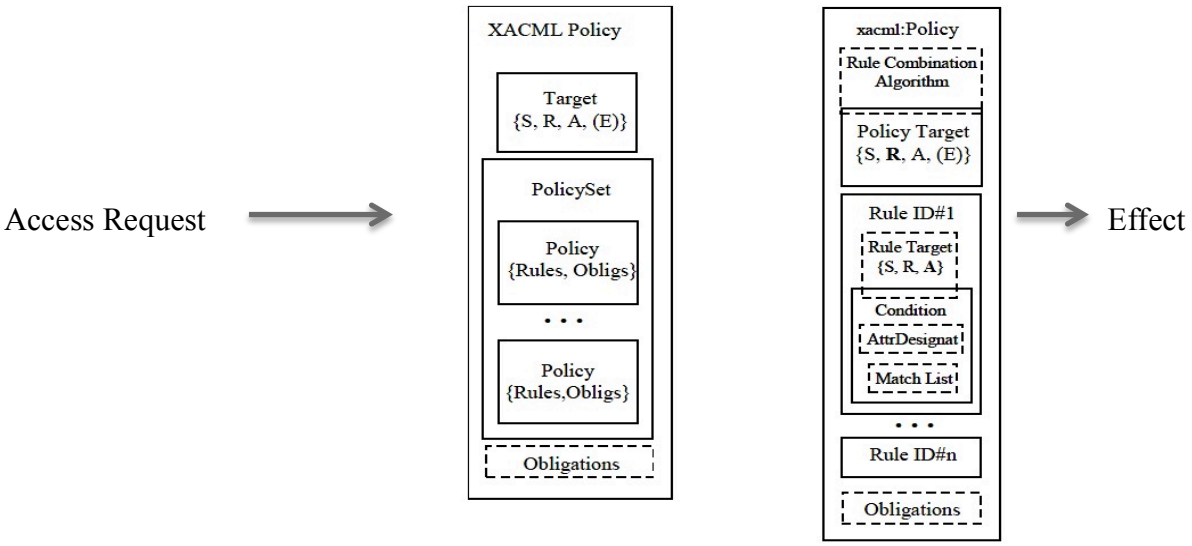


Figure 4.2: XACML Policy Model (Demchenko, 2009)

The XACML profile of RBAC (OASIS, 2004) defines a profile for the use of XACML to meet the requirements of RBAC. In this specification, roles are expressed as XACML subject attributes, except in role assignment where roles are resource attributes. Use of separate roles also allows for support of hierarchical RBAC. The assignment of various role attributes to users and the enabling of those attributes within a session are outside the scope of the XACML PDP. Role assignment entities may, however, use an XACML role assignment policies to determine which users are allowed to have various role attributes enabled, and under what conditions. These role assignment policies are a different set from the permission policy instances used to determine the access permissions associated with each role. The role assignment policies are used only when the XACML Request comes from a role assignment entity. A role assignment or enablement entity is a system entity or entities

responsible for issuing role attributes to users and for enabling those attributes for use during a given session and for each role, one role and one permission policy set are defined (OASIS, 2004). While XACML provides the means to express and enforce a policy, it does not specify how to request and retrieve the required attributes i.e. specify a protocol for communication between the PEP and PDP. SAML is a highly suitable candidate for this protocol.

4.2 SAML

Security Assertion Markup Language (SAML) is an XML-based standard for exchanging authentication, authorization and attribute data between security domains that have established trust relations. SAML provides assertion and protocol elements that may be used for retrieval of attributes for use in an XACML Request Context.

Figure 4.3 shows a SAML Attribute Assertion which includes the name of the attribute issuer, an optional digital signature for authenticating the attribute, an optional subject identity to which the attribute is bound, and optional conditions for use of the assertion that may include a validity period during which the attribute is to be considered valid. Such an assertion is suitable for storing attributes in an Attribute Repository, for transmitting attributes between an Attribute Authority and an Attribute Repository, and for transmitting attributes between an Attribute Repository and a PEP or XACML Context Handler.

For querying an on-line Attribute Authority for attributes, and for holding the response to that query, SAML defines Attribute Query and Attribute Response elements. SAML major application areas include SSO, attribute-based authorization and web services security. The means by which lower-level communication or messaging protocols (such as HTTP or

SOAP) are used to transport SAML assertion or protocol messages is defined by the SAML bindings (Demchenko, 2009).

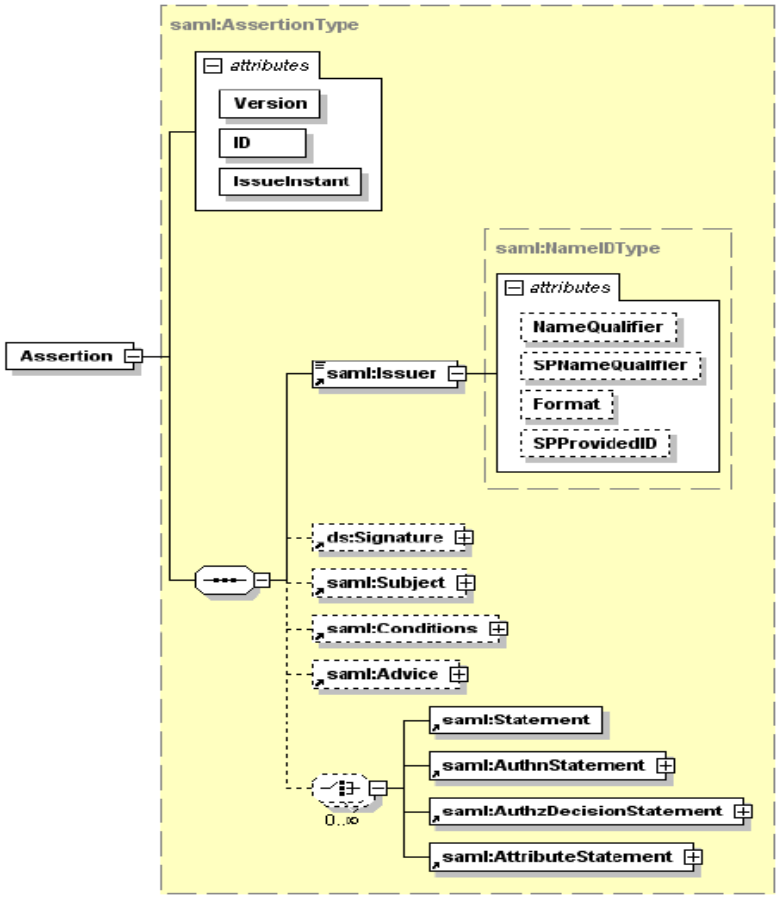


Figure 4.3: SAML Assertion (Demchenko, 2009)

4.2.1 Aligning XACML with SAML

The SAML 2.0 profile of XACML (OASIS, 2005) defines how to use SAML 2.0 to protect, store, transport, request, and respond with XACML schema instances and other information needed by an XACML implementation. SAML 2.0 enhancements include features derived from the Liberty Alliance Identity Federation Framework (ID-FF) V1.2 specifications that were contributed to the OASIS Security Technical Committee in 2003,

capabilities present in the Internet2's Shibboleth architecture, and enhancement requests resulting from experience with numerous deployments of SAML V1.x in the industry. Thus SAML 2.0 represents the convergence of SAML 1.1, Liberty ID-FF 1.2 and Shibboleth 1.3.

SAML 2.0 protocol uses security tokens containing assertions to pass information about a principal (usually an end user) between a SAML authority (i.e., an identity provider) and a SAML consumer (i.e., a service provider). This profile requires no changes or extensions to XACML, but does define extensions to SAML. SAML 2.0 complements XACML functionality in many ways:

- Use of SAML 2.0 Attribute Assertions with XACML, including the use of SAML Attribute Assertions in a SOAP Header to convey Attributes that can be consumed by an XACML PDP
- Use of SAML to carry XACML authorization decisions, authorization decision queries, and authorization decision responses
- Use of SAML to carry XACML policies, policy queries, and policy query responses
- Use of XACML authorization decisions or policies as Advice in SAML Assertions
- Use of XACML responses in SAML Assertions as authorization tokens.

Figure 4.4 illustrates how SAML protocol and assertions and, XACML Request/Response messages can be used in a typical policy based decision-making.

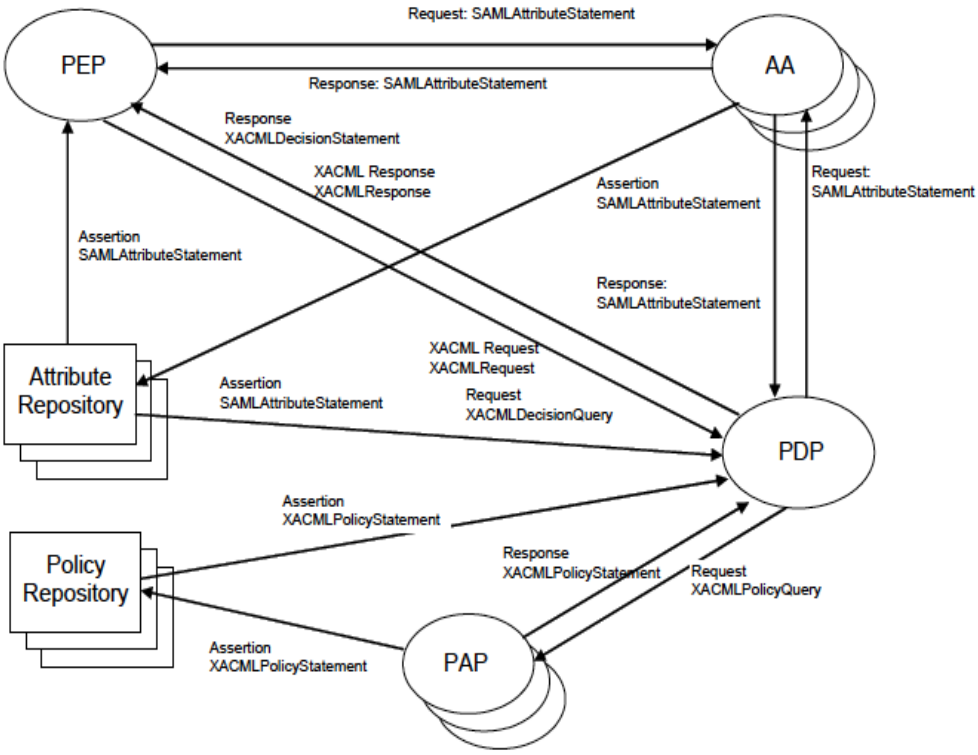


Figure 4.4: Integration of SAML with XACML (Demchenko, 2009)

SAML attribute assertions may be used as input to authorization decisions made according to the XACML standard specification. Since the SAML attribute format differs from the XACML attribute format, mapping of these attributes must be carried out. The XACML attribute profile facilitates this mapping by standardizing naming, value syntax, and additional attribute metadata. SAML attributes generated in conformance with this profile can be mapped automatically into XACML attributes and used as input to XACML authorization decisions.

From Figure 4.4, the service requester initiates request to access a specific resource. The PEP sends the resource access request to an XACML PDP in a XACML authorization decision query. The PEP may obtain attributes in one of the following ways:

- Directly from an online Attribute Authority (an entity that binds attributes to

identities) using an *Attribute Query*. This query returns an Attribute Statement in the SAML response.

- From a repository where they were stored previously by an Attribute Authority in the form of SAML Attribute Statements.

The XACML Policy Decision Point evaluates the resource access request and decides additional attributes are needed from the AA or attribute repository. This allows the XACML Policy Decision Point to augment the resource access request with additional attributes. The Attribute Authority creates an assertion of an attribute statement in the Attribute Repository, which also makes the attribute statement available to the XACML Policy Enforcement Point or to the XACML Policy Decision. The XACML Policy Administration Point finds relevant policies from the XACML Policy Repository and creates a policy statement assertion. With the availability of relevant policies and attributes, the XACML Policy Decision Point is able to respond to the XACML Policy Enforcement Point with a XACML authorization decision statement. The SAML profile of XACML provides means to write assertions regarding the identity, attributes, and entitlements of a subject, and defines a protocol for exchanging these assertions between entities.

4.3 Trust Model

Trust establishment between Service consumers, Service Providers (SP) and Identity Providers (IdP) is of very high importance in scenarios where the Service Providers and Service Consumers are strangers. Such a trust model is referred as IdP/SP model. SAML is the reference XML-based standard for implementing the IdP/SP model that addresses several

security scenarios and supports many security technologies. The power of SAML is that it can establish trust relationship between entities with different security mechanisms. SAML is different from other security systems due to its approach of expressing assertions about a subject that other applications within a network can trust. According to the identity management model (IDM), SAML uses the IdP and SP concepts as defined in a generic Authentication and Authorization Infrastructure (AAI) such as Liberty ID-FF, Shibboleth. (Celesti, Tusa, Villari, & Puliafito, 2013) Figure 4.5 shows the exchange of attributes between the Service Provider (SP), Identity Provider (IdP) and Attribute Authority (AA).

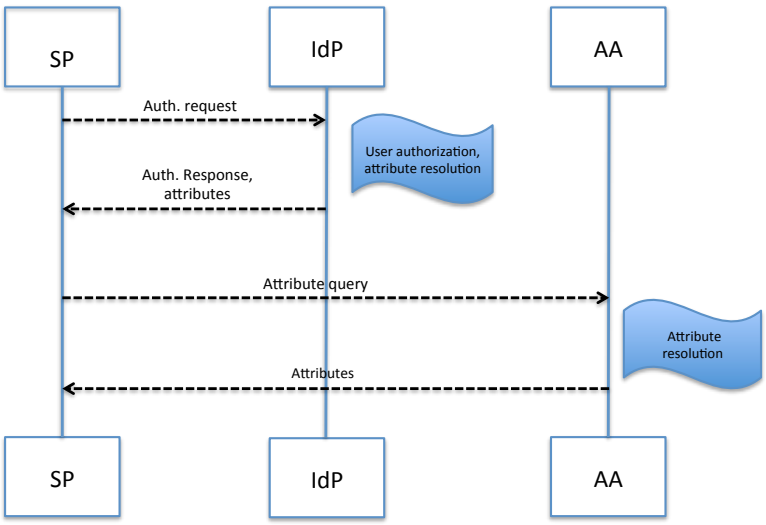


Figure 4.5: Attribute Exchange between SP, IdP and AA

4.4 RBAC

Role-Based Access Control (RBAC) (Ferraiolo & Kuhn, 1992) simplifies the administration and management of privileges using roles that can be updated without updating the privileges for every user on an individual basis. Permissions are associated with these roles and users are assigned to appropriate roles based on factors such as their

responsibilities and qualifications. Users can be easily reassigned roles. Roles can be granted new permissions, and organized in role hierarchies to reflect the organization's lines of responsibility and authority. The RBAC model is as shown in Figure 4.6.

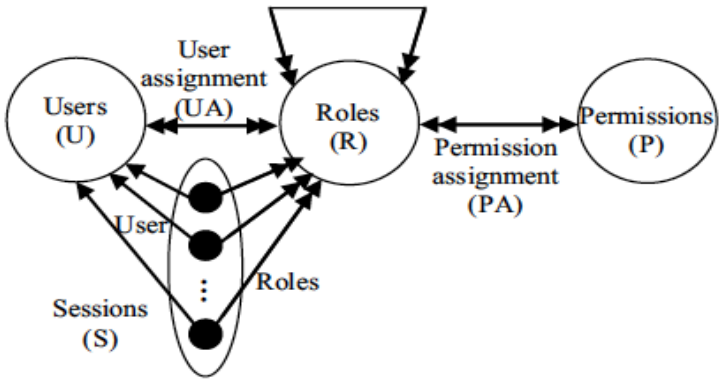


Figure 4.6: RBAC and its Relationships (Ferraiolo & Kuhn, 1992)

The RBAC model taxonomy consists of four models – core, hierarchical, statically and dynamically constrained RBAC. Core RBAC covers the basic set of features that are included in all RBAC systems. Hierarchical RBAC adds the concept of a role hierarchy (partial ordering of roles). Constrained RBAC includes static and dynamic separation of duties (SOD) properties. Statically constrained RBAC adds constraint relations imposed on role assignment relations while Dynamically constrained RBAC imposes constraints on the activation of sets of roles that may be included as an attribute of a user's subjects (NIST, 2006).

Based on research carried out by Ferraiolo & Kuhn (1992), the RBAC model was outlined as a more appropriate system of control in civilian government or commercial organizations than either the multilayer security of MAC or the user-centered security model of DAC. RBAC addresses the requirements of multi-user and multi-application systems in large organizations, and simplifies administration of access rights. While RBAC scales better, it

still is not suitable for highly dynamic applications, where unique roles have to be created for all combinations of security labels and constraints (Burmester, 2012). Because of RBAC's relevance, it has been widely investigated and several extensions as well as possible applications have been proposed e.g., TRBAC, X-GTRBAC (Damiani, Bertino, Catania, & Perlasca, 2007).

4.5 ABAC

Attribute-Based Access Control (ABAC) is a logical access control methodology where authorization to perform a set of operations is determined by evaluating attributes associated with the subject, object, requested operations, and, in some cases, environment conditions against policy, rules, or relationships that describe the allowable operations for a given set of attributes. Authorization is defined for subject descriptors, consisting of several attribute conditions. See Figure 4.7.

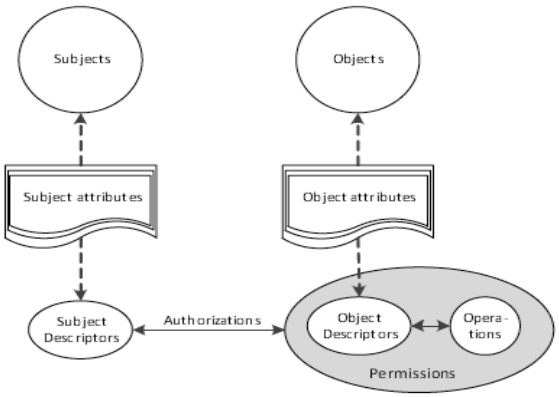


Figure 4.7: ABAC Framework

Permissions consist of a combination of an object descriptor that contains a set of attributes and attribute conditions, and an operation on the object(s) specified by the descriptor. Environment attributes such as, the time of day, temperature can also be used to

control access. ABAC can encompass the functionality of RBAC by treating identities or roles as attributes. Compared to RBAC, ABAC captures dynamic environment attributes that are temporal. It can also enforce fine-grained policies, specifically when the number of subject and object attributes becomes large. However, in highly dynamic real-time applications, the event space that determines the attribute values can be very large thus making any attempt to capture real-time availability scenarios non scalable (Burmester, 2012). ABAC enables object owners or administrators to apply access control policy without prior knowledge of the specific subject and for an unlimited number of subjects that might require access and rules do not have to be modified as subjects are added (NIST, 2013a).

4.6 TBAC

Task-Based Access Control (TBAC) differs from traditional access controls and security models in many respects. Instead of having a system-centric view of security, TBAC approaches security modeling and enforcement at the application and enterprise level. It lays the foundation for a new breed of "active" security models i.e. security modeling and enforcement from the perspective of activities or tasks. In an active approach to security management, permissions are constantly monitored and activated and deactivated in accordance with emerging context associated with progressing tasks (such as in workflows). Figure 4.8 shows the concepts, features, and components that make TBAC an active security model, which includes the following: (Thomas & Sandhu, 1997)

- Modeling of authorizations in tasks and workflows as well as the monitoring and management of authorization processing and life-cycles as tasks progress
- Use of both type-based as well as instance and usage-based access control

- Maintenance of separate protection states for each authorization-step
- Dynamic runtime check-in and checkout of permissions from protection states as authorization-steps are processed.

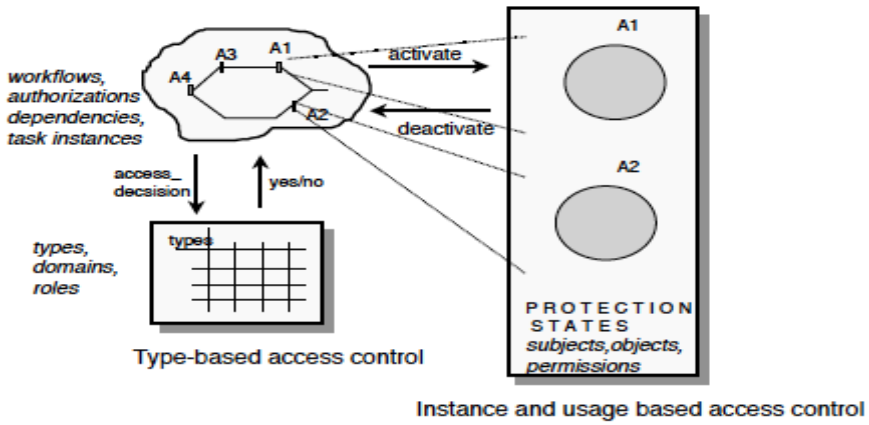


Figure 4.8: TBAC and its Relationships (Thomas & Sandhu, 1997)

Permissions are controlled and managed in such a way that they are turned-on only in a just-in-time fashion and synchronized with the processing of authorizations in progressing tasks (Thomas & Sandhu, 1997).

4.7 Conceptual Categorization Framework

To define access control requirements for Cloud Computing systems, the use of classic systems engineering processes has lead to the adoption of existent or modified access control approaches and, the existing access control requirements are usually adjusted to fit the limitations of the mechanism at hand, leading to limitations in policy specification. Due to the inadequacy of contemporary implementations in fulfilling the new security requirements set by cloud systems, Gouglidis & Marvridis (2009) proposed a conceptual categorization framework, as shown in Figure 4.9, for identifying access control requirements in Cloud

Computing environments.

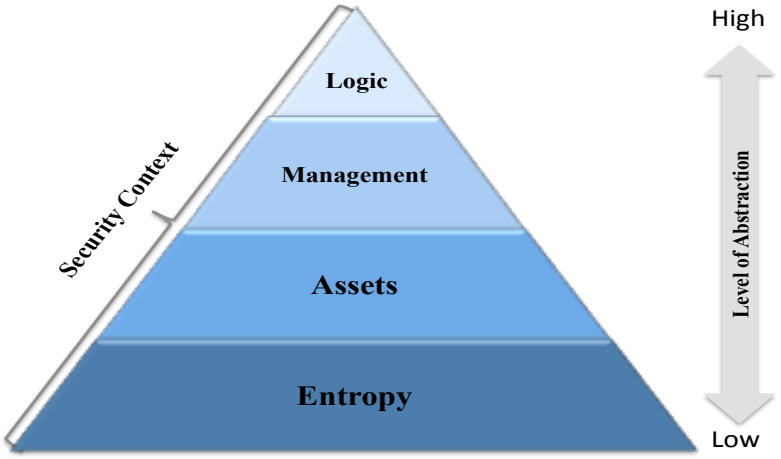


Figure 4.9: Conceptual Categorization Layers (Gouglidis & Mavridis, 2009)

The conceptual categorization identifies and groups requirements into four different abstraction layers. Classification of the layers is as shown in Figure 4.10.

- *Entropy layer:* The entropy layer identifies requirements from the virtual and geographic dispersion of objects in a system. The entropy level can be relatively high or low based on the type of cloud model deployed e.g. for applications deployed with public or hybrid model, the entropy is relatively high and, it is low for private deployment model. In addition, the hosts used to provide the assets can also be characterized by high dispersion with the public or hybrid model and low for private model. Issues like authentication can be examined under the entropy layer.
- *Assets layer:* The asset layer identifies requirements from the type of shared objects within the boundary of the entropy layer. An asset can either be of software (service & data) or hardware type (computation, storage & equipment).
- *Management layer:* The management layer defines requirements from policy management. It is used to fulfill the need for capturing security issues raised from the

management of policies and trust relationships among objects. The distribution level of a system as defined in the entropy layer affects the management of its policies. In enterprise applications using Cloud Computing technologies centralized management is required whereas in public and hybrid models where collaboration is required, management of policies is required to be distributed and applied among participating organizations. Application of management operations can either be static or dynamic. Operations considered here include problem identification, conflict resolution, privilege revocation and trust management.

- *Logic layer:* The logic layer incorporates requirements that are not handled by the former layers. The logic layer can be classified into two – the model and execution classes. Both subclasses have similar requirements, i.e., support of collaborations, workflows and co-operations to tackle highly dynamic environments.

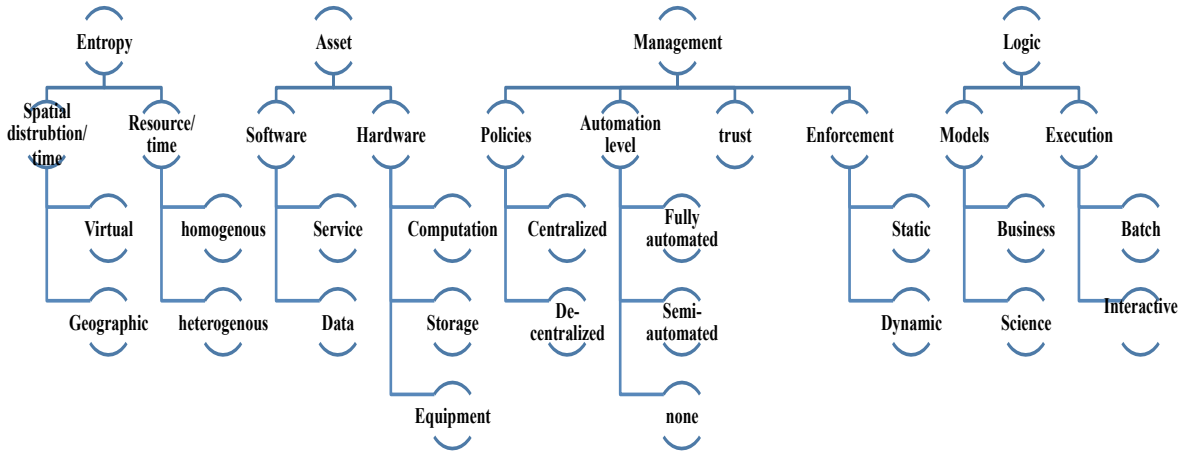


Figure 4.10: Conceptual Categorization Classification (Gouglidis & Mavridis, 2009)

In order to define the requirements for an access control system, suitable for the cloud,

characteristics exposed by the security infrastructure layers in the cloud (application, host, network) were incorporated. These characteristics may vary depending on the use cases that need to be supported by a specific system.

4.8 XACMUT

XACml MUTation (XACMUT) (Antonia Bertolino, Lonetti, & Marchetti, 2012) is a framework that performs the complete process of mutation analysis i.e. apply a set of mutation operators to a given XACML policy, execute a given test suite on the policy and its mutants, compare the obtained results and provide statistics about fault detection effectiveness.

4.8.1 Mutant Operators

Mutation analysis has been used for measuring the effectiveness of a test suite. In this thesis, we apply mutation analysis. With mutation operators, the policy under test is modified to derive a set of faulty policies (mutants) each containing a fault. These operators emulate faults in the specification of XACML functions and in the definition order of the rules. Some mutation operators implemented in the XACMUT tool include but are not limited to the following operators. Others are as shown in Figure 4.11.

- *Policy Set Target True (PSTT):* It removes the Target of each PolicySet ensuring that the PolicySet is applied to all requests.
- *Policy Set Target False (PSTF):* It modifies the Target of each PolicySet such that the PolicySet is never applied to a request.
- *Policy Target True (PTT):* It removes the Target of each Policy ensuring that the

Policy is applied to all requests.

- *Policy Target False (PTF):* It modifies the Target of each Policy ensuring that the Policy is never applied to a request.
- *Rule Target True (RTT):* It removes the Target of each rule ensuring that the Rule is applied to all requests.
- *Rule Target False (RTF):* It modifies the Target of each rule such that the Rule is never applied to a request.
- *Rule Condition True (RCT):* It removes the condition of each Rule ensuring that the Condition always evaluates to True.
- *Rule Condition False (RCF):* It manipulates the Condition values or the Condition functions ensuring that the Condition always evaluates to False.
- *Change Policy Combining Algorithm (CPC):* It replaces the existing policy combining algorithm with another policy combining algorithm. The set of considered policy combining algorithms is for deny-overrides, permit-overrides, first-applicable, only-one-applicable.
- *Change Rule Combining Algorithm (CRC):* It replaces the existing rule combining algorithm with another rule combining algorithm. The set of considered rule combining algorithms is f deny-overrides, permit-overrides, first-applicable.
- *Change Rule Effect (CRE):* It changes the rule effect by replacing Permit with Deny or Deny with Permit.

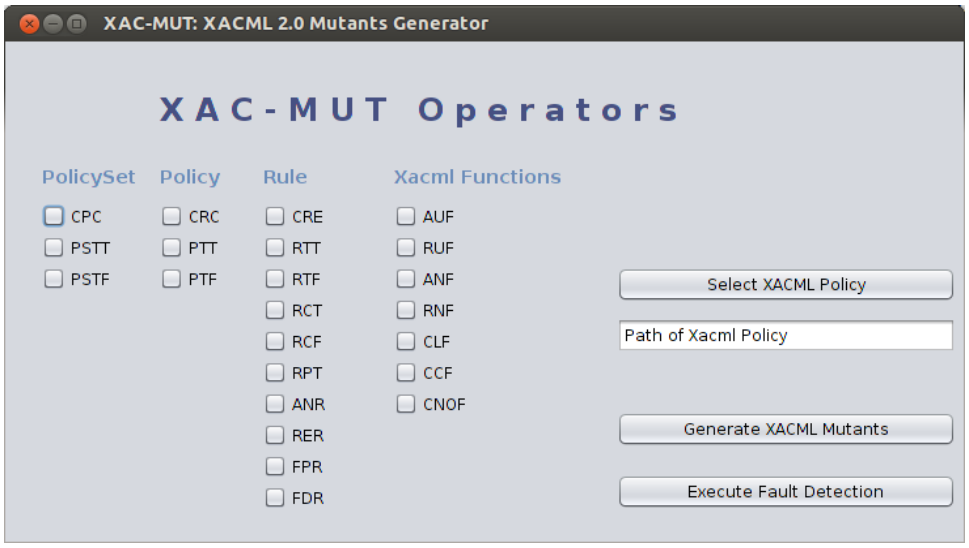


Figure 4.11: XACMUT 2.0 Mutant Generator Screenshot (Antonia Bertolino, Lonetti, & Marchetti, 2012)

4.8.2 XACMUT Architecture

The main architecture of XACMUT and its components are as shown in Figure 4.12.

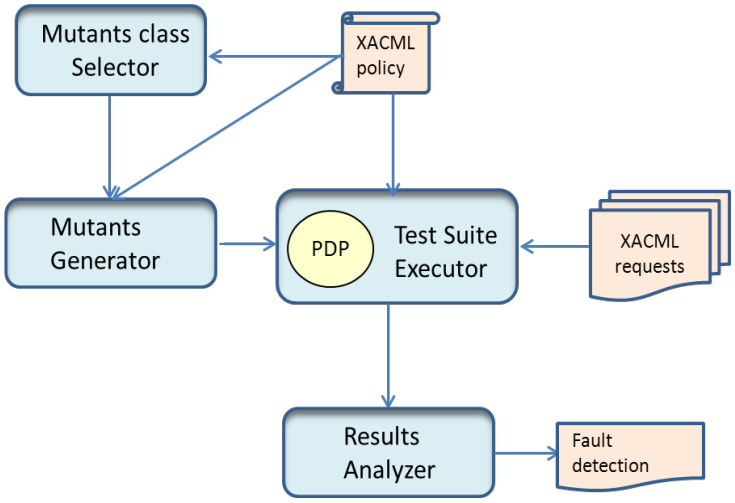


Figure 4.12: XACMUT Architecture (Antonia Bertolino, Lonetti, & Marchetti, 2012)

- *Mutants Class Selector:* This component is provided by an interface for the user interaction as shown in Figure 4.11. The user can select the XACML policy to be used, select the mutation operators to be applied, select the set of XACML requests

that will be used, execute the policy mutants against a test suite, and verify which mutants have been killed by the application of the test suite.

- *Mutants Generator:* This component generates the mutants for a given XACML policy using mutation operators selected by the user.
- *Test Suite Executor:* This component executes the XACML requests provided by the user on the original XACML policy and on the generated set of mutated policies. For requests execution, this component integrates a PDP engine (specifically the Sun PDP), which is able, given a policy and a request, to provide the corresponding result (Permit, Deny, NotApplicable or Indeterminate).
- *Results Analyzer:* This component takes as input all the results obtained by the execution of the test suite on the original XACML policy and on its set of mutants and elaborates the fault detection effectiveness. It works as follows: for each request the result obtained by its execution on the original XACML policy is compared with those obtained on its mutants set. If the results are different, the mutant is classified as killed. The component provides as output the list of mutants killed and alive, and the percentage of fault detection effectiveness obtained by the requests execution.

4.9 ASP

Answer set programming (ASP) (Lifschitz V. , 2008) is a form of declarative programming oriented towards difficult, primarily NP-hard, search problems. In ASP, search problems are reduced to computing stable models, and answers set solvers – programs for generating stable models – are used to perform search. The search algorithms used in the design of many answer set solvers are enhancements of the Davis-Putnam-Logemann-

Loveland procedure, and they are somewhat similar to the algorithms used in efficient SAT (Satisfiability Checking) solvers.

ASP provides a common basis for formalizing and solving various problems, but it is distinct from others solvers in that it focuses on knowledge representation and reasoning: its language is an expressive nonmonotonic language based on logic programs under the stable model semantics (Lifschitz M. G., 1988), which allows elegant representation of several aspects of knowledge such as causality, defaults, and incomplete information. What distinguishes ASP from other nonmonotonic formalisms is the availability of several efficient implementations, answer set solvers, such as DLV, SMODELS1 and CLASP3, which led to practical nonmonotonic reasoning that can be applied to industrial-level applications.

4.10 Concluding Remark

This chapter covers background information on frameworks that are used throughout this thesis. It introduces the standard access control mechanisms (ABAC, RBAC & TBAC), whose advantages are combined to design our proposed framework. It also identifies SAML as the protocol for the exchange of assertions as well as defines the process of aligning SAML with XACML (policy language used) to enforce access control policies on resources in the cloud. In order to identify the requirements for an access control system suitable for the cloud, the conceptual categorization framework is introduced. Since the identity of users in the cloud is mostly unknown, a trust mechanism based on IDM model is presented. Our framework is implemented using XACMUT and ASP.

In the next chapter, the proposed policy-based framework is presented.

Chapter 5: Proposed Policy-Based Management Framework

In this chapter, a new architecture suitable for the cloud is proposed. We first define the access control requirements for a generic Cloud Computing scenario based on Gouglidis & Marvridis' proposed conceptual categorization framework described in Section 4.7 and then, make a comparison of existing access control models used for the cloud environment to determine their applicability. The results of our analysis are used to develop a policy-based framework suitable for the cloud.

5.1 Identifying Access Control Requirements for Cloud Security

A cloud scenario is used to identify a suitable access control requirement for the cloud. The cloud environment involves a hospital that wishes to utilize public clouds to support an integrated care delivery model for its patients. It deployed an Electronic Medical Record (EMR) application hosted in the cloud in form of Software as a Service (SaaS) to be used by Hospitals, Patients, Medical Personnel, Primary Health Care Networks, Pharmacies, Insurance Companies and Government, which are in different domains, through the Internet. A cloud service provider hosts the EMR application and users from all participating parties are required to collaborate with each other, manage their data and request on-demand use of services within a single application and database. Direct call to web services can be made by organizations with existing medical record applications to integrate them with the EMR application. Organizations without existing medical record applications can use the web user interface to access the EMR application. Doctors, patients can also have access through the

web or various mobile devices from anywhere. The operational environment is illustrated in Figure 5.1.

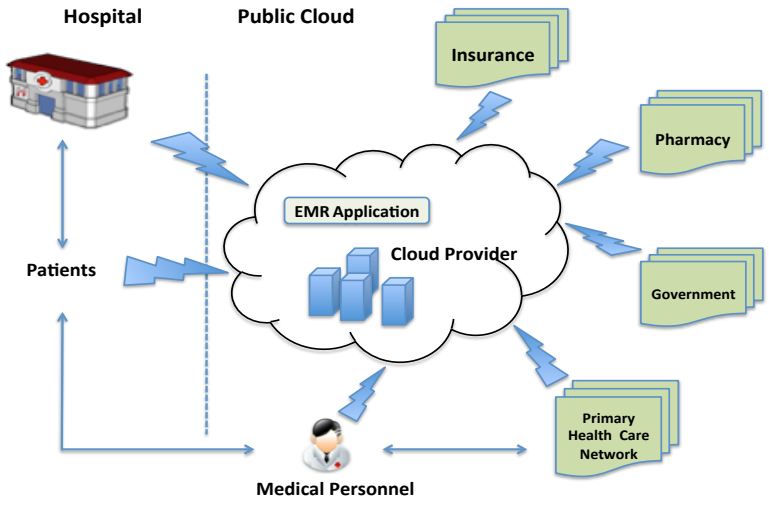


Figure 5.1: Generic Cloud Scenario Operational Environment

From this scenario, how can the right amount of a patient's medical information be safely disclosed to the right personnel at any point in time considering the security and legislative concerns governing a patient's medical record e.g. laws such as HIPAA, PIPEDA.

This scenario requires new technical protocols and innovative solutions that are able to address multi-party security requirements according to predefined policies. From a user side, there will be an explosion of identities and devices. There will be the need for identity aggregation on devices, as well as for analyzing the impact of locations that provide environment transparency and/or disclosure. Using the conceptual categorization framework in Section 4.7requirements identified under each layer are as follows:

- *Entropy layer:* Requirements include limiting access to resources. Here, the public model of deployment is being used, which is characterized by high dispersion of collaborating parties. Therefore, proper authentication of users is important as domains may make use

of different authentication protocols. To cope with the complexity of the entropy layer, the access control system must be able to support attributes that are dispersed across domains to overcome heterogeneity issues.

- *Asset layer:* The cloud service provider exposes the SaaS application as a set of services that can be realized using technologies such as web services. Access control in a Cloud Computing environment is mostly coarse-grained at the service level, as it only requires permission, denial and restriction to services. However, in this scenario, fine-grained access control must be considered at the data level because collaboration among participating parties is required, which only authorizes users to access specified assets if they are legitimate users. Therefore, the access control system must be able support fine-grained access control on data.

  Furthermore, multi-tenancy must be supported to guarantee that data are protected from each other throughout collaboration. Multi-tenancy has been much associated with PaaS or IaaS. However, SaaS applications too have various degrees of multi-tenancy depending on the how much of the SaaS application layer is designed to be shared. The highest degree of multi-tenancy allows the database schema to be shared and supports customization of the business logic, workflow and user-interface layers. It is important to ensure that the cloud delivery model 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.

- *Management layer:* In the cloud, typical issuers come from different organizations with independent administrative authorities. Therefore, centralized authority may not be suitable. In the specified scenario, since collaboration is required and users may be

unknown, a decentralized authority is needed. A decentralized authority is required where the different organizations retain administrative control over their resources.

Moreover, enterprises and corporations are usually composed of different working areas or departments. Each area may have its own enforcement points to authorize access to their resources. However, some authorization decisions may depend on the information belonging to other areas or domain, which cannot be directly accessed due to privacy issues. In this sense, instead of recovering the required information, the authorization decision is delegated to the areas that could handle it, and the results of such delegated decisions are combined to form an appropriate decision. With a decentralized authority, different tenants can delegate their authorization decisions to other tenants to form the overall access control decision. Also, since the number of interacting users can be quite large, manual provisioning and de-provisioning of access rights can be tedious, insufficient and impractical. Dynamic assignment of access right is a feature that can be added to the access control system.

- *Logic layer:* Permissions defined based on objects and their relationships can be complex due to the number of users, resources, data types and access types involved. To manage access rules, the authorization system must be able to define conditional access rules, evaluate the context associated with them dynamically, and provide an effective means of managing permissions to support dynamic collaborations and workflows. Furthermore, the principles of control that includes semantic constraints such as separation of duties, least privilege, and contextual constraints should be supported.

Overall, the following characteristics were identified as required in the generic cloud scenario described above – decentralization, fine-grained access control, multi-tenancy, trust/credential federation, constraint specification, collaboration, active and policy

specification.

5.2 Comparison of Existing ACMs for Cloud Security

In this section, we review a number of access control models such as the RBAC (Reeja, 2012), X-GTRBAC (Damiani, Bertino, Catania, & Perlasca, 2007), (Danwei, Xiuli, & Xunyi, 2009), MTAS (Calero, Edwards, Kirschnick, Wilcock, & Wray, 2010), MT-RBAC (Tang & Sandhu, 2013), S-RBAC (Li, Liu, Wei, Liu, & Liu, 2010), RB-MTAC (Yang, Lai, & Lin, 2013) and TRBAC (Narayanan & Giine, 2011) that have been proposed for the cloud to determine their applicability to the cloud scenario discussed in Section 5.1 The criteria used are based on the characteristics identified and the evaluation is based on the level of fulfillment of these requirements by these access control models. **Table 5.1** illustrates our evaluation of these models using the following scale: 0 – none, 1 – low, 2 –moderate, and 3 – high.

Table 5.1: Comparison of Existing Access Control Models for Cloud Computing

CHARACTERISTICS	RBAC	X-GTRBAC	UCON	MT-RBAC	MTAS	S-RBAC	RB-MTAC	TRBAC
Decentralized	0	0	0	3	0	0	0	0
Fine-grained	1	3	3	1	1	1	1	3
Multi-tenancy	0	0	0	3	3	3	3	0
Trust/Credential Federation	0	0	3	3	0	0	0	0
Constraint Specification	1	2	3	3	2	2	2	2
Collaboration	0	0	0	3	0	0	0	3
Policy Specification & Enforcement	2	2	2	3	2	2	2	3
Active (workflows)	0	0	0	0	0	0	0	3

From **Table 5.1**, we deduce that most of the existing models reviewed were extensions of RBAC, which are centralized and slightly coarse-grained. A number of the models are now looking to ensure multi-tenancy but in most cases, the identities of the users need to be known. In terms of collaboration, which is a requirement for modern enterprises, a large number of these models do not provide cross-tenant authorization and also, none of the models except TRBAC took workflows into consideration.

RBAC is widely deployed for organizations, it provides simpler access administration and user permission review, but it is not sufficiently granular to restrict data access only to the proper subjects, e.g., a patient record authorized for interns in a hospital may make it accessible to all interns and creating more roles to make permissions fine-grained can result in role explosion. Also, role inheritance in RBAC does not fully reflect the same process in real organizations. For example, if a role R1 (i.e., a higher role) inherits from another role R2 (i.e., a lower role) this means that R1 inherits the full permissions set of R2, whereas in real life organizations, a higher role only inherits a partial permission set of a lower role.

In addition, access control policies in RBAC are static; making it inflexible in dynamically changing environment and this hinders the application of RBAC to achieve a more fine-grained access control required for Cloud Computing. RBAC is not able to consider contextual information in access control decision-making. However, various variants of RBAC have been proposed to address these insufficiencies.

5.3 Proposed Access Control Architecture

Figure 5.2 depicts the proposed access control architecture, which comprises the following architectural components:

- *Context Handler* acts as a communication bus by creating a mutual context between components,
- *Policy Decision Point (PDP)* makes an access control decision by evaluating the request against the available policies,
- *Policy Enforcement Point (PEP)* forwards the received requests to PDP and enforces the obligations returned from PDP,
- *Policy Administration Point (PAP)* makes the policies available to PDP
- *Policy Information Point (PIP)* retrieves the attribute values requested by context handler.
- *Trust Manager:* This service responds to requests for attributes in form of SAML attribute queries.
- *Role Enablement Authority:* This is responsible for assigning roles to users and for enabling roles for use within a user's session, Policies are used to determine which users are allowed to enable which roles and under which conditions. The roles are expressed using attributes.

Among these components, PEP and PDP are of particular importance as they are the components in which state information can be maintained.

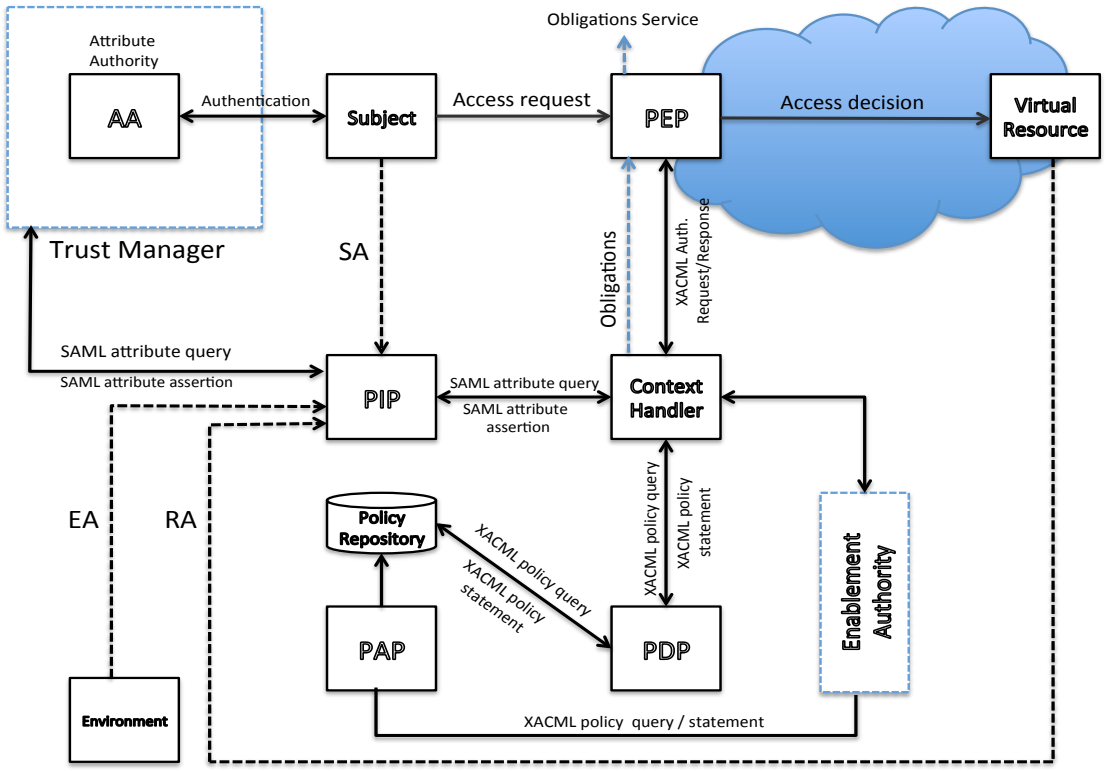


Figure 5.2: Proposed Access Control Architecture

To define our architecture, the core of our approach involves the process of extracting user attributes using SAML v2.0 and mapping into XACML attributes as described in Section 4.2.1, to be used for making authorization decisions. In the context of this thesis, the SAML profile of XACML specified by OASIS as described in Section 4.2is used to convey detailed information about the subject, resource, and action from the service gateway (PEP) to the centralized authorization service (PDP) and to convey back the authorization decision.

To express roles as XACML attributes, the core and hierarchical role based access control (RBAC) profile of XACML, which describes the use of XACML to meet the requirements of RBAC as shown in Section 4.1is used. Dynamic roles can then be assigned based on the Role Enablement Authority (REA) component using defined XACML role assignment

policies.

In cloud systems, in many cases, the service providers and the service consumers are strangers. Since they do not have pre-established trust between them, the service provider must be able to authenticate the unfamiliar users and then determine whether the requestors have enough permission to access the requested services. When the IDP/SP model described in Section 4.3is applied to the cloud, the Service Provider can request the Service Consumers' attributes from the Identity Provider (Trusted Authority) that is part of an Attribute Authority (AA). SAML Attribute Authorities are particularly useful when there is an existing SAML identity federation with established policies and trust. Users' attributes can be retrieved during single sign-on process and merged with the other attributes, so that applications receive them in a standard way and there is no need to modify them to make them consume those extra attributes.

Decision-making using the authorization framework occurs after a client requests for a resource, the request is then intercepted by the PEP in the authorization engine and sent to the PDP to begin evaluation. When the PDP receives the evaluation request from PEP, it first collects information needed by calling the PIP. The PIP issues an Attribute Query to the (Trust Manager) Identity provider attribute service for attributes about that user, which may or may not include the user's identity. The identity provider sends an attribute assertion containing trusted information about the user to the PIP. Each PIP then returns a normalized representation of the collected attributes. The attributes are then grouped as a single set of attributes per entity and are stored, so that the PDP can access them when evaluating their policies. The PDP combines the decisions, using a policy combination algorithm to render a final decision, and returns the decision to the PEP. Context Handler gets the authorization decision, and transforms it into a format, which can be accepted by PEP. Then PEP enforces

the authorization decision.

5.4 Proposed Access Control Model for Cloud

This section presents the proposed model designed for cloud environment. This model is decentralized, allowing for the evaluation and enforcement of access control at run-time and it is capable of applying policies on resources resident in the cloud. In order to address all the characteristics identified in Section 5.1as requirements for cloud access control model, we hereby introduce an attribute-driven, role and task based access control framework, which can be implemented using the architecture described in Figure 5.2.

In this framework, authorization is in two folds – first, dynamic roles are created using rules applied on the attributes of tenants/subjects to authorize tenants/subjects to roles. Second, permissions to access resources are granted to tasks mapped to roles for a fine-grained and active access control.

5.4.1 Components of the Proposed Model

The proposed access control model will utilize the characteristics of the standard ABAC (Burmester, 2012), RBAC (Ferraiolo & Kuhn, 1992) and TBAC (Thomas & Sandhu, 1997) models thereby combining their benefits to synergize the advantages of each. Below are important features and components of the proposed model. See Figure 5.3.

- *Role-based Access:* The characteristics that were defined demand both the support of both a centralized and a decentralized access control among different domains and the dynamic joining of new ones. The advantage of roles is considered in our model due to the convenience they bring to administration of permissions, hierarchies and temporal

constraints in centralized architectures.

- *Attribute-based Access:* With the strength from attribute-based approach, access control policy can be applied without prior knowledge of the specific subject using their attributes. With attributes, the model can cope better in a decentralized environment as this feature makes it possible to provide inter-domain access to users in a collaborative way without the need for them to be known by the resource a priori. Attribute-based access is also used to support finer-grained access control through the use of context.
- *Task-based Access:* Tasks being the basic logic units of business processes is considered here to help overcome the problem of resource heterogeneity. Inclusion of tasks in our model provides an active access control model where permissions are activated or deactivated according to the current task or process state, which is a requirement for modern enterprises. The introduction of tasks also aids collaboration.
- *Trust Mechanism:* In the cloud environment, users wishing to access the cloud resource are mostly unknown but they have attributes that have been created in other domains by identity providers. Different service providers and identity providers have come together to form a circle-of-trust (CoT) based on some agreement. A trust mechanism is established as described in the trust model in Section 4.3to retrieve user attributes upon authentication.
- *Constraints:* To satisfy the requirements of the logic layer, the model should be able to satisfy some preconditions when authorizations are granted such as granting an access right to specific subjects and separation of duties. Obligation rules must also be followed in any action to constrain the way requests are executed.

- *Dynamic Roles:* This is a feature added to this model to support a more flexible form of roles assignment that does not have the limitations of the traditional static roles called dynamic roles. The roles created by the security administrator has an associated set of attributes that must be satisfied by the subjects who are to be assigned to that role and they are determined at runtime, as requests for access are made. The process of assigning roles to users is made dynamic using expressions of the tenant or subject attributes rather than their unique identifiers. Inter-domain attribute exchanges and attribute mappings are provided through the trust model described in Section 4.3
- *Two-fold Authorization:* This is the algorithm of authorization that takes into account not only access rules, but also internal states (roles, tasks, constraints and attributes) of the authorization subject/tenant. In this model, permissions are authorized both to roles and to tasks based on the tenant or subject attributes. As a result, different subjects will observe different instances of the same cloud service in different sessions. The idea is to leverage the RBAC features regarding permissions assignments based-roles.

  In addition, subjects can get permissions through tasks when they execute a workflow, thereby supporting task dynamic constraints. This prevents subjects from accessing resources when not executing the corresponding tasks, and thus satisfying the confidentiality requirements of modern enterprises. Here, with the roles assigned to subjects, they can access resources and accomplish their duties through tasks. Authorization information will be inferred from access control data structures, such as subject-role assignment and task-role assignment relations.

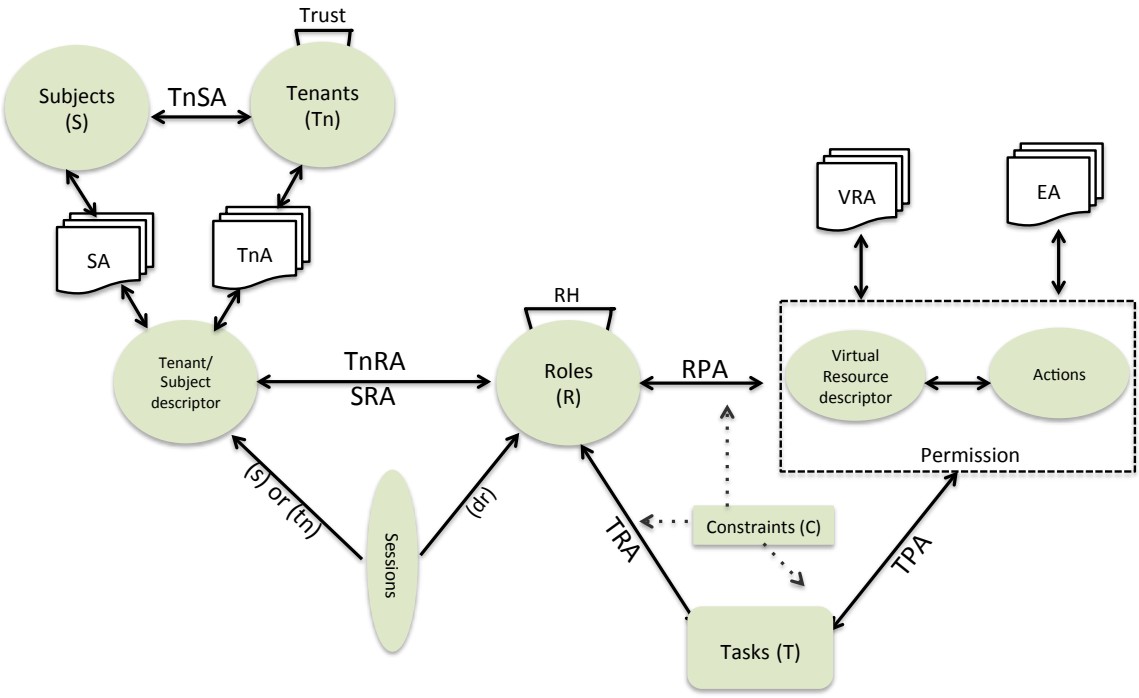


Figure 5.3: Proposed Access Control Model

5.4.2 Elements

Elements identified in this model include Tenants, Subjects, Virtual Resources, Environments, Roles, Tasks, Permissions, Actions and Constraints.

- *Tenants (Tn):* Tenants are clients (enterprises or individual) of the cloud service provider (CSP) who use the cloud services.
- *Subjects (S):* A subject is an entity that takes action on resources. They can be users, which are either employees or clients of a cloud tenant, applications or devices. The identity and characteristics of subjects can be defined based on attributes such as IP address, clearance level, organizational unit, and email address.
- *Virtual Resources (VR):* A virtual resource is an entity in the cloud that is acted upon by a subject, e.g., files and databases. Their attributes e.g., file name, extension and

URI, can be leveraged to make access control decisions.

- *Environment (E):* This describes the context under which access may be provided. The environment is associated with a set of attributes e.g., access time, device type, and location.
- *Roles (R):* This is a job function associated with a tenant. It is also provided to each subject based on the activities they are to perform. A role links a tenant or subject to certain tasks providing access rights to the needed virtual resource. Each tenant or subject is assigned a role for initial coarse-grained access control based on the attributes.
- *Tasks (T):* They are fundamental units of business processes and may include several subtasks. Tasks are mapped to roles and access rights are determined for fulfilling assigned tasks for a fine-grained and active access control.
- *Permissions (P):* This is the authorization to perform an action on a virtual resource. It is a set of access policies for fine-grained access control. Permission is associated with a single tenant or subject. A tenant or subject may have multiple permissions.
- *Actions (A):* They are requests made by tenants or subjects to be performed over virtual resources, e.g., read, write, modify and print.
- *Constraints (C):* They are conditions that must be satisfied for permissions to be granted. They are used to describe different security policies, e.g., separation of duties, least privilege, and delegation.
- *Sessions (SS):* A session refers to the set of events that occurs from when a tenant or subject connects to the cloud system till when they disconnects from it. It provides conditions for supporting dynamic role assignment. During each session, a tenant or subject is assigned to a set of roles. The active role will be changed dynamically

among assigned roles for each interaction. In multi-tenant cloud environments, the subject and active roles of a session are not necessarily from the same tenant. Session profiles record all events that happened in each session, and they can be described by a number of properties. The common properties are the start time, end time, location, record of activity and system status.

5.4.3 Assignment Relations

The following are possible pairs of entity assignments in the access control mechanism:

- *Tenant-Role Assignment (TnRA):* In the tenant-role assignment, many tenants can be dynamically assigned to roles based on their attributes.
- *Subject-Role Assignment (SRA):* In the subject-role assignment, many subjects can be dynamically assigned to roles based on their attributes.
- *Task-Role Assignment (TRA):* In the task-role assignment, many tasks can be assigned to roles.
- *Role-Permission Assignment (RPA):* With the role-permission assignment policy, a many permission-role assignment relations can be assigned. Role permission assignment depends on the role, virtual resource, action, permission and constraint.
- *Task-Permission Assignment (TPA):* In task-permission assignment, a many permission to task assignment relations can be assigned. Task permission assignment depends on the current task, virtual resource, action, permission and constraint. Access rights are determined for fulfilling assigned tasks.

In this framework, permissions possessed by roles and permissions required by tasks are described separately and the assignment of tasks to roles is thus constrained; same roles may

not have the same tasks depending on the context and only users assigned to specific tasks are allowed to carry out that task. For instance if two subjects have the same role based on the RPA, but one of the subject is not defined in the TRA, the subject will not be allowed to execute such task since there is no TPA assignment relation.

5.4.4 Used Annotations

Formally, we define sets TnA, SA, VRA, EA, R, T, A, and P as a set of tenant attributes, subjects attributes, virtual resource attributes, environment attributes, roles, tasks, actions, and permissions, respectively. We define the sets of relations as follows:

- $|TnRA| \subseteq |TnA| \times |R|$ ---------- (1).

  The definition (1) represents the tenant-role assignment relations mapping the tenant attributes to the roles they are member of. Here, roles will be dynamically assigned to tenants based on their attributes using specified policies.

- $|SRA| \subseteq |SA| \times |R|$ ---------- (2).

  The definition (2) represents the subject-role assignment relations mapping the subject attributes to the roles they are member of. Here, roles will be dynamically assigned to subjects based on their attributes using specified policies.

- $|TRA| \subseteq |T| \times |R|$ ---------- (3).

  The definition (3) represents the task-role assignment relations mapping tasks to roles they are assigned to.

- $|RPA| \subseteq |R| \times |P|$ ---------- (4).

  The definition (4) represents the permission-role assignment relations mapping roles to permissions they are authorized to. $|P| = 2^{(VRA \times A)}$, where permission sets (P) can

be obtained from the Cartesian product of virtual resources (VRA) and the actions (A).

- $|TPA| \subseteq |T| \times |P|$ ---------- (5).

  Definition (5) represents the task permission assignment relations mapping tasks to permissions. It defines the set of permissions required to execute tasks. $|P| = 2^{(VRA\ X\ A)}$, where permission sets (P) can be obtained from the Cartesian product of virtual resources (VRA) and the actions (A).

- *Role Hierarchy:* $|RH| \subseteq |R| \times |R|$ ---------- (6).

  Definition (6) defines inheritance relationship among roles. For instance, if r_1 and $r_2 \in R$ and role hierarchy (RH) denotes that r_1 is superior to r_2. As a result, r_1 automatically inherits the permissions of r_2.

- *Session (SS):* denoted as $SS \rightarrow 2^{|R|}$ ---------- (7).

  Definition (7) represents the mapping function from session set (SS) to the power set of roles.

- *Attribute Distribution*: Access control decisions are made from policies based on the attributes of entities. Suppose TnA, SA, VRA and EA are the set of tenant attributes, subject attributes, virtual resource attributes and environment attributes respectively, which can be identified by the cloud services, then their attributes are defined in (8), (9), (10) and (11) as follows:

  - $|TnA| = \{tna_1 .. tna_w\} \mid w \geq 1$ ---------- (8).
  - $|SA| = \{sa_1 .. sa_x\} \mid x \geq 1$ ---------- (9).
  - $|VRA| = \{vra_1 .. vra_y\} \mid y \geq 1$ ---------- (10).
  - $|EA| = \{ea_1 .. ea_z\} \mid z \geq 1$ ---------- (11).

Where w, x, y and z are integers to represent the maximum number of attributes for each entity.

5.4.5 Policy Specification

In this model, five policy types can be defined:

- *Tenant-Role Assignment Policy:* A policy that contains rules to assign roles to a tenant.
- *Subject-Role Assignment Policy:* A policy that contains rules to assign roles to a subject.
- *Task-role Assignment Policy:* Policies that contain rules to assign tasks to roles
- *Role Permission Policy:* A policy that contains all the permissions associated with a role.
- *Task Permission Policy:* A policy that contains all the permissions associated with a task.

Each role in the system is represented by one role assignment policy and one permission policy. Likewise, each task in the system is represented by one task-role assignment policy and one task permissions policy.

5.4.5.1 Tenant / Subject-Role Assignment Policy Specification

Roles are assigned to tenants or subjects dynamically based on the policies defined, making manual assignment unnecessary. Whenever a user authenticates to the cloud service, their attributes are retrieved using appropriate formats and exchange mechanisms as described in Section 4.2The application site can then use the attributes to assign roles to the

tenant or subject. The process of deriving an access control decision relies on the role assignment algorithms. The algorithms will take the tenant attributes, subject attributes and constraints as inputs, to produce a matching role. The role assignment policy for tenants can be described using a tuple of five elements (tenant attributes (TnA), virtual resource attributes (VRA), action (A), permission (P), constraint (C)), and a tuple of 6 elements for subject-role assignment policy – (tenant attributes (TnA), subject attributes (SA), virtual resource attributes (VRA), action (A), permission (P), constraint (C)). In these policies, role is the virtual resource being protected. For instance, the tuple for tenant-role assignment policy can be read as: A tenant with attributes (e.g., organization name) can be assigned a particular role considering the constraints. The system can be modeled with the following elements:

Tenant-Role Assignment Policy = {TnA, VRA, A, P, C}.

i. *TnA={organization, individual}.*
ii. *VRA = {administrator | technical director | engineering technician | production manager |media producer}.*
iii. *A = {enable, disable}.*
iv. *P={enable role, disable role}.*
v. *C={time}.*

Below are some rules:

r1. All users from National Broadcasting Corporation (NBC) with email address ending with admin.com are assigned to the administrator role.

r2. Only five users are allowed to be in the administrator role at any point in time.

The rule r1 (TnA = “NBC”, SA = “admin.com”, VRA = “administrator role”, A= “enable”, P = “permit”) can be written in XACML as shown below using the UMU-XACML policy editor (UMU, 2009). In this policy, role is the resource that is being protected.

Table 5.2: Subject-Role Assignment <PolicySet> for 'Administrator Role'

```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <PolicySet xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
PolicyCombiningAlgId="" PolicySetId="" xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os cs-xacml-schema-
policy-01.xsd">
3. <Target />
4. <Policy PolicyId="PPS:administrator:role" RuleCombiningAlgId="">
5.   <Target />
6.   <Rule Effect="Permit" RuleId="urn:oasis:name:tc:xacml:2.0:demo:ruleid:1">
7.   <Description>All users from National Broadcasting Corporation (NBC) with email address ending with admin.com are assigned
the administrator role.</Description>
8.     <Target>
9.     <Subjects>
12.        <Subject>
13.       <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
14.          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">NBC</AttributeValue>
13.          <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
DataType="http://www.w3.org/2001/XMLSchema#string" Issuer="" MustBePresent="true" />
14.        </SubjectMatch>
15.       </Subject>
16.       <Subject>
17.       <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
18.          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">admin.com</AttributeValue>
19.          <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
DataType="http://www.w3.org/2001/XMLSchema#string" Issuer="NBC" MustBePresent="true" />
20.        </SubjectMatch>
21.       </Subject>
22.      </Subjects>
23.      <Resources>
24.       <Resource>
25.        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
26.         <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">administrator</AttributeValue>
27.         <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
DataType="http://www.w3.org/2001/XMLSchema#string" Issuer="NBC" />
28.        </ResourceMatch>
29.      </Resource>
30.      </Resources>
31.      <Actions>
32.       <Action>
33.        <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
34.          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">enable</AttributeValue>
35.          <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string" />
36.        </ActionMatch>
37.       </Action>
38.      </Actions>
39.    </Target>
40.  </Rule>
41. </Policy>
42. </PolicySet>
```

Line 1 to 5 define the name of the policy set, policy instance and target to which the subject, resource, and action of this policy applies.

Line 6 defines a rule in this example and its effect is "Permit," if this rule is evaluates to "True".

Lines 7 through 42 define the decision requests to which the tenant, subject, resource, and action this rule applies. Lines 9 through 22 describe the rule that applies the decision requests to the subject. Lines 23 through 30 describe the rule that applies the decision requests to a specific resource value, which must match the <ResourceMatch> element. The <ResourceMatch> element specifies a matching function in the MatchId attribute, a string value "&xml;string" and a pointer to a specific "resource attribute" in the request context, by means of the <ResourceAttributeDesignator> element. The match function is used to compare the string value with value of the "resource attribute" in the request context. This rule applies to decision request, only if the match returns "True". Lines 31 through 38 define the action match in a similar format to a resource match. It defines the match of "action attribute" in the request context to string value "enable".

Line 39 closes the Target element.

Line 40 closes the Rule element.

Line 41 closes the Policy element.

Line 42 closes the Policy Set element.

5.4.5.2 Policy Specification for Task-Role Assignment

The process of deriving an access control decision also relies on the task-role assignment algorithms. It contains the tuple (Role, Task, Action, Constraint, Permission). The algorithms will take the role and constraints as inputs, to produce matching tasks. The example below contains task-role assignment <PolicySet> for a task associated with the doctor's role. To write the policy in XACML, the role is the subject attribute and task is the virtual resource being protected. See an example in Table 5.3.

Table 5.3: Task-Role Assignment <PolicySet> for Administrator Role

```
<?xml version="1.0" encoding="UTF-8"?>
<PolicySet xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
PolicyCombiningAlgId="" PolicySetId="" xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os cs-xacml-
schema-policy-01.xsd">
  <Target />
  <Policy PolicyId="PPS:administrator:task" RuleCombiningAlgId="">
    <Target />
    <Rule Effect="Permit" RuleId="urn:oasis:name:tc:xacml:2.0:demo:ruleid:2">
      <Description>Users with the role "administrator" can schedule appointments</Description>
      <Target>
        <Subjects>
          <Subject>
            <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
              <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">administrator role</AttributeValue>
              <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:subject:role"
DataType="http://www.w3.org/2001/XMLSchema#string" Issuer="" MustBePresent="true" />
            </SubjectMatch>
          </Subject>
        </Subjects>
        <Resources>
          <Resource>
            <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
              <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">schedule appointment</AttributeValue>
              <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
DataType="http://www.w3.org/2001/XMLSchema#string" Issuer="" />
            </ResourceMatch>
          </Resource>
        </Resources>
        <Actions>
          <Action>
            <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
              <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">assign task</AttributeValue>
              <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string" />
            </ActionMatch>
          </Action>
        </Actions>
      </Target>
    </Rule>
  </Policy>
</PolicySet>
```

5.4.5.3 Policy Specification for Role Permission Policy

In our proposed model, role permission specifies whether the subject has access on the virtual resource. It defines access control policies using the tuple (Role, Virtual Resource, Action, Permission, Constraint). This can be illustrated using the rules specified below:

r1. An administrator can access NBC.dropbox.com space only between the hours (6am – 8pm).

Role Permission Policy = {R, VRA, A, P, C}

i. *R = {Administrator}*

ii. *VRA = {NBC.dropbox.com}*

iii. *A = {access}*

iv. *P = {grant access}*

v. *C={6am – 8pm}*

To specify the policy using XACML, the subject attribute is the role.

Table 5.4: Role Permission <PolicySet> for Administrator Role

```
<?xml version="1.0" encoding="UTF-8"?>
<PolicySet xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" PolicyCombiningAlgId="" PolicySetId="" xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os cs-
xacml-schema-policy-01.xsd">
 <Target />
 <Policy PolicyId="RPS:administrator:role:" RuleCombiningAlgId="">
  <Target />
  <Rule Effect="Permit" RuleId="urn:oasis:name:tc:xacml:2.0:demo:ruleid:3">
   <Description>An administrator can access the NBC.dropbox.com space between 8am to 6pm.</Description>
   <Target>
    <Subjects>
     <Subject>
      <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">administrator role</AttributeValue>
       <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:subject:role "
DataType="http://www.w3.org/2001/XMLSchema#string" Issuer="" MustBePresent="true" />
      </SubjectMatch>
     </Subject>
    </Subjects>
    <Resources>
     <Resource>
      <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">NBC.dropbox.com</AttributeValue>
       <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
DataType="http://www.w3.org/2001/XMLSchema#string" Issuer="" />
      </ResourceMatch>
     </Resource>
    </Resources>
    <Actions>
     <Action>
      <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">grant access</AttributeValue>
       <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string" />
      </ActionMatch>
     </Action>
    </Actions>
   </Target>
 <Condition>
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
     <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-greater-than-or-equal">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
       <EnvironmentAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"
```

```
DataType="http://www.w3.org/2001/XMLSchema#time" />
          </Apply>
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">8am</AttributeValue>
        </Apply>
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-less-than-or-equal">
          <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
            <EnvironmentAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"
DataType="http://www.w3.org/2001/XMLSchema#time" />
          </Apply>
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">6pm</AttributeValue>
        </Apply>
      </Apply>
    </Condition>
  </Rule>
  <PolicyIdReference>PPS:administrator:role</PolicyIdReference>
 </Policy>
</PolicySet>
```

5.4.5.4 Policy Specification for Task Permission Policy Set

The task permission <PolicySet> contains the permissions associated with the tasks for a role. Permission depends on the task, virtual resource, action, permission, and constraint elements. The subject attributes will be the task. For example, below is a policy that allows scheduling of appointments.

Table 5.5: Task Permission <PolicySet> for Administrator Role

```
<?xml version="1.0" encoding="UTF-8"?>
<PolicySet xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" PolicyCombiningAlgId="" PolicySetId="" xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os
cs-xacml-schema-policy-01.xsd">
 <Target />
 <Policy PolicyId="" RuleCombiningAlgId="">
  <Target />
  <Rule Effect="Permit" RuleId="urn:oasis:name:tc:xacml:2.0:demo:ruleid:1">
   <Description>Permission to schedule an appointment.</Description>
   <Target>
    <Subjects>
     <Subject>
      <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Schedule
appointment/AttributeValue>
        <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:subject:target-namespace "
DataType="http://www.w3.org/2001/XMLSchema#string" Issuer="" MustBePresent="true" />
      </SubjectMatch>
     </Subject>
    <Resources>
     <Resource>
      <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">schedule sheet</AttributeValue>
       <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
```

```
DataType="http://www.w3.org/2001/XMLSchema#string" Issuer="" />
          </ResourceMatch>
        </Resource>
      </Resources>
      <Actions>
        <Action>
          <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">write</AttributeValue>
            <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string" />
          </ActionMatch>
        </Action>
      </Actions>
    </Target>
  </Rule>
  <PolicyIdReference>PPS:administrator:role</PolicyIdReference>
 </Policy>
</PolicySet>
```

5.4.6 Constraints

Constraints are important in access control systems. The following are constraints that can be enacted using the proposed model:

- *Least privilege:* This requires that each subject be granted the most restrictive set of privilege needed to perform authorized tasks, e.g., occurs when a subject is able to activate more than one rule. For a doctor to diagnose a patient, he will need the permission to read the patient's data, lab results and chart, and not modify them. Providing access to these resources represent the least access.
- *Delegation:* This involves the transfer of access temporarily to another. It allows a tenant /subject to assign task to another. Delegation can happen in supervision, workflows and approval classes, e.g., Sam, a physician, will like his trainee physician to apply medical treatments in emergencies.
- *Spatial and Temporal access:* This can be used to include location and time constraints over access rights, e.g., Night nurses of XYZ hospital are only allowed read access patient's data between 8pm and 6am.

- *Separation of duty (SOD):* Assigning different users with the responsibilities of recording, maintaining and authorizing transactions, e.g., a user can be assigned to authorize the work of another and users cannot perform mutually exclusive tasks. SOD can be static (i.e., defined from a matrix of users) or dynamic (based on user attributes) for a particular process chain, e.g., trainee physicians are not authorized to apply any medical treatment unless a practitioner physician approves.
- *Cardinality:* This is an attribute that specifies how many times a task can either be assigned or assumed, e.g., a subject from a particular tenant should not be allowed to make more than 10 access requests.

5.5 Concluding Remark

This chapter provides a set of considerations that are important for a dynamic, fine-grained and active access control system for the cloud environment. In this chapter, a review of existing access control models used for cloud security was carried out in order to determine their applicability to the cloud environment. Also, we discussed policy types that can be specified using our access control model and, some specific ways in which XACML can be used to specify these policies in the cloud environment. The next chapter presents the implementation of the framework using a case study.

Chapter 6: Implementation

This chapter provides an implementation of the proposed access control model to express constraints on resources in the cloud. First, policies are carefully specified because correct implementation and enforcement of the policies are based on the premise that the policy specifications are correct. The implementation then takes the access requests and policies as input to determine if access to a resource is permitted or denied. The response to this evaluation is the output. Therefore, both the policy specifications and implementations are subjected to verification to ensure that they truly capture the security requirements. To implement the policies in our framework, our approach adopts the Answer Set Programming (ASP) introduced in Section 4.9to formulate XACML. With ASP, we model a Policy Decision Point (PDP) that loads the policies and evaluates access requests against these policies. This allows us to leverage the features of ASP solvers in performing various logical reasoning to verify constraint such as separation of duty.

In the next section is a case study implementation to verify the access control framework.

6.1 A Case Study

This section introduces an illustrative example for a Claims Management System based in the cloud. It is the case of an enterprise ABC, with approximately 10 insurance companies operating directly in 25 different countries that moved its insurance claims application to the cloud in response to increase in claim events and decrease in underwriting profits, to allow for flexibility of use on a pay-per-use basis. The enterprise also uses other cloud-based business applications (e.g., saleforce.com, SharePoint) and Gmail for web based email management. The data captured by the claims application is to be used by the employees of the insurance companies, clients, brokers, call centers and contractors. In this case, since the

application is no longer hosted within the premises of ABC, sensitive data needs to be shielded from unauthorized users, considering the challenges that may stem from the use of multiple cloud service providers. Using our proposed access control model, appropriate compliance required for data access in the cloud applications can be realized by providing an access control point to interface the cloud services. To implement our proposed access control framework, we illustrate our encoding on the auto claim process within the enterprise involving the following elements:

- Tenants, TnA = {*Call Centre Services (CCS), Zyco Auto Company, ABC*},
- Subjects, SA = {*email add, role, clearance*}
- Roles, R = {*Call Center Agent, Underwriter, Auto Contractor, Appraiser, Claims Manager*},
- Tasks involved in the claim are, T = {*check insurance coverage, repair auto, process claim*},
- Virtual resources used, VRA = {*Demographic data, Auto data, Financial data*},
- Actions that can be carried out, A= {*read, write*}.
- Permissions (P) are obtained from the Cartesian product of virtual resources and actions (P = VR X A).

From this case study, some of the rules specified is as shown in the table below:

Table 6.1: Rules Data Table

#	Tenants (TnA)	Subjects (SA)	Roles (R)	Tasks (T)	Virtual Res. (VRA)	Actions (A)	Permissions (P)	Constraints (C)
1.	Call Centre Services (CCS)	Any	Call Center Agent	Check insurance coverage	Demographic, Auto claims & Financial data	Read	Read Demographic data Read Auto claims data Read Financial data	Cannot be assigned an Underwriter role

#	Tenants (Tn)	Subjects (S)	Roles (R)	Tasks (T)	Virtual Res. (VR)	Actions (A)	Permissions (P)	Constraints (C)
2.	ABC	Underwriter role, Reliable Clearance	Underwriter	Check insurance coverage	Demographic & Financial data	Read, write	Read Demographic data Write Demographic data Read Financial data	Cannot be assigned a Claims manager role
3.	Zyco Auto Company	Any	Auto Contractor	Repair auto	Auto claims data	Read	Read Auto claims data	Contractors within Canada only
4.	ABC	Email add ending with ABCapp.com	Appraiser	Check insurance coverage	Demographic data	Read	Read Demographic data	Not more than 10 records daily
5.	ABC	Claims Mgr. Role, Secret Clearance	Claims Manager	Process claim	Financial data	Read, write	Read Financial data Write Financial data	Between 6am and 8pm

For example, some role permission rules and policies that can be extracted from the table above are as follows:

R1: Any subject with the role *Call Center Agent* can read a client's *Demographic data*.

R2: Any subject with the role *Appraiser* can read a client's *Demographic data,* but cannot have access a client's *Demographic data* more than 5 times daily.

P1: Any subject with the roles *Call Center Agent*, *Underwriter* and *Appraiser* can read the *Demographic data*, but an *Appraiser* cannot access the data more than 5 times daily.

R3: Any subject with the role *Underwriter* can read or write to a client's *Demographic data.*

R4: An *Underwriter* cannot change a client's *Demographic data* on weekends.

P2: An underwriter can read or write a client's demographic data but cannot write to the demographic data on weekends.

6.2 Measurement Criteria

To evaluate the framework, we investigate that the framework can detect various types of inconsistencies by checking for the:

- *Correctness of policy specification:* To assure that correct specification of the access control policy. We test different cases of the policies by introducing faults.
- *Correctness of implementation:* This verifies the policy properties in order to assure that the implementation of the access control policy does not authorize subjects' access requests that are not permitted by the rules of the access control policy being implemented. Properties such as inconsistency, incompleteness, conflicting and unreachability properties are checked. For instance, we check whether:
    - A subject cannot have an active role that is not authorized for the user.
    - A subject can only perform an action on a virtual resource if there exists a role that is included in the subject's active role set and there exist a permission that is assigned to the role such that the permission authorizes the performance of the action on the resource.
    - A subject can only perform a task only if there exists a task that is included in the subject's active task-role set and there exist a permission that is assigned to the task such that the permission authorizes the performance of the action on the task.

- Check to detect whether any specified combinations of rules does not grant access not allowed by the access control policy.

- *Policy Coverage:* Check whether each individual policy element is involved when a request is evaluated.
- *Integrity:* Check constraints defined are fulfilled, e.g., in separation of duty, a subject that is a member of group A is not allowed to access resources for members of group B if a constraint is defined.

6.3 Implementation Strategy

Policies in our framework are specified using XACML, (a logic based language), which has become the de facto standard for specifying and enforcing access control policies. XACML describes three classes of objects: rules, policies and policy sets. The applicability of an access request to a target is realized from the aggregation of rules, policies and policy sets after matching the values of the subject, resource and action attributes in the target with the values of the attributes in the request. If they match then the policy is considered relevant to the request and is evaluated.

Managing access control policies are often error prone, especially when multiple access control policies are involved. To evaluate the applicability of a policy on an access request, we determined the results of the evaluation of the rules, policies and then policy sets. XACML rules are the most basic component of a policy. They have a goal effect, which is either Permit or Deny. They have a domain of applicability, and conditions under which they can yield Indeterminate. The domain of applicability (target) is specified as a series of

predicates about the environmental data that must all be satisfied for the rule to yield its goal effect.

Since multiple rules or policies may apply to an access request, rule or policy combining algorithms are used to determine the response to an access request. Standard combining algorithms are defined for Deny-overrides, Permit-overrides, First-applicable and Only-one-applicable. For instance, these can be represented by four symbols ($\oplus$, Θ, $\otimes$, $\varnothing$) for combining rules or policies.

- *Permit-overrides:* $p \oplus q$ always yields permit if either p or q yield permit.
- *Deny-overrides:* $p \Theta q$ always yields deny if either p or q yield deny
- *First-applicable:* $p \varnothing q$ yields p's answer unless that answer is not applicable, in which case it yields q's answer.
- *Only-one-applicable:* $p \otimes q$ requires that one of p or q yield not applicable and then yields the other half's answer.

If no policy or policy set applies, then the result is "NotApplicable", but if more than one policy or policy set is applicable, then the result is "Indeterminate". When exactly one policy or policy set is applicable, the result of the combining algorithm is the result of evaluating the single applicable policy or policy set. Therefore, the result of combining rules or policies yields Permit (p), Deny (d), NotApplicable (n) or Indeterminate (i).

A conflict exists between rules or policies if one or more is applicable to a request. The probability of conflict increases if a virtual resource has multiple owners or multiple policies associated with it. Basically, in XACML, conflicting decisions never occur since all policies are combined with a particular combining algorithm that returns only one decision. For instance, a policy *P1* may say that subject *U1* of a tenant *Tn1* can perform an action *A1* on a

resource *R1* any time. Another policy *P2* may say that any subject cannot perform an action *A1* on a resource *R1* between 12am-6am. Since these policies conflict, a decision is required to know whether it is a conflict or not. If this is not a conflict then either *P1* is allowed to supersede *P2* or *P2* is allowed to supersede *P1,* and if this is a conflict, changes will have to be made to the policies.

The strategy taken to implement policies that can be specified in our framework is as follow:

i. *Define a set of XACML policies:*

From the rules defined in Table 6.1 above, they state whether an access can be granted to the resources or not using the different ways for specifying the policy types in our framework as described in Section 5.4.5.

ii. Apply mutation to each policy to introduce faults:

After specifying the policies, to obtain an experimental basis for our model, a fault model for the security policy is defined by adapting mutation analysis on the policies. XACMUT (Antonia Bertolino, Lonetti, & Marchetti, 2012) as described in Section 4.8is used to generate mutants of the XACML policy using mutation operators. The set of mutation operators emulates the introduction of both syntactic (i.e., modify policy elements) and semantic (i.e., change logical constructs) faults. The evaluation criterion being considered here is the correctness of policy specification and implementation of access control policies. Policy tests are conducted by supplying typical test inputs/cases (requests) and subsequently checking test outputs (responses) against expected ones to reveal security flaws or inconsistencies between the policy and the implementation.

The program under test is iteratively mutated to produce numerous mutants, each containing one fault. A test input is independently executed on the original program and each

mutant program. If the output of a test executed on a mutant differs from the output of the same test executed on the original program, then the fault is detected and the mutant policy is "killed" by the request; otherwise, the mutant policy is not killed. In policy mutation testing, the program under test, test inputs, and test outputs correspond to the policy, requests, and responses, respectively. For instance, from the case study in Section 6.1if the actual response to the rule #3 yields a permit when the target of the rule and the access request matches, then the expected request response to the faulty policy should yield deny if a mutant operator is applied.

The fundamental premise of mutation testing is that if the policy contains a fault, there will usually be a set of mutants that can only be killed by a test case that also detects that fault. In other words, the capability to detect small, minor faults such as mutants implies the capability to detect complex faults. The higher the percentage of killed mutants, the more effective the test suite is at detecting faults. This helps to determine the correctness of policy specification, and that each element is involved during evaluation.

iii. *Transform XACML policies into ASP logic programs:*

XACML policies are transformed into ASP formulas for policy analysis by introducing the syntax of logic programs and answer set semantics. The ASP technique can be used to solve combinatorial problems efficiently to detect access control policies incompleteness property, conflicting property and unreachability property.

We provide an ASP representation example in Table 6.2 that turns an XACML policy into formulas under the stable model semantics. By using the DLV (an ASP solver) and SPARC, several XACML policy analysis services such as policy verification, comparison, and inconsistency checking can be automated. DLV is one of the well-developed solvers for ASP programs. To use an ASP solver, the following were installed:

1. Java Runtime Environment (JRE). The system was tested on Java version 1.7.0_67.
2. The SPARC to ASP translator. SPARC offers explicit constructs to specify objects, relations, and their sorts.
3. An ASP solver (DLV- a deductive database system, based on disjunctive logic programming).

Table 6.2: ASP Representation Example using DLV + SPARC

```
sorts
#tenant_attributes={call_center_services, zyco_auto_company, ABC}.
#subject_attributes={call_center_agent, underwriter,auto_contractor, appraiser, claims_manager}.
#resource_attributes={demographic_data, auto_data, financial_data}.
#action_attributes={read, write}.
#effect={permit,deny,indet}.
#ruleid=[r][1..4].
#policyid=[p][1..2].
#cond={#days, #number_of_access}.
#days=[day][1.7].
#number_of_access=[1..5].

predicates
target(#ruleid,#tenant_attributes,#subject_attributes,#resource_attributes,#action_attributes).
decision(#ruleid,#effect).
decision_from(#policyid,#ruleid,#effect).
decision_p(#policyid,#effect).
tenant(#ruleid,#tenant_attributes).
subject(#ruleid,#subject_attributes).
resource(#ruleid,#resource_attributes).
action(#ruleid,#action_attributes).
effect(#ruleid,#effect).
cond(#cond).

rules
target(r1, call_center_agent, demographic_data, read).
effect(r1,permit).
target(r2, appraiser, demographic_data, read).
effect(r2,permit).
cond(r2, 1 >= C <= 5).

target(r3, underwriter, demographic_data, read | write).
effect(r3,permit).
target(r4, underwriter, demographic_data, write).
effect(r4,deny).
cond(d6 >= C <= d7).

decision(R,S,RE,A,C,E) :- target(R,S,RE,A), effect(R,E), cond(R,C).
-decision(R,E) :- not decision(R,E).

%p1
decision_from(p1,r1,E) :- decision(r1,E).
-decision_from(p1,r1,E) :- not decision_from(p1,r1,E).
```

```
decision_from(p1,r2,E) :- decision(r2,E).
-decision_from(p1,r2,E) :- not decision_from(p1,r2,E).

:~ decision_from(p1,R,permit).
decision_p(p1,E) :- decision_from(p1,R,permit), decision_from(p1,R,E).
-decision_p(p1,E) :- not decision_p(p1,E).

:~ decision_from(p1,R,deny).
decision_p(p1,E) :- decision_from(p1,R,deny), decision_from(p1,R,E).
-decision_p(p1,E) :- not decision_p(p1,E).

%p2
decision_from(p1,r3,E) :- decision(r3,E).
-decision_from(p1,r3,E) :- not decision_from(p1,r3,E).

decision_from(p2,r4,E) :- decision(r4,E).
-decision_from(p2,r4,E) :- not decision_from(p2,r4,E).

:~ decision_from(p2,R,permit).
decision_p(p2,E) :- decision_from(p2,R,permit), decision_from(p2,R,E).
-decision_p(p2,E) :- not decision_p(p2,E).

:~ decision_from(p2,R,deny).
decision_p(p2,E) :- decision_from(p2,R,deny), decision_from(p2,R,E).
-decision_p(p2,E) :- not decision_p(p2,E).
```

iv. Policy Analysis:

Here, we execute a set of test cases on the policy and its mutants. The problem of verifying a security property *P* against an XACML description can be cast into the problem of checking whether the program $II \cup \{\neg P\} \cup II_{\text{config}}$ has no answer sets, where *II* is the program corresponding to the XACML description and II_{config} is the following program that generates arbitrary configurations. If no answer set is found, then this implies that the property is verified. Otherwise an answer set returned by the answer set solver serves as a counterexample that indicates why the description does not entail *P*. This helps to find the flaws in the description. For example, consider the verification of role permission policy set p_1. We want to verify that an Appraiser cannot read the demographic data of a client more than 5 times daily. The property can be represented as follows.

? – decision(appraiser, demographic_data, read, -C >=5, permit).

Given the corresponding ASP program of p1, the negation of the property and *II*config, the

ASP solver returns no answer set, from which we can conclude that the property is verified. If an answer set was found, it reflects a flaw of the policy.

6.4 Results

We conduct experiments on 5 XACML policies and our empirical results show that our framework effectively identifies security leakage. Table 6.3 summarizes the characteristics of each policy. Columns 1-11 show the policy name, number of (policies, rules, condition, permit rules, deny rules, mutants generated, test suite), percentage of mutants killed, number of correct implementation, and policy coverage, respectively.

Table 6.3: Policies Used and Results from Analyzing Each Policy

Policy	#Policy	#Rules	#Cond	#Permit	#Deny	#Mutants	#Test suite	%Mutant killed	Correctness of Implementation	Policy coverage
Demo1	2	4	2	1	1	33	10	100%	100%	100%
Demo2	4	9	5	3	1	20	10	100%	100%	100%
Demo3	3	7	2	1	2	25	10	100%	100%	100%
Demo4	2	5	4	2	0	37	10	100%	100%	100%
Demo5	3	6	2	2	1	28	10	100%	100%	100%

Mutant Operators:
CPC (Change Policy Combining Algorithm, CRC (Change Rule Combining Algorithm), PTT (Policy Target True), PTF (Policy Target False), RTT (Rule Target True), RTF (Rule Target False), ANR (Add New Rule), CRE (Change Rule Effect).

The policies describe both the permit and deny policies. This helps in checking if the policies permit or deny as specified. Whenever changes are made to the security property being checked, it is expected that the response to the policies should change from the response to the original policies. About 10 test suites were used for each policy and it mutants to detect inconsistencies in policy specification and implementation. The high

percentage of mutants killed, policy covered, and implemented shows the capability of the framework in detecting security leakage. This indicates that if the access control model is implemented, it can prevent access from unauthorized users.

We compared our model with the other models identified from Table 5.1, and found that our model is able to address all the characteristics identified for a secure cloud access control. See Table 6.4.

Table 6.4: Our Proposed Model in Comparison with Existing Models

CHARACTERISTICS	RBAC	X-GTRBAC	UCON	MT-RBAC	MTAS	S-RBAC	RB-MTAC	TRBAC	Our Model
Decentralized	0	0	0	3	0	0	0	0	3
Fine-grained	1	3	3	1	1	1	1	3	3
Multi-tenancy	0	0	0	3	3	3	3	0	3
Trust/Credential Federation	0	0	3	3	0	0	0	0	3
Constraint Specification	1	2	3	3	2	2	2	2	3
Collaboration	0	0	0	3	0	0	0	3	3
Policy Specification & Enforcement	2	2	2	3	2	2	2	3	3
Active (workflows)	0	0	0	0	0	0	0	3	3

6.5 Related Work

A number of researches are being carried out to develop access control frameworks for cloud services. Recently, Calero et al., (2010) presented a multi-tenancy authorization system (MTAS), which extends the well-known role-based access control model by building trust among collaborating tenants. Singh & Singh (2013) proposed a new architecture of RBAC using ontology to keep a backup of data being sent to the cloud server but does not give an

implementation of the model. A model to ensure a two-stage security at the API level was proposed by Sirisha & Kumari (2010). However, the model only ensures that only registered users from white listed domains can access the cloud service. Li, Liu, Wei, Liu, Liu & Liu (2010) proposed RBAC for SaaS systems by enhancing the S-RBAC model but the method proposed in the paper has some imperfections that need more in-depth research. For example, permissions always cannot be assigned to a user permanently, so the access control model must provide the time constraints mechanism to achieve temporary assignment.

Task and role based access control (TRBAC) was considered a viable model for cloud environment and was found to dynamically validate access permissions for users based on the assigned roles and tasks performed by users with an assigned role. However, Ma, Wu, Zhang & Li, (2012) noted that TRBAC is centralized and does not consider local and global access control integration and their communication in a distributed environment. Also, with TRBAC, it is hard to make classification of tasks and, it could not deal with some business rules such as delegation and closing account, which are common and important to support efficient execution of business activities. Extended generalized temporal role based access control (X-GTRBAC) model, an enhanced hybrid version of the X-RBAC and GTRBAC models, was also found suitable. X-GTRBAC relies on the certification provided by trusted third parties (e.g., public key infrastructure (PKI) Certification Authority) to assign roles to users and also considers the context (e.g., time, location) to directly affect the level of trust associated with a user (as part of user profile) to make access decisions. (Meghanathan, 2013). In order to achieve multi-tenant access control, Calero et al., (2010) proposed a multi-tenancy authorization system (MTAS) by extending the RBAC model with a coarse-grained trust relation among collaborating tenants. Multitenant collaborations are enabled in MTAS by bridging two tenants with a cross-tenant trust relation. In addition, Tang & Sandhu (2013)

proposed a family of multi-tenancy role based access control (MT-RBAC) models by extending RBAC model with the components of tenants and issuers to address multi-tenant authorization for collaborative cloud services. MT-RBAC aims to enable fine-grained cross-tenant resource access by building tenant-level granularity of trust relations. Yang et al., (2013) used identity management and the RBAC model to design a role-based multi-tenancy access control (RB-MTAC) with consideration of multi-tenant and multi-user cloud environment. In RB-MTAC, each user has its own access control list (ACL) and when the user logs in, the system will determine the access privileges according to the ACL of the user.

Danwei et al., (2009) developed an access control architecture based on the usage control (UCON) authorization model. This model manages concepts such as subject, object, right, obligation, condition, and attribute. The main contribution of this proposal is the inclusion of a negotiation model in the authorization architecture to enhance the flexibility of access control for cloud services. Then, when the access requested doesn't match access rules, it provides users with a second access choice through negotiation in certain circumstances, rather than refusing access directly. This work is still in a conceptual stage, so its usability in a cloud environment has yet to be demonstrated.

Some recent work examines XACML-based access control model with SAML for security assertion for a Cloud Computing framework (Khan, A. R., 2012). XACML is the policy language used due to its expressivity and flexibility for specifying access control policies. XACML is an XML-based, well-established standard to define and enforce policies. The main components of the XACML architecture include components as described in the standard Internet Engineering Task Force architecture which includes a Policy Enforcement

Point (PEP), a Policy Decision Point (PDP), a Policy Information Point (PIP), a Policy Administration Point (PAP), and an obligations service. While XACML provides the means to express and enforce a policy, it does not specify how to request and retrieve the required attributes i.e., specify a protocol for communication between the PEP and PDP. Security Assertion Markup Language (SAML) is a highly suitable candidate for this protocol (Markus Lorch, Seth Proctor & Rebekah Lepro, 2003).

SAML is an XML-based standard for exchanging authentication, authorization and attribute data between security domains that have established trust relations. SAML provides assertion and protocol elements that may be used for retrieval of attributes for use in an XACML Request Context. For querying an on-line Attribute Authority for attributes, and for holding the response to that query, SAML defines Attribute Query and Attribute Response elements. SAML major application areas include single sign on, attribute-based authorization and web services security. The means by which lower-level communication or messaging protocols (such as HTTP or SOAP) are used to transport SAML assertion or protocol messages is defined by the SAML bindings (Demchenko, Yuri, 2009). Manoj & Sekaran, (2013) proposed an architecture for the distributed access control (DAC) in the Cloud Computing paradigm, taking into account the access control requirements of the cloud service providers and consumers. Also, a workflow model for the proposed access control architecture was given. However, there is no discussion on how policies can be specified or authorization achieved.

In this thesis, we present a scalable, decentralized and fine-grained access control based on the characteristics of RBAC and ABAC and TBAC. Our framework allows distributed access control of data stored in the cloud so that only authorized users with valid attributes

can access them.

6.6 Concluding Remark

In our implementation approach, we use mutation analysis in order to improve the security tests. Policy tests are used because they are capable of detecting security faults in the policy specifications. To analyze the model, we utilized the ASP solver, a formal verification tool to demonstrate analysis and test case generation for the formal specifications of the model.

Conflicts that may result as multiple rules overlap i.e. an access request may match several rules were addresses by the combining algorithm used in XACML, allowing us to verify the policy security properties and detect policy violations.

Chapter 7: Conclusion and Future Work

This chapter reviews the contributions of this thesis and discusses the issues that need to be addressed in the future work. Today's enterprises are very attracted to Cloud Computing due to its cost saving factor. However, security is a hindering factor affecting its adoption. Security has become a core ingredient of nearly most modern software and information systems. The adoption of consistent, well-defined security policies for access control can be shown to support business innovation, cost effectiveness and competitiveness. In order to effectively address security aspects for cloud systems, more convenient and mature access control mechanisms need be designed. We hereby contribute to this field by developing an access control framework for Cloud Computing security.

7.1 Contributions

We present an attribute-driven task-role based access control framework for the cloud. We focus on using XACML for specifying policies in our framework due to its expressiveness, and demonstrated how our methodology could be applied to an enterprise using cloud services. The major contributions of this thesis are as follows:

- Designed an attribute-driven role and task based access control model that allows for the evaluation and enforcement of policies at run-time while being flexible, scalable and decentralized.
- Compared existing access control models for the cloud to determine their applicability and identified the requirements for designing an access control solution specific for cloud scenario.

7.2 Future Work

In future, approaches to address the heterogeneity among the policies from different cloud domains is another area that can be explored since extensive collaborations exist among services provided by different clouds, which might have different security mechanisms. In addition, since trust relations is used to achieve authorization of unknown users, further research is required to determine the feasibility of trust models. Therefore, an administration model and enhancements for the trust model will be developed.

References

Aken, V. (2005). Management Research as a Design Science: Articulating the Research Products of Mode 2 Knowledge Production in Management. Br. Journal Management, 16 (1), pp. 19-36.

Antonia Bertolino, S. D., Lonetti, F., & Marchetti, E. (2012). "XACMUT; XACML 2.0 Mutants Generator." Software Testing, Verification and Validation, IEEE 2013 6th International Conference on. Luxemborg. ISBN 978-1-4799-1324-4. Pg 28-33.

Arakawa, J., & Sasada, K. (2011). A Decentralized Access Control Mechanism using Authorization Certificate for Distributed File Systems. 6th International Conference on Internet Technology and Secured Transactions, 11-14 December, pp. 148-153. Abu Dhabi, United Arab Emirates.

Asma, A., Chaurasia, M. A., & Mokhtar, H. (2012). Cloud Computing Security Issues. International Journal of Application or Innovation in Engineering & Management, 1 (2), pp. 141-147.

Atayero, A. A., & Feyisetan, O. (2011). Security Issues in Cloud Computing. Journal of Emerging Trends in Computing and Information Sciences, 2 (10), ISSN 2079-8407, pp. 546-552.

Benson et al. (1999). Realizing the Potential of C41: Fundamental Challenges. In National Academies Press, ISBN 0-309-06485-6, pp.1-298.

Bishop, M. (2002). Computer Security. Addidon-Wesley. ISBN 0-201-44099-7, pp. 1-1136.

Burmester, M. (2012). T-ABAC: An Attribute-Based Access Control Model for Real-Time Availability in Highly Dynamic Systems. Florida State University, Department of Computer Science, Florida, pp. 1-6.

Calero, J. M., Edwards, N., Kirschnick, J., Wilcock, L., & Wray, M. (2010). Toward a Multitenancy Authorization System for Cloud Services. IEEE Computer and Reliability Society, pp. 48-51.

Celesti, A., Tusa, F., Villari, M., & Puliafito, A. (2013). Security and Cloud Computing: Intercloud Identity Management Infrastructure. University of Messima, Dept. of Mathematics, Faculty of Engineering, Italy, pp. 1-3.

Chow, Richard (2009). Controlling Data in the Cloud: Outsourcing Computation without Outsourcing Control. CCSw'09 ACM, pp. 86-89.

Cloud Security Alliance Information. (2013). Information Security Policy. Retrieved June 23, 2013, from Cloud Security Alliance Information: www.cloudsecurityalliance.com/information-security-policy

Coi, Juri Luca De & Olmedilla, Daniel (2005). A Review of Trust Management Security and Privacy Policy Languages. L3S Research Center, University of Hannover, Hannover, Germany, pp. 1-8.

CSA. (2011). Critical Areas to focus in Cloud Computing V. 3.0. Cloud Security Alliance, pp. 1-177.

Damiani, M., Bertino, E., Catania, B., & Perlasca, P. (2007). A Spatially Aware RBAC. ACM Transactions on Information and Systems Security, 10 (1). 2.

Damianou, N., & et.al. (2002). Tools for Domain-based Policy Management of Distributed Systems. In Proceedings of the IEEE/IFIP NOMS 2002: Network Operations and Management Symposium, pp. 203-217. Florence, Italy.

Damianou, N., Dulay, N., Lupu, E., & Sloman, M. (2001). The Ponder Policy Specification

Language. Proceedings of Policy 2001: Workshop on Policies for Distributed Systems and Networks. 29-31, pp. 18-39. Bristol, UK: Springer-Verlag LNCS 1995. 122.
Danwei, C., Xiuli, H., & Xunyi, R. (2009). Access Control of Cloud Service Based on UCON. Nanjing University of Posts & Telecommunications, pp.559-564.
Davy, S., & Barrett, K. (2005). Policy-Based Architecture to Enable Autonomic Communications - A Position Paper, vol. 2, issue 4.
Demchenko, Y. (2009). Security Languages for Access Control and Authorisation: SAML and XACML Languages Overview. Technical Report, pp. 1-44.
Dijk, M. V., & Juels, A. (2010). On the Impossibility of Cryptography Alone for Privacy-Preserving Cloud Computing. In Proceedings of the 5th USENIX conference on Hot Topics in Security (HotSec'10). USENIX Association, Berkeley, CA, USA, pp. 1-8.
Diver, Sorcha (2007). Information Security Policy - A Development Guide for Large and Small Companies. SANS Institute, pp. 1-43.
Doelitzscher, F., Reich, C., & Sulistio, A. (2010). Designing Cloud Services Adhering to Government Privacy Laws. Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on. Bradford, pp. 930-935.
ENISA. (2009). Cloud Computing: Benefits, Risks, and Recommendations for Information Security. European Network and Information Security Agency. Report, pp.1-125.
Fang Hao, e. a. (2010). Secure Cloud Computing with a Virtualized Network Infrastructure. IEEE Symposium on Security and Privacy, vol 7, pp. 1-7.
Ferraiolo, D., & Kuhn, D. (1992). Role Based Access Control. In Proceedings of 15th National Computer Security Conference, Baltimore, MD, pp. 554-563.
Fujitsu. (2010). Personal Data in the Cloud: A Global Survey of Consumer Attitudes. Technical Report, Fujitsu Research Institute, pp 1-13.
Garg, Anshula, & Mishra, Pradeeep. (2012). Methodologies for Access Control and their Interactions. International Journal of Engineering Research and Applications (IJERA), 2 (5), pp. 342-345.
Gartner (2008). Assessing Security Risks of Cloud Computing. Gartner, pp. 1-6.
Google (2014, April 16). Trends. Retrieved April 16, 2014, from Google: www.google.ca/trends
Gouglidis, A., & Mavridis, I. (2009). On the Definition of Access Control Requirements for Grid and Cloud Computing Systems. Networks for grid applications, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 25. ISBN 978-3-642-11732-9. Springer Berlin Heidelberg, pp. 19.
Grundy, M. A., & Muller, I. (2010). An Analysis of the Cloud Computing Security Problem. APSEC 2010 Cloud Workshop. Sydney, Australia, pp. 1-6.
Henver, Alan, March, Salvatore T., Park, Jinsoo, & Ram, Sudha (2004). Design Science in Information System Research. MIS Quarterly, 28 (1), pp. 75-105.
Herath, T., & Rao, H. (2009). Encouraging Information Security Behaviours in Organizations: Role of Penalties, Pressures and Percieved effectiveness. Decision Support System, 47, pp. 154-165.
IDC. (2009). Cloud Computing 2010: An IDC Update. IDC Executive Telebriefing, IDC, pp. 1-23.
Jude, M. (2001). Policy-based Management: Beyond the Hype. Business Communciations Review, pp. 52,54,56.
Juniper Networks. (2013). An Integrated security Solution for the Virtual Datacenter and

Cloud. White paper, Juniper Networks Inc, pg. 1-12.
Kagal, Lalana, Finin, Tim, & Hendler, Jim (2005). Policy Management for the web. 14th International Word Wide Web Conference, pp. 62. Chiba, Japan. 123
Kailash et al, P. (2012). Integrating the Trusted Computing Platform into the Security of Cloud Computing. International Journal of Advanced Research in Computer Science and Software Engineering, 2 (2).
Karadsheh, L. (2012). Applying Security Policies and Service Level Agreement to IaaS Service Model to Enhance Security and Transition. Computers & Security – SciVerse ScienceDirect , pp. 315-326.
Karn, Bernice (2011). Data Security - The Case against Cloud Computing. Cassel Brock Lawyers, pp. 1-23.
Khan, Abdul Raouf (2012). Access Control in Cloud Computing Environment. ARPN Journal of Engineering and Applied Sciences, 7 (5), pp. 613-615.
Khoo, B., Harris, P., & Hartman, S. (2010). Information Security Governance of Enterprise Information Systems: An Approach to Legislative Compliant. International Journal of Management & Information Systems, 14 (3), pp. 49-56.
Kumar, V., & Kumar, D. S. (2013). Access Control Framework for Social Networking Systems Based on Present Access Control Policies. International Journal of Engineering Research & Technology (IJERT), 2 (5), pp. 917-921.
Li, D., Liu, C., Wei, Q., Liu, Z., & Liu, B. (2010). RBAC-based Access Control for SaaS Systems. Northern Eastern University, Software College. IEEE, pp. 1-4.
Li, X.-Y., Shi, Y., Yu-Guo, & Ma, W. (2010). Multi-tenancy Based Access Control in Cloud. Beijing Jiatong University, School of Computer and Information, pp. 1-4.
Lifschitz, M. G. (1988). The Stable Model of Semantics for Logic Programming. Proceedings of International Logic Programming Conference and Symposium, pp. 1070-1080. MIT Press.
Lifschitz, V. (2008). What is Answer Set Programming? In Proceedings of the AAAI Conference on Artificial Intelligence, pp. 1594-1597. MIT Press.
Liu, S.-C., & Wu, T.-H. (2010). Enterprise Information Infrastructure Constructed by Integral Planning. International Journal of Digital Society, 1 (3).
Ma, G., Wu, K., Zhang, T., & Li, W. (2012). A Flexible Policy-Based Access Control Model for Workflow. Electrical Review, North China Electric Power University, pp. 67-71.
Markus Lorch, Seth Proctor, Rebekah Lepro (2003). "First Experiences Using XACML for Access Control in Distributed Systems." ACM workshop on XML security, Fairfax, VA, USA. Pg. 25-37
Martin, Jean-Christophe (1999). Policy-Based Networks. Sun BluePrints OnLine, pp. 1-17.
Meghanathan, N. (2013). Review of Access Control Models for Cloud Computing. Jackson. David C. Wyld (Eds): ICCSEA, SPPR, CSIA, WimoA - 2013. CS & IT-CSCP 2013, pp. 77-85.
Mel, & Grance (2011). The NIST Definition of Cloud Computing, NIST Special Publication 500-299, National Institute of Standards and Technology, Technology Administration U.S. Department of Commerce, pp. 1-7.
Meyers, M. (2002). CISSP Certification Passport. Mcgraw Hill Companies, ISBN 0072225785, pp. 1-422.
Michelberger, P., & Labodi, C. (2012). After Information Security – Before a Paradigm Change (A Complex Enterprise Security Model). 9 (4), pp. 101-116.
Min, Y.-G., Shin, H.-J., & Bang, Y.-H. (2012). Cloud Computing Security and Access

Control Solutions. Journal of Security Engineering, pp. 135-141.
Morgan Stanley. (2011). Cloud Computing takes off. Blue Paper, Morgan Stanley Research Global. Alpha Wise, pp. 1-104.
Morsy, M. A., Grundy, J., & Muller, I. (2010). An Analysis of the Cloud Computing Security Problem. APSEC 20120 Cloud Workshop, (pp. 1-6). Sydney, Australia. 124
Narayanan, H. A., & Giine, M. (2011). Ensuring Access Control in Cloud provisioned Healthcare Systems. In 2011 IEEE Proceedings on Consumer Communications and Networking Conference (CCNC), pp. 247-251. Las Vegas, NV.
NIST. (2006). Assessment of Access Control Systems. Interagency report 7316, National Institute of Standards and Technology, Technology Administration U.S. Department of Commerce, pp. 1-51.
NIST. (2011). Cloud Computing Reference Architecture. Specification, National Institute of Standards and Technology, pp. 1-35. NIST. (2013a). Guide to Attribute Based Access Control (ABAC) definition and Draft Consideration (Draft). NIST Special Publication 800-162, National Institute of Standards and Technology, pp. 1-43.
NIST. (2013b). NIST Cloud Computing Security Reference Architecture. NIST Special Publication 500-299, National Institute of Standards and Technology, U.S. Department of Commerce, pp. 1-204.
OASIS. (2013). eXtensible Access Control Markup Language (XACML) Version 3.0. OASIS, pp. 1-154.
OASIS. (2005). SAML 2.0 profile of XACML v2.0. Specification, OASIS standard, pp. 1-21. Location: http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-saml-profilespec-os.pdf.
OASIS. (2004). XACML Profile of RBAC. Specification, pp. 1-21. Location: http://docs.oasis-open.org/xacml/cd-xacml-rbac-profile-01.pdf
Owen, Morne (2009). An Enterprise Information Security Model for a Micro Finance Company: A Case Study. Nelson Mandela Metropolitan University, pp.1-128.
Paladi, N., Gehrmann, C., & Morenius, F. (2013). State of the Art and Hot Aspects in Cloud Data Storage Security. SICS Technical Report T2013:01, Swedish Institute of Computer Science and Ericsson Research, pp. 1-24.
Patel, S. C., Umrao, L. S., & Singh, R. S. (2012). Policy Based Framework for Access Control in Cloud Computing. International Conference on Recent Trends in Engineering & Technology (ICRTET2012), pp. 142-146.
Pearson, S. (2012). Privacy, Security and Trust for Cloud Computing. Technical Report. HP labs. HPL-2012-80R1, pp. 1-57.
Perlin et al, A. (2010). An Entity-centric Approach for Privacy and Identity Management in Cloud Computing. Reliable Distributed Systems. 29th IEEE Symposium on. New Delhi, pp. 177-183.
Reddy, V. K., & Reddy, L. (2011). Security Architecture of Cloud Computing. International Journal of Engineering, Science and Technology (IJEST), 3, pp. 7150.
Reeja, S. (2012). Role Based Access Control Mechanism in Cloud Computing using Cooperative Secondary Authorization Recycling Method. International Journal of Emerging Technology and Advanced Engineering, 2 (10).
Ribeiro, C., Zuquete, A., Fereira, P., & PauloGuedes. (2000). SPL: An Access Control Language for Security Policies with Complex Constraint. IST/INESC, Portugal. 6.
Saleh et al, M. (2011). A New Comprehensive Framework for Information Security Risk Management. Applied Computing and Informatics, 9, pp. 107-118.

Samarati, P., & Vimercati, S. D. (2001). Access Control: Policies, Models and Mechanisms. Universit'a di Milano, Dipartimentodi Tecnologie dell'Informazione, Crema, Italy, pp. 137-195. 125

Shahram et al, G. (2012). Information Security Management on Performance of InformationSystems Management. Journal of Basic and Applied Scientific Research, 2 (3), ISBN 2090-4304, pp. 2582-2588.

Sharma, D. (2011). Enterprise Information security Management Framework (EISMF). Thesis (S.M in Engineering and Management), Massachusetts Institute of Technology, pp. 124-130.

Siewe, F. (2005). A Compositional Framework for the Development of Secure Access Control Systems. PHD Thesis, De Monfort University, Software Technology Laboratory, Faculty of Computing Sciences and Engineering, England, pp. 1-225.

Singh, P., & Singh, S. (2013). A New Advance Efficient RBAC to Enhance the Security in Cloud Computing. India, pp. 1-7.

Sirisha, A., & Kumari, G. (2010). API Access Control in Cloud using the RBAC Model. Trendz in Information Sciences and Computing (TISC), 2010 IEEE Conference on. Chennai. ISBN 978-1-4244-9007-3. Pg. 135-137.

Subashini, S., & Kavitha, V. (2011). A survey on Security Issues in Service Delivery Models of Cloud Computing. Journal of Network and Computer Applications , 34 (1), pp. 1-11.

Takeda, H., Veerkamp, P., & Yoshikawa, H. (1990). Modeling Design Process. Al Magazine, 11 (4), pp. 37.

Tang, B., & Sandhu, R. (2013). A Multi-tenant RBAC Model for Collaborative Cloud Services. Eleventh Annual Conference on Privacy, Security and Trust. IEEE, pp. 229-238.

Thomas, R., & Sandhu, R. (1997). Task-based Authorization Controls (TBAC): A Family of Models for Active and Enterprise-oriented Authorization Management. Proceedings of the IFIP WG11.3 Workshop on Database Security. Aug 11-13, pp. 1-16. California: Chapman & Hall.

UMU (2009). UMU-XACML Editor. (U. O. Murcia, Producer) Retrieved Feb 28, 2014, from UMU-XACML Editor: umu-xacmleditor.sourceforge.net

Veiga, A. D., & Eloff, J. (2006). An Information Security Governance Framework. Inf. Sys. Management. 24, 4 (October 2007), pp. 361-372.

Waller et al, (2011). Policy Based Management for Security in Cloud Computing. STA 2011 Workshops: IWCS 2011 and STAVE 2011, Loutraki, Greece, June 28-30, 2011. Proceedings, pp. 130-137.

Waller, Adrian (2004). Policy Based Network Management. Thales Research & Technology, UK, pp. 1-19.

Whittman, M., & Mattford, H. (2009). Principles of Information Security (3rd Edition ed.). (C. Technology, Ed.), ISBN 978-1-4239-0177-8, pp. 1-550.

Wood et al., (2009). The Case for Enterprise-ready Virtual Private Clouds., pp. 1-5.

Yang, S.-J., Lai, P.-C., & Lin, J. (2013). Design Role-Based Multi-tenancy Access Control Scheme for Cloud Services. International Symposium on Biometrics and Security Technologies, IEEE, ISBN 978-0-7695-5010-7, pp. 273-279.

Zargar, S. T., Takabi, H., & Joshi, J. B. (2011, September 20-21). DCDIDP: A Distributed, Collaborative, and Data-driven Intrusion Detection and Prevention Framework for Cloud Computing Environments. Collaborative Computing, Networking,

Applications and Worksharing (CollaborateCom), 2011 International Conference on, IEEE, ISBN 978-1-46-0683-6, pp. 332-341. 126
Zhao, Hang (2012). Security Policy Definition and Enforcement in Distributed Systems. PhD Thesis, Columbia University, Graduate School of Arts & Sciences, pp. 1-151.
Zhao, W., & Gao, F. (2012). Design of Dynamic Fine-grained Role-based Access Control Strategy. Proceedings of IEEE CCIS2012, pp. 275-278.
Zhu, J., & Wen, Q. (2012). SaaS Access Control Research Based on UCON. Fourth International Conference on Digital Home, pp. 331-334. IEEE Computer Society.
Zissis, D., & Lekkas, D. (2012). Addressing Cloud Computing Security Issues. Future Generation Computers (28), pp. 583-592.

Appendix: Access Control Policy Schema

Access-control-xacml-2.0-policy-schema-os.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:oasis:names:tc:xacml:1.0:policy"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xacml="urn:oasis:names:tc:xacml:1.0:policy"
elementFormDefault="qualified" attributeFormDefault="unqualified">
	<!-- -->
	<xs:element name="PolicySet" type="xacml:PolicySetType"/>
	<xs:complexType name="PolicySetType">
	  <xs:sequence>
		<xs:element ref="xacml:Description" minOccurs="0"/>
		<xs:element ref="xacml:PolicySetDefaults" minOccurs="0"/>
		<xs:element ref="xacml:Target"/>
		<xs:choice minOccurs="0" maxOccurs="unbounded">
			<xs:element ref="xacml:PolicySet"/>
			<xs:element ref="xacml:Policy"/>
			<xs:element ref="xacml:PolicySetIdReference"/>
			<xs:element ref="xacml:PolicyIdReference"/>
		</xs:choice>
		<xs:element ref="xacml:Obligations" minOccurs="0"/>
	  </xs:sequence>
	  <xs:attribute name="PolicySetId" type="xs:anyURI" use="required"/>
	  <xs:attribute name="PolicyCombiningAlgId" type="xs:anyURI" use="required"/>
	</xs:complexType>
	<!-- -->
	<xs:element name="PolicySetIdReference" type="xs:anyURI"/>
	<xs:element name="PolicyIdReference" type="xs:anyURI"/>
	<!-- -->
	<xs:element name="PolicySetDefaults" type="xacml:DefaultsType"/>
	<xs:element name="PolicyDefaults" type="xacml:DefaultsType"/>
	<xs:complexType name="DefaultsType">
	  <xs:sequence>
		<xs:choice>
			<xs:element ref="xacml:XPathVersion"/>
		</xs:choice>
	  </xs:sequence>
	</xs:complexType>
	<!-- -->
	<xs:element name="XPathVersion" type="xs:anyURI"/>
	<!-- -->
	<xs:element name="Policy" type="xacml:PolicyType"/>
	<xs:complexType name="PolicyType">
	  <xs:sequence>
		<xs:element ref="xacml:Description" minOccurs="0"/>
		<xs:element ref="xacml:PolicyDefaults" minOccurs="0"/>
		<xs:element ref="xacml:Target"/>
		<xs:element ref="xacml:Rule" minOccurs="0" maxOccurs="unbounded"/>
		<xs:element ref="xacml:Obligations" minOccurs="0"/>
	  </xs:sequence>
	  <xs:attribute name="PolicyId" type="xs:anyURI" use="required"/>
	  <xs:attribute name="RuleCombiningAlgId" type="xs:anyURI" use="required"/>
	</xs:complexType>
```

```
<!-- -->
<xs:element name="Description" type="xs:string"/>
<!-- -->
<xs:element name="Rule" type="xacml:RuleType"/>
<xs:complexType name="RuleType">
  <xs:sequence>
            <xs:element ref="xacml:Description" minOccurs="0"/>
            <xs:element ref="xacml:Target" minOccurs="0"/>
            <xs:element ref="xacml:Condition" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="RuleId" type="xs:anyURI" use="required"/>
  <xs:attribute name="Effect" type="xacml:EffectType" use="required"/>
</xs:complexType>
<!-- -->
<xs:simpleType name="EffectType">
  <xs:restriction base="xs:string">
            <xs:enumeration value="Permit"/>
            <xs:enumeration value="Deny"/>
  </xs:restriction>
</xs:simpleType>
<!-- -->
<xs:element name="Target" type="xacml:TargetType"/>
<xs:complexType name="TargetType">
  <xs:sequence>
            <xs:element ref="xacml:Subjects"/>
            <xs:element ref="xacml:Resources"/>
            <xs:element ref="xacml:Actions"/>
  </xs:sequence>
</xs:complexType>
<!-- -->
<xs:element name="Subjects" type="xacml:SubjectsType"/>
<xs:complexType name="SubjectsType">
  <xs:choice>
            <xs:element ref="xacml:Subject" maxOccurs="unbounded"/>
            <xs:element ref="xacml:AnySubject"/>
  </xs:choice>
</xs:complexType>
<!-- -->
<xs:element name="Subject" type="xacml:SubjectType"/>
<xs:complexType name="SubjectType">
  <xs:sequence>
            <xs:element ref="xacml:SubjectMatch" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<!-- -->
<xs:element name="AnySubject"/>
<!-- -->
<xs:element name="Resources" type="xacml:ResourcesType"/>
<xs:complexType name="ResourcesType">
  <xs:choice>
            <xs:element ref="xacml:Resource" maxOccurs="unbounded"/>
            <xs:element ref="xacml:AnyResource"/>
  </xs:choice>
</xs:complexType>
<!-- -->
<xs:element name="AnyResource"/>
```

```
<!-- -->
<xs:element name="Resource" type="xacml:ResourceType"/>
<xs:complexType name="ResourceType">
  <xs:sequence>
            <xs:element ref="xacml:ResourceMatch" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<!-- -->
<xs:element name="Actions" type="xacml:ActionsType"/>
<xs:complexType name="ActionsType">
  <xs:choice>
            <xs:element ref="xacml:Action" maxOccurs="unbounded"/>
            <xs:element ref="xacml:AnyAction"/>
  </xs:choice>
</xs:complexType>
<!-- -->
<xs:element name="AnyAction"/>
<!-- -->
<xs:element name="Action" type="xacml:ActionType"/>
<xs:complexType name="ActionType">
  <xs:sequence>
            <xs:element ref="xacml:ActionMatch" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<!-- -->
<xs:element name="SubjectMatch" type="xacml:SubjectMatchType"/>
<xs:complexType name="SubjectMatchType">
  <xs:sequence>
            <xs:element ref="xacml:AttributeValue"/>
            <xs:choice>
                      <xs:element ref="xacml:SubjectAttributeDesignator"/>
                      <xs:element ref="xacml:AttributeSelector"/>
            </xs:choice>
  </xs:sequence>
  <xs:attribute name="MatchId" type="xs:anyURI" use="required"/>
</xs:complexType>
<!-- -->
<xs:element name="ResourceMatch" type="xacml:ResourceMatchType"/>
<xs:complexType name="ResourceMatchType">
  <xs:sequence>
            <xs:element ref="xacml:AttributeValue"/>
            <xs:choice>
                      <xs:element ref="xacml:ResourceAttributeDesignator"/>
                      <xs:element ref="xacml:AttributeSelector"/>
            </xs:choice>
  </xs:sequence>
  <xs:attribute name="MatchId" type="xs:anyURI" use="required"/>
</xs:complexType>
<!-- -->
<xs:element name="ActionMatch" type="xacml:ActionMatchType"/>
<xs:complexType name="ActionMatchType">
  <xs:sequence>
            <xs:element ref="xacml:AttributeValue"/>
            <xs:choice>
                      <xs:element ref="xacml:ActionAttributeDesignator"/>
                      <xs:element ref="xacml:AttributeSelector"/>
```

```
                </xs:choice>
            </xs:sequence>
            <xs:attribute name="MatchId" type="xs:anyURI" use="required"/>
        </xs:complexType>
        <!-- -->
        <xs:element name="AttributeSelector" type="xacml:AttributeSelectorType"/>
        <xs:complexType name="AttributeSelectorType">
            <xs:attribute name="RequestContextPath" type="xs:string" use="required"/>
            <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
            <xs:attribute name="MustBePresent" type="xs:boolean" use="optional" default="false"/>
        </xs:complexType>
        <!-- -->
        <xs:element name="ResourceAttributeDesignator" type="xacml:AttributeDesignatorType"/>
        <xs:element name="ActionAttributeDesignator" type="xacml:AttributeDesignatorType"/>
        <xs:element name="EnvironmentAttributeDesignator" type="xacml:AttributeDesignatorType"/>
        <!-- -->
        <xs:complexType name="AttributeDesignatorType">
            <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
            <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
            <xs:attribute name="Issuer" type="xs:string" use="optional"/>
            <xs:attribute name="MustBePresent" type="xs:boolean" use="optional" default="false"/>
        </xs:complexType>
        <!-- -->
        <xs:element name="SubjectAttributeDesignator" type="xacml:SubjectAttributeDesignatorType"/>
        <xs:complexType name="SubjectAttributeDesignatorType">
            <xs:complexContent>
                <xs:extension base="xacml:AttributeDesignatorType">
                    <xs:attribute name="SubjectCategory" type="xs:anyURI" use="optional"
default="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"/>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <!-- -->
        <xs:element name="AttributeValue" type="xacml:AttributeValueType"/>
        <xs:complexType name="AttributeValueType" mixed="true">
            <xs:sequence>
                <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
            <xs:anyAttribute namespace="##any" processContents="lax"/>
        </xs:complexType>
        <!-- -->
        <xs:element name="Function" type="xacml:FunctionType"/>
        <xs:complexType name="FunctionType">
            <xs:attribute name="FunctionId" type="xs:anyURI" use="required"/>
        </xs:complexType>
        <!-- -->
        <xs:element name="Apply" type="xacml:ApplyType"/>
        <xs:element name="Condition" type="xacml:ApplyType"/>
        <!-- -->
        <xs:complexType name="ApplyType">
            <xs:choice minOccurs="0" maxOccurs="unbounded">
                <xs:element ref="xacml:Apply"/>
                <xs:element ref="xacml:Function"/>
                <xs:element ref="xacml:AttributeValue"/>
```

```
                <xs:element ref="xacml:SubjectAttributeDesignator"/>
                <xs:element ref="xacml:ResourceAttributeDesignator"/>
                <xs:element ref="xacml:ActionAttributeDesignator"/>
                <xs:element ref="xacml:EnvironmentAttributeDesignator"/>
                <xs:element ref="xacml:AttributeSelector"/>
            </xs:choice>
            <xs:attribute name="FunctionId" type="xs:anyURI" use="required"/>
            <!-- Legal types for the first and subsequent operands are defined in the accompanying table -->
        </xs:complexType>
        <!-- -->
        <xs:element name="Obligations" type="xacml:ObligationsType"/>
        <xs:complexType name="ObligationsType">
            <xs:sequence>
                <xs:element ref="xacml:Obligation" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
        <!-- -->
        <xs:element name="Obligation" type="xacml:ObligationType"/>
        <xs:complexType name="ObligationType">
            <xs:sequence>
                <xs:element ref="xacml:AttributeAssignment" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="ObligationId" type="xs:anyURI" use="required"/>
            <xs:attribute name="FulfillOn" type="xacml:EffectType" use="required"/>
        </xs:complexType>
        <!-- -->
        <xs:element name="AttributeAssignment" type="xacml:AttributeAssignmentType"/>
        <xs:complexType name="AttributeAssignmentType" mixed="true">
            <xs:complexContent mixed="true">
                <xs:extension base="xacml:AttributeValueType">
                    <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <!-- -->
</xs:schema>
```