Deckblatt der Dissertation

Vorderseite

Multi-Layered Policy Generation and Management in Clouds

Dissertation
zur Erlangung des mathematisch-naturwissenschaftlichen Doktorgrades
"Doctor rerum naturalium"
der Georg-August-Universität Göttingen

im Promotionsprogramm PCS
der Georg-August University School of Science (GAUSS)

vorgelegt von
Faraz Fatemi Moghaddam
aus Tehran

Göttingen, 2017

Georg August University, Göttingen,
Institute of Computer Science
Goldschmidtstrasse 7,
37077 Götingen,
Germany.

Tel:	+49 (551) 39 – 172000
Fax:	+49 (551) 39 – 14403
Email:	office@informatik.uni-goettingen.de
Web:	www.informatik.uni-goettingen.de

Comitte Members:
Prof. Dr, Ramin Yahyapour
Prof. Dr. Dieter Hogrefe

Dedication

To my beloved wife, Dr. Pardis Najafi,
for her hidden strength, endless support and constant love…

To my sweet baby boy, Ryan

To my selfless mother, Fatima

and to the soul and bright memory of my late father,
Dr. Enayat Fatemi Moghaddam

Acknowledgment.

I would like to thank my supervisor, *Prof. Dr. Ramin Yahyapour*, for the patient guidance, encouragement and advice he has provided throughout my time as his student. I have been extremely lucky to have a supervisor who cared so much about my work, and who responded to my questions and queries so promptly

My sincere gratitude is reserved for *Dr. Philipp Wieder* for his invaluable insights and suggestions during the project. I appreciated his guidance, support and willingness to take time to discuss my research.

Very special thanks to *GWDG* and *CleanSky EU Project* for giving me the opportunity to carry out my doctoral research and for their financial support.

I would also like to take this opportunity to thank *Prof. Dr. Dieter Hogrefe* and *Ms. Martina Brücher* for their very helpful supports and suggestions.

Special thanks got to: *Prof. Dr. Jens Grabowski, Prof. Dr. Xiaoming Fu*, *Prof. Dr. Delphine Reinhardt, Prof. Dr. Carsten Damm*, *Prof. Dr. Bernd Stock*, *Dr. David Koll*, *Dr. Sven Bingert*, *Dr. Song Yang*, *Dr. Oliver Wannenwetsch*, *Dr. Vanessa End*, *Fei Zhang, Alessio Silvestro, Dr. Sachin Sharma, Dr. Fabian Schneider, Nitinder Mohan, Amir Reza Fazely, Dr. Edwin Yaqub,* and *Hossein Salahi*.

This research has been supported by CleanSky ITN Project (Grant No. 607584) funded by the Marie-Curie-Actions within the 7th Framework Program of the European Union (EU FP7).

Abstract

The long awaited Cloud computing concept is a reality now due to the transformation of computer generations. However, security challenges are most important obstacles for the advancement of this emerging technology. A well-established policy framework is defined in this thesis to generate security policies which are compliant to requirements and capabilities. Moreover, a federated policy management schema is introduced based on the policy definition framework and multi-level policy application to create and manage virtual clusters with identical or common security levels. The proposed model consists in the design of a well-established ontology according to security mechanisms, a procedure which classifies nodes with common policies into virtual clusters, a policy engine to enhance the process of mapping requests to specific node as well as associated cluster and matchmaker engine to eliminate inessential mapping processes. The suggested model has been evaluated according to performance and security parameters to prove the efficiency and reliability of this multi-layered engine in cloud computing environments during policy definition, application and mapping procedures.

Keywords: Cloud Computing; Security; Security Management; Policy Management; Access Control; Policy Mapping; Privacy; Ontology.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

ABGS	Attribute-Based Group Signature
ABS	Attribute-Based Signature
CAP	Cloud Access Policy
CC	Cloud Customer
CP	Cloud Provider
CSDS	Cloud Security Distributary Set
CSON	Cloud Security Ontology
CSRT	Cloud Security Rooted Tree
CU	Cloud User
EAP	Extensible Authentication Protocol
ECC	Elliptic Curve Cryptosystem
FSP	Federated Security Policies
HABE	Hierarchical Attribute-Based Encryption
KP-ABE	Key Policy Attribute-Based Encryption
LaaS	Law-as-a-Service
MLO	Multi-Level Ontology
OWL	Web Ontology Language
PANA	Protocol for Carrying Authentication for Network Access
PD	Policy Database
PE	Policy Engine
PEaaS	Policy Engine-as-a-Service
PLC	Policy Layer Constructor
QoS	Quality of Service
RAE	Ring Analysis Engine

SLA	Service Level Agreement
SLC	Security Level Certificate
SSO	Single Sign-On
TAP	Temporary Access Policy
UAP	User Access Policy
VC	Virtual Cluster
VM	Virtual Machine
W3C	World Wide Web Consortium
WS-Policy	Web Service Policy Framework
XACML	Extensible Access Control Markup Language

About the Author

Faraz Fatemi Moghaddam is working as a scientist at Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen (GWDG), Georg- August-Universität, and also research fellow in EU FP7 CleanSky ITN Project. He was graduated in 2009 from Azad University of Tehran with a diploma in Software Engineering and also post graduated in 2013 from Staffordshire University.

His research interest lies in the areas of security and privacy challenges in cloud-based environments. He has also worked in NeroCloud research group and HAWK university of applied science as a researcher and lecturer and published several papers regarding to security issues in clouds. At GWDG, he currently holds ESR researcher position of CleanSky project. More details about his experiences and publications can be found at: https://www.linkedin.com/in/farazfatemimoghaddam/.

Chapter 1

1. INTRODUCTION

1.1.Introduction

Cloud Computing is an emerging open standard model, which can enable ubiquitous computing built around core concepts such as virtualization, processing power, distribution and elastic scaling to provide a shared pool of configurable computing resources [1]. The most often claimed advantages of cloud include offering on-demand IT resources, improved availability, dynamic resource provisioning and cost reduction. Whilst at first glance the value proposition of cloud-based services to carry out deployment and managing large scale data services is strong [2], there are many challenges that need to be overcome to make clouds an ideal platform for scalable analytics [3].

In fact, there are critical obstacles such as security and confidentiality, availability, transfer bottlenecks, performance unpredictability, reputation fate sharing and quick scaling [4] for the advancement of cloud computing as a widely accepted technology (Fig 1.1).

One of the most challenging issues in virtualized data centers is to provide appropriate levels of security and protection for resources according to the sensitivity and requirements of cloud customers [5][6][7]. Indeed, each individual customer needs to be granted reliable security level(s) based on defined details of Service Level Agreement (SLA).

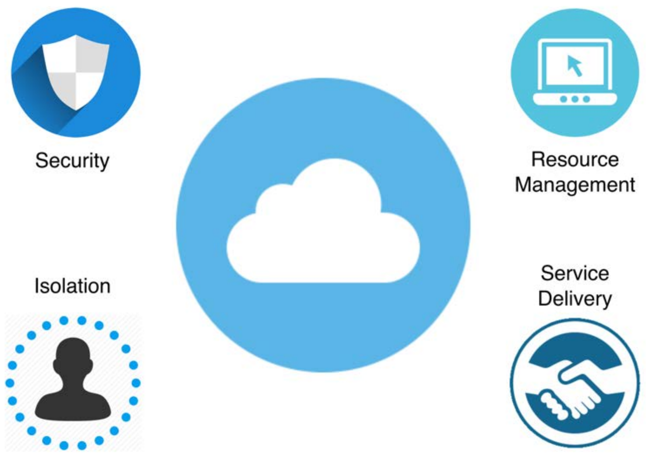


Fig. 1.1. Main Challenges in Cloud Computing Environments

By an efficient and reliable policy management model, the protection of virtualized nodes is ensured as well as decrease of processing time for manipulating sensitive and also non-sensitive nodes. These policies need to follow a structural framework in all levels of definition, generation, application and management.

Typically, cloud providers confront two main challenges in policy-based multi-level models: The separation of services from high-level security constrains about the access and usage of resources, and the scalability of policy management and policy mapping processes according to the isolation and availability concepts in clouds-based environments [8]. The latter, which is the focus of this work, refers to the capability that is provided in cloud-based data centers to provision security policies for each virtualized node according to the requirements of customers and to classify these nodes based on defined policies where the scalability of this policy-based framework is ensured in all levels.

1.2. Research Questions

To have a better understating about the objectives of a policy-based cloud computing three main entities are defined:

- *Cloud Provider (CP)*: A service provider that offers cloud-based resources and services.
- *Cloud Customer (CC)*: An organization or a company that uses cloud services for employments or subscribers (*e.g.* universities, hospitals, *etc.*).
- *Cloud User (CU)*: Defined end-users (*i.e.* subscribers) that use cloud-based services offered by *Cloud Customer* according to the internal contracts.

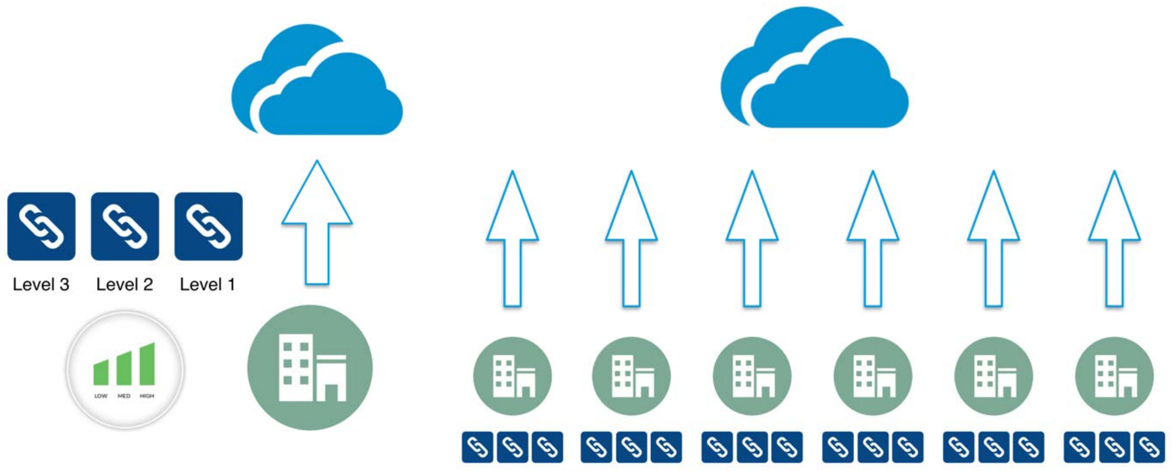


Fig. 1.2a Fig. 1.2.b

Fig. 1.2. Multi-Level Policy Management in Clouds

Each CC needs several security levels based on sensitivity of resources and specific requirements to distribute the resources to CUs. The security levels are defined according to security capabilities of CP (Fig 1.2a). In fact, CPs provide security resources as on-demand services to be applied to data stored in cloud storages (*i.e.* Security-as-a-Service). With increasing the number of these specific security levels by different cloud customers (Fig 1.2b), a challenging issue is risen. How to manage access requests by CUs according to defined policies by CCs? Accordingly, there main challenges should be considered (Fig 1.3):

- How to create different security rings (*i.e.* levels) for CCs (*e.g.* University) based on the requirements, sensitivity of data and the capabilities of CPs?
- How to manage data and resources to apply policies according to defined security rings?
- How to manage user access requests based on defined policies by CCs?

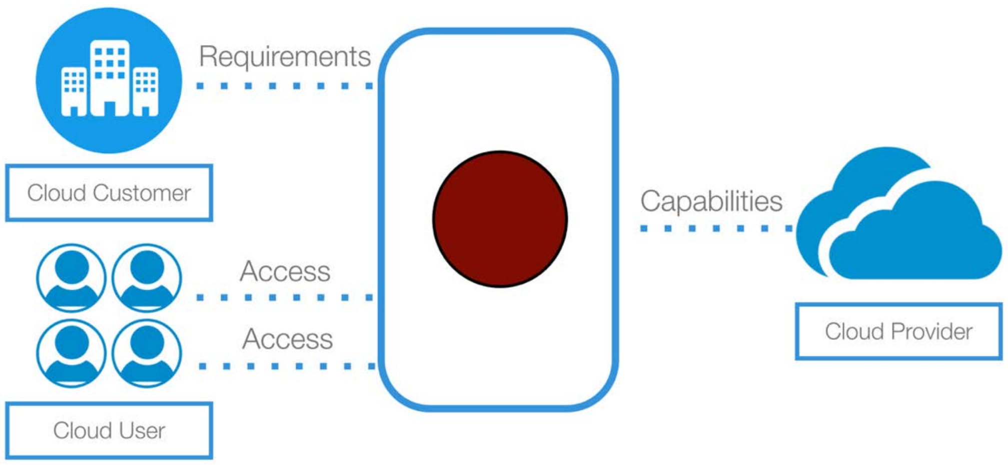


Fig. 1.3. The Necessity of Policy Management in Clouds

1.3. Research Aim

The main aim of this thesis is to introduce an object-oriented standard to create and manage multiple security levels based on the capabilities of the service provider and requirements of cloud customers. In fact, each individual customer is granted the appropriate security level based on the declared requirements in order to enhance reliability in cloud computing environments.

The approach we propose is based on semantic policy clustering to classify nodes with same or common security policies in an aggregate virtualized cluster for federating defined policies according to their characteristics. By this federation, the processing time for each policy mapping is reduced due to the elimination of gratuitous and avoidable matching jobs. We define a well-established policy framework to define security policies which are compliant to requirements and capabilities. Moreover, a federated policy management schema is introduced based on the policy definition framework and policy clustering to create and manage virtual clusters with identical or common security levels. The proposed model consists in the design of a well-established ontology according to security mechanisms, a procedure which classifies nodes with common policies into virtual clusters, a policy engine to enhance the process of mapping requests to specific node as well as associated cluster and matchmaker engine to eliminate inessential mapping processes.

1.4. Objectives and Challenges

The main concerns regarding the described models are the discovery, interoperability, and compatibility of security requirements based on the characteristics of current distributed networks and cloud-based environments [9]. Furthermore, the scalability and flexibility of mapping and the semantic analysis of policies regarding different capabilities of service providers and requirements of customers are the other challenging issues in the process of policy generation, application and management [10]. These challenges are classified in policy definition, policy application and policy mapping processes as follows:

- The process of offering security capabilities of CPs to CCs as on-demand services. Also, the offers need to be updated based on added or revoked capabilities of CPs.
- The process of policy definition and generation based on efficient mapping between requirements and capabilities.
- The simultaneous syntactic and semantic analysis of security level.
- The process of security ring (level) establishment according to requirements of customers.
- The process of applying security policies to resources based on capabilities.
- The scalability and flexibility of security levels by different CCs.
- The process of managing access requests by CUs according to defined policies.

The research was done in three phases to provide an efficient and reliable policy generation and management schema for Multi Security Level Cloud Computing (MSLCC). The aim of each phase is to define, apply and map policies based on the requirements of cloud customer and sensitivity of data.

- Phase 1 (Policy Generation): Providing a structural multi-level ontology to define and manage security levels according to the capabilities of service provider, constraints and requirements of cloud customers.

- Phase 2 (Policy Application): Introducing a reliable resource management and scheduling algorithm to apply defined policies to data based on established security level in the first phase.
- Phase 3 (Policy Mapping): Designing an efficient access management schema to map access request of cloud users according to defined policies by cloud customers in associated security ring.

1.5. Conclusion

To enhance the quality on managing security policies in cloud-based environments and to provide efficient, secure and reliable matching between security requirements of customers and capabilities of service providers, a multi-layered policy engine is introduced in this thesis. A well-established policy framework has been defined to generate security policies which are compliant to requirements and capabilities. Moreover, a federated policy management schema has been introduced based on the policy definition framework and multi-level policy application to create and manage virtual clusters with identical or common security levels. The model has been evaluated according to performance and security parameters and proved that this multi-layered policy engine enhances the reliability and efficiency of managing security polices in cloud computing environments during policy definition, policy application and policy mapping procedures.

The remaining parts of this paper are organized as follows: a tutorial of policy-based cloud computing is presented in Section 2 as well as multi-layered policy in chapter 3. Chapter 4 introduces a structural resource management and scheduling algorithm to apply multi-layered security policies based on the defined ontology. Chapter 5-7 set out three different scenarios for multi-level policy engine in authentication, access control and re-encryption protocols. The comprehensive performance and security analysis of policy-based model in different scenarios are presented in each chapters and this is followed by the conclusion in Chapter 8.

Chapter 2

2. MOTIVATION AND RELATED WORKS

2.1.Security in Clouds

Cloud computing is an emerging technology which is known to be the advanced generation of on-demand IT services over broad networks. This unprecedented evolution utilizes the concepts of virtualization, processing power, isolation, connectivity and distribution [10] to store and share computer resources via Internet. Despite the considerable advantages of cloud-based services such as elasticity, resource pooling, layer-based responsibilities and lower service delivery, maintenance and upgrade costs [11], there are some remarkable security and privacy concerns that have affected the reliability of cloud computing environments [12]. These issues led to the appearance of several researches and solutions and have become the leading cause of impeding the development of cloud-based services in industries with sensitive data.

Typically, security and privacy issues in clouds have been classified to three main parts: Identity management and authentication procedures, data protection in cloud-based data centers, and managing accesses according to defined policies [13]. The latter, which is the focus of this work, refers to the capabilities of service provider to protect resources in data centers according to sensitivity of data and requirements of cloud customers. In fact, an efficient using of security mechanisms according to the capabilities of service provider and requirements of cloud customers for a secure and reliable data protection and access control in cloud environments is normally expressed by defined data policies [14].

Accordingly, a tutorial of policy-based cloud computing is presented is this chapter by issues and key challenges that should be addressed using existing technologies and innovative mechanisms such as policy management engine that provides different levels of security in cloud data centers based on capabilities, requirements and constraints which are also included various security protocols, mechanisms and algorithms in clouds.

2.2.Policy Management in Clouds

The rapid growth of using cloud-based services in various industries is impossible to deny, as it has enhanced the reliability and efficiency for accessing shared pools of configurable computing resources. This growth is the result of providing the considerable advantages of storing and maintaining computing resources in unlimited storages with the most cost efficient method, business continuity and scalability [4]. Despite these substantial benefits, there are some remarkable information policy concerns such as security, privacy and access management that have affected the reliability of cloud computing environment. Applying an appropriate level(s) of security to data stored in cloud-based storages is one of the most challenging issues in this area, based on the details defined in SLA [15][16] and sensitivity of data [17].

The establishment of a multi-level security architecture based on the data sensitivity and requirements of cloud customers is the most appropriate solution to avoid considerable and unnecessary processing power consumption for manipulating both sensitive and non-sensitive data in the same level of security [9]. On the other hand, managing multiple security levels based on the capabilities of the cloud provider and the requirements of cloud customers is a potential issue due to the elasticity, isolation and scalability concepts in cloud-based environments.

The most common approach to express high-level security constraints is based on the the usage of languages and metadata for the specification of security policies [14]. By these language-based approaches, cloud providers are able to announce and provide security capabilities as well as matching these capabilities to customers' requirements [18]. In fact, policies are defined according to syntactic and also semantic matchmaking of requirements and capabilities to ensure the security of nodes and to map access requests to defined policies efficiently. The challenging issue in this case is to manage applied policies in different virtualized nodes considering scalability, isolation and elasticity concepts in cloud computing environments.

The adoption of policy-based multi-level security management in clouds needs an appropriate and structural ontology to represent, generate, apply and manage policies according to core concepts of virtualized data centers [19]. Several policy management engines and frameworks have been proposed following different approaches in various application domains. These models have typically focused only in access policies [20] include privacy and user-based concepts or Federated Security Policies (FSP) contain multiple security concepts such as encryption, access control, authentication and transport.

One of the most popular web-based semantic languages has been used in several policy-based schemas is Web Ontology Language (OWL) [21]. A complex knowledge and relations between requirements, capabilities and constraints were represented in OWL [22] with an efficient reasoning support, sufficient expressive power and convenience of expression [23]. Also, several OWL-Based subsidiaries were presented to improve the restricted expressivity of OWL. OWL-L [24], OWL-S [25] and F-OWL [26] are some of OWL-based models there have been introduced to provide an independent domain policy specifications based on deontic constructs and to allow several types of policies such as right, prohibition, dispensations and obligations. Furthermore, to improve the problem of allowing customers of classic multi-agents and distributed networks to define different security policies as well as predictability and controllability assurance of each components, KAoS [27] was proposed as an OWL-based policy language model. In KAoS, policies are classified to four main categories: *Positive-Authorization*, *Negative-Authorization*, *Positive-Obligation* and *Negative-Obligation* that are associated with service properties (*i.e.* policy services such as policy enforcement and domain services such as hierarchical grouping of users) [28]. However, the main issue of OWL and its subsidiaries is the compatibility of these models to modern distributed networks specially in elasticity and isolation concepts.

Ponder [29] is another policy management framework based on object oriented concepts that was developed by imperial college. This framework includes general architecture and policy deployment model associated with several extensions for both access control and protection managements. It also allows customers to generate events, constraints, constants and other reusable elements as associated parts of security policies as well as allowing instantiation of typed policy specification for parameterization support of policies. The most important disadvantages of Ponder is the lack of generality by using several basic policy types and compositing each of them with different syntax. Accordingly, several Ponder-based deployment models were introduced [21,22] to address the instantiation, distribution and enabling of policies and also disabling, unloading and deletion of policies.

Web Service Policy Framework (WS-Policy) [32] is an extendable general purpose framework that has been recommended by World Wide Web Consortium (W3C). An associated syntax was defined in this framework to describe the policies of entities with a broad range of service capabilities and requirements in web-based models. WS-Policy involves several subsets according to the different structures of service domains. For instance, WS-Trust [33] was defined to change security tokens into different formats by an interoperable manner in order to establish and assess the presence of participants in secure message exchange. Also, WS-SecurityPolicy [34] and WS-SecureConversation [35] were defined to describe the security specifications of WS-Trust by improving the performance of frequent communications and using a shared symmetric and a pair of asymmetric keys from the security context respectively. The other popular WS and XML-based schema to describe the policy, request the authorization decision and respond with the authorization decision is XACML (Extensible Access Control Markup Language) [36]. This standard defines a policy enforcement point for interacting with a policy decision point. The comparison between WS-based and XACML standards have been extended in [17].

WS-Policy and associated subsets have been extended by several researchers to be used in policy engines of multi-domain services. In overall, WS standards have been used in policy engines to provide QoS assertion models which are generic, domain-independent and expressible across different layers and service roles [37]. One of the WS-based semantic models was proposed in [38] for generating security policies specifically for cloud computing environments to a enable a flexible and powerful matchmaking process between customers and providers security requirements. This model uses several terms and concepts to model security features within a policy as well as providing compatibility of semantic framework with syntactic polices. Furthermore, using an automated negotiation framework [39] based on WS-Policy is another solution to support participant security policies for communication, negotiation and SLA creation.

To enable more satisfactory discovery results that better fit the requirements of cloud customers, WS-Policy was extended through an efficient ontology and rule reasoning [40]. Hence, a set of rules was defined security policies associated a developed rule-based engine improve policy evaluation and policy mapping. The drawback of this rule-based policy management was the considerable processing time for taking the overheads of policy representation with an ontology language and for transferring them to a rule-based structure.

According to the most prominent characteristic of cloud-based models for providing on-demand services, semantic rules have been used during the establishment of policy framework for an efficient and reliable policy generation and mapping [41]. Several works such as LaaS (Law-as-a-Service) [42] were proposed for cloud service providers on law-aware semantic cloud policy infrastructure to deploy their cloud resources and services based on OWL ontologies and stratified Datalog rules with negation for policy exceptions. Also, PEaaS (Policy Engine-as-a-Service) [43] was suggested based on WS-Policy to provide multi-level policies in clouds according to Protection ontology. These policies create create standard or dedicated security rings (*i.e.* levels) regarding to the capabilities of service providers, constraints and requirements of cloud customers. However, the heterogeneous characteristics of these services together with the dynamicity inherent in clouds, hinders the formulation of an effective and interoperable set of policies that is adoptable for the underlying domain of applications [44].

Hence, establishment of ontological templates for the semantic representation of security policies is needed to facilitate the definition of appropriate security policies using a generic and extensible RDF [45]. Therefore, policies should analyzed syntactically and also semantically in different service layers and service roles to express capabilities and requirements based on SLA and to complement the existing service selection and negotiation framework [37]. The most challenging concern in simultaneous syntactic and semantic mapping is the flexibility of the generated policies according to the predictable and un-predictable variations of capabilities, constraints and requirements [46].

The other challenging issue in configuration of security policies is the possibility of confliction in the course of unexpected occurs for policy management and enforcement. Indeed, the structure of policy generation framework needs to be reliable and efficient enough to provide accurate detection of conflict between policies and the security of cloud digestion [47]. This concept has been more extended in federated cloud networks to rely on a service manifest that specifies global polices [48]. These extended frameworks enable automated deployment and configuration of network security functions across different clouds. However, the complexity of these federated frameworks is still challenging according to multi-structure policies in virtualized data centers [41,42].

Overall, the main concerns regarding to semantic policy-based security management models are: (1) to provide efficient, secure and reliable matching between security requirements of customers and capabilities of service providers according to interoperability and isolation concepts in cloud-based environments [51], and (2) to ensure the scalability and flexibility of mapping due to the large number of defined polices and access requests [52]. These two concerns reduce the reliability of using cloud services according to different policy-based aspects such as identity management, data protection and user revocation.

2.3. Policy-Based Identity Management

The fundamental goal of any identity management model is to ensure a reliable authentication of subscribed users according to the defined policies of different cloud servers and to protect information from un-authorized accesses. There is a wide variety of methods, techniques, models, and administrative capabilities used to propose and design identity management models [53] and each system has its own attributes, methods, and functions. The importance of these identity management models is more evident for cloud providers and customers according to the characteristics of cloud-based services that work with shared open environments. Therefore, several studies and researches were performed to improve the reliability and efficiency of managing identities in clouds.

In recent years most of Single Sign On (SSO)-enabled access management models are based on web applications such as SAML [54] and OAuth [55] for allowing users and application services access to web resources. OpenID [56] is one of the most popular relevant federated authentication technologies that allows cloud users to use a single identity for accessing various services from different cloud servers and for elimination of managing different identities by cloud users. However, OpenID relies on an ID provider to generate a unique identity for each user [57]. Therefore, the server has to connect to the ID provider on the Internet during authentication of cloud users and it leads to a high level of time and computation load [58].

Shibboleth [59] is another federated identity management model which is similar to OpenID, for allowing users to authenticate to different services using just one piece of information. Shibboleth is an open source implementation of federated identity based management model where the identity providers provide information and the service providers consume this information giving access to content or services [60]. However, the most challenging concern of Shibboleth is to provide different levels of authentication based on the sensitivity of data in various cloud servers. In fact, mapping between federated identity information with different levels of security in cloud servers based on defined policies is still the main issue in these types of federated identity management models [57][58].

The other solution was proposed as Kerberos [61] by using distribution of authentication tickets to provide a generic access control protocol and reliable SSO. The most drawback of Kerberos is the lack of privacy solutions in the model that was tried to solve as an extension in several models such as KAMU [62] or PrivaKerb [46]. Kerberos-based models use an operation mode (*cross-realm*) to be compatible for federated environments, Nevertheless, these models consist a completely independent infrastructure aside those already established for the access to web application services and the network access service [63]. Hence, Kerberos *cross-realm* federations have not been widely deployed [64]. Using an interaction between Kerberos and Extensible Authentication Protocol (EAP) [65] protocols was the other solution the enhance weaknesses of *cross-realm* federations. Using EAP-based pre-authentication mechanism [63] and also using Protocol for Carrying Authentication for Network Access (PANA) [66] to bootstrap dynamic Kerberos credentials on the service providers [67] are the most popular efforts to enhance Kerberos *cross-realm* federations. However, the necessity for deployment of Kerberos entity on every organization and providing SSO within each organization's boundaries are the most considerable inconvenient of proposed models.

To solve the problem of compatibility and deployment in Kerberos-based models, Leandro et al. [68] uses a multi tenancy authorization system to deliver access control based on concerns about the privacy of data. The proposed model was built around Shibboleth core concepts with authorization and authentication mechanisms to emphasize on self-governing and control of trusted third parties, according to the digital identity federation [69]. This method was followed by [70] by adding stand-alone identity management features to the federated model. However, it has been proven [71] that misuse of user identity information in self-governed and stand alone identity federation could happen via SSO services in IDP and SP, which could lead to identity theft (*i.e.* the main concern in federated identity management systems). Thus, Bhargav-Spantzel et al. [72] recommended two mechanisms to protect the misused of identity information: distributing user identity information amongst several self-governed entities and using zero-knowledge proofs techniques to prevent identity theft within an IDP or SP. Although, the recommended mechanisms reduced the chance of identity theft, there are still serious concerns about the process mapping requests from revoked identities in stand alone identity federations [73]

Kalra and Sood [74] proposed an Elliptic Curve Cryptosystem-based (ECC) algorithm to provide a mutual authentication protocol for secure communication of embedded devices and cloud servers in association with HTTP cookies. The evaluation of this model proved that it was robust against multiple security attacks. However, managing ECC keys for different cloud servers in this model takes considerable processing power for manipulating sensitive and non-sensitive access policies in different cloud servers [75].

Apart from ECC, several cloud-based authentication models were designed with various techniques such as Biometrics-Based Authentication [76], Certificate-less Anonymous Authentication [74][75], User Behavior Analysis-based [77] and ID-Based identity management [78] with the same issue that is the lack of congruency in different cloud servers with distinguished access policies. Indeed, the necessity of a policy-based identity management in different cloud servers with various security levels is undeniable according to rapid growth of cloud providers.

Using policies to establish different security levels in traditional and also modern distributed networks allows to manage the processing power for manipulating sensitive and also non sensitive resources. Hence, several policy-based languages and models are suggested to classify resources based on sensitivity. WS-Policy [32] is an extendable general purpose framework associated with a defined syntax to describe the policies of entities and a broad range of service requirements and capabilities in a web services-based system. This XML-based framework has been extended by various researchers according to service requirements such as security services. Di Modica and Tomarchio proposed a semantic secure policy matching based on WS-Policy framework in [38] and [41] for service oriented and cloud-based architectures. In the cloud-based model, the capabilities of the cloud service provider and the requirements of the cloud customer were defined within policies adopted to the WS-Policy framework.

There are several security standards that are extended from WS-Policy architecture such as "WS-Trust" [33] or "WS-PolicyAttachment" [79]. "WS-Trust" is an OASIS standard for changing security tokens from one format to another in an interoperable manner in order to establish and assess the presence of participants in a secure message exchange. Also, "WS-PolicyAttachment" was expressed to define two mechanisms for associating policies with the subjects they apply and to represent the way of attaching WS-Policy descriptions end points. Furthermore "WS-SecurityPolicy" [34] and "WS-SecureConversation" [35] were extended from WS-Policy architecture to describe the security specifications of WS-Trust and to improve the performance of frequent communications by using a shared symmetric and pair of asymmetric keys from the security context respectively. The comparison between WS-based standards is available in [17].

The challenging issue in policy-based resource classification is to map access requests from cloud users based on defined security policies of resources. In fact, a policy-based identity management needs to analyze policies syntactically and also semantically and map access requests based on the established security mechanisms of each node [12]. Hence, policy management is one of the most challenging key points of identity management in multi-level virtualized resources. On the other hand, the processes of scheduling, analyzing and mapping access request tasks according to the policies need to be considered in policy-based identity management. Providing a federated authentication schema for different cloud servers needs an efficient authentication task manager to administrate access requests based on defined policies. CSA is a multi-level adaptive authentication schema in clouds that was proposed [80] to dictate the efforts of protocol participants by identifying a legitimate user's requests and placing them at the top of the authentication process queue. In fact, a multi-objective scheduling model for authentication tasks was suggested to prevent DoS attacks in multi-level cloud servers. Although, the process of authentication task management in CSA was based on risk identification, not on defined policies in multi-level authentication.

According to the previous research results, a scalable policy-based identity management is presented in this chapter to address two main problems: (1) Lack of coincidence in identity management models based on defined policies and various security levels in different cloud servers, (2) Lack of multi-objective authentication task management according to the defined policies in multi-level authentication procedures.

2.4. Policy-Based Data Protection and Re-Encryption

Using cryptographic models are the most common solutions to ensure data and resource protection in virtualized environments. To guarantee the reliability of these encryption models and to make sure the data confidentiality and fine-grained access control in cloud computing environments, stored data and resources needs to be re-encrypted periodically or based on special mechanisms such as revoked user-based or manual re-encryption [81].

Managing the process of re-encryption is a challenging issue that involves many limitations such as time management, resource confidentiality, and level of access. Therefore, an efficient re-encryption management may increase the reliability and the rate of security in cloud computing environments.

The most popular re-encryption models are based on attributes for managing and monitoring security of resources. These attributes are defined as properties of re-encryption class to classify resources based on sensitivity and priority. Hierarchical Attribute-Based Encryption (HABE) is one of the suggested models [82] that use data consistency and data confidentially attributes for high performance and full delegation re-encryption process. The main drawback of this model is the dependency of the HABE performance on reliability of cloud infrastructure. This means, the correctness of the re-encryption process is completely dependent on the rate of security in cloud infrastructure.

This problem was solved in R3 model by using a time-based re-encryption approach [83], in this model the underlying cloud infrastructure was not necessarily reliable in order to ensure correctness. Furthermore, the time difference between cloud server and data owner is an important issue in time-based re-encryption models that was solved in R3 with appropriate clock synchronization.

The performance of time-based re-encryption was improved by Liu et al. [83] to determine a period of time according to defined parameters for re-encrypting stored data, generating new key and automatic expiring of revoked user's access. In this model, concepts of attribute-based re-encryption and proxy re-encryption were combined with sets of time attributes. Therefore, only users whose attributes satisfy the access structure and whose access rights are effective in the access time can recover corresponding data.

One of the other attribute-based re-encryption models was Key Policy-Attribute Based Encryption (KP-ABE) that was proposed by Park et al. (2006). In this approach, internal nodes are threshold gates and leaf nodes are associated with attributes that are used to encrypt data. This model was improved [85] by adding some techniques such as Typed-Based Proxy Re-Encryption [86] and bilinear mapping for providing selectively delegate decryption right using Typed-Based Proxy Re-Encryption. The main problem of this model was the dependency of KP-ABE on specific attributes that decreased the compatibility of this model in virtualized infrastructure and cloud-based environments. In fact, this model uses single level re-encryption policies and this mechanism declined semantic mapping between policies and capabilities.

To solve the problem of single level policies for reliable re-encryption, several multi-level policy management schemas were proposed. Di Modica and Tomarchio (2011) suggested one of the first policy-based classification approach's in clouds that leverages on the semantic technology to enrich standardized security policies with an ad-hoc content and to enable machine reasoning which is then used for both the discovery and the composition of security-enabled services. In this model, requirements and capabilities for cloud customers and providers are defined within policies which are adopted to policy intersection mechanism provided by WS-Policy [32].

WS-Policy is a recommended framework from W3C for policy specification of Web Services that includes policies that are defined as a collection of alternatives contain assertions to specify well-established characteristics for using selection of various services (*e.g.* requirements, capabilities or behaviors).

Overall, the main concerns in current re-encryption models in clouds are dependency of suggested models on specific attributes in property-based models that has been not adopted to virtualized infrastructure and lack of scalability and flexibility in semantic mapping of policies in policy-based re-encryption models.

2.5. Policy-Based User Revocation

The problem of managing user revocation requests in policy-based cloud computing is the other focus of this work. In fact, each of revocation requests should be mapped to defined policies of associated resources in the request for evaluation of the user revocation process and updating defined security policies.

Most of current user revocation models is based on Attribute-Based Signature (ABS). In fact, the most challenging issue in encryption-based user revocation models is to re-encrypt data and manage associated keys after a user is revoked from cloud services [87]. One of these revocable ABS models was proposed by Escala *et al.* [88] proved to be adaptive secure in the standard model. This schema assigns a randomly selected identity to each user in addition to the attributes associated with an external entity to keep a secret verification key and a list of revoked user identities. Also, the verification key is used to trace a signature to the signer. However, this model conflicts with the unlink-ability and anonymity properties of ABS schemes.

Using an external party as a mediator to manage instantaneous user revocation [89] or structural timestamps associated with the attribute private key [90] are two extra features that was proposed for ABS-based user revocation models. The main drawback of these features is the potential overhead and performance impact due to the lack of immediate user revocation process.

To decrease these overheads, Attributed-Based Group Signature (ABGS) schemas are proposed to provide anonymity for users in a group and generate a signature on behalf of the group [91]. The validation process can only verify the correctness of the signature, and whether it is produced by a valid user in the group. However, this scheme relies on the group manager to link a signature to a signer before the signature is revoked.

Panda [92] is a public auditing mechanism for shared data that was proposed with an efficient user revocation mechanism. The idea of proxy re-signatures was used in this model to allow the cloud to resign blocks on behalf of existing users during user revocation, so that existing users do not need to download and re-sign blocks by themselves. Also, Panda uses a public verifier to audit the integrity of shared data without retrieving the entire data from the cloud, even if the cloud re-signs a part of shared data. This main drawback of this model is the considerable processing power for manipulating sensitive and also non-sensitive data after the process of user revocation. In fact, the whole associated resources need to be updated to ensure the security of cloud after a user is removed from accessing.

To decrease the processing power and provide an efficient user revocation model, each revocation request needs to be processed according to the defined security policies. In fact, security and privacy policies (*i.e.* encryption, signature, access control and authentication) for each associated cloud node specifies whether the additional manipulations and processes are needed or not [93].

2.6. Use-Case

In this section a general use-case is described to express the research question more significant. Assume that there is a Medical Center (*e.g. MCenter*) that aims to use cloud-based services from a service provider (*e.g. CloudX* company) for the subscribers. Accordingly, the main entities of contract are defined as follows:

- Cloud Provider: *CloudX*
- Cloud Customer: *MCenter*
- Cloud Users: *Patients, Doctors, Nurses, Administration Staff, Financial Department Staff, Management, Statistical Department Staff, IT Department Staff, Insurance Companies, etc.*

According to the sensitivity of data, the customer needs three levels of security. The security requirement list is provided from customer as follows:

Table 2.1. MCenter Security Requirements

	Level 1	Level 2	Level 3
Name	Low	Medium	High
Encryption	No	Yes	Yes (Strong)
Shared	Yes	Yes (Limited)	No
Discretionary Access	Yes	Yes	No
Authentication	No	Yes	Yes (Double)
Integrity	Yes	Yes	Yes
Digital Signature	Yes	Yes	Yes
Geo-Control	No	Yes	Yes
Content-Depend	No	No	Yes
Context-Based	Yes	Yes	Yes
Temporal Isolation	No	No	Yes
View-Based	No	No	Yes
Attributes	No	Yes	Yes
Role-Based	No	Yes	Yes

The main aim of a policy-management engine is to provide these security levels for *MCenter*. In fact, the first question of this research is to map the requirements of cloud customers to the capabilities of service provider efficiently according to the constraints and current mechanisms.

Furthermore, imagine several cloud customers (*e.g.* companies, universities, medical centers, *etc.*) with different requirements according to the sensitivity of data and resources. The second aim of a policy management engine is to consider scalability and isolation concepts for providing various dedicated security levels according to the different requirements of cloud customers. In fact, the Policy Engine needs to create security level based on requirements and capabilities, to manage these levels based on constraints, isolation and scalability concepts, and to apply real time actions according to defined policies.

The most common approach to express high-level security constraints is based on the the usage of languages and metadata for the specification of security policies [20]. By these language-based approaches, cloud providers are able to announce and provide security capabilities as well as matching these capabilities to customers' requirements. In fact, policies are defined according to syntactic and also semantic matchmaking of requirements and capabilities to ensure the security of nodes and to map access requests to defined policies efficiently. The challenging issue in this case is to manage applied policies in different virtualized nodes considering scalability, isolation and elasticity concepts in cloud computing environments.

The approach we propose is based on semantic policy clustering to classify nodes with same or common security policies in an aggregate virtualized cluster for federating defined policies according to their characteristics. By this federation, the processing time for each policy mapping is reduced due to the elimination of gratuitous and avoidable matching jobs. We define a well-established policy framework to define security policies which are compliant to requirements and capabilities. Moreover, a federated policy management schema is introduced based on the policy definition framework and policy clustering to create and manage virtual clusters with identical or common security levels. The proposed model consists in the design of a well-established ontology according to security mechanisms, a procedure which classifies nodes with common policies into virtual clusters, a policy engine to enhance the process of mapping requests to specific node as well as associated cluster and matchmaker engine to eliminate inessential mapping processes.

2.7. Summary

A tutorial of policy-based cloud computing was presented is this chapter by issues and key challenges that should be addressed using existing technologies and innovative mechanisms such as policy management engine that provides different levels of security in cloud data centers based on capabilities, requirements and constraints which are also included various security protocols, mechanisms and algorithms in clouds.

Overall, the main concerns regarding to semantic policy-based security management models is: (1) to provide efficient, secure and reliable matching between security requirements of customers and capabilities of service providers according to interoperability and isolation concepts in cloud-based environments [51], and (2) to ensure the scalability and flexibility of mapping due to the large number of defined polices and access requests [52].

In next chapters, a well-established policy management framework has been defined to generate security policies which are compliant to requirements and capabilities. Furthermore, the effects of this policy-based framework in various scenarios such as identity management, user revocation and data protection have been examined to enhance the reliability and efficiency of cloud computing as an emerging technology.

Chapter 3

3. CLOUD SECURITY ONTOLOGY

3.1. Introduction

In this chapter, a structural policy management engine is introduced to enhance the reliability of managing different policies in clouds and to provide standard as well as dedicated security levels (rings) based on the capabilities of the cloud provider and the requirements of cloud customers. Cloud security ontology (CSON) is an object oriented framework defined to manage and enable appropriate communication between the potential security terms of cloud service providers. CSON uses two super classes to establish appropriate mapping between the requirements of cloud customers and the capabilities of the service provider. It also provides standard and dedicated security rings through simultaneous syntactic and semantic analysis. In comparison with current models, the proposed ontology enhances reliability and efficiency in order to establish appropriate and structural policy management in cloud computing environments.

3.2. Related Works

Web Ontology Language (OWL) is one of the most popular semantic web languages that was used in policy-based models to represent complex knowledge and relations between capabilities and requirements. OWL and its subsidiaries were used in several policy frameworks regarding to the computational logic analysis features. HP labs proposed a policy framework (Rei) [94] to provide an independent domain policy specification based on deontic constructs and F-OWL reasoned [26] and to allow several specification policies (i.e. right, prohibition, dispensations and obligations). Rei supports two main meta policies: Default Meta Policy (to describe the default behaviour of the policy) and Meta-Meta Policy (to allow the setting of precedencies based on default meta policies).

KAoS [27] was another OWL-based ontology language that was introduced to allow customers of distributed and multi-agent systems to define arbitrary policies and to assure predictability and controllability of each components [28]. These policies are divided into four main categories: *Positive-Authorization*, *Negative-Authorization*, *Positive-Obligation* and *Negative-Obligation.* Each of them is associated with service properties that are classified to policy services (based on specification, conflict resolution, management and policy enforcement) and domain services (enabling the hierarchical grouping of users and computational entities to easier administrate policies) [28].

WS-Policy [79] is an extendable general purpose framework associated with a defined syntax to describe the policies of entities and a broad range of service requirements and capabilities in a web services-based system. This XML-based framework has been extended by various researchers according to service requirements such as security services. Di Modica and Tomarchio proposed a semantic secure policy matching based on WS-Policy framework in [41] and [38] for service oriented and cloud-based architectures. In the cloud-based model, the capabilities of the cloud service provider and the requirements of the cloud customer were defined within policies adopted to the WS-Policy framework.

There are several security standards that are extended from WS-Policy architecture such as “WS-Trust” [33] or “WS-PolicyAttachment” [40]. “WS-Trust” is an OASIS standard for changing security tokens from one format to another in an interoperable manner in order to establish and assess the presence of participants in a secure message exchange. Also, “WS-PolicyAttachment” was expressed to define two mechanisms for associating policies with the subjects they apply and to represent the way of attaching WS-Policy descriptions end points. Furthermore “WS-SecurityPolicy” [34] and “WS-SecureConversation” [35] were extended from WS-Policy architecture to describe the security specifications of WS-Trust and to improve the performance of frequent communications by using a shared symmetric and pair of asymmetric keys from the security context respectively. The comparison between WS-based standards is available in [17].

Sriharee et al., (2004) extended WS-Policy architecture through an efficient ontology and rule reasoning to enable more satisfactory discovery results that better fit the requirements of cloud customers. In this model, a set of rules was established as a policy and a rule-based engine was developed to more conveniently process the policy evaluation. However, taking the overheads of policy representation with an ontology language and transferring them to a rule-based structure is a challenging issue, taking considerable processing power for both policy generation and application processes.

Using semantic rules in the establishment of policy framework is the most appropriate solution to map between requirements and capabilities semantically to generate efficient and reliable policies. Hashmi et al., (2014) suggested a negotiation frameworks based on these semantic rules which conducts an automated negotiation between service provider and customer to generate protocol independent and support participant polices for communication, negotiation and SLA creation. Moreover, Chhetri et al., (2010) presented a policy-cantered QoS meta-model and a QoS assertion-model by syntactic and semantic generic analysis of policies in different service layers and service roles. This allows to express requirements, capabilities and constraints based on defined details of SLA and to complement the existing service selection and negotiation frameworks. The most challenging concern in semantic mapping is the flexibility of the generated policies according to the predictable and un-predictable variations of capabilities, constraints and requirements [46].

In overall, the main concerns regarding the described models are the discovery, interoperability, and compatibility of security requirements based on the characteristics of current distributed networks and cloud-based environments [9]. Furthermore, the scalability and flexibility of mapping and the semantic analysis of policies regarding different capabilities of service providers and requirements of customers are the other challenging issues in the process of policy generation, application and management [10]. Thus, in this chapter, an efficient ontology is presented to establish dedicated security levels for cloud customers based on the policies defined in accordance with the requirements, capabilities and constraints of each entity.

3.3. Cloud Security Ontology

Cloud security ontology (CSON) is an object oriented framework that is defined to manage and enable appropriate communication between potential security terms of cloud service provider. In fact, the main aim of CSON is to manage (*i.e.* launch, revoke, etc.) security terms based on capabilities of cloud provider, to provide an on-demand security service for cloud customers associated with a policy engine and an API according to the requirements, and to establish a validity engine for syntactic and semantic mapping between requirements, capabilities and constraints. Figure 3.1 shows a detailed overview of the CSON architecture:

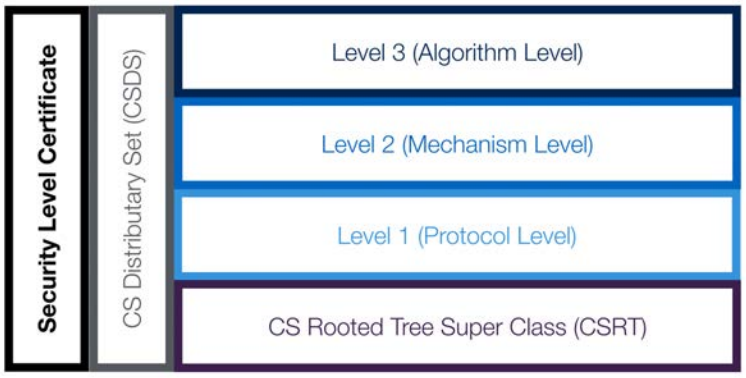

Security Level Certificate
CS Distributary Set (CSDS)
Level 3 (Algorithm Level)
Level 2 (Mechanism Level)
Level 1 (Protocol Level)
CS Rooted Tree Super Class (CSRT)

Fig 3.1. General Architecture of CSON

CSON includes 3 levels of security terms, two super classes and one output class to create and manage security levels in clouds. The CS Rooted Tree super class (CSRT) is designed to manage security capabilities of cloud service provider by classifying algorithms in 3 levels according to inheritance concepts:

- **Protocol Level (Level 1)** is the highest level of CSRT, and contains the main security protocols for the establishment of security rings. A high level root for various security algorithms is defined in this level to specify which protocols are offered by cloud service provider. Access Control, Cryptography, Key Management, Transport, Authentication and Signature are the main protocols in this level.
- **Mechanism Level (Level 2)** is an interface level between the highest level of CSRT and different security algorithms offered by service provider. The main role of this level is to divide each protocol into several mechanisms and to categorize security algorithms into these mechanisms to provide the appropriate relations between the highest and the lowest level of architecture.
- **Algorithm Level (Level 3)** is the lowest level of the CSRT and includes various security algorithms offered by cloud service provider. The capabilities of the service provider are updated according to the availability or unavailability of security algorithms in this level. Figure 3.2 shows an example of CSRT in each level.

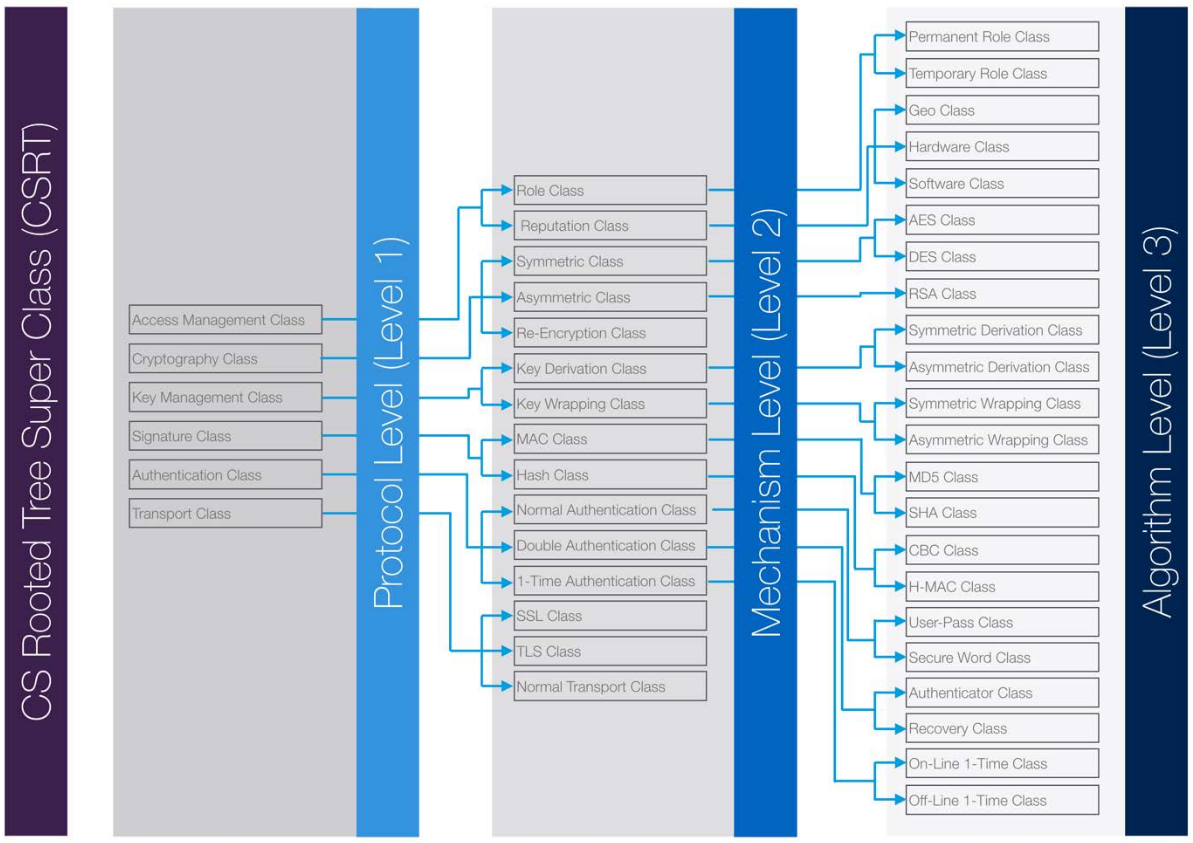


Fig 3.2. CS Rooted Tree Super Class Architecture

According to this classification and based on the inheritance concepts, the processes of rendering and revocation of security capabilities are limited to the lowest level of CSON in cloud-based data centres. In fact, CSRT represents all available security capabilities of the service provider by scanning the rooted tree. Hence, there is no necessity for rescanning the availability of security algorithms in created object.

Cloud Security Distributary Set (CSDS) is the second super class in CSON that has been designed to provide a semantic mapping between capabilities of service providers and customer requirements (Figure 3.3). CSDS includes four main concepts - *Scope, Essential, Structure* and *Objective* - within several sub-concepts.

Security Scope is a general notion that defines the owner (Cloud Customer) and the target (Cloud User) for the security ring. The Cloud Customer (CC) is typically an organization or a company that uses cloud services for employees or subscribers (*e.g.* Universities, Hospitals, etc.) while Cloud Users (CU) are defined end-users that use cloud-based services offered by CC according to the internal contracts (*e.g.* Students or Lecturers in a University). Hence, the scope of SCDS defines the creator or owner of security rings and potential targets according to SLA.

Objective of CSDS includes the security requirements that are mentioned by cloud customers and a defined set of security purposes (*i.e.* Confidentiality, Integrity, Authorization, Authentication, and Non-Repudiation) that represent a justification for the security ring.

Security Structure is the most important and highest level abstraction, constituting security policies and high level sub-policies according to the availability in CSRT. The structure of CSDS shows which security characteristics are used as main policies in the security ring based on the capabilities of the service provider. These policies are settled on the main protocols (*e.g.* user authentication, access control, cryptography, etc.) and subsets in lower levels. Furthermore, high level sub policies are also defined according to the selected policies. Sub policies in CSON are divided into two main categories: High Level Sub Policies and Low Level Sub Policies. High Level Sub Policies are clarified during the ring definition and affect all data subscribed in one security level with the same attribute (*e.g.* IP addresses from Germany for Geo control, reputation mechanism and access control protocol in a German university). Low Level Sub Polices are specifically assigned by admins or cloud users for each data in the same security ring after the process of ring definition (*e.g.* student, lecturer, supervisor and staff roles in permanent algorithm, role mechanism and access control protocol in a university).

The last property of CSDS is *Security Essential*, which provides information about intransitive physical and virtual resources in order to apply defined policies to data (*e.g.* encryption) and to retrieve data based on access requests (*e.g.* decryption). Furthermore, the billing information of the established ring (per a certain amount of data) is specified in this section.

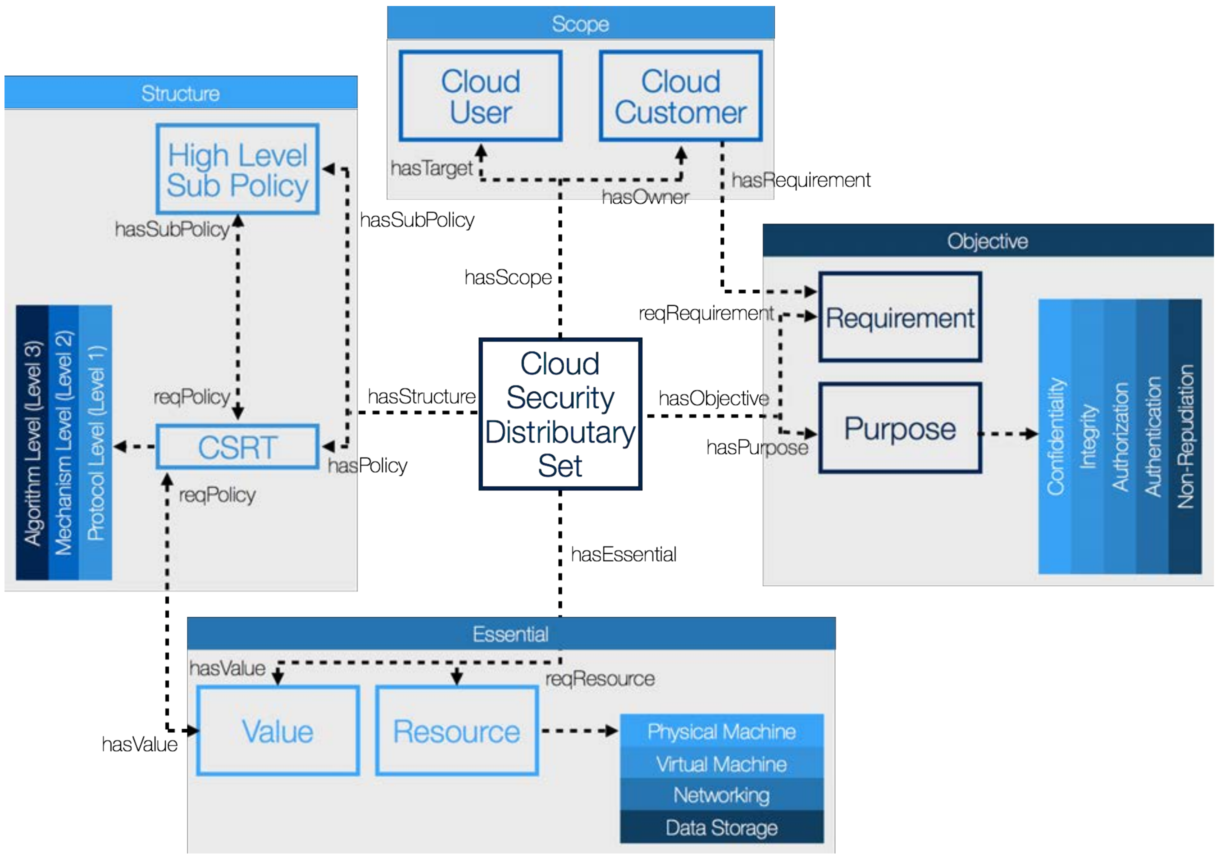


Fig 3.3. Cloud Security Distributary Set: High Level Class

Security Level Certificate (SLC) is the output of CSON including all the details about a security ring established in the cloud servers. In fact, SLC is a security-based SLA that describes subscribed security-as-a-services mechanisms to a cloud customer based on requirements and sensitivity of data. The structure of SLC is extended from WS-Policy and includes two main parts: header and body.

The *Header* of SLC introduces the scope, objectives and essentials of an established security level according to a validated CSDS object. These properties provide a general overview of the security level, owner and users, purposes, value and resources that are related to the ring, in terms of the following relationships:

- A cloud security level is owned by only one cloud customer (*hasOwner* property) while supporting several cloud users (*hasTarget* property).
- A cloud security level includes one to five security purposes (*hasPurpose* property) according to several requirements declared by cloud customer (*reqRequirement* property).
- A cloud security level estimates several cloud-based physical and virtual resources for the policy application process per a certain amount of data (*reqResource* property).

- A cloud security level has a determined value according to the requested resources and billing calculations per a certain amount of data (*hasValue* property).

The header of SLC is written according to CSON and is based on the WS-Policy structure. An example of a security level certificate header can be found below:

```
<wsp:Policy>
…namespace definition…
   <wsp:SLC rdf:ID="A543D3527B3">
      <wsp:Header>
         <security:Scope>
            <security:hasOwner rdf:ownerID="A3453"/>
            <security:hasTarget rdf:target="cu-ns#owner|cu-ns3#regEmail#@gwdg.de"/>
         </security:Scope>
         <security:Objective>
            <security:haspurpose rdf:purpose="confidentiality#integrity#authentication"/>
            <security:reqRequirement rdf:requestID="T353S387"/>
         </security:Objective>
         <security:Essential>
            <security:hasValue rdf:SLAresourceID="sla-ns#owner#slcID"/>
            <security:reqResource rdf:resource="VC-ns#VM-ns#Pr-ns#DS-ns"/>
         </security:Essential>
      </wsp:Header.
      <wsp:Body>
      …
      </wsp:Body>
   </wsp:SLC>
</wsp:Policy>
```

The SLC file starts with a specific *SLC_ID* that belongs to a cloud customer and has several targets: cloud users associated with the customer or users with an organizational registered email address. Furthermore, the key purposes of the security level and the set of customer requirements according to the request ID are shown as objectives of the SLC. Finally, the value of the developed SLC and the resources required to apply these policies per data are settled down in the essential part of the header.

The *Body* of SLC provides details about selected security mechanisms, policies and high-level sub policies of a security ring. In fact, all security terms with constant properties in a security ring are defined according to the capabilities of the service provider as well as the requirements of cloud customers based on the WS-Policy structure. The relations between algorithms are defined as follows:

- A cloud security level includes at least one security algorithm for each protocol according to the classification defined in CSRT (*hasPolicy* Property).
- High level sub-policies should be defined in the process of ring definition (*hasSubPolicy* Property) for associated algorithms in each cloud security level (*hasSubPolicy-reqPolicy* Property).
- Low level sub-policies will be defined during the process of policy application.

The body of SLC is established according to CDDS and CSRT and is based on the WS-Policy structure. An example of a security level certificate body for access management and cryptography protocols is shown below:

```
<wsp:Policy>
…namespace definition…
   <wsp:SLC rdf:ID="A543D3527B3">
      <wsp:Header>
      …
      </wsp:Header.
      <wsp:Body>

         <security:AccessManagementProtocol rdf:ID="AccessManagementRequirement">
            <security:RoleMechanism rdf:ID="RoleClassRequirement">
               <security:PermanentAlgorithm rdf:resource="PermanentRoleClass">
                  <rdf:HLSP_Role rdf:HLSP_Role_1="Lecturer" rdf:HLSP_Role_2="Student"
               rdf:HLSP_Role_3="Professor" rdf:HLSP_Role_4="Staff" rdf:HLSP_Role_5="Admin"/>
               </security:PermanentAlgorithm>
            </security:RoleMechanism>
            <security:ReputationMechanism rdf:ID=ReputationClassRequirement">
               <security:GeoAlgorithm rdf:resource="GeoClass">
                  <rdf:HLSP_Geo rdf:HLSP_Geo_1="Germany" rdf:HLSP_Geo_2="World"/>
               </security:GeoAlgorithm>
            </security:ReoutationMechanism>
         </security:AccessManagementProtocol>

         <security:CryptographyProtocol rdf:ID="CryptographyRequirement">
            <security:SymmetricMechanism rdf:ID="SymmetricRequirement">
               <security:AESClass rdf:resource="AESClass">
                  <rdf:HLSP_AES rdf:HLSP_AES_KeySize="256" rdf:HLSP_AES_Mode="CBC"/>
               </security:AESClass>
            </security:SymmetricMechanism>
            <security:ReEncryptionMechanism rdf:ID=REEncryptionRequirement">
               <security:ManualReEncryption rdf:resource="ManualReClass">
                  <rdf:HLSP_MRE rdf:HLSP_MER_1="Admin" rdf:HLSP_MER_2="Owner"/>
               </security:ManualReEncryption>
            </security:ReEncryptionMechanism>
         </security:CryptographyProtocol>
         <security:SignatureProtocol rdf:ID="SignatureRequirement">
            …
         </security: SignatureProtocol>
         <security:KeyManagementProtocol rdf:ID="KeyManagementRequirement">
            …
         </security:KeyManagementProtocol>
         <security:AuthenticationProtocol rdf:ID="AuthenticationRequirement">
            …
         </security: AuthenticationProtocol>
         <security:TransportProtocol rdf:ID="TransportRequirement">
            …
         </security: TransportProtocol>
      </wsp:Body>
   </wsp:SLC>
</wsp:Policy>
```

Regarding the defined security level, the permanent role and geo reputation classes were defined as access management protocols. Accordingly, five main roles and two main geo reputations are clarified as potential high level sub policies. The data owner or admin can choose between these properties for each data in the security level. Furthermore, AES symmetric algorithm has been chosen as the cryptography protocol with CBC mode and 256bits key size as high level sub-policies associated with the manual re-encryption algorithm defining two roles (*i.e.* admin and data owner) as sub-policies. Other protocols are also selected with the same procedure in each algorithm and high level sub-policies.

3.4. Security Level Clarification

The process of clarifying security levels has the requirements of the cloud customer and the capabilities of the service provider as inputs and SLC as the output. An efficient mapping between requirements, capabilities and constraints is done in this process to provide a security level certificate for cloud customers in the policy management engine.

The architecture of our proposed framework is based on four components: Policy Engine (PE), Policy Database (PD), SLA Engine and Ring Analysis Engine (RAE). A cloud service provider uses the policy management engine to offer two types of security levels to PD based on the defined or potential policies (fig. 3.4):

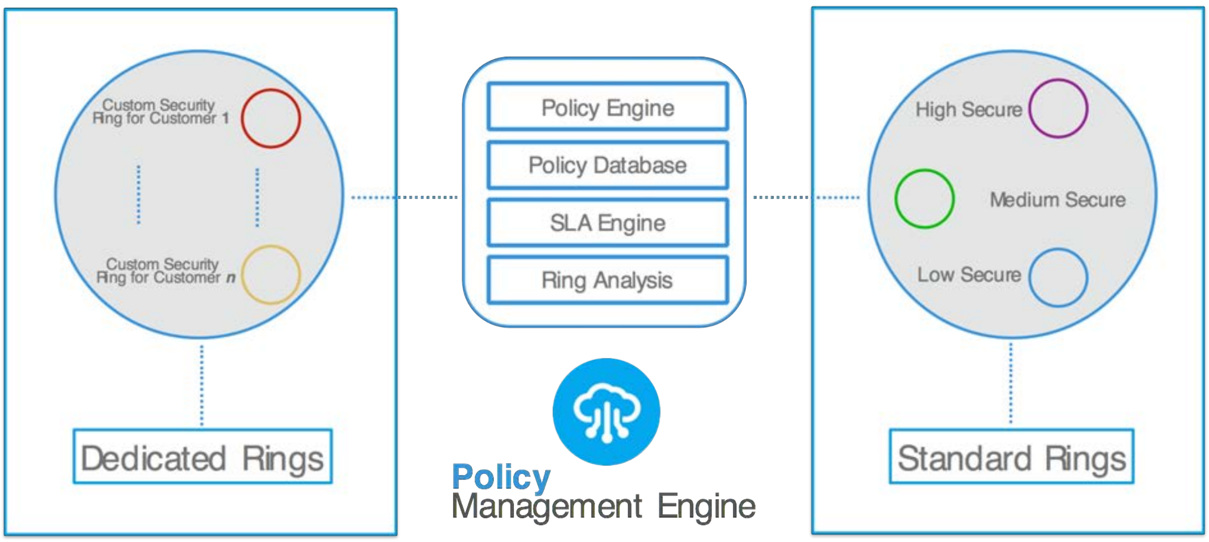


Fig 3.4. Standard and Dedicated Rings associated with Policy Management Engine

- *Standard Rings*: include sets of standard policies in package form that are offered by the service provider for a selection based on desired security level and data sensitivity.
- *Dedicated Rings*: include sets of potential security policies according to the capabilities offered by the service provider to let customers to create dedicated custom security rings based on their requirements.

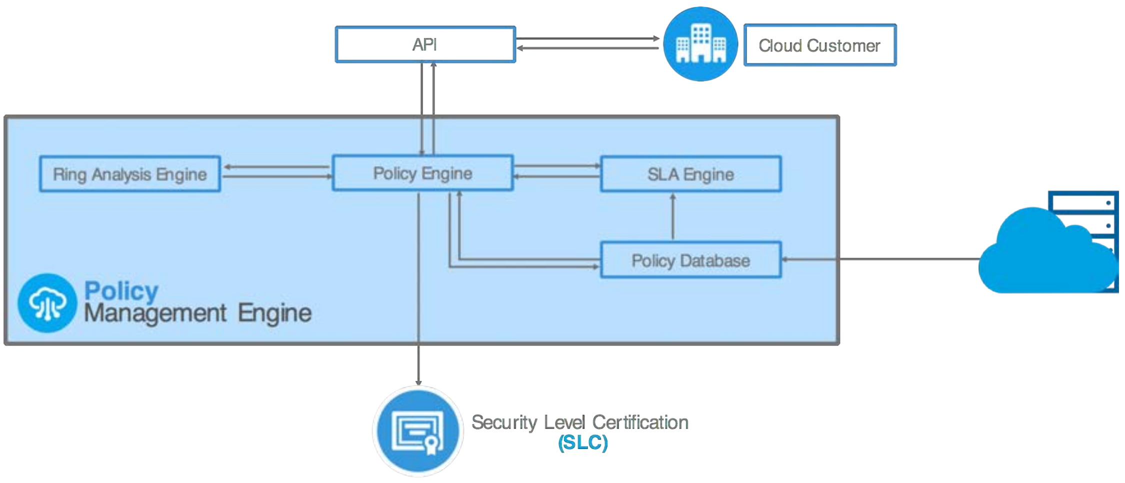


Fig 3.5. Proposed Policy Management Engine Framework

The policy database would be periodically updated according to the new or revoked security capabilities of the service provider, mutually interacting with the SLA engine about these updates. The policy engine is the main component of the framework and is responsible for policy matching and other policy-based procedures.

This is done by receiving service requests and requirements from cloud customer, processing the request, establishing appropriate connections with the ring analysis engine to semantically and syntactically validate the security level, receiving SLA-based details from SLA engine and generating the final SLC (Figure 3.5).

Algorithm 3.1.Selection of Standard Rings

Input: Set P: Let $P = \{p_1, p_2, \ldots, p_n\}$ represents all standard policy packages.

Output: SLC_j

```
1  $\exists\, p_i \in P : p_i = \{sp_1, sp_2, \ldots, sp_m\}$
       //where $sp_j$ ( $j \in \{1,2,\ldots,m\}$ ) donates a defined Security Protocol based on CSRT in standard policy package $i$.
2  $CSDS\ ring_j$
     $= new\ CSDS\ (CC_u, req_{id}, purp_{1,2,..,h})$;
       // an object is created from cloud security distributary set super class with User ID, Request ID and several purposes as properties of default constructor.
3  $Select\ (ring_j, p_i)$;
       // a policy package is chosen by cloud customer for package lists according to the requirement. This selection can be manually done by cloud customer or automatically suggested by RAE according to the defined purposes in object creation.
3  $For\ (int\ x = 1;\ x \leq\ m;\ ++x)\ \{$
       $bool\ SPCA = Check\_Avai(ring_j.sp_x)$;
       $if\ (SPCA\ ==\ False) then\ \{$
         $ring_j.req\ =\ False$;
         $Break; \}$
       $bool\ SPCU = Check\_Updates(ring_j.sp_x)$;
         $if\ (SPCU\ =$
         $=\ True) then\ Update(ring_j.sp_x)\ ; \}$
       // The availability of each algorithm in selected policy package is checked from PD and required updates are applied.
4  $if\ (ring_j.req\ ==\ true) then\ set(tg_{1,2\ldots t})$;
   $for\ (int\ x = 1;\ x \leq\ m;\ ++x)\ \{$
     $if (ring_i.sp_m.subpolicy_{bool} == true) then$
       $set(ring_i.sp_m.subpolicy_{1,2,..k}); \}$
       // Targets of the ring are set by cloud customer. Moreover, if selected policies need high level sub policies, cloud customer should define based on the necessity.
5  $send\ (ring_j, RAE)$;
   $set(res_{1,2,\ldots r})$;
       // Ring analysis engine reviews the selected policies and sub polices and determines the required resources per data for policy application approximately.
6  $send\ (ring_j, SLA)$;
   $set(ring_j, value)$;
       // Based on the policies, high level sub policies and approximate resource requirements, SLA engine determines the value of ring per data.
7  $Send\ (ring_j, CC_u)$;
   $if\ ring_j.Agreement\ ==$
    $true)\ then\ Generate(SLC_j, ring_j)$;
       // If all the ring details (including policies, sub policies, billing details, etc.) are agreed by cloud customer ring is created and associated SLC is generated by PE.
```

There are two types of service requests from cloud customers according to their requirements: *Standard Ring Requests* and *Dedicated Ring Requests*. In the first case, the request is processed in PE by checking updates in PD and SLA Engine. RAE analyses the ring syntactically and semantically to validate the ring, check required high level sub-policies and estimate required resources for policy application. Algorithm 3.1. details the process of generating standard rings.

The second type of service requests includes dedicated rings to establish custom security levels based on potential security policies according to the capabilities of the service provider. Typically, the service provider updates the list of potential security algorithms in the PD according to new or revoked capabilities and allows cloud customers to choose these algorithms themselves.

The establishment of dedicated security rings may be done by RAE or cloud customers depending on different scenarios. In the first scenario, the PE receives information regarding the cloud customer, sensitivity of data and desired privacy details through an online survey-based form. According to the received data, RAE tries to match the requirements and capabilities to create a dedicated security ring. For this purpose, RAE generates 5 indexes according to the defined purposes (*i.e.* Confidentiality, Integrity, Non-Repudiation, Authorization, Authentication). The value of each index is determined by the data collected in the survey-based form (Algorithm 3.2).

Therefore, RAE matches the best algorithm of each protocol to the value of purpose indexes and sends this to the cloud customer for confirmation. Each algorithm in CSRT has a security purpose set $SPS_s(Conf_s, Int_s, NRep_s, Autho_s, Authen_s)$ that is compared with PI_i for match-making purposes.

Algorithm 3.2. Determining Values of Purpose Indexes

Input: Set Q: Let $Q = \{q_1, q_2, \dots, q_n\}$ represents all questions in survey-based form.

Output: $PI_i(Conf_i, Int_i, NRep_i, Autho_i, Authen_i)$

// The value of each purpose index is between [0,5] where 5 in the highest security value.

```
1  $PurpseIndex\ PI_i = PurposeIndex(Default);$              //An object is created from Purpose
                                                            Index set class by default value for
                                                            each index.
2  $for\ (j = 0; j \leq n;\ ++j)\{$                          //Each question of survey-based form
   $Check(q_j);$                                             is analyzed by RAE according to the
   $if(q_j == a_1) then\ Conf_i = Conf_i + Conf(a_1);$       default index policies and the value of
   $if(q_j == a_2) then\ Int_i = Int_i + Int(a_1);$          each index is determined.
   ...
   $if(q_j == a_5) then\ Authen_i = Authen_i + Authen(a_1);\}$
3  $Finalize\ (PI_i);$                                       //The value of PI set is finalized and
                                                            sent for match making processes.
```

In the second scenario, cloud customers define security levels by selecting algorithms from CSRT based on their requirements. In fact, cloud customers play the main role in this scenario and apply for dedicated rings through custom selections. For this purpose, an object is created from CSRT class within the algorithms selected by the cloud customer. The main duty of RAE in this scenario is to analyse the object, to check whether the selected algorithms are semantically matched and to find any conflicting algorithms. The following example shows unmatched and conflicting algorithms in different protocols:

```
<security:CryptographyProtocol rdf:ID="CryptographyRequirement">
   <security:SymmetricMechanism rdf:ID="SymmetricRequirement">
      <security:AESClass rdf:resource="AESClass"/>
   </security:SymmetricMechanism>
</security:CryptographyProtocol>
<security:KeyManagementProtocol rdf:ID="KeyManagementRequirement">
   <security:AsymmetricDrivation rdf:ID="AsymmetricDerivationRequirement"/>
   <security:AsymmetricWrap rdf:ID="AsymmetricWarpRequirement"/>
</security:KeyManagementProtocol>
```

According to WS-Policy code, the received CSRT object includes a symmetric cryptography resource within two asymmetric key management resources that are unmatched. RAE tries to find these unmatched algorithms in three ring analysis rounds:

1st Ring Analysis Round

In the first round, the algorithm level is checked to find conflicting algorithms (*e.g.* use AES and DES simultaneously). If RAE finds these types of conflicts, it prioritizes them by assigning $PRI \in \{1, 2, \ldots, n\}$ as a prioritize index where 1 is assigned to the highest secure algorithm (Fig 3.6A). If there are no other un-matched algorithms, all of the matched algorithms are awarded 1 as their PRI (Fig 3.6B).

2nd Ring Analysis Round

All of the algorithms with $PRI = 1$ are analyzed at this round in mechanism level. If RAE finds any conflicting algorithms, it tries to test other priorities to solve the conflict. For example, an unmatched selection is found by RAE in Authentication Level as follows:

$Normal\ Authentication\ \{Secure\ Word\ (PRI = 1), User - Pass\ (PRI = 2)\}$
$Double\ Authentication\ \{Authenticator\ (PRI = 1)\}$

The conflict between *Normal Authentication* and *Double Authentication* classes is solved by changing priorities in *Normal Authentication* Class as follows (Fig 3.6C):

$Normal\ Authentication\ \{User - Pass\ (PRI = 1), Secure\ Word\ (PRI = 2)\}$
$Double\ Authentication\ \{Authenticator\ (PRI = 1)\}$

If the conflict is not solved through reprioritization, the PRI value of unmatched algorithms is changed based on higher security (Fig 3.6D). For example:

$Symmetric\ Cryptography\ \{AES\ (PRI = 1), DES\ (PRI = 2)\}$
$Asymmetric\ Cryptography\ \{RSA\ (PRI = 1)\}$

In this example the conflict between symmetric and asymmetric mechanism are not solved by changing the priority index on AES and DES. Therefore, the PRI should be changed in mechanism level as follows:

$Symmetric\ Cryptography\ \{AES\ (PRI = 1), DES\ (PRI = 3)\}$
$Asymmetric\ Cryptography\ \{RSA\ (PRI = 2)\}$

3rd Ring Analysis Round

The third round of ring analysis includes conflicting algorithms with $PRI = 1$ in Protocol level. In fact, RAE checks whether there are not any unmatched algorithms in protocol level same as in the previous example about cryptography and key management conflict above. In that case, RAE tries to use another prioritization to solve the problem (Fig 3.6E) as follows:

$$Cryptography\ \{AES\ (PRI = 1), RSA\ (PRI = 2), DES\ (PRI = 3)\}$$
$$Key\ Management\ \{Asymmetric\ Wrap\ (PRI = 1), Asymmetric\ Derivation\ (PRI = 1)\}$$

By reprioritization the cryptography protocol is changed to:

$$Cryptography\ \{RSA\ (PRI = 1), AES\ (PRI = 2), DES\ (PRI = 3)\}$$
$$Key\ Management\ \{Asymmetric\ Wrap\ (PRI = 1), Asymmetric\ Derivation\ (PRI = 1)\}$$

If the conflict at protocol level is not solved by reprioritizing algorithms, RAE sends the conflict report to cloud customer to re-establish the CSRT object. Otherwise, all algorithms with $PRI > 1$ are eliminated from the object (Fig 3.6F) and remaining algorithms are sent to cloud customer for final confirmation and application of high-level sub-policies.

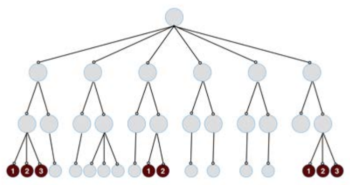


Fig. 3.6A

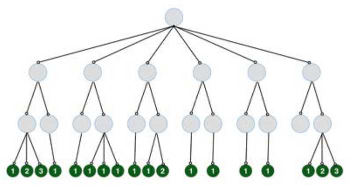


Fig. 3.6B

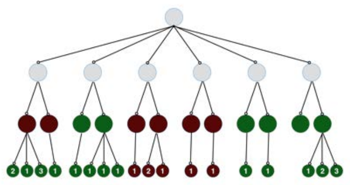


Fig. 3.6C

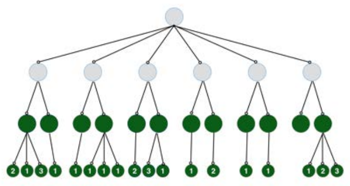


Fig. 3.6D

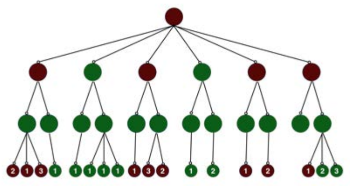


Fig. 3.6E

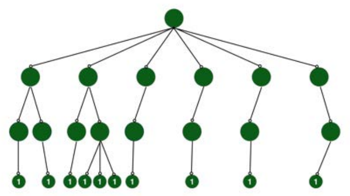


Fig. 3.6F

Fig 3.6. Performance of RAE on Dedicated Ring Establishment

3.5. Evaluation and Case Studies

A prototype of the ring establishment framework was developed to evaluate the performance of the aforementioned approach for creating standard and dedicated security levels in cloud-based environments. WS-Policy standard was used to generate a Security Level Certificate and to analyze the defined policies syntactically and semantically. Accordingly, three simple case studies are presented for selecting standard security packages and creating dedicated security levels through prioritization and matchmaking algorithms.

3.5.1. Case Study 1. (Standard Ring Establishment)

In the first case study, the provider offers 4 security packages to the cloud customer as standard rings. These standard security packages enable cloud customers to establish regular security levels based on requirements and data sensitivity. After the selection, high-level sub-policies and target cloud users are defined by cloud customers. The created CSDS object is then sent to the SLA engine and RAE to determine its value and the estimated required resources. Finally, SLC is generated following the CSDC analysis and confirmation of cloud customers. Table 3.1 details the offered security packages.

Table 3.1. Standard Security Levels offered by a Cloud Provider in Case Study 1

	Ring 1 (Public Ring)	Ring 2 (Unlisted)	Ring 3 (Sen. Unlisted)	Ring 4 (Private)
Cryptography	-	RSAClass ManualREClass	RSAClass DESClass TimeREClass	AESClass TimeREClass ManualREClass
Authentication	UserPassClass	UserPassClass	UserPassClass AuthenticatorClass	UserPassClass AuthenticatorClass
Signature	-	-	MD5	SHA
Transport	-	-	TLS	TLS
Key Manage	-	ASYWrapClass	ASYWrapClass SyWrapClass	SyWrapClass SyDerivationClass
Access Control	PublicClass	PermanentRoleClass TemporaryRoleClass	PermanentRoleClass GeoClass	OwnerClass SoftwareClass

Based on the defined packages and data sensitivity, a university in Germany (UNIGER) is defined as a cloud customer who tries to use cloud-based services with standard policy packages for data. UNIGER needs 3 levels of security: A Public Ring for all public data, without any special privacy concerns, an Unlisted Ring for certain semi-public data that can be accessible to everyone inside or outside of the university having session links or keys, and a Sensitive Unlisted Ring for high sensitive data that is only accessible by defined cloud users with special conditions and high-level sub-policies. For instance: A high level sub-policy for the permanent role class in Ring 3:

```
<rdf:HLSP_Role rdf:HLSP_Role_1="Lecturer" rdf:HLSP_Role_2="Student"
rdf:HLSP_Role_3="Professor" rdf:HLSP_Role_4="Staff" rdf:HLSP_Role_5="Admin"/>
```

a high level sub-policy for the geographical class in Ring 3:

```
<rdf:HLSP_Geo rdf:HLSP_Geo_1="Germany" rdf:HLSP_Geo_2="World"/>
```

and high level sub-policies for cryptography (encryption and re-encryption algorithms) class in Ring 3:

```
<rdf:HLSP_DES rdf:HLSP_DES_KeySize="256" rdf:HLSP_DES_Mode="CBC"/>
<rdf:HLSP_RSA rdf:HLSP_RSA_KeySize="1024"/>
<rdf:HLSP_MRE rdf:HLSP_TIME_1="1" rdf:HLSP_Time_2="7" rdf:HLSP_Time_3="30"/>
```

The process of defining low-level sub-policies (LLSP) is performed by cloud users in the run-time after the ring establishment. In this case study, UNIGER establishes a security level with 5 permanent roles and 2 geographical roles as HLSP, cloud users being able to assign one or more of these policies as LLSP during the run-time for each data. After defining HLSPs, the SLA engine calculates the value of these services based on HLSPs (for example cryptography key size) and RAE estimates the required resources security ring. The SLC is generated in the final step, after the confirmation of the cloud customer based on the SLA details.

3.5.2. Case Study 2. (PE Dedicated Ring Establishment)

The second case study evaluated the performance of the RAE when matching capabilities and requirements from a survey-based form. Accordingly, a survey-based questionnaire was designed to collect requirements from cloud customers and to determine the value of the security purposes for the matching process. In this case study, the match levels are defined as: Perfect Match, Close Match, Possible Match and No Match, going from the highest to the lowest match between the requirements, the value of security purposes and the algorithms offered. Figure 3.7 shows the process of matching in each of the security protocols according to CSRT, as well as the overall CSRT process of matching based on the value of the security purposes.

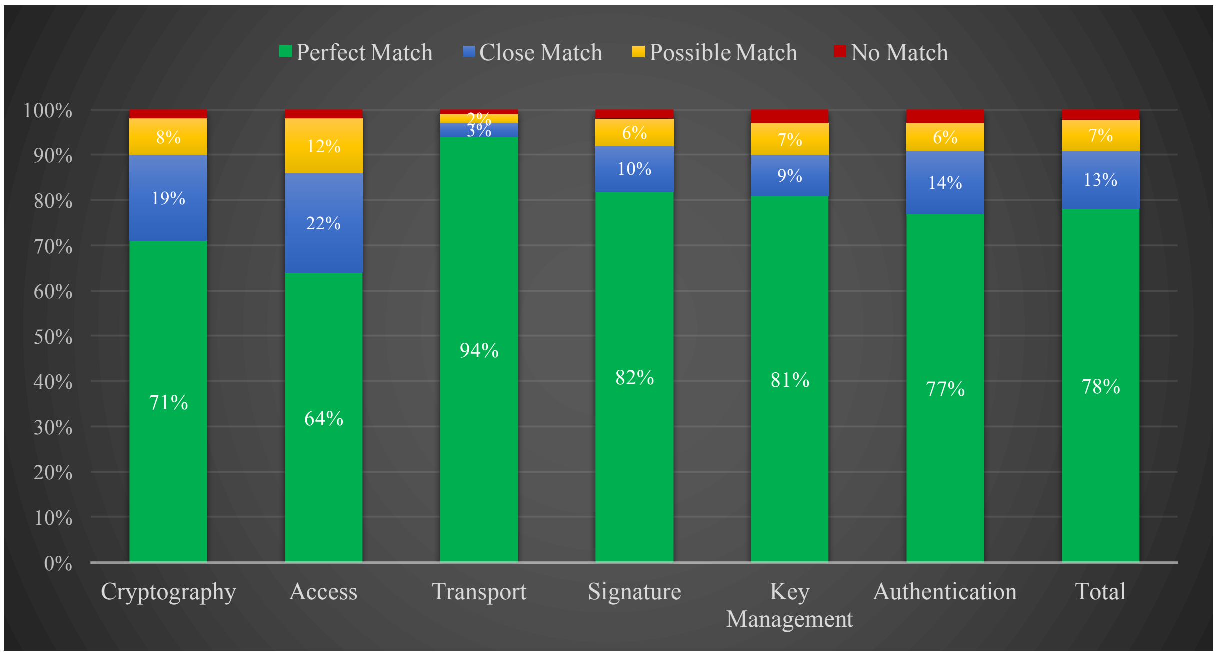


Fig. 3.7. Match Results between Requirements, Value of Security Purposes and Algorithms Offered

According to the results, 78% of the requirements were matched to the most suitable security algorithm offered in CSRT. The minimum value of the Perfect match is related to the access management protocol due to the variety of algorithms and the concepts of reputation and role-based access control models.

However, there are 34% close or possible matches in the access protocol that can be used based on the requirements. As expected, the transport protocol receives the highest value of perfect matches due to the popularity of the SSL and TLS mechanisms, while the overall "No Match" results contain only 2% of the total matches.

3.5.3. Case Study 3 (Dedicated Ring Establishment with Ring Analysis Rounds)

In the third case study, the performance of RAE during the ring analysis rounds was evaluated. For this purpose, 200 objects were created from CSRT based on random and knowledge-based selections from security algorithms and transferred to RAE for analysis processes. The reason for the random selection was to evaluate the syntactic and semantic performance of RAE during the ring analysis rounds. The aim was to find out how efficient this analysis was, on a scale from basic errors of a random selection (*i.e.* without any knowledge) to a reliable security level establishment. Figure 3.8 shows the performance of RAE in the ring analysis rounds.

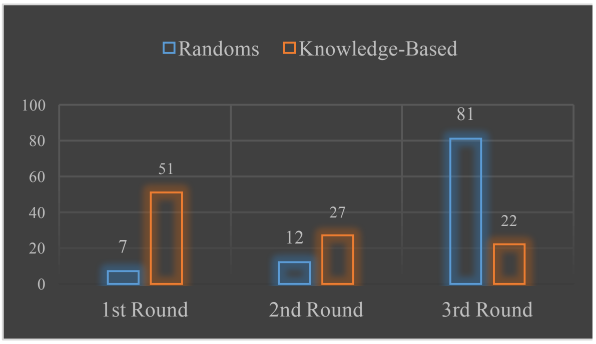


Fig 3.8A. Analysis Round Required for the Ring Establishment Confirmation from RAE

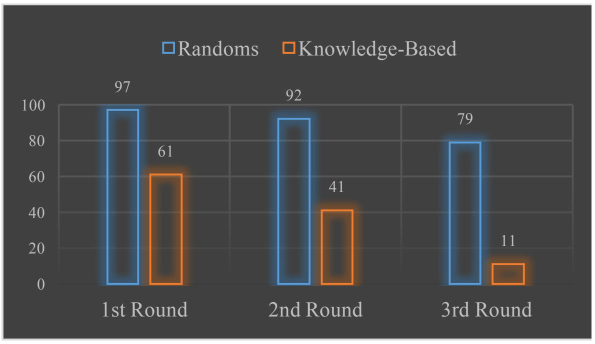


Fig 3.8B. Conflicts Found in Each Round of the Ring Analysis Process

Fig 3.8. Performance of RAE in the Ring Analysis Rounds

According to Figure 3.8A, 81% of the random objects needed all three analysis rounds to find conflicts. In fact, conflicts were found in 97%, 92% and 79% of the random objects in the analysis rounds 1 to 3 respectively (Figure 3.8B).

A total of 93% of the random objects were settled by RAE and only 7% of the random objects were send to PE for further changes. As expected, 51% of the knowledge-based objects needed only the first round of evaluation for improvement, with only 11% of these objects having errors in the final round and 2% being sent to PE for further communication with the cloud customer. Overall, the results show that RAE could find and fix conflicts in 95% of the random or knowledge-based objects without any extra negotiation between cloud customers and the policy engine.

3.5.4. Comparison between Similar Models and Ontologies

Table 3.2 shows the advantages of the suggested model in comparison with similar policy management models. The most important characteristic of the proposed model is providing simultaneous syntactic and semantic analysis in order to establish standard or dedicated security levels through reliable mapping between the requirements of cloud customers and the capabilities of the cloud provider.

Table 3.2. Comparison between CSON and other Policy Management Models

	IETF	Ponder	KaoS	Rei	Modica	CSON
Language-Based	Yes	Yes	Yes	Yes	Yes	Yes
Object-Oriented	No	Yes	No	No	No	Yes
Syntactic & Semantic Analysis	No	No	No	No	Yes	Yes
Security Level Certificate (SLC)	No	No	No	No	No	Yes
Ring Establishment	No	No	No	No	No	Yes
Standard and Dedicated Rings	No	No	No	No	No	Yes
Ring Analysis	Yes	Yes	No	Yes	Yes	Yes
WS-Policy & XML	No	Yes	Yes	No	Yes	Yes
Multi-Level Sub-Policies	No	No	No	No	No	Yes
Cloud Security Rooted Tree (CSRT)	No	No	No	No	No	Yes
Cloud Security Distributary Set	No	No	No	No	No	Yes

3.6. Conclusion

In this chapter, a structural policy management engine was introduced to enhance the reliability of managing different policies in clouds and to provide standard as well as dedicated security levels (rings) based on the capabilities of the cloud provider and the requirements of cloud customers.

Cloud security ontology provides a special cloud-based security structure to establish standard and dedicated security levels in a cloud computing environment through syntactic and semantic analysis of the security requirements and capabilities. Accordingly, CSRT and CSDS were designed as two super classes to map requests and offers, as well as to establish appropriate communications between the Policy Engine, the Policy Database and the SLA Engine. Furthermore, RAE as a ring analysis component justifies the security levels and evaluates the process of mapping syntactically and semantically. SLC as the output of this model defines standard or dedicated security levels for various cloud customers.

Despite the considerable benefits of the proposed model, the complexity of various policies is a challenging issue that should be solved by layering mechanisms of defined policies. In the next chapter, a layering procedure has been added to the defined ontology to enhance the process of matching between capabilities and requirements, to improve the process of applying policies to data and resources, and to establish an efficient mapping process between access requests and defined policies in cloud-based environments.

Chapter 4

4. MULTI-LAYERED POLICY APPLICATION

4.1.Introduction

As discussed before, the most common approach to express high-level security constraints is based on the the usage of languages and metadata for the specification of security policies. By these language-based approaches, cloud providers are able to announce and provide security capabilities as well as matching these capabilities to customers' requirements [18]. In fact, policies are defined according to syntactic and also semantic matchmaking of requirements and capabilities to ensure the security of nodes and to map access requests to defined policies efficiently. The challenging issue in this case is to manage applied policies in different virtualized nodes considering scalability, isolation and elasticity concepts in cloud computing environments [95].

The approach we propose is based on semantic policy clustering to classify nodes with same or common security policies in an aggregate virtualized cluster for federating defined policies according to their characteristics. By this federation, the processing time for each policy mapping is reduced due to the elimination of gratuitous and avoidable matching jobs. We define a well-established policy framework to define security policies which are compliant to requirements and capabilities. Moreover, a federated policy management schema is introduced based on the policy definition framework and policy clustering to create and manage virtual clusters with identical or common security levels. The proposed model consists in the design of a well-established ontology according to security mechanisms, a procedure which classifies nodes with common policies into virtual clusters, a policy engine to enhance the process of mapping requests to specific node as well as associated cluster and matchmaker engine to eliminate inessential mapping processes.

4.2.Related Works

The adoption of policy-based multi-level security management in clouds needs an appropriate and structural ontology to represent, generate, apply and manage policies according to core concepts of virtualized data centers [19]. Several policy management engines and frameworks have been proposed following different approaches in various application domains. These models have typically focused only in access policies [20] include privacy and user-based concepts or Federated Security Policies (FSP) contain multiple security concepts such as encryption, access control, authentication and transport. One of the most popular web-based semantic languages has been used in several policy-based schemas is Web Ontology Language (OWL) [21]. A complex knowledge and relations between requirements, capabilities and constraints were represented in OWL [22] with an efficient reasoning support, sufficient expressive power and convenience of expression [23]. Also, several OWL-Based subsidiaries were presented to improve the restricted expressivity of OWL. OWL-L [24], OWL-S [25] and F-OWL [26] are some of OWL-based models there have been introduced to provide an independent domain policy specifications based on deontic constructs and to allow several types of policies such as right, prohibition, dispensations and obligations. Furthermore, to improve the problem of allowing customers of classic multi-agents and distributed networks to define different security policies as well as predictability and controllability assurance of each components, KAoS [27] was proposed as an OWL-based policy language model. In KAoS, policies are classified to four main categories: *Positive-Authorization*, *Negative-Authorization*, *Positive-Obligation* and *Negative-Obligation* that are associated with service properties (*i.e.* policy services such as policy enforcement and domain services such as hierarchical grouping of users) [28]. However, the main issue of OWL and its subsidiaries is the compatibility of these models to modern distributed networks specially in elasticity and isolation concepts.

Ponder [29] is another policy management framework based on object oriented concepts that was developed by imperial college. This framework includes general architecture and policy deployment model associated with several extensions for both access control and protection managements. It also allows customers to generate events, constraints, constants and other reusable elements as associated parts of security policies as well as allowing instantiation of typed policy specification for parameterization support of policies. The most important disadvantages of Ponder is the lack of generality by using several basic policy types and compositing each of them with different syntax. Accordingly, several Ponder-based deployment models were introduced [21,22] to address the instantiation, distribution and enabling of policies and also disabling, unloading and deletion of policies.

Web Service Policy Framework (WS-Policy) [32] is an extendable general purpose framework that has been recommended by World Wide Web Consortium (W3C). An associated syntax was defined in this framework to describe the policies of entities with a broad range of service capabilities and requirements in web-based models. WS-Policy involves several subsets according to the different structures of service domains. For instance, WS-Trust [33] was defined to change security tokens into different formats by an interoperable manner in order to establish and assess the presence of participants in secure message exchange. Also, WS-SecurityPolicy [34] and WS-SecureConversation [35] were defined to describe the security specifications of WS-Trust by improving the performance of frequent communications and using a shared symmetric and a pair of asymmetric keys from the security context respectively. The other popular WS and XML-based schema to describe the policy, request the authorization decision and respond with the authorization decision is XACML (Extensible Access Control Markup Language) [36]. This standard defines a policy enforcement point for interacting with a policy decision point. The comparison between WS-based and XACML standards have been extended in [17].

WS-Policy and associated subsets have been extended by several researchers to be used in policy engines of multi-domain services. In overall, WS standards have been used in policy engines to provide QoS assertion models which are generic, domain-independent and expressible across different layers and service roles [37]. One of the WS-based semantic models was proposed in [38] for generating security policies specifically for cloud computing environments to a enable a flexible and powerful matchmaking process between customers and providers security requirements. This model uses several terms and concepts to model security features within a policy as well as providing compatibility of semantic framework with syntactic polices. Furthermore, using an automated negotiation framework [39] based on WS-Policy is another solution to support participant security policies for communication, negotiation and SLA creation.

To enable more satisfactory discovery results that better fit the requirements of cloud customers, WS-Policy was extended through an efficient ontology and rule reasoning [40]. Hence, a set of rules was defined security policies associated a developed rule-based engine improve policy evaluation and policy mapping. The drawback of this rule-based policy management was the considerable processing time for taking the overheads of policy representation with an ontology language and for transferring them to a rule-based structure.

According to the most prominent characteristic of cloud-based models for providing on-demand services, semantic rules have been used during the establishment of policy framework for an efficient and reliable policy generation and mapping [41]. Several works such as LaaS (Law-as-a-Service) [42] were proposed for cloud service providers on law-aware semantic cloud policy infrastructure to deploy their cloud resources and services based on OWL ontologies and stratified Datalog rules with negation for policy exceptions. Also, PEaaS (Policy Engine-as-a-Service) [43] was suggested based on WS-Policy to provide multi-level policies in clouds according to Protection ontology. These policies create create standard or dedicated security rings (*i.e.* levels) regarding to the capabilities of service providers, constraints and requirements of cloud customers.

However, the heterogeneous characteristics of these services together with the dynamicity inherent in clouds, hinders the formulation of an effective and interoperable set of policies that is adoptable for the underlying domain of applications [44]. Hence, establishment of ontological templates for the semantic representation of security policies is needed to facilitate the definition of appropriate security policies using a generic and extensible RDF [45]. Therefore, policies should analyzed syntactically and also semantically in different service layers and service roles to express capabilities and requirements based on SLA and to complement the existing service selection and negotiation framework [37] The most challenging concern in simultaneous syntactic and semantic mapping is the flexibility of the generated policies according to the predictable and un-predictable variations of capabilities, constraints and requirements [46].

The other challenging issue in configuration of security policies is the possibility of confliction in the course of unexpected occurs for policy management and enforcement. Indeed, the structure of policy generation framework needs to be reliable and efficient enough to provide accurate detection of conflict between policies and the security of cloud digestion [47]. This concept has been more extended in federated cloud networks to rely on a service manifest that specifies global polices [48]. These extended frameworks enable automated deployment and configuration of network security functions across different clouds. However, the complexity of these federated frameworks is still challenging according to multi-structure policies in virtualized data centers [41,42].

Overall, the main concerns regarding to semantic policy-based security management models is: (1) to provide efficient, secure and reliable matching between security requirements of customers and capabilities of service providers according to interoperability and isolation concepts in cloud-based environments [51], and (2) to ensure the scalability and flexibility of mapping due to the large number of defined polices and access requests [52]. Accordingly, a multi-layered federated policy-based resource classification framework has been presented in this paper to classify and manage security levels in clouds and to provide efficient mapping between requests and policies.

4.3. Multi-Layered Policy Generation

A multi-layered security policy is built around two main concepts: *Foundation* and *Structure*. The *Foundation* of each policy includes potential security protocols, mechanisms and algorithms that are used to establish dedicated security levels according to the capabilities of service provider and requirements of cloud customers, as well as, the *Structure* of policies is the way to define and manage policies and sub-policies according to the *Foundation*.

4.3.1. Policy Foundation (Multi-Level Ontology)

The foundation of security policies is based on a structural ontology to provide efficient mapping between capabilities of the service provider, sensitivity of resources and requirements of cloud customers. In fact, this ontology is established to describe security-related concepts (*e.g.* protocols, mechanism, algorithm, credential, *etc.*) and to define the characterization of policies in terms of requirements and capabilities according to object oriented and inheritance concepts. Multi-Level Ontology (MLO) is the proposed ontology based on WS-Policy [32] that covers core concepts of security domain to address security aspects at high levels [41]. MLO defines the concepts of security policy in the context of suggested model by two main super classes: *Protocol* and *Basis*.

The *Protocol* super class is based on main protocols of security levels (*i.e.* authentication, access control, transport, encryption and key-management) to define policies for classification of virtualized resources in cloud-based environments. All protocols are offered by cloud services providers regarding to the three levels of policy matrix super class in Protection Ontology [43]: Protocols, Mechanisms and Algorithms. Furthermore, each protocol is associated with several defined properties as sub-polices (*e.g.* key-size for encryption protocol, authenticator emails for two-factor authentication protocol and authorized locations for geo-based access control protocol).

According to the capabilities of service provider, protocols are defined and also updated by adding or revocation of algorithms in policy matrix super class. MLO uses *hasProtocol*, *hasPolicy* and *hasSubPolicy* to define security policies structurally (Figure 4.1) and the created objects from the *Protocol* super class use WS-Policy formalization for description.

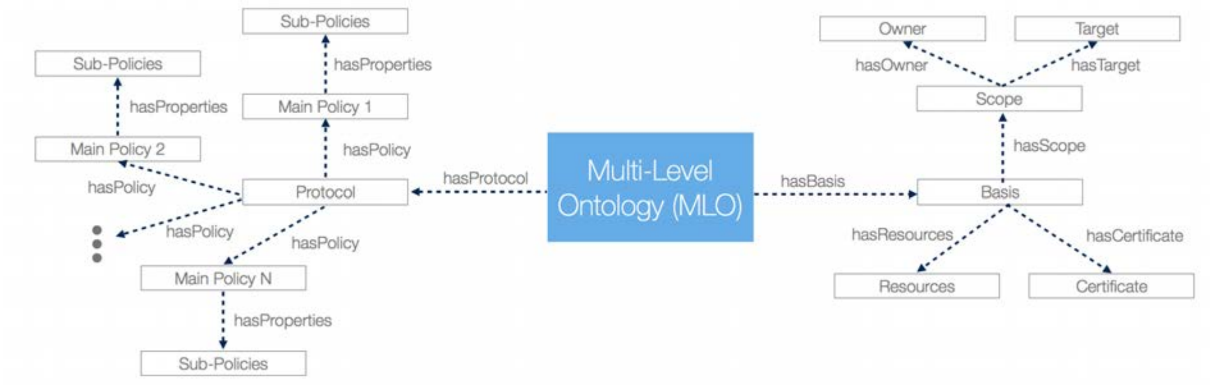


Fig. 4.1. Multi-Level Ontology: Top Level Class

In the following an access control protocol according to role and geo-based mechanisms and associated sub-policies has been defined:

```
<security:AccessControlPolicy rdf:ID="AccessControlRequirement">
   <security:RolePolicy rdf:ID="PrRoleClass">
      <security:PermanentRole rdf:resource="Permanent">
         <rdf:PR_Role rdf:PR_Role_1="Lecturer"
                      rdf:PR_Role_2="Student"
                      rdf:PR_Role_3="Admin"
                      rdf:PR_Role_4="HRStaff"
                      rdf:PR_Role_5="Manager"/>
      </security:PermanentRole>
   </security:RolePolicy>

   <security:ReputationPolicy rdf:ID=RepRequirement">
      <security:GeoRep rdf:resource="GeoRepClass">
         <rdf:REP_Geo rdf:REP_Geo_1="US"
                      rdf:REP_Geo_2="Canada"/>
       </security:GeoRep>
   </security:ReoutationPolicy>
</security:AccessControlPolicy>
```

This access policy is defined by a permanent role-based policy with 5 accepted properties (*i.e.* sub-policies) associated with a geographical reputation policy with two target locations. Each of policies is mapped to a semantic resource to enable reliable matchmaking (Figure 4.2).

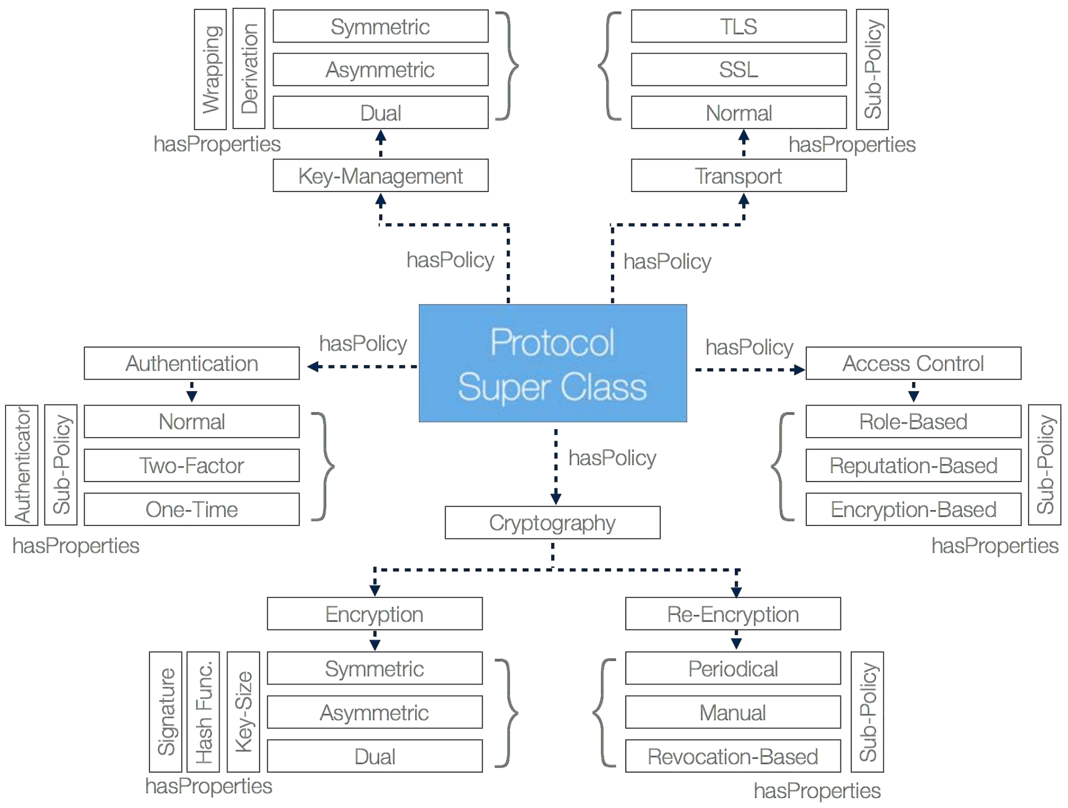


Fig 4.2: Protocol Super Class

The basis of multi-level ontology is established around essential information of security level according to defined policies. *Basis Scope* involves general notion for the owner of dedicated level (*i.e.* cloud customer: An organization or a company that uses cloud services for employments or subscribers) and potential targets (*i.e.* cloud users: Defined end-users that use cloud-based services offered by cloud customers according to the internal contracts).

Basis Resources properties, define required information about intransitive physical and virtual resources in order to apply defined policies to data (*e.g.* encryption) and to retrieve data based on access requests (*e.g.* decryption). Finally, *MLO Certificate* is the validation property of the policy for checking the reputation of policy during policy application and policy mapping processes.

4.3.2. Policy Structure

Policy Structure aims to minimize the processing time of applying policies and mapping them to access requests and to enhance the reliability of managing security levels according to the defined details of *Policy Foundation*. In fact, a structural framework is introduced for layering policies based on defined *Protocols* and associated *Basis* to increase the reliability and efficiency of policy application, management and mapping. The detailed mathematical description of the problem is given as follows:

Assumes that there are N cloud nodes in the data center, denoted as $\{S_1, S_2, \dots, S_N\}$, and the defined policy capability set of node S_α with $\alpha \in \{1,2,\dots,N\}$ is $PCS(S_\alpha) = \{C_1, C_2, \dots, C_X\}$ and all capabilities are associated with a capability set as follows: $\forall C_x\ (x \in [1, X])$: $C_x = \{C_{type}, C_{res}, C_{cert}, sp_1, sp_2, \dots\}$ where $C_{type}, C_{res}, C_{cert}$ and sp are protocols (*i.e.* authentication, access control, cryptography, *etc.*), algorithm semantic resource, protocol certificate and sub-policies respectively. Given M policy application tasks waiting to be processed, denoted as $\{PA_1, PA_2, \dots PA_M\}$ and each PA_m is mapped to the defined multi-level ontology set, $MLO(PA_m) = \{cert, scope, res, p_1, p_2, \dots, p_U\}$ with several security policies where $cert, scope$ and res are associated certificate, policy scope and estimated resources respectively. Other notations used in the proposed model are shown in Table 4.1.

Table 4.1. The Notations of Proposed Model

Notations	Description	Notations	Description
S	Cloud Node (Server)	PCS	Policy Capability Set
C	Capability of Node	C_{type}	Capability Protocol Types
C_{res}	Semantic Resource	C_{cert}	Protocol Certificate
sp	Sub-Policy	PA	Policy Application Task
MLO	Multi-Level Ontology Set	$cert$	Policy Certificate
$scope$	Policy Scope	res	Estimated Resources for Application
VC	Virtual Cluster	$MapFunction$	Mapping Application Tasks
$VCClass$	Virtual Cluster Class	APP	Approval Flag
$TokenClass$	Token Class	AT	Access Token

To process each policy application request, all of the defined policies of data need to be checked and mapped to the most compatible cloud node efficiently:

$$\begin{aligned} Process(PA_m) &= MapFunction\big(MLO(PA_m), PCS(S_\alpha)\big) \\ &= \left(\sum_{j=1}^{X}\left(\sum_{i=1}^{U} MapFunction\left((PA_m.p_i), (S_\alpha.C_j)\right)\right)\right) \end{aligned}$$

The main objective of layering policies is to minimize the processing time of policy application requests regarding to the details of *Policy Foundation*.

$$\begin{aligned}\min Process(\{PA_1, PA_2, \dots PA_M\}) &= \min \sum_{z=1}^{M} MapFunction\big(MLO(PA_z), PCS(S_\alpha)\big) \\ &= \min\left(\sum_{z=1}^{M}\left(\sum_{j=1}^{X}\left(\sum_{i=1}^{U} MapFunction\left((PA_z.p_i),(S_\alpha.C_j)\right)\right)\right)\right)\end{aligned}$$

The most appropriate solution for minimizing the processing time is to define multi-layered structure as well as distributing policy application tasks according to the capabilities of cloud nodes. Accordingly, a Policy Layer Constructor (PLC) is defined to create multi-layered policies from created objects of MLO. The main duty of PLC is to classify and create layers from each policy object according to associated protocols and sub-policies.

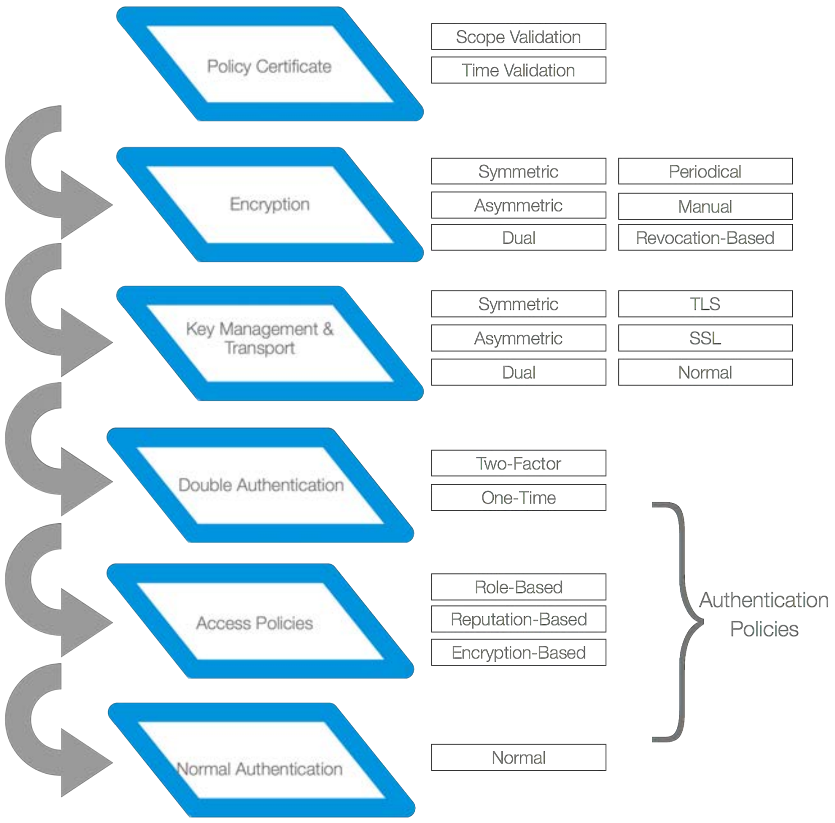


Fig 4.3. Multi-Layered Policy Structure

Figure 4.3 shows the structure of layering policies based on associated protocols and algorithms. The certificate and scope of the policy are settled in the highest level policy-layering, as well as, time-validation. Furthermore, protocols and algorithms in each policy are classified based of the estimated required resources and processing time in MLO object. Typically, encryption protocols are in the highest level and normal authentication algorithms are settled in the lowest level of policy application.

The combination of *Policy Foundation* and *Policy Structure* helps to define policies efficiently based on data sensitivity and requirements of cloud customers, as well as, enhancing the process of applying policies to data according to the security capabilities of cloud nodes and settled protocols in created MLO objects.

Algorithm 4.1 shows the performance of PLC to create layers for each policy according to the policy certificate and defined protocols.

Algorithm 4.1. Layering Policy Application Requests

Input: $\{PA_1, PA_2, \dots PA_M\}$: Set of Policy Application Request Tasks.
Output: $\{PA'_1, PA'_2, \dots PA'_M\}$: Set of Layered Policy Application Request Tasks.

```
1:    $for\ m = 1,2,\dots,M\ \{$
        $Process\ (PA_m);$
2:    $if\left(\begin{matrix}(MLO(PA_m).cert.(time \wedge scope) = valid) \wedge \\ (\prod_{i=1}^{U}(MLO(PA_m).p_t.cert) = valid)\end{matrix}\right)\ then\ (PA'_m = PA_m);$
      $add(PA'_m);$
      $PA'_m.Layer(Certification) = newLayer(PA'_m.cert, PA'_m.scope)$
3:    $PA'_m.Layer(Encryption) = newLayer(PA'_m.EncLayer);$
      $forall\ (P_u \in PA'_m.\{p_1, p_2, \dots, p_U\})$
       $if\ (p_u.type = Encrpytion$
                         $\vee\ ReEncryption), then\ add(p_u, PA'_m.EncLayer);$
4:    $PA'_m.Layer(Key/Trans) = newLayer(PA'_m.Key/TransLayer);$
      $forall\ (P_u \in PA'_m.\{p_1, p_2, \dots, p_U\})$
      $if\ (p_u.type = KeyManage$
                         $\vee\ Transport), then\ add(p_u, PA'_m.Key/TransLayer);$
5:    $PA'_m.Layer(DoubleAuth) = newLayer(PA'_m.DAuthLayer);$
      $forall\ (P_u \in PA'_m.\{p_1, p_2, \dots, p_U\})$
      $if\ (p_u.type$
      $= DoubleAuthentication), then\ add(p_u, PA'_m.DAuthLayer);$
6:    $PA'_m.Layer(AccessManagement) = newLayer(PA'_m.AccessLayer);$
      $forall\ (P_u \in PA'_m.\{p_1, p_2, \dots, p_U\})$
      $if\ (p_u.type = AccessManagement), then\ add(p_u, PA'_m.AccessLayer);$
7:    $PA'_m.Layer(NormalAuth) = newLayer(PA'_m.AuthLayer);$
      $forall\ (P_u \in PA'_m.\{p_1, p_2, \dots, p_U\})$
      $if\ (p_u.type = Authenticaion), then\ add(p_u, PA'_m.AuthLayer);\}$
8:    $Send\ (\{PA'_1, PA'_2, \dots, PA'_M\}, Classifier);$
```

In the first step all of the policy certification flags (*i.e.* validation, scope and timestamps) for each policy application request task are checked. The invalid or expired policy certificates are removed from the queue for re-considering and updating processes. Following this elimination, a certification layer object is created from certification class for all remaining tasks. The rest of policy layering process is based on policy types. Encryption and re-encryption policies take the deepest layers of policies, as well as normal authentication policies are settled in the highest layer according to the policy structure and foundation concepts. The process of policy definition and layering (Figure 4.4) is done based on four main components as follows:

- *Policy Engine*: To define security polices according to capabilities of service provider and requirements of cloud customers.

- *Policy Database*: To store defined polices and security level certificates.
- *Policy Layer Constructor*: To create multi-layered polices based on policy types and certificates.

- *Validity Engine*: To check and update invalid or expired policies application request tasks according to the certification details.

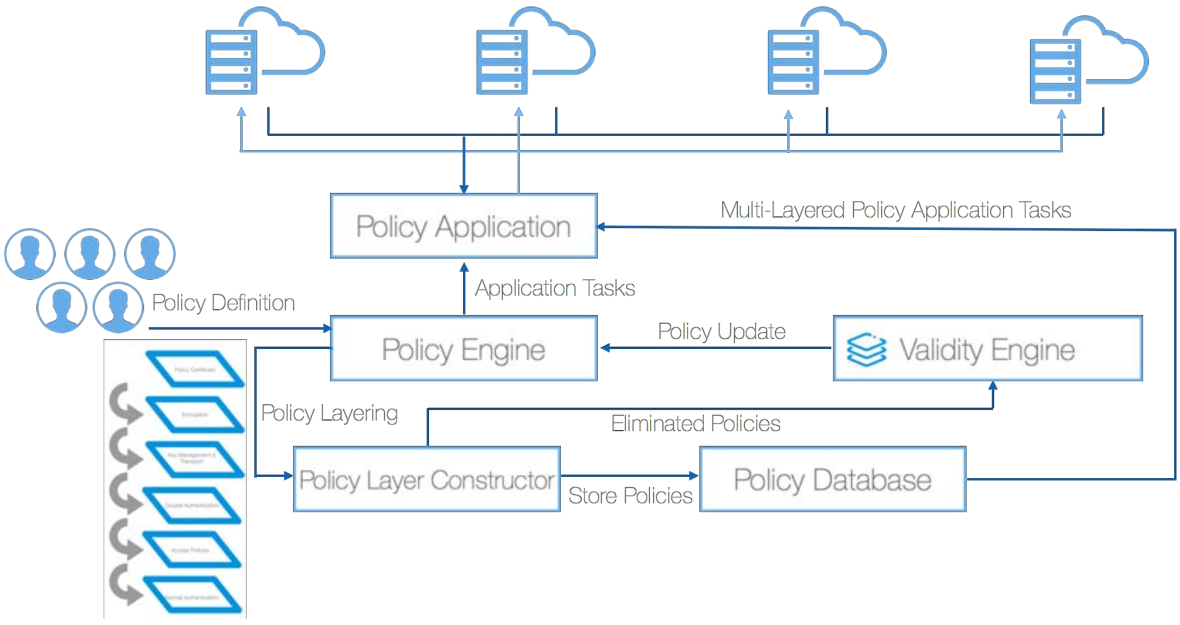


Fig 4.4. Process of Policy Layering

Security policies are defined by administrators or cloud customers in Policy Engine. After policy definition, an object is created from PFO based on *Basis* and *Protocol* super classes. The defined policy is sent to *Policy Layer Constructor* for classification and layer creation. Following the layering process, the multi-layered policy objects and associated sub-policies are stored in *Policy Database*. In addition, the eliminated polices are sent to *Validity Engine* for updating and sent back to *Policy Engine* for policy layering after successful update process.

4.4. Policy Application

The process of policy application is based the security capabilities of cloud nodes and efficient distribution of multi-layered policy application tasks on the most appropriate cloud nodes. To achieve this purpose, multi-level virtual clusters are defined to classify cloud nodes according to the security capabilities.

Let $VC = \{VC_1, VC_2, \dots VC_{\pi+\delta}\}$ represents all virtual clusters with common security capabilities, the former π items of which are the first level clusters and the rest δ items are low level clusters. Clusters in the lowest level contains nodes with same capabilities and in upper levels are mixed from cloud nodes and lower level clusters. Figure 4.5 shows a four-level virtual cluster rooted tree in detail.

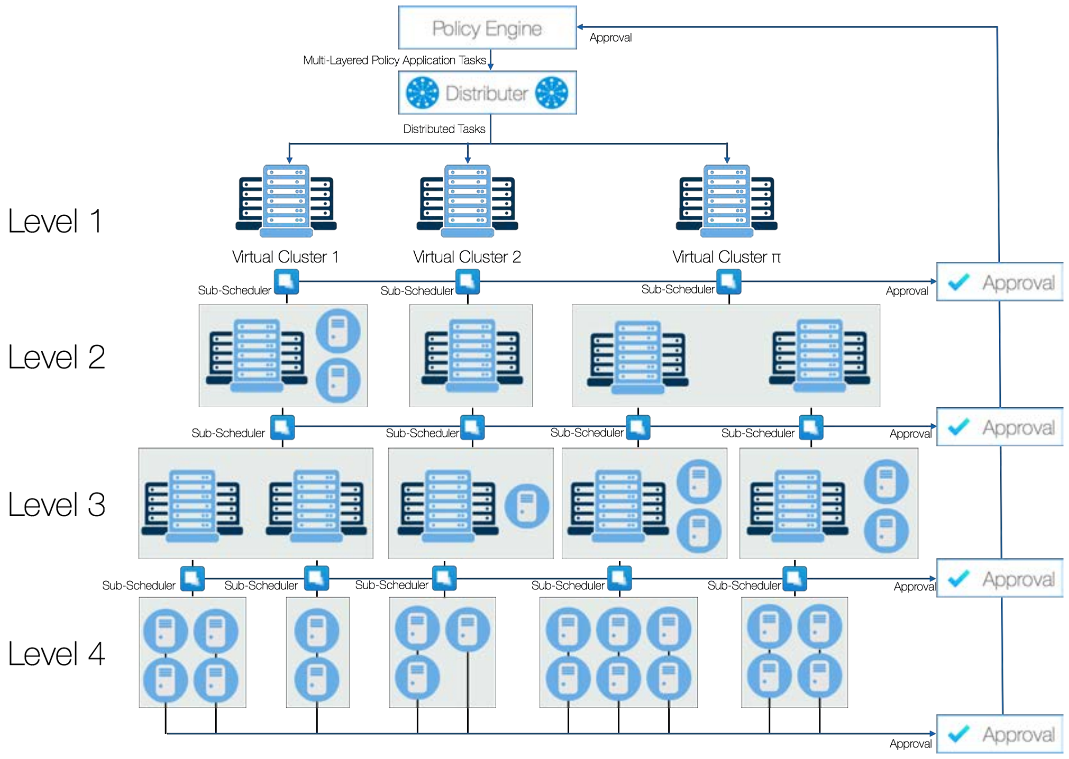


Fig 4.5. Multi-Level Clustering based on Security Capabilities

To categorize cloud nodes according to the common capabilities a classification process is introduced to create the first level of clustering by several virtual clusters with common security capabilities (Process 1 to 3 in Algorithm 4.2).

In fact, each VC in the highest level contains cloud nodes with common security protocols according to Multi-Layer Ontology. Thus, the output of the first step is $\{VC_1, VC_2, \dots VC_\pi\}$.

The second step of classification phase is based on security algorithms. Each created clusters in the first level is evaluated to classify cloud nodes according to the defined algorithms in MLO (Process 4 to 6 in Algorithm 4.2). From the third step, cloud nodes are classified based on sub-policies (Process 7 to 9). This process is continued until it reaches to the lowest level of classification architecture (Process N).

Algorithm 4.2. Cloud Nodes Classification Phase

Input: $\{S_1, S_2, \ldots S_N\}$: Total Cloud Nodes.
Output: $\{VC_1, VC_2, \ldots VC_{\pi+\delta}\}$: Virtual Cluster Set.

```
1:    $for\ m = 1,2,\ldots,N;$
          $for\ \forall\ C_n\ \in PCS(S_m)$ where $n = 1,2,\ldots,X;$
2:          $if\ \exists VC_j : \forall\ C_n : \sum C_n.C_{type}$ is matched,
              $then, send(S_m) to\ VC_j;$
3:    $else, if\ \exists C_n : C_n.C_{type}\ \notin \{VC_1, VC_2, \ldots, VC_{\pi-f}\},$
              $then, VCClass\ VC_\varphi = New\ VCClass(\sum C_n.C_{type});$
4:    $for\ x = 1,2,\ldots,\pi;$
          $for\ \forall\ S_\vartheta : S_\vartheta \in VC_x;$
            $for\ \forall\ C_n\ \in PCS(S_\vartheta)$ where $n = 1,2,\ldots,N;$
5:            $if\ \exists VC_j : \forall\ C_n : \sum C_n.C_{alg}$ is matched,
              $then, send(S_\vartheta) to\ VC_\gamma;$
6:            $else, if\ \exists C_n : C_n.C_{alg}\ \notin \{VC_1, VC_2, \ldots, VC_{\pi+f'}\},$
              $then, VCClass\ VC_\varphi = New\ VCClass(\sum C_n.C_{alg});$
7:    $for\ x = 1,2,\ldots,\pi;$
          $for\ \forall\ VC_\varepsilon : VC_\varepsilon \in VC_x;$
           $for\ \forall\ S_\vartheta : S_\vartheta \in VC_\varepsilon;$
            $for\ \forall\ C_n\ \in PCS(S_\vartheta)$ where $n = 1,2,\ldots,N;$
8:            $if\ \exists VC_\varepsilon : \forall\ C_n : \sum C_n.sp_\alpha$ is matched,
              $then, send(S_\vartheta) to\ VC_\sigma;$
9:            $else, if\ \exists C_n : C_n.sp_\alpha\ \notin \{VC_1, VC_2, \ldots, VC_{\pi+f''}\},$
              $then, VCClass\ VC_\varphi = New\ VCClass(\sum C_n.sp_\alpha);$
⋮         Creation of new levels until reaching to the lowest level
$N$:      $VC = \{VC_1, VC_2, \ldots VC_{\pi+\delta}\};$
```

The proposed model uses a *Distributer* component on the top level and several *Sub-Schedulers* in each virtual clusters. The process of applying multi-layered security policies to data is done as follows:

Step 1: *Policy Engine* receives the updated layered policy application request tasks from *Layer Constructor* and *Policy Database*: $\{PA'_1, PA'_2, \ldots PA'_M\}$ and sends to the *Distributer* component.

Step 2: The main duty of *Distributer* is to classify policy application tasks to the most appropriate high level cluster according to the capabilities of the cluster. Each virtual cluster is associated with a capability set $C(VC_z) = \{C_1, C_2, \ldots, C_\gamma\}$ where $z \in \{1,2,\ldots,\pi+\delta\}$ and $\gamma = (\sum_1^i PCS(S_i) + \sum_1^j C(VC_j))$ includes all of the capabilities of sub-clusters and sub-nodes. The *Distributer* compares the multi-level ontology set of policies with the capability set of high-level clusters:

$$Compare\big(MLO(PA'_m), C(VC_z)\big) = Compare\Big(\big(\forall p_u \in MLO(PA'_m)\big), \big(\forall C_u \in C(VC_z)\big)\Big)$$

The policy application task is assigned to the high-level virtual cluster with all of the required capabilities. If there are several options, the task is sent to the cluster with lower workload. Furthermore, if there is not any cluster with all of the capabilities, the task is assigned to a cluster with most common capabilities and policies.

Step 3: Each virtual cluster has a Sub-Scheduler component to manage policy application tasks and classify them according to the capabilities of associated clusters and sub-nodes. The algorithm of scheduling application tasks is as follows:

Step 3.1: A classified policy application task set $\{PA'_1, PA'_2, \dots PA'_D\}$ is received form the *Distributer* or higher *Sub-Scheduler* where $1 < D < M$.

Step 3.2: The *Sub-Scheduler* compares the multi-level ontology set of policies with the capability set of subset clusters or sub-nodes according to security algorithms:

$$\forall\, PA'_d \in \{PA'_1, PA'_2, \dots PA'_D\}:$$
$$Compare\big(MLO(PA'_d), \{C(VC_{1,2,\dots}), PCS(S_{1,2,\dots})\}\big) =$$
$$Compare\Big(\big(\forall p_u \in MLO(PA'_d)\big), \big(\forall C_u \in \{C(VC_{1,2,\dots}), PCS(S_{1,2,\dots})\}\big)\Big)$$

The policy application task is assigned to the most appropriate sub-cluster or sub-node with all of the required capabilities. If the task is assigned to the most appropriate cloud node, the process of mapping defined policy is performed (Algorithm 4.3).

According to this algorithm, each policy is applied to the data according to several layers. In the first step the policy application is processed in the cloud node as well as calling all associated components in the next step. Step 3 re-checks the policy certification for validating the request. From step 4 to 8, *Normal Authentication*, *Access Commands*, *Double Authentication*, *Key Management* and *Encryption* policies are applied respectively according to layers of policy.

If the process of mapping fails in each step, the loop is broken and the task is sent back to the *Policy Engine* for updating as well as eliminating the task from *Distributer* set. Also, if all of the defined policies are applied successfully, the confirmation flag is sent to the *Distributer*.

Step 3.3: If there is not appropriate cloud node for the application task, the task is sent to the most appropriate virtual cluster in the lower level.

Step 3.4: In the lower level, Step 3.2 and 3.3 are repeated until the most appropriate cloud node is found for the policy application task.

Step 4: According to the first step, if there is not any cluster with all of the required capabilities, the task is assigned to a cluster with most common capabilities and policies. Therefore, the cluster applies all of the common policies to the task and sends a *Conditional* flag to the *Distributer* for re-assigning the task to another cluster to apply remaining policies. Accordingly, Step 3.2 and 3.3 are repeated to achieve the *Approved* flag.

Algorithm 4.3. Policy Mapping Process in Selected Cloud Node

Input: $MLO(PA_m) = \{cert, scope, res, p_1, p_2, \ldots, p_U\}$: Policy Application Task
Output: $APP(PA_m)$: Approval Flag of Applied Policy

```
3.2.1:  $Process\ (PA_m);$
3.2.2:  $\Big((\forall p_i \in PA_m) \& \ \big(\forall C_j \in PCS(S_\alpha)\big)\Big) : MapFunction\Big((PA_m.p_i), (S_\alpha.C_j)\Big)$
3.2.3:  $if \left(\prod_{u=1}^{U} (PA_m.p_u.cert = expired)\right) then, APP(PA_m) = false!$
        $esle\ (PolicyCertification = Pass);$
3.2.4:  $for(u = [1, U])$
        $forall\begin{pmatrix} (PA_m.p_u.type = Authentication) \\ (PA_m.p_u.res = Normal) \end{pmatrix}$
        $if\ \big(MapFunction(S_\alpha.C_j.C_{res}, PA_m.p_u) = false\big),$
        $APP(PA_m) = false!$
        $else\ (PolicyNormalAuth = Pass);$
3.2.5:  $for(u = [1, U])$
        $forall(PA_m.p_u.type = Access)$
        $if\ \big(MapFunction(S_\alpha.C_j.C_{res}, PA_m.p_u) = false\big),$
        $APP(PA_m) = false!$
        $else\ (PolicyAccess = Pass);$
3.2.6:  $for(u = [1, U])$
        $forall\begin{pmatrix} (PA_m.p_u.type = Authentication) \\ (PA_m.p_u.res = Double) \end{pmatrix}$
        $if\ \big(MapFunction(S_\alpha.C_j.C_{res}, PA_m.p_u) = false\big),$
        $APP(PA_m) = false!$
        $else\ (PolicyDoubleAuth = Pass);$
3.2.7:  $for(u = [1, U])$
        $forall(PA_m.p_u.type = Key/Trans)$
        $if\ \big(MapFunction(S_\alpha.C_j.C_{res}, PA_m.p_u) = false\big),$
        $APP(PA_m) = false!$
        $else\ (PolicyKeyTrans = pass);$
3.2.8:  $for(u = [1, U])$
        $forall(PA_m.p_u.type = Encrpytion)$
        $if\ \big(MapFunction(S_\alpha.C_j.C_{res}, PA_m.p_u) = false\big),$
        $APP(PA_m) = false!$
        $else\ (PolicyEncryption = pass);$
3.2.9:  $if\ (APP(PA_m) \neq false) = send\big(APP(PA_m)\big)$
```

4.5. Policy Mapping

One of the most challenging issues in policy-based security models is to provide efficient and reliable mapping between access requests and defined polices. The special characteristic of proposed multi-layered ontology helps to improve the process of access management by eliminating unnecessary mapping processes.

The process of mapping between policies and access requests is done in several phases.

Phase 1 (Basic Authentication): An access token is generated for the cloud user with basic authentication policies of the system:

$$TokenClass\ AT_u = New\ TokenClass(U_x, NormalAuth);$$

where U_x is the user information. After the successful authentication, access to the basic level without additional policies is granted. For each new access request, the access engine checks the associated polices and map to the requested node if all of the policies are satisfied.

Phase 2 (Multi-Layered Policy Mapping): In this phase each layer of requested policy is checked step by step to map the access request to data efficiently.

Step 1: In the first step all of the policy certification flags are checked as follows:

$$if\left(\prod_{u=1}^{U}(Policy_m.p_u.cert = expired)\right) then, = AccessDenied!$$

In fact, if even one of the certificates from defined polices is invalid or expired, the process of access is discontinued and other polices are not checked anymore.

Step 2: After checking the certificate, main policies and sub-policies are checked based on defined layers. Hence, if the request is against one policy, the access is denied and lower level polices are not checked anymore. Accordingly, the unnecessary mapping processes are eliminated from access task management queue. An access policy checking is mapped as follows:

```
$for(u = [1, U])$
$forall(\ Policy_m.p_u.p_{type} = Access)$
  $if\ (AccessMap((\forall sp \in Policy_m.p_u), (U_x.Capability)) = false)\ then, AccessDenied!$
$else\ ((PolicyAccess = Pass) \wedge (AT_u = AT_u.addCap(Policy_m.p_u)))\ ;$
```

The session access token is updated according to the capabilities of requester to enhance the process of policy mapping.

Step 3: The process of policy mapping is continued for the next access requests regarding to the stored capabilities of access token. In fact, only un-checked capabilities are evaluated in next requests until the token is valid.

```
$for(u = [1, U])$
$forall\left((\ Policy_m.p_u.p_{type} = Authentication) \wedge (Policy_m.p_u.p_{res} = Double)\right)$
        $if\ ((AT_u.Capability.isChecked(Policy_m.p_u.p_{res}) = Valid))$
          $then, (PolicyAuthentication = Pass)$
```

$else\ if\ \big(AuthenticationMap\big((\forall sp \in Policy_m.p_u),(U_x.Capability)\big) = false\big)$
$then, AccessDenied!$
$else\ \Big((PolicyAuthentication = Pass) \wedge \big(AT_u = AT_u.addCap(Policy_m.p_u)\big)\Big);$

Step 4: If the access request fulfills all associated policy requirements based on the stored access token or consequence policy mapping, the access is granted by the *Access Management Engine*.

4.6. Evaluation and Discussion

In order to incarnate the superiorities of multi-layered policy generation, application and management in cloud-based environments, we give a security and performance analysis of the suggested model in this section. Accordingly, several scenarios are described to evaluate the reliability and efficiency of this model during policy definition, policy application and policy mapping as follows:

4.6.1. Performance Analysis

To evaluate the performance of proposed schema several experiments were performed. In the first case study, the efficiency of multi-layered policy definition and multi-level scheduling were examined. Hence, three types of policy definition and policy application environments were defined: (1) *Single Layered Policies with Single Level of Scheduling*, (2) *Multi Layered Policies with Single Level of Scheduling* and (3) *Multi Layered Policies with Multi Level of Scheduling*. The simulation environment was designed based 200 nodes with different security capabilities as well as, same size data associated with random policy application tasks from a list of 50 defined security levels.

The total processing time by increasing the number of policy application tasks were evaluated in the first experiment (Figure 4.6A). The policy application tasks are increased from 100 to 3000 requests according to defined security levels and three types of policy layering and scheduling levels. According to the results the multi-layered policies with multi-level of scheduling (*i.e.* proposed model) performed considerably efficient by increasing the number of tasks. In the first 500 requests, the total processing time was approximately in a same range for all scenarios. However, the difference was significantly appeared by increasing the number of task requests as far as 61% difference between the proposed model and single layered policies with single level of scheduling scenario and has proved the efficiency of suggested model in applying security policies to data.

In the second case study (Figure 4.6B), the performance of managing access requests according to defined policies in proposed model was evaluated according to the defined policy definition and application environments: *Single Layered Policies*, *Multi-Layered Policies without Token* and *Multi-Layered Policies with Token*. Hence, response times for 200 access requests in each environment by one user were examined. According to the results the response time for multi-layered policies with token is significantly less than two other models due to the elimination of unnecessary processes for policy mapping. The range of response time for multi layered policies with token is between $[200,1100]\ ms$, as far as, $[800, 1700]\ ms$ for single layered policies.

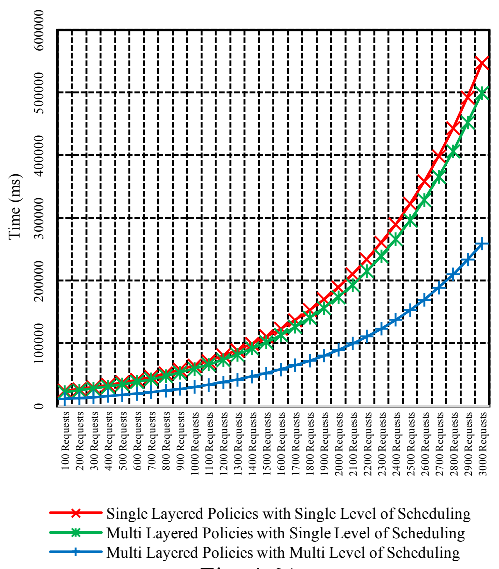


Fig. 4.6A.

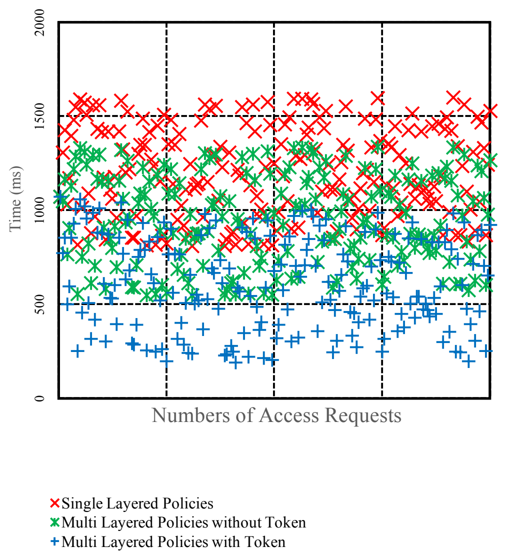


Fig. 4.6B.

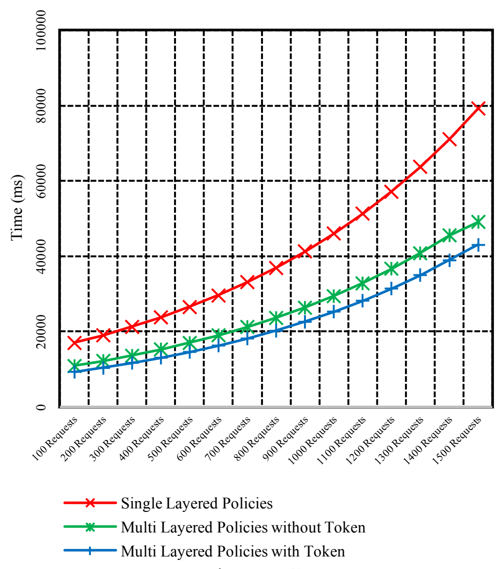


Fig. 4.6C.

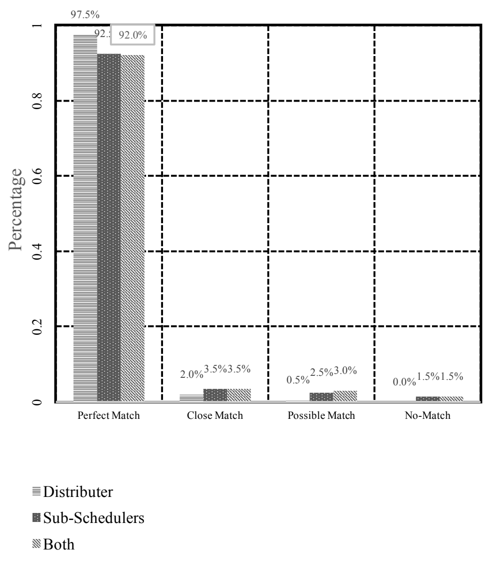


Fig. 4.6D.

Fig 4.6. Performance Analysis of Multi-Layered Policy Management Model with Multi-Level of Scheduling

In the third experiment the performance of multi-layered policy management during semantic policy mapping by invalid access requests was evaluated. In the scenario, numbers of users requested 100% invalid access requests. In fact, all of requests were against certificates or defined access, authentication and encryption policies.

Figure 4.6C shows the results of total processing time for invalid access requests. Based on the results, the multi-layered policy with token schema processed invalid requests considerably less that other models by elimination of un-necessary policy mapping processes.

Indeed, several access requests were terminated in the higher levels and there was not any necessity to process lower policies in multi-layered policy schema.

The final experiment was regarding to the performance of *Distributer* and *Sub-Scheduler* components for efficient scheduling of policy application tasks. Accordingly, four types of parameters were defined: *Perfect Match* (the most appropriate cloud node based on capabilities and current workload), *Close Match* (the most appropriate cloud node based on capabilities but not current workload), *Possible Match* (the possible cloud node but not the most appropriate one) and *No-Match* (there is not a match between the cloud node and application task requirements). The experiment environment involves 4 levels of scheduling with 50 nodes associated with defined capabilities and current workload parameter. The aim of this experiment was to assign 200 policy application tasks to these nodes and to check whether it was the best choice or not. The experiments checked the performance of *Distributer*, associated *Sub-Scheduler* and both of them. According to the results in figure 3.6D, the *Distributer* assigned 97,5% of tasks to the perfect cluster and none of the tasks was assigned to the completely no-match cluster. Also. 92,5% of tasks were distributed into the most appropriate nodes by *Sub-Schedulers*. There were 1,5% no-match results that were because of the variety of capabilities in different nodes. Furthermore, 92% of tasks were assigned to the most appropriate cluster and node, as well as, 3,5% and 3% close and possible matches according the capabilities of nodes and current workload.

Table 4.2. Security Analysis of Multi-Layered Policy Generation

Security Concern	Category	Solution Method
Validity of Policy	*Foundation*	Policy Certificate was defined for confirming the validity of policies based on time flags and owners.
Revoked Capabilities	*Foundation*	Multi-level policy ontology was used to manage added or revoked capabilities of cloud provider by inheritance concepts.
Flexible Policies	*Foundation*	Properties (Sub-Policies) were introduced to change defined value without any limitation and security concerns.
Reliable Classification	*Structure*	Using *Policy Foundation Set* with defined Policy Type to classify policies efficiently.
Invalid Authentication	*Structure*	Establishing two levels of authentication after checking policy certificate and access policies to ensure about the identity of requester.
Invalid Access	*Structure*	The access request is terminated whenever one of the policies is not satisfied and there is not any necessity to check lower level policies and sub-policies.
Resource Protection	*Structure*	Using cryptography mechanism in the lowest layer of policies to decrease the processing power for encryption and decryption as well as providing associated keys in the last step.
Reliable Layering	*Structure*	Using a stand-alone component to create layers for policies before storing in policy database according to defined structure.
Semantic Mapping	*Structure*	Establishing a validity engine for mapping defined policies to access requests with a semantic and reliable method.
Validity of Certificates	*Structure*	Updating the polices engine about the validity of certificates after each policy mapping.

4.6.2. Security Analysis

The security of multi-layered policy definition has been ensured in two main categorizes: reliability of *Policy Foundation* and security of *Policy Structure*. Defined polices were flexible in both security mechanisms and associated properties to provide reliable layering according to the capabilities of service provider and requirements of cloud customers.

Table 4.2 shows the security analysis of our proposed model regarding to defined objectives and possible treats. The analysis proves this multi-layered schema may resist against possible treats by various methods in both *Foundation* and *Structure* parts.

The security of multi-level policy application and policy mapping were based on reliable classification of nodes and efficient distribution of policy application tasks. In fact, defined polices were classified efficiently to provide secure mapping between request and capabilities. Furthermore, all access requests were evaluated with taking minimum processing power and maximum security.

Finally, the process of token management was considered to control access confirmations from associated level securely for prevention of un-authorized accesses. Table 4.3 shows the security analysis of suggested model regarding to defined objectives and possible treats. The analysis shows this multi-layered schema may resist against possible treats by various methods and solutions in classification and policy management phases.

Table 4.3. Security Analysis of Policy Management

Security Concern	Category	Solution Method
Reliable Assigning	*Policy Application*	All policy application tasks are assigned to the most appropriate cloud node for policy application according to the multi-level clustering.
Failure in Policy Application	*Policy Application*	Without a valid approval, the Policy Engine does not remove the policy application task from the queue. Therefore, if there is a failure in policy application procedures, the Distributer re-assigns the task to the most appropriate node.
Failure in Security Algorithm	*Policy Application*	Only affects the associated clusters and other clusters are isolated. Furthermore, the failure may solved in only higher levels and all lower levels are affected due to the advantages of multi-level policies.
Reliable Access Management	*Policy Mapping*	Mapping all the associated polices layer by layer according to the capabilities of requester
Validity of Policies	*Policy Mapping*	Re-checking the certificate of associated policy to confirms the validity of policy before access request processing.
Security of Tokens	*Policy Mapping*	Checking and updating the content of token during each access request.
Un-Secure Mapping	*Policy Mapping*	The process of mapping requests to policies is done in several levels from lower to higher security algorithms. Thus, it's impossible to map the access requests without satisfaction of all associated policies.

4.6.3. Competitive Analysis

Table 4.4 shows the competitive analysis of suggested model according to the objectives of a multi-layered policy management in cloud computing environments. Based on the comparison, the suggested model provides a multi-level and also multi-layered policy engine to establish a syntactic and also semantic mapping between polices and access requests. Furthermore, the flexibility of policies is increased by using inheritance concepts and multi level sub-policies. Finally, additional and un-necessary policy mapping processes are eliminated from the model by using a multi-layered policy checking.

Table 4.4. Competitive Analysis of Proposed Model

Competitive Analysis	Rei [94][94][94][94]	Di Modica	Veloudis	PEaaS	Proposed
Structure	Policy	Policy	Policy	Policy	Policy
Environment	Web	Cloud	Cloud	Cloud	Cloud
Multi-Level	Yes	Yes	Yes	Yes	Yes
Multi-Layer	No	No	Yes	No	Yes
Syntactic & Semantic Analysis	No	Yes	Yes	Yes	Yes
Layering Engine	No	No	No	No	Yes
Flexible Polices	No	No	No	No	Yes
WS-Policy Language	No	Yes	Yes	Yes	Yes
Sub-Policies and Properties	No	No	No	Yes	Yes
Policy Certificate	No	Yes	No	No	Yes
Mapping Elimination	No	No	No	No	Yes
Dual Authentication Layering	No	No	No	No	Yes

4.7. Conclusion

To enhance the quality on managing security policies in cloud-based environments and to provide efficient, secure and reliable matching between security requirements of customers and capabilities of service providers [96], a multi-layered policy engine was introduced in this chapter. A well-established policy framework was defined to generate security policies which are compliant to requirements and capabilities.

A multi-layered security policy was built around two main concepts: *Foundation* and *Structure*. The *Foundation* of each policy includes potential security protocols, mechanisms and algorithms that are used to establish dedicated security levels according to the capabilities of service provider and requirements of cloud customers, as well as, the *Structure* of policies is the way to define and manage policies and sub-policies according to the Foundation.

The process of policy application is based the security capabilities of cloud nodes and efficient distribution of multi-layered policy application tasks on the most appropriate cloud nodes. Hence, a federated policy management schema is introduced based on the policy definition framework and multi-level policy application to create and manage virtual clusters with identical or common security levels. The model was evaluated according to performance and security parameters and proved that this multi-layered policy engine enhances the reliability and efficiency of managing security polices in cloud computing environments during policy definition, policy application and policy mapping procedures.

One of the most important effects of our policy management system is to provide secure authentication and access control procedures based on sensitivity of resources, capabilities of service provider and constraints. The next chapter specifies our policy management model in a cloud-based authentication and authorization schema to enhance the efficiency and reliability of mapping access requests to associated resources with defined policies.

Chapter 5

5. Policy-Based Identity Management

5.1. Introduction

One of the most challenging security issues in clouds that has led to the appearance of several researches and solutions is to ensure reliable accesses to different cloud servers based on various policies in each server. In fact, service providers needs to manage access requests and map them to resources according to defined policies from cloud customers or service providers [49].

Using a federated identity management schema is the most popular solution for managing accesses to different cloud servers with single identity. In recent years most cloud services have adopted OpenID [56] or Shibboleth [59] as the most independent and flexible authentication and identity management models in cloud-based platforms. The proliferation of these identity federations has allowed cloud users belonging to one network (known as home organization) to access the services provided by other networks (known as remote organizations), all members of the same federation [97].

Therefore, there isn't any necessity for cloud users to re-introduce their credentials for each access in different cloud servers. The most important characteristic of identity management models is to provide a framework with fast-authentication mechanisms [12], low access time and reduced authentication data exchanges between different service access requests [98]. Although the establishment of multiple security mechanisms in each node enhance the security of resources and reduces considerable processing power for manipulating sensitive and also non-sensitive data [43], the authentication data exchange and access time for cloud users in identity management models are also affected. In particular, two important concerns in cloud-based identity management models are still challenging:

- Managing defined policies in different virtualized nodes according to capabilities of service providers, requirements of resource owner and constraints [99].

- Mapping access requests to cloud-servers based on established security mechanisms and defined policies of each node [100].

In this chapter, a policy-based user authentication model is presented to provide a reliable identity management mechanism for establishing multiple access policies in different virtualized nodes and mapping access requests to defined policies accordingly capabilities of cloud servers and requirements of resources. Accordingly, a structural policy language and policy engine are introduced for policy generation, application and management. Moreover, a policy match framework is described for managing identities and mapping access requests to virtualized resources based on defined policies.

5.2.Related Works

The fundamental goal of any identity management model is to ensure a reliable authentication of subscribed users according to the defined policies of different cloud servers and to protect information from un-authorized accesses. There is a wide variety of methods, techniques, models, and administrative capabilities used to propose and design identity management models [53] and each system has its own attributes, methods, and functions. The importance of these identity management models is more evident for cloud providers and customers according to the characteristics of cloud-based services that work with shared open environments. Therefore, several studies and researches were performed to improve the reliability and efficiency of managing identities in clouds.

In recent years most of Single Sign On (SSO)-enabled access management models are based on web applications such as SAML [54] and OAuth [55] for allowing users and application services access to web resources. OpenID [56] is one of the most popular relevant federated authentication technologies that allows cloud users to use a single identity for accessing various services from different cloud servers and for elimination of managing different identities by cloud users. However, OpenID relies on an ID provider to generate a unique identity for each user [57]. Therefore, the server has to connect to the ID provider on the Internet during authentication of cloud users and it leads to a high level of time and computation load [58].

Shibboleth [59] is another federated identity management model which is similar to OpenID, for allowing users to authenticate to different services using just one piece of information. Shibboleth is an open source implementation of federated identity based management model where the identity providers provide information and the service providers consume this information giving access to content or services [60]. However, the most challenging concern of Shibboleth is to provide different levels of authentication based on the sensitivity of data in various cloud servers. In fact, mapping between federated identity information with different levels of security in cloud servers based on defined policies is still the main issue in these types of federated identity management models [57][58].

The other solution was proposed as Kerberos [61] by using distribution of authentication tickets to provide a generic access control protocol and reliable SSO. The most drawback of Kerberos is the lack of privacy solutions in the model that was tried to solve as an extension in several models such as KAMU [62] or PrivaKerb [46]. Kerberos-based models use an operation mode (*cross-realm*) to be compatible for federated environments. Nevertheless, these models consist a completely independent infrastructure aside those already established for the access to web application services and the network access service [63]. Hence, Kerberos *cross-realm* federations have not been widely deployed [64]. Using an interaction between Kerberos and Extensible Authentication Protocol (EAP) [65] protocols was the other solution the enhance weaknesses of *cross-realm* federations. Using EAP-based pre-authentication mechanism [63] and also using Protocol for Carrying Authentication for Network Access (PANA) [66] to bootstrap dynamic Kerberos credentials on the service providers [67] are the most popular efforts to enhance Kerberos *cross-realm* federations. However, the necessity for deployment of Kerberos entity on every organization and providing SSO within each organization's boundaries are the most considerable inconvenient of proposed models.

To solve the problem of compatibility and deployment in Kerberos-based models, Leandro et al. [68] uses a multi tenancy authorization system to deliver access control based on concerns about the privacy of data. The proposed model was built around Shibboleth core concepts with authorization and authentication mechanisms to emphasize on self-governing and control of trusted third parties, according to the digital identity federation [69]. This method was followed by [70] by adding stand-alone identity management features to the federated model. However, it has been proven [71] that misuse of user identity information in self-governed and stand alone identity federation could happen via SSO services in IDP and SP, which could lead to identity theft (*i.e.* the main concern in federated identity management systems). Thus, Bhargav-Spantzel et al. [72] recommended two mechanisms to protect the misused of identity information: distributing user identity information amongst several self-governed entities and using zero-knowledge proofs techniques to prevent identity theft within an IDP or SP. Although, the recommended mechanisms reduced the chance of identity theft, there are still serious concerns about the process mapping requests from revoked identities in stand alone identity federations [73]

Kalra and Sood [74] proposed an Elliptic Curve Cryptosystem-based (ECC) algorithm to provide a mutual authentication protocol for secure communication of embedded devices and cloud servers in association with HTTP cookies. The evaluation of this model proved that it was robust against multiple security attacks. However, managing ECC keys for different cloud servers in this model takes considerable processing power for manipulating sensitive and non-sensitive access policies in different cloud servers [75].

Apart from ECC, several cloud-based authentication models were designed with various techniques such as Biometrics-Based Authentication [76], Certificate-less Anonymous Authentication [74][75], User Behavior Analysis-based [77] and ID-Based identity management [78] with the same issue that is the lack of congruency in different cloud servers with distinguished access policies. Indeed, the necessity of a policy-based identity management in different cloud servers with various security levels is undeniable according to rapid growth of cloud providers.

Using policies to establish different security levels in traditional and also modern distributed networks allows to manage the processing power for manipulating sensitive and also non sensitive resources. Hence, several policy-based languages and models are suggested to classify resources based on sensitivity. WS-Policy [32] is an extendable general purpose framework associated with a defined syntax to describe the policies of entities and a broad range of service requirements and capabilities in a web services-based system.

This XML-based framework has been extended by various researchers according to service requirements such as security services. Di Modica and Tomarchio proposed a semantic secure policy matching based on WS-Policy framework in [38] and [41] for service oriented and cloud-based architectures. In the cloud-based model, the capabilities of the cloud service provider and the requirements of the cloud customer were defined within policies adopted to the WS-Policy framework.

There are several security standards that are extended from WS-Policy architecture such as “WS-Trust” [33] or “WS-PolicyAttachment” [79]. “WS-Trust” is an OASIS standard for changing security tokens from one format to another in an interoperable manner in order to establish and assess the presence of participants in a secure message exchange. Also, “WS-PolicyAttachment” was expressed to define two mechanisms for associating policies with the subjects they apply and to represent the way of attaching WS-Policy descriptions end points. Furthermore “WS-SecurityPolicy” [34] and “WS-SecureConversation” [35] were extended from WS-Policy architecture to describe the security specifications of WS-Trust and to improve the performance of frequent communications by using a shared symmetric and pair of asymmetric keys from the security context respectively. The comparison between WS-based standards is available in [17].

The challenging issue in policy-based resource classification is to map access requests from cloud users based on defined security policies of resources. In fact, a policy-based identity management needs to analyze policies syntactically and also semantically and map access requests based on the established security mechanisms of each node [12]. Hence, policy management is one of the most challenging key points of identity management in multi-level virtualized resources. On the other hand, the processes of scheduling, analyzing and mapping access request tasks according to the policies need to be considered in policy-based identity management.

Providing a federated authentication schema for different cloud servers needs an efficient authentication task manager to administrate access requests based on defined policies. CSA is a multi-level adaptive authentication schema in clouds that was proposed [80] to dictate the efforts of protocol participants by identifying a legitimate user’s requests and placing them at the top of the authentication process queue. In fact, a multi-objective scheduling model for authentication tasks was suggested to prevent DoS attacks in multi-level cloud servers. Although, the process of authentication task management in CSA was based on risk identification, not on defined policies in multi-level authentication.

According to the previous research results, a scalable policy-based identity management is presented in this chapter to address two main problems: (1) Lack of coincidence in identity management models based on defined policies and various security levels in different cloud servers, (2) Lack of multi-objective authentication task management according to the defined policies in multi-level authentication procedures.

5.3. Problem Description

As described in previous section, the main aim of proposed model is to manage identities based on defined policies in cloud servers. Each virtualized node in cloud-based data center is associated with set of policies. These polices are classified in several protocols according to Protection Ontology [43].

The classification of security policies are based on three main parts: Resource Protection (including cryptography and key management policies), Confidential Transport (including signature and transport policies) and Identity Management (including authentication and access control policies). The latter, which is the focus of this work, refers to the capabilities that are provided to ensure the reliable access mapping between requests and policies by managing identities based on capabilities of service provider and requirements of cloud users.

Assume that there are N virtualized node (server) in the cloud-based data center, denoted as $\{S_1, S_2, \dots, S_N\}$, and the current authentication policy set of node S_n with $s \in \{1,2,\dots,N\}$ is $P(S_n) = \{p_1, p_2, \dots, p_M\}$. Given I registered users' access requests waiting to be processed, denoted as $\{U_1, U_2, \dots, U_I\}$, and each U_i is associated with specific identity set (authentication and authorization set)

$$AA(U_i) = \left\{ \begin{matrix} ID_i, h(PW_i), \left(h(ID_i) \oplus h(PW_i)\right), \left(AP_1, h(AR_1), \left(h(AP_1) \oplus h(AR_1)\right)\right)_i, \dots, \\ \left(AP_j, h(AR_j), \left(h(AP_j) \oplus h(AR_j)\right)\right)_i \end{matrix} \right\}$$

where ID, PW, AP and AR are user ID, user basic password, access policy and access response respectively. Other notations used in this model are shown in Table 5.1.

There are several authentication and authorization (access) policies that are defined for each node to enhance the security level of the node in comparison between other nodes. The authentication policies are focus on confidentiality and integrity of resources, while the authorization policies are based on privacy and access management features of cloud resources. To provide a semantic mapping between requests and policies, each of authentication and authorization policies of a specific node need to be evaluated according to the characteristic of cloud user.

Table 5.1. The Notations of Proposed Model

Notations	Description	Notations	Description
S_i	Node (Server) ID	$\parallel$	String Concatenation Operation
$P(S_n)$	Set of Policies for Node n	$h(\,)$	One-Way Hash Function
U_i	Current User	$\oplus$	Exclusive-Or Function
p_i	Policy Type	TS	Timestamp Generated by the Cloud User
SP	Sub-Policy	TS'	Timestamp Generated by the Node
$AA(U_i)$	Identity Set (Authentication & Authorization)	ΔTS	Time Difference of the Timestamps
ID_i	User ID	$AccessSession$	Access Session Class
PW_i	Basic Password for User i	TK_i	Access Token for User i
AP_{j_i}	Access Policy for Policy j and User i	e	Access Token Status Property
AR_{j_i}	Access Response for Policy j and User i	$NAR_{(i,n)}$	Node Access Request for User i and Node n
x_n	Secret Key for Node n	RQ	Response Queue
$HP(S_n)$	Hashed Value Set of Policies for Node n	CQ	Checking Queue
$X(S_n)$	Secret Policy Value for Node n	$X'(S_n)$	Secret Value based on Access Request
AST	Algorithm Session Time	TM	Task Management Class

The objective of suggested model is to map elements of the policy set for each node to appointed access responses for cloud users to provide decisive access permit. For instance, consider a cloud provider with different services (*e.g.* storage, platform, software, *etc.*) and each service has dedicated security policies (*e.g.* two factor authentication for storage and one-time single pass for software). The main problem is to address the process of mapping security requirements of these cloud services to defined authentication and authorization capabilities of the cloud user in identity set. Overall, the access request of specific node is granted if and only if the following equation is applied to the request:

$$\forall p_i \in P(S_n) : \left(\exists \left(AP_j, AR_j\right)_i : \left\{\left(p_i = \left(AP_j\right)_i\right) \wedge \left(\left(AR_j\right)_i = true\right)\right\}\right)$$

In fact, cloud user needs to provide additional authorization and authentication capabilities for nodes with higher security policies. The proposed model tries to manage access requests and map between access policies and authentication capabilities of cloud users according to the equation.

5.4. Proposed Model

Using an agent-based authentication model [101][102][103] to send access requests, to search on policy queues and to match access requests to a specific defined policy may seem like a plausible solution for achieving the goal. However, this agent-based authentication process in not scalable and takes lots of processing power to map between requirements and capabilities. Thus, the design of our proposed model is based on a different manner. Our schema uses a framework with four components to define, store, check and match policies with identity details. Figure 5.1 shows the overall architecture of our model in details.

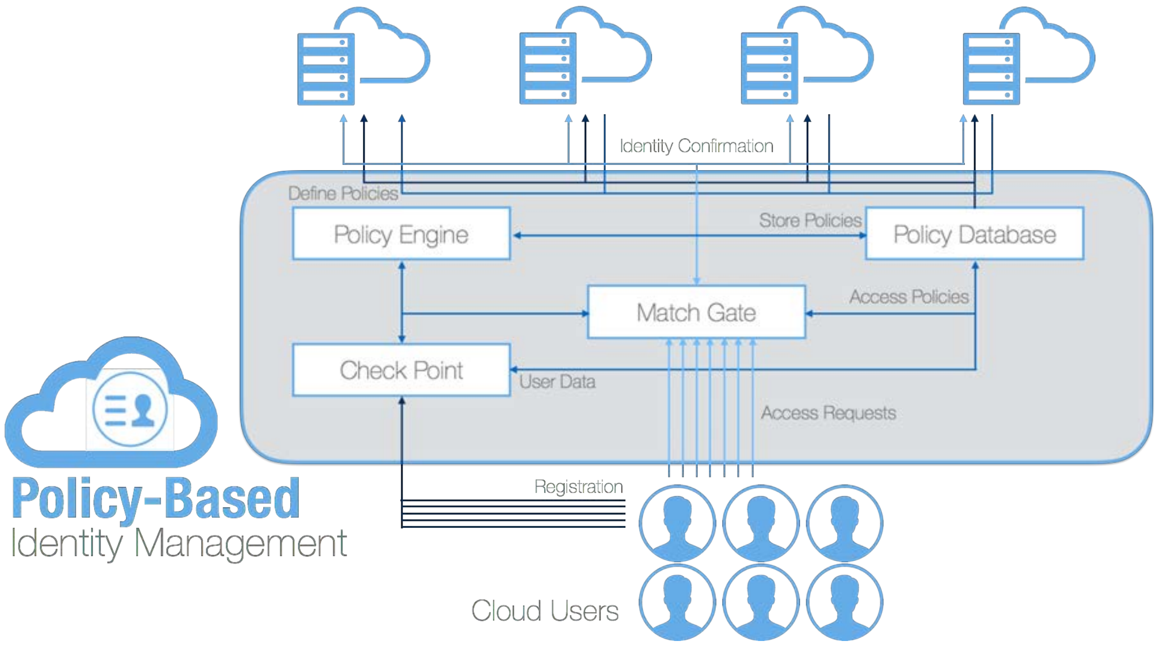


Fig. 5.1. Architecture of Policy-Based Identity Management

5.4.1. Policy Engine

The main duty of Policy Engine is to define and generate authentication and authorization policies based on the structural Protection Ontology [43] for cloud customers according to security requirements. Protection Ontology is a policy language based on WS-Policy [32] as a recommended W3C language for defining various security levels in cloud-based environments.

Protection ontology classifies security algorithms to three main levels: Protocol, Mechanism and Algorithm. In the proposed model authentication and authorization capabilities of service provider are offered according to this classification. Figure 5.2 shows an example of this classification according to the authentication and authorization capabilities of service provider. This structural classification helps to apply different security mechanisms to virtualized nodes and creates security levels based on requirements of cloud users and sensitivity of resources.

Each of the offered algorithms is associated to a structural semantic resource for security level establishment according to the concepts of WS-Policy and Protection Ontology. A security-based SLA is the output of security ring (level) establishment and is defined as Security Level Certificate (SLC).

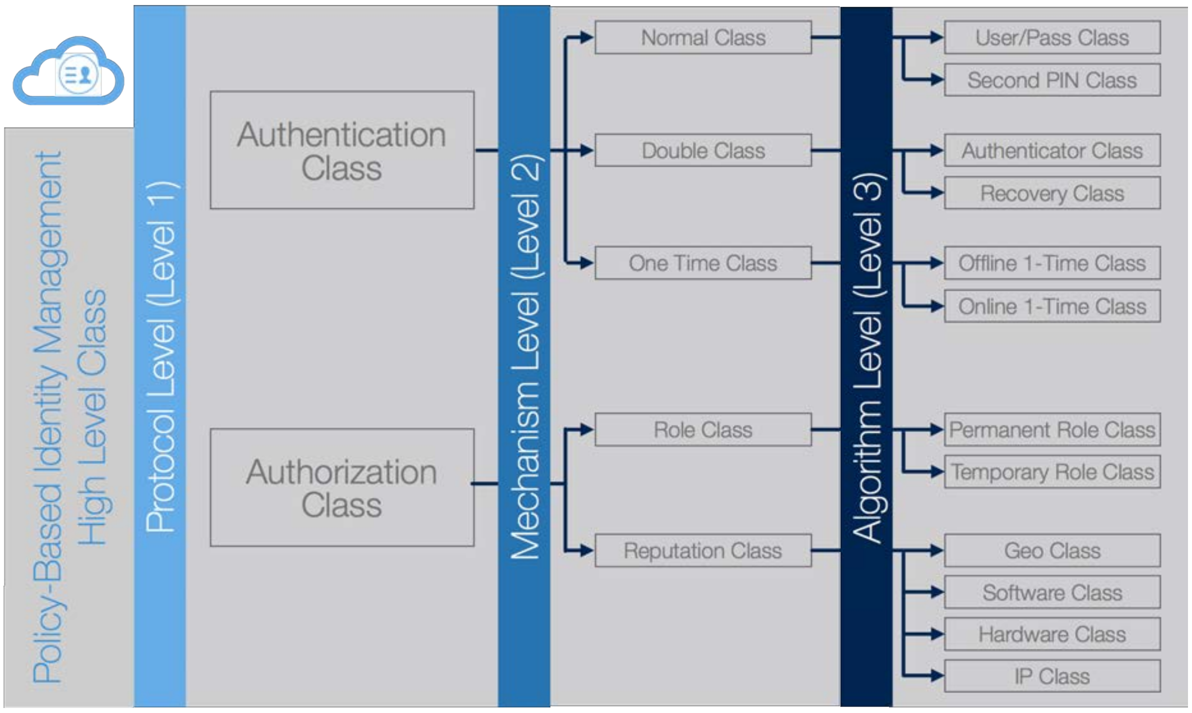


Fig. 5.2. Policy-Based Identity Management Ontology: Multi Level Class

In the following an example of ring definition for virtualized nodes according to WS-Policy and described classification is given:

```
<wsp:Policy>
…namespace definition…
   <wsp:SLC rdf:ID="SLC3GRSE23GHD5">
      <wsp:Header>
      …
      </wsp:Header.
      <wsp:Body>

         <security:AuthenticationProtocol  rdf:ID="AuthenticationRequirement">
            <security:DoubleMechanism rdf:ID="DoubleClassRequirement">
               <security:AuthenticatorAlgorithm rdf:resource="AuthenticatorClass">
                 <rdf:HLSP_Authenticator rdf:HLSP_Authenticator_1="App" rdf:HLSP_Authenticator_2="Email"
                 rdf:HLSP_Authenticator_3="Phone">
                 </security:AuthenticatorAlgorithm>
            </security:DoubleMechanism>
         </security: AuthenticationProtocol>

         <security:AuthorizationProtocol  rdf:ID="AccessManagementRequirement">
            <security:RoleMechanism rdf:ID="RoleClassRequirement">
               <security:PermanentAlgorithm rdf:resource="PermanentRoleClass">
                 <rdf:HLSP_Role rdf:HLSP_Role_1="Lecturer" rdf:HLSP_Role_2="Student"
                 rdf:HLSP_Role_3="Professor" rdf:HLSP_Role_4="Staff" rdf:HLSP_Role_5="Admin"/>
                 </security:PermanentAlgorithm>
            </security:RoleMechanism>
            <security:ReputationMechanism
             rdf:ID=ReputationClassRequirement">
               <security:GeoAlgorithm
                rdf:resource="GeoClass">
                    <rdf:HLSP_Geo rdf:HLSP_Geo_1="Germany" rdf:HLSP_Geo_2="World"/>
               </security:GeoAlgorithm>
            </security:ReoutationMechanism>
         </security:AuthorizationProtocol>

      </wsp:Body>
   </wsp:SLC>
</wsp:Policy>
```

In this example, the authentication policy of the virtualized node uses double authentication with authenticator algorithm as second password generator. The second factor methods of this authenticator is based on various delivery such as application, email and text message. Furthermore, an authorization mechanism is defined based on classical role-based access model with several sub-policies (*i.e.* defined roles) associated with modern reputation access management to check the geographical characteristics of access requests. The process of mapping between security requirements of virtualized nodes to access requests of cloud users is based on defined policies of SLC.

As described the main duty of policy engine is to define and generate authentication and authorization policies for different virtualized nodes according to the sensitivity of nodes and capabilities of service provider. The process of policy application is done by policy engine based on generated SLC as follows:

Step 1. Policy engine sends SLC ID to node n to apply defined policies of SLC to the node.

Step 2. According to the associated SLC, node n calls semantic resources of SLC to create $P(S_n)$.

$$for\left(i = 0\ to \sum_{\alpha=0}^{R}(rdf\!:Algorithm)_\alpha\right)$$

$$\begin{Bmatrix} p_i = \left(p_i \parallel add\left(SP_\mu\right)\right) \\ X(p_i) = h(p_i) \\ HP(S_n) = add\left(X(p_i)\right) \end{Bmatrix}$$

where R is the total of semantic algorithm resources and $\mu = \sum_{\varepsilon=0}^{Count(HLSP)}(rdf\!:HLSP)_\varepsilon$ are the defined sub-policies for each algorithm based on the SLC. Also, the hashed value of each policy p_i is stored in the set $HP(S_n)$.

Step 3. $X(S_n) = h(x_n) \oplus h(p_1) \oplus h(p_2) \dots \oplus h(p_n)$
where x_n and $X(S_n)$ are the secret key and the secret value for node n respectively.

Step 4. Send $\{(P(S_n), HP(S_n)), X(S_n)\}$ to Policy Database.
The SLC, policy set, hashed policy set and secret value of node n are sent to policy database.

5.4.2. Policy Database

All of the generated SLCs, defined policies and sub policies for clouds servers are stored in the Policy Database component. Each SLC can be assigned to several nodes with similar or different sub-policies according to the security requirements, resource sensitivity and capabilities of service provider. Moreover, the final policy set with low level sub policies, hashed policy set and secret value for each node are stored in the policy database for matchmaking process during access control.

5.4.3. Policy Check Point

The check point component creates, updates and manages identities for accessing to different nodes. Identities are defined in registration phase, updated in checking phase and managed in access control phase. In recent years, two types of registration progresses are performed in web-based models:

- Normal Registration: The creation of personal identity within cloud provider with User ID, Password and other personal details.
- Third-Party Registration: Using registered identities in social media or other cloud providers (*e.g.* Google ID or Facebook ID, *etc.*).

During the registration phase by each of these models, an Identity set (Authentication and Authorization) object $AA(U_i)$ is created from identity set class for user U_i. The basic identity set with the lowest identity details is associated with the user ID and password:

$$I(U_i) = \{ID_i, h(PW_i), (h(ID_i) \oplus h(PW_i))\}$$

By the basic identity set, cloud users can access to the nodes with the lowest security level in cloud environment. However, three types of authentication and authorization access policies need to be defined and added to the identity set based on polices and capabilities of service provider:

- User Access Policies (UAP): These types of policies are defined by cloud users according to capabilities of cloud provider. For instance, cloud user can establish second password with an authenticator application or email.
- Cloud Access Policies (CAP): These types of policies are awarded to cloud users by the provider or admin after an identity validation (*e.g.* a role-based identity in a university).
- Temporary Access Policies (TAP): These types of policies are based on dynamic parameters such as location, hardware and time.

An access policy is defined in identity set according to the characteristics of policy by a triplex set as follows:

$$\left(AP_j, h(AR_j), \left(h(AP_j) \oplus h(AR_j)\right)\right)_i$$

where AP_j and AR_j refer to semantic resource access policy (*e.g.* two factor authentication by Email) and access responses (*e.g.* confirmed email address) respectively. Therefore, the authentication set for U_i are updated based on defined UAP, CAP and TAPs as:

$$AA(U_i) = \left\{ \begin{array}{c} ID_i, h(PW_i), \left(h(ID_i) \oplus h(PW_i)\right), \left(AP_1, h(AR_1), \left(h(AP_1) \oplus h(AR_1)\right)\right)_i, \dots, \\ \left(AP_j, h(AR_j), \left(h(AP_j) \oplus h(AR_j)\right)\right)_i \end{array} \right\}$$

5.4.4. Policy Match Gate

The proposed identity management model for mapping accesses requests to defined policies is based on the performance of policy match gate. Given I registered users' access requests waiting to be processed, denoted as $\{U_1, U_2, \dots, U_I\}$, and each U_i is associated with a specific authentication set $AA(U_i)$. The main aim of Match Gate is to process access requests and to map between these requests and defined polices for each node according to the identity set.

To provide an efficient policy mapping algorithm, a session class is defined by policy match gate for creation of access session objects according to the capabilities of cloud users. The objects from this class ($AccessSession$ class) use several security functions and parameters to ensure about the reliable mapping between capabilities and security policies. After the registration phase in the check point component, cloud users are able to sign in to cloud computing environment by their basic internal or external login information. A successful basic login lets the policy match gate to create a session object from the access session class for basic or additional security checking. The process of using this object for identity management is in number of steps as follows:

Step 1. An object is created by the basic login from $AccessSession$ class with basic parameters.

$$AccessSession\ ASU_i = new\ AccessSession\left(ID_i, h(PW_i), TS, \left(h(ID_i) \oplus h(PW_i)\right), TK_i, e\right)$$

where TK_i is a basic token access for U_i and is valid if login details are matched with $A(U_i)$ and e is a Boolean property that shows the status of TK_i whether is enabled or disabled.

Step 2. The basic value of TK_i after the first login lets the cloud user to access basic nodes with lowest security level. In this level policy match gate checks if ΔTS and e are still valid, the access of cloud user to the root nodes are granted. The basic value of TK_i is calculated as follows:

$$TK_i = \left(h\left(h(ID_i \parallel TS) \oplus h(PW_i)\right)\right)$$

Step 3. When the cloud user requests for accessing to basic nodes, the match gate calculates Node Access Request (NAR) as follows:

$$if\big((e = true)\wedge(\Delta TS = Valid)\big)\ then\ \{NAR_{(i,n)} = \big(TK_i, Enc(TK_i, x_n)\big)\}$$

where Enc is AES-256 encryption function with the secret key for node n. The checking phase confirms the identity of user and the value of NAR is sent from match gate to requested node.

Step 4. Server n receives the request from Match Gate and access is granted if the difference between timestamps and the following equation is valid:

$$if\ \Big((\Delta TS = Valid)\wedge\big(TK_i = Dec(TK_i, x_n)\big)\Big)\ then\ Access\ is\ Granted$$

This calculation helps to check if the secret key of node n is still valid or not. In fact, the validated identity from match gate can access to request node if the secret value of node is valid. If the validity of the equation is not confirmed, Match Gate should update the secret key of server n in database.

Step 5. If the cloud user requests for accessing to nodes with the defined security policies and higher privacy levels, further identity details are requested from Match Gate based on the defined policies. Thus, Match Gate checks $P(S_n)$ from policy database and asks U_i if UAP or TAP policies are needed for authentication and authorization checking. Also, the user database is checked by Match Gate for CAP policies for only authorization checking if needed. Each of the requested access details should be provided by the cloud user (*i.e.* UAP and TAP) or the user database (*i.e.* CAP) and ASU_i is updated according to the provided details:

$$ASU_i.AddAccessCapability\big(AP_1, h(AR_1), \big(h(AP_1)\oplus h(AR_1)\big), AST_1\big);$$

$$ASU_i.AddAccessCapability\big(AP_2, h(AR_2), \big(h(AP_2)\oplus h(AR_2)\big), AST_2\big);$$

$$\vdots$$

$$ASU_i.AddAccessCapability\ \Big(AP_j, h\big(AR_j\big), \Big(h\big(AP_j\big)\oplus h\big(AR_j\big)\Big), AST_j\Big);$$

where AST is the Algorithm Session Time that shows the maximum validity of confirmed access response. For instance, the valid time for confirmed second password is longer that 1-time password. By each of the additional identity details the value of TK_i is updated as follows:

$$TK_i = TK_i \oplus \Big(h\big(AP_j \parallel TS\big)\oplus h\big(AR_j\big)\Big)$$

Step 6. After updating the value of Tk_i and confirming the identity of cloud user by additional identity request and according to the capabilities of user, the match gate sends server access requests to the requested node as follows:

$$X'(S_n) = \left(h(x_n) \oplus h(AP_1) \oplus h(AP_2) \dots \oplus h(AP_j)\right)$$
$$if\left((e = true) \wedge (\Delta TS = Valid)\right)\ then\ \{NAR_{(i,n)} = \left(TK_i, Enc(TK_i, x_n), h\left(X'(S_n)\right), TS'\right)\}$$

Step 7. Server n receives the request from Match Gate and access is granted if the ΔTS and the following equations are valid:

$$if\ \left((\Delta TS = Valid) \wedge \left(TK_i = Dec(TK_i, x_n)\right) \wedge \left(h\left(X(S_n)\right) = h\left(X'(S_n)\right)\right)\right)$$
$$then\ Access\ is\ Granted$$

This calculation checks the validity of timestamp, the validity of secret key and finally the confirmed application and mapping process of defined policies by checking the validity of secret value.

Step 8. If the user requests to access to a node with common policies that were confirmed by match gate before and the Algorithm Session Time for the access response is still valid for the policy, just un-checked policies are evaluated and the is no necessity to re-check previous policies. In fact, every functions and properties of ASU_i are confirmed and stay reusable until the algorithm session time for that authentication or authorization algorithm is still valid. For instance, the session time for double authentication is less than single authentication and U_i needs to be double-authenticated again after the session time for double authentication is over while the session time for basic authentication is still valid. Also, re-authentication for some authorization access policies (*e.g.* Geographical or Software authenticators) or One-Time passwords need to be checked periodically or continuously. These valid session times are defined as sub-policies in the ring establishment stage based on Protection Ontology.

5.4.5. Match Gate Task Management

One of the most challenging issues in policy-based mapping models is to manage access request tasks according to defined polices in different types of workloads. In fact, the scalability of suggested models needs to guaranteed in the case predictable of dramatic changing workloads. To achieve this purpose, all of the access request tasks need to be evaluated semantically to classify requests based on the current status of the access session object.

The process of match gate task management is performed in three phase: classification, initialization and scheduling. In the first phase of analysis, access requests with created session objects are separated from new requests.

Let $T = \{t_1, t_2, \dots t_{N+C}\}$ represents all access request tasks, the former N items of which are new requests and the rest C items are requests with created ASU objects. $TaskManagment\ (TM)$ class is defined to create objects including a $C \times 5$ matrix as follows:

$$\begin{array}{c|ccccc} & TK_i & e & UAP & CAP & TAP \\ \hline t_{N+1} & & & & & \\ t_{N+2} & & & & & \\ \vdots & & & & & \\ t_{N+C} & & & & & \end{array}$$

where e is the status of TK_i showing whether is enabled or disabled and UAP, CAP and TAP are the numbers of remaining cloud, user and temporary access policies for each cloud user respectively.

The first created object from TM class has the highest priority for processing. Also, the second object is created from new objects of basic authentications, $t_i \in \{t_1, t_2, \dots, t_N\}$ and if $N > C$, next objects are created from TM with lower priorities.

The process of managing requests is done according to the priority of objects from TM, the validity of TK_i obtained by checking the value of e and the number of remaining tasks in each cell of matrix. Algorithm 4.1 shows the process of access task management in details. The scheduling process is done in z rounds according to the number of created matrix objects and the tasks are scheduled in response and checking queues for authentication requests from users and the user database in clouds respectively.

Algorithm 5.1. Access Task Management

```
Input: $TM = \{TM_1, TM_2, \dots, TM_z\}$ : the set of created objects from $TM$ class.
Output: Scheduled task queue.

1.      Initialize Matrix $TM_1$ to $TM_z$.
        $\exists t_\beta \in T$ where $T = \{t_{N+1}, t_{N+2}, \dots t_{N+C}\}$
        $for\ \delta = 1,2,\dots,z, \{if\ TM_\delta = Enabled, Assign\ (t_\beta\ to\ TM_\delta) and\ Break\}$
2.      Analysis of First Matrix Object.
        $for\ (\vartheta = 0\ to\ C)$, check all of the items in $TM_1$ with enabled $e$ and send remained $UAP$ and
        $TAP$ tasks to Response Queue ($RQ$) and $CAP$ task to Checking Queue ($CQ$).
3.      Update First Matrix Object.
        $for\ (\vartheta = 0\ to\ C)$, check all of the items in $TM_1$ with disabled $e$, delete row details and update
        $TM_1$ status.
4.      Analysis of Second Matrix Object.
        $for\ (\vartheta = 0\ to\ C)$, check all of the items in $TM_2$ with enabled $e$ and send remained $UAP$ and
        $TAP$ tasks to Response Queue ($RQ$) and $CAP$ task to Checking Queue ($CQ$).
5.      Update Second Matrix Object.
        $for\ (\vartheta = 0\ to\ C)$, check all of the items in $TM_2$ with disabled $e$, delete row details and update
        $TM_2$ status.
⋮       ⋮
$x$.    Analysis of $z$th Matrix Object.
        $for\ (\vartheta = 0\ to\ C)$, check all of the items in $TM_z$ with enabled $e$ and send remained $UAP$ and
        $TAP$ tasks to Response Queue ($RQ$) and $CAP$ task to Checking Queue ($CQ$).
$x + 1$. Update Second Matrix Object.
        $for\ (\vartheta = 0\ to\ C)$, check all of the items in $TM_z$ with disabled $e$, delete row details and update
        $TM_z$ status.
```

After the classification process in the first phase and initializing Matrix TM_1 to TM_z in the second phase, tasks need to be classified according to authentication and authorization characteristics. In the task analysis phase, authentication tasks including UAP, TAP policies are scheduled in response queue that needs appropriate access response from cloud user.

On the other hand, most of authorization tasks are CAP-based and checked in checking queue to receive data from user database and the rest are based on UAP that are scheduled in response queue. After the scheduling process, the match gate updates the status of the task according to the number of remaining UAP, TAP and CAP tasks and the value of e for the task is changed. If there is any remaining authentication and authorization tasks, the row will be deleted from associated task management matrix. Figure 4.3 shows classification, initialization and analysis phases in details.

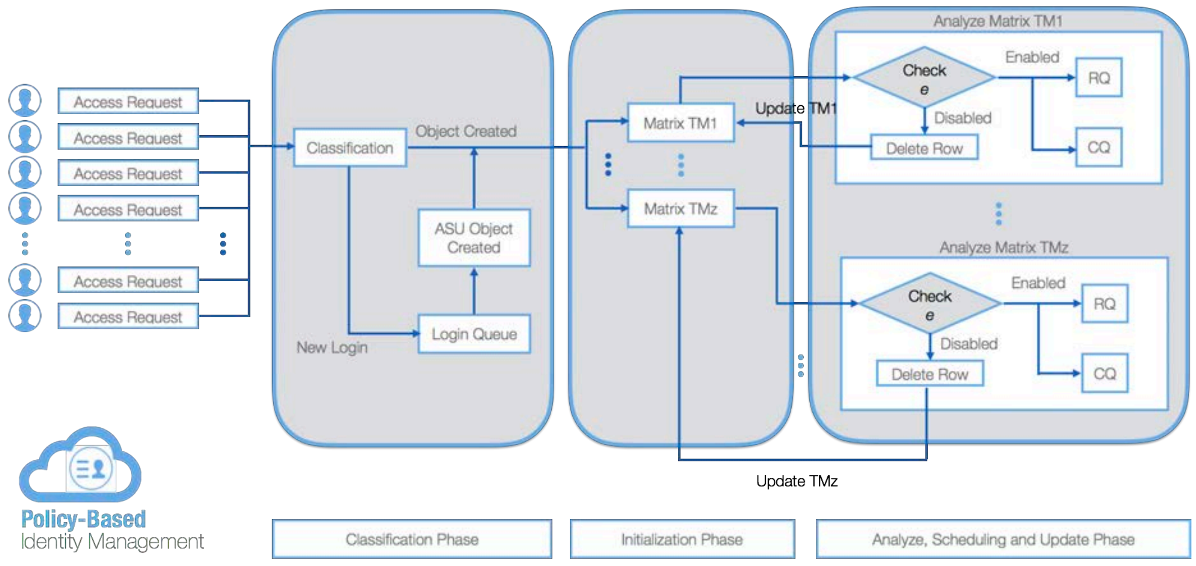


Fig. 5.3. Match Gate Task Management Schema

5.5. Discussion

In order to incarnate the superiorities of this schema in cloud-based environments, we give a security and performance analysis of the proposed model in this section. Accordingly, several scenarios are described as follows:

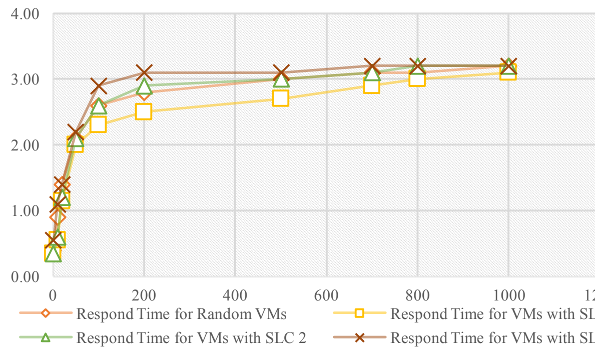
5.5.1. Case Study 1

In the first case study the performance of check point and match gate components are evaluated. Accordingly, 3 Security Level Certificates are defined (Table 4.2) and several virtual machines are established with different security sub-policies based on SLCs. Also, cloud users try to access these machines regarding to identification capabilities and security requirements.

Table 5.2. Defined Authentication and Access Protocols in SLCs

Protocol	SLC 1 (Low)	SLC 2 (Medium)	SLC 3 (High)
Authentication	User/Pass	User/Pass	User/Pass
Authentication	-	Second Pass	Second Pass
Authentication	-	-	One-Time Pass
Authorization	Permanent	Permanent	Permanent
Authorization	Temporary	Temporary	-
Authorization	-	Geo	Geo
Authorization	-	-	Software

The first aim of this experiment is to analyze the scalability of proposed model. This scalability experiment is classified to three part: (1) Number of virtual servers with different sub-policies, (2) Number of cloud users with access request, and (3) Simultaneous increase in both numbers of virtual servers and cloud users.



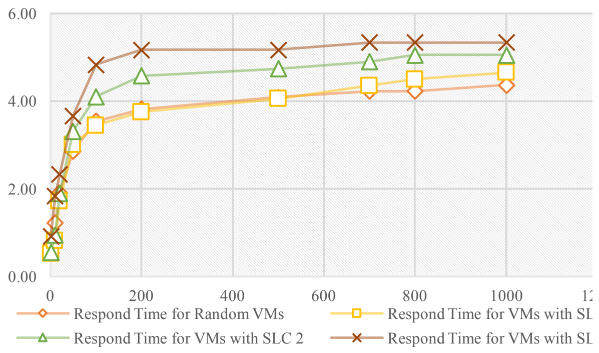


Fig. 5.4A. Respond Time By Increasing Number of VMs   Fig. 5.4B. Respond Time By Increasing Number of Users

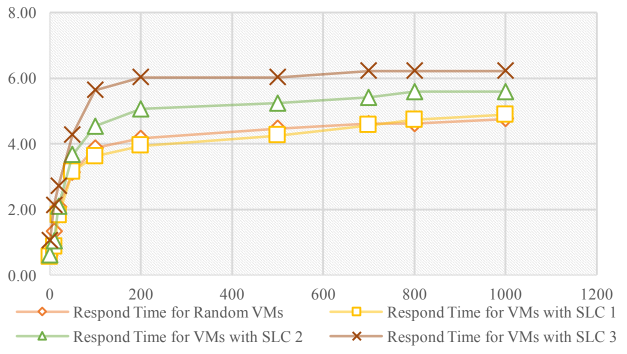


Fig. 5.4C. Respond Time By Simultaneous Increase in both Numbers of Virtual Servers and Cloud Users

Fig. 5.4. Respond Time for Proposed Identity Management Model

In the first experiment (Fig. 5.4A), the respond time for access requests by increasing the number of VMs with various sub-policies were examined. According to the results, the respond time was risen dramatically before the number of VMs were reached to 200 VMs. Nevertheless, this increased were continued with less intensity after 200 VMs and proves that the model is scalable in confronting with large number of servers in cloud-based environments. Furthermore, the respond time by increasing the number of access requests (Fig. 5.4B) had a same behavior specially in large number of users.

Finally, the last experiment (Fig. 5.4C) shows the respond time in simultaneous increase of access request and VMs with different defined sub-policies. According to the experiment results, the respond time for access requests was increased sharply in first stage and after a certain amount of requests and VMs, the increase slope remained approximately constant. As was expected, the sharp increase in requests for SLC3 is higher than other requests because of the complexity of defined polices in high secure levels. Moreover, the random requests for accessing to cloud servers as a real world experience shows better results than SLC2 and SLC3 and proves that the proposed model is scalable in large number of access requests or cloud servers with different access policies.

5.5.2. Case Study 2

In the second experiment the performance of match gate in different types of workloads was evaluated. Accordingly, the total process time for processing 500 access requests to VMs with SLC3 (high secure VMs with more authentication and authorization policies) was examined in the first step. The aim of this case study is to examine the effects of static, continuously changing, dramatic increase and predictable increase workloads on the performance of match gate task management. The experiment was in 6 rounds based on different types of workloads. Figure 5.5 shows the results in details.

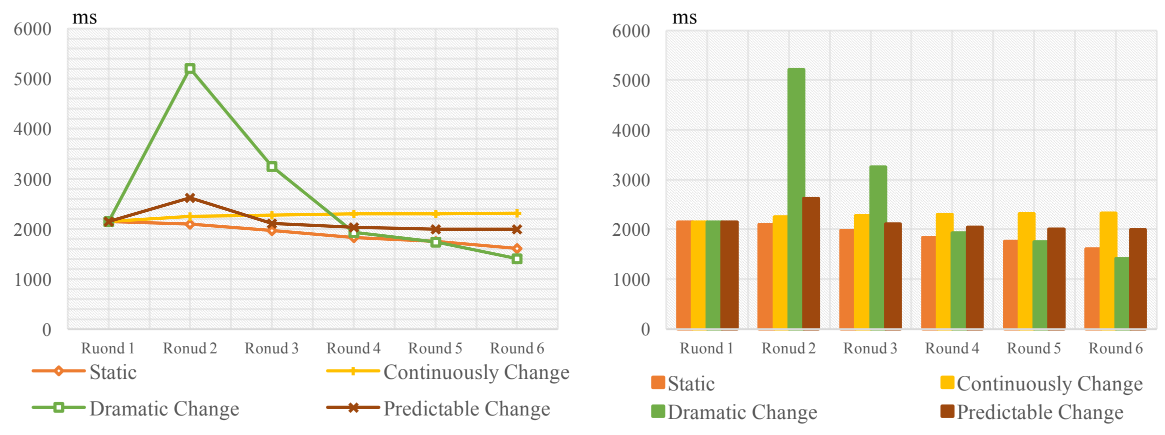


Fig. 5.5. Effects of Different Workload on the Performance of Match Gate Task Management

In the static workload, the number of user accesses was same in all rounds. However, the total processing time was decreased slightly due to the common policies in different VMs. Thus, there was not any necessity to re-check common policies. As expected, in the dramatic increase of users requests, the total processing time was risen dramatically and in the respective rounds the total processing time was reduced considerably to the normal range. This change was less in predictable change due to the predictable scheduling in associated task processing.

Finally, the rate of change in continuously increase of requests is significantly slight. That was because two different effects: increase due to the number of requests and decrease due to the common policies in different VMs. Overall, the results show the performance of match gate task management for semantic mapping of access polices to request was scalable enough in different types of workloads.

5.5.3. Case Study 3

The most important characteristics of the proposed model is the ability to map between defined polices and access requests. This mapping process is in both authentication and policy definition sides. To analyze the performance of the schema, a scenario was designed with a service provider with 3 SLCs (Table 5.2) and 20 VMs with different sub-policies (10 SLC1-based, 6 SLC2-based and 4 SLC3-based VMs). 100 users with different identification capabilities and authorization properties (*e.g.* role, geo, etc.) requested to access established VMs by match gate component. Each user tried to login basically in the first step and requested to access several VMs with different SLCs and sub-policies randomly.

In total, there were 2000 access requests from users (includes 1087 requests for accessing to VMs for the first time, 654 requests for accessing to VMs without new identification response, 168 request with expired session time and 91 un-authorized requests). Fig. 5.6 shows the performance analysis of Match Gate. According to the results, 100% of the granted accesses or prevented requests were syntactically and semantically analyzed with successful results. In fact, the proposed model guarantees the correctness of access management process by semantic mapping between user identities and defined policies.

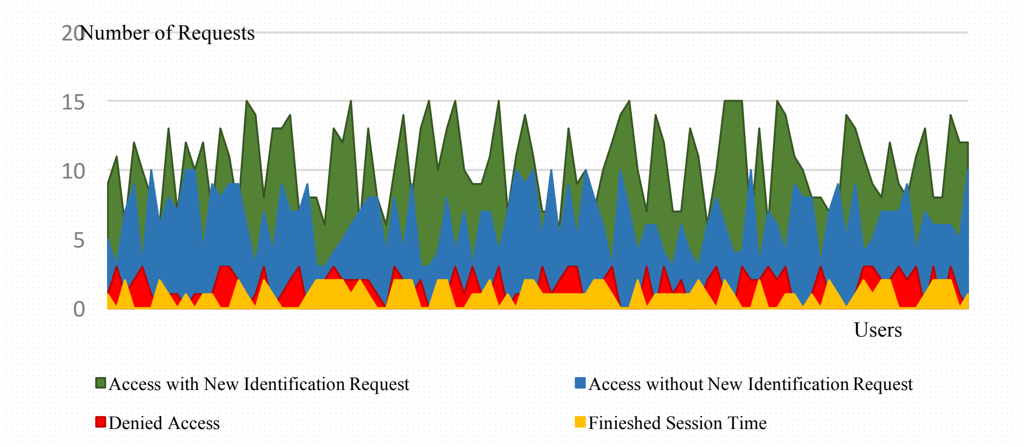


Fig. 5.6. Performance Analysis of Match Gate

5.5.4. Security Analysis

The security of policy-based user authentication model is granted in 3 main parts:

1. Security of Defined Policies: All of the policies are re-checked periodically to specify new or revoked polices of cloud servers according to timestamps.

2. Security of User Data: All of user authentication capabilities are secured and capable to resist against possible attacks and unpredicted events.

3. Security of Mapping Process: The process of mapping between access requests and defined policies are secured by using appropriate solutions to control timestamps, resist against possible attacks and map in a secure and semantic manner.

Table 5.3 shows the security analysis of the proposed model according to defined objectives. The analysis shows that this schema can resist against possible attacks by various methods and solutions during policy definition, policy application, user registration, identity management and mapping between access requests and defined polices. Each of these solutions were tested and several scenarios were designed to evaluate the security of suggested schema.

Table 5.3. Security Analysis of Proposed Model

Security Concern	Classification	Solution Method
Policy Definition	Policies	Using WS-Policy Definition of Secret Policy Value for Server n $X(S_n) = h(x_n) \oplus h(p_1) \oplus h(p_2) \dots \oplus h(p_n)$
Policy Update	Policies	Checking Servers Secret Key in Lowest Level of Security $TK_i = Dec(TK_i, x_n)$ Checking Secret Policy Value for Policy Checking $h(X(S_n)) = h(X'(S_n))$
User Registration	User	Using Token-Based Solution $TK_i = \left(h\left(h(ID_i \parallel TS) \oplus h(PW_i)\right)\right)$
Access Request	Identity Management	Boolean Value of e for Checking Current Status of User Updating Access Token by New Capabilities $TK_i = TK_i \oplus \left(h(AP_j \parallel TS) \oplus h(AR_j)\right)$
Mapping Requests	Identity Management	Generation of Confirmation Tag from Match Gate and Send to Cloud Server $NAR_{(i,n)} = (TK_i, Enc(TK_i, x_n), h(X'(S_n)), TS')$
Insider Attack	Identity Management	Dual Checking in Policy Match Gate and Cloud Server $TK_i = TK_i \oplus \left(h(AP_j \parallel TS) \oplus h(AR_j)\right)$ $h(X(S_n)) = h(X'(S_n))$
Impersonation Attack	Identity Management	Update Timestamp by Dual XOR to hashed Access Capabilities and Access Respond. $TK_i = TK_i \oplus \left(h(AP_j \parallel TS) \oplus h(AR_j)\right)$
Reply Attack	Identity Management	Encrypt and Decrypt the Access Tag from Match Gate to Cloud Server by Unique Secret Value of Cloud Sever based on Defined Access Policies.

5.5.5. Competitive Analysis

Table 5.4 shows the competitive analysis of the proposed model according to the objectives of an identity management schema in cloud-based environments. According to the results, the proposed model is able to provide a multi-level authentication process by policy definition in different cloud servers and dual authenticator in both match gate and cloud server to ensure about the security and efficiency of this process.

Table 5.4. Competitive Analysis of Proposed Model

Objectives	OpenID	Sood	Kumari	Lu	Proposed Model
Policy-Based	Yes	Yes	No	No	Yes
Multi-Level Authentication	Yes	No	Yes	Yes	Yes
Light Time & Computation Load	No	No	No	Yes	Yes
Multi-Level Token-Based	No	No	No	No	Yes
Cloud Server Dual Authenticator	No	No	No	No	Yes
Multi-Level Policies by Protection Ontology	No	No	No	No	Yes
Authentication Task Manager	Yes	No	Yes	No	Yes
Scalability	Yes	No	No	No	Yes
Semantic Analysis of Access Request	No	Yes	No	No	Yes

Furthermore, the proposed model is significantly scalable in comparison with most of authentication models to have a better performance in face with large number of access request and defined policies. Finally, this schema uses semantic analysis of access request with an efficient authentication task management and token-based identity management to ensure about the security and reliability of this challenging issue in cloud computing environments.

5.6. Conclusion

According to the previous research results, a scalable policy-based identity management was presented in this chapter to address two main problems: (1): Lack of coincidence in identity management models based on defined policies and various security levels in different cloud servers, (2): Lack of multi-objective authentication task management according to the defined policies in multi-level authentication procedures [99].

Therefore, a policy-based user authentication model was presented to provide a reliable identity management mechanism for establishing multiple access policies in different virtualized nodes and mapping access requests to defined policies based on capabilities of cloud servers and requirements of resources. Moreover, a structural policy language and policy engine were introduced for policy generation, application and management.

The proposed model provides an authentication schema with 4 main components (*i.e.* Policy Engine, Policy Database, Check Point and Match Gate) to define access policies by cloud servers, to apply policies according with Protection Ontology, to manage user identities and to map access request by cloud users with defined polices semantically.

This model was evaluated with performance, security and competitive analysis, and the reliability and efficiency of the suggested schema was assured for managing user identities in different cloud servers with various levels of security.

In addition to the effects of our policy-management model in authorization and authentication procedures, there is a challenging issue in resources protection specially during user revocation, failure or other unpredictable scenarios. In fact, the security of associated data and resources should be granted by service provider according to defined policies. In the next two chapters a reliable re-encryption and effective user revocation models are introduced based on our policy management system to ensure the security of data in case of unpredictable scenarios.

Chapter 6

6. Policy-Based Re-Encryption Schema

6.1. Introduction

Using cryptographic models are the most common solutions to ensure data and resource protection in virtualized environments. To guarantee the reliability of these encryption models and to make sure the data confidentiality and fine-grained access control in cloud computing environments, stored data and resources needs to be re-encrypted periodically or based on special mechanisms such as revoked user-based or manual re-encryption [81].

Managing the process of re-encryption is a challenging issue that involves many limitations such as time management, resource confidentiality, and level of access. Therefore, an efficient re-encryption management may increase the reliability and the rate of security in cloud computing environments. Hence, a multi-level re-encryption model based on policy management has been presented in this chapter to ensure data security in cloud computing environments. The proposed model uses a policy-based ontology to generate, manage and apply re-encryption policies based on the characteristics of resources, sensitivity of data and capabilities of service provider.

6.2. Related Works

The most popular re-encryption models are based on attributes for managing and monitoring security of resources. These attributes are defined as properties of re-encryption class to classify resources based on sensitivity and priority. Hierarchical Attribute-Based Encryption (HABE) is one of the suggested models [82] that use data consistency and data confidentially attributes for high performance and full delegation re-encryption process. The main drawback of this model is the dependency of the HABE performance on reliability of cloud infrastructure. This means, the correctness of the re-encryption process is completely dependent on the rate of security in cloud infrastructure.

This problem was solved in R3 model by using a time-based re-encryption approach [83], in this model the underlying cloud infrastructure was not necessarily reliable in order to ensure correctness. Furthermore, the time difference between cloud server and data owner is an important issue in time-based re-encryption models that was solved in R3 with appropriate clock synchronization.

The performance of time-based re-encryption was improved by Liu et al. [83] to determine a period of time according to defined parameters for re-encrypting stored data, generating new key and automatic expiring of revoked user's access. In this model, concepts of attribute-based re-encryption and proxy re-encryption were combined with sets of time attributes. Therefore, only users whose attributes satisfy the access structure and whose access rights are effective in the access time can recover corresponding data.

One of the other attribute-based re-encryption models was Key Policy-Attribute Based Encryption (KP-ABE) that was proposed by Park et al. (2006). In this approach, internal nodes are threshold gates and leaf nodes are associated with attributes that are used to encrypt data. This model was improved [85] by adding some techniques such as Typed-Based Proxy Re-Encryption [86] and bilinear mapping for providing selectively delegate decryption right using Typed-Based Proxy Re-Encryption.

The main problem of this model was the dependency of KP-ABE on specific attributes that decreased the compatibility of this model in virtualized infrastructure and cloud-based environments. In fact, this model uses single level re-encryption policies and this mechanism declined semantic mapping between policies and capabilities.

To solve the problem of single level policies for reliable re-encryption, several multi-level policy management schemas were proposed. Di Modica and Tomarchio (2011) suggested one of the first policy-based classification approach's in clouds that leverages on the semantic technology to enrich standardized security policies with an ad-hoc content and to enable machine reasoning which is then used for both the discovery and the composition of security-enabled services. In this model, requirements and capabilities for cloud customers and providers are defined within policies which are adopted to policy intersection mechanism provided by WS-Policy [32]. WS-Policy is a recommended framework from W3C for policy specification of Web Services that includes policies that are defined as a collection of alternatives contain assertions to specify well-established characteristics for using selection of various services (*e.g.* requirements, capabilities or behaviors).

Overall, the main concerns in current re-encryption models in clouds are dependency of suggested models on specific attributes in property-based models that has been not adopted to virtualized infrastructure and lack of scalability and flexibility in semantic mapping of policies in policy-based re-encryption models.

6.3. Proposed Model

To establish a policy-based re-encryption schema, several components need to be defined for generating, storing, managing and applying policies to cloud-based resources according to capabilities of service provider, time limitation and other constraints. Hence, Policy-Based Re-Encryption Schema (POBRES) is proposed, including 4 main components to define and manage policies in cloud computing environments (Fig. 6.1).

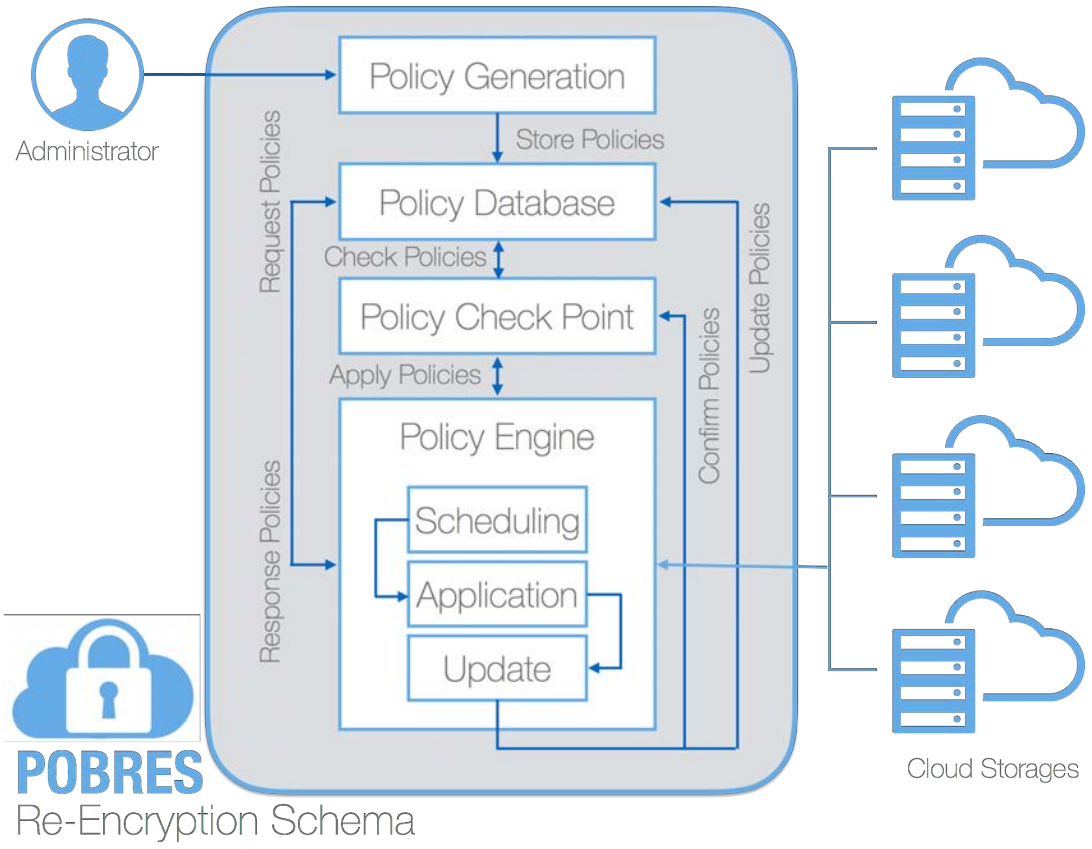


Fig 6.1. Policy-Based Re-Encryption Schema (POBRES)

6.3.1. Policy Generation Component (PGC)

The main duty of this component is to define re-encryption policies based on requirements of cloud customers, capabilities of service provider and settled details in SLA. In fact, PGC uses a structural ontology to expound and clarify policies and sub-policies.

This ontology is based on three levels of capabilities, scope and credential to define main re-encryption policy structure with associated sub-policies, potential resources for application process and reputation details for managing process respectively. Figure 6.2 shows the structure of POBRES ontology: *high level class* in details.

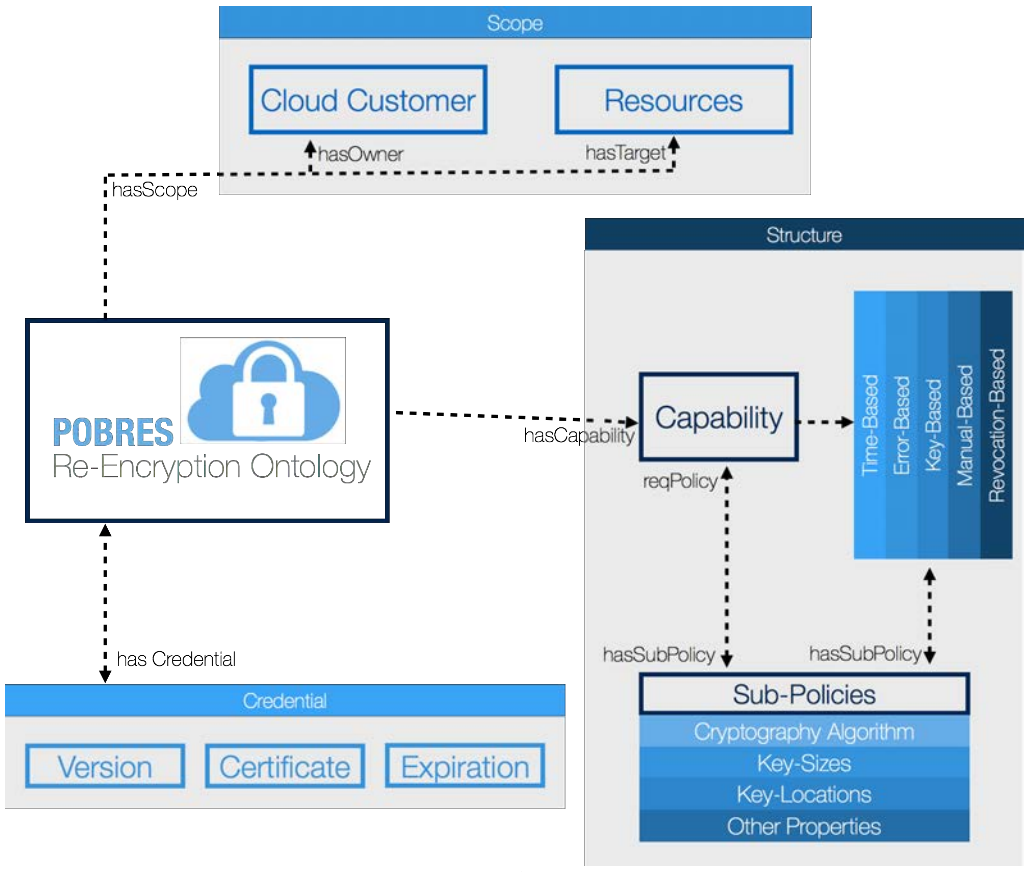


Fig 6.2. POBRES Ontology: *High Level Class*

POBRES ontology is classified to three main sub-classes to define re-encryption policies:

- *Structure*: Including re-encryption capabilities of service provider, properties and other sub-policies and encryption characteristics of security level. The structure of POBRES is based on WS-Policy [32] as a recommended policy language standard by W3C.

In the following, an example of POBRES certificate is given:

```
<wsp:Policy>
…namespace definition…
   <wsp:POBRES rdf:ID="#hf4h#hd5f">
      <wsp:Scope>
      …
      </wsp:Scope>
      <wsp:Credential>
      …
      </wsp:Credential>
      <wsp:Structure>

        <security:ReEncryption  rdf:ID="ReEncryptionRequirement">
           <security:Capability rdf:ID="CapabilityRequirement">
                <rdf:HLSP_Capability
                rdf:HLSP_Time="240"
                rdf:HLSP_Error="#443#352#404#252" rdf:HLSP_Manual="#T363#y5y3"
             </security:Capability>
          </security: ReEncryption>

        <security:Cryptography rdf:ID="CryptographyRequirement">
           <security:Encryption rdf:ID="EncryptionRequirement">
                <rdf:HLSP_Encryption
             rdf:HLSP_Algorithm="AES"
             rdf:HLSP_KeySize="#256" rdf:HLSP_KeyLocation="#TE43hs3g"
           </security:Encryption>
         </security: Cryptography>

       </wsp:Structure>
    </wsp:POBRES>
</wsp:Policy>
```

In this example an AES encryption algorithm with 256bits key size associated with time, error and manual re-encryption methods are defined for security level of stored resources in cloud storages.

- *Scope*: Defines actual and potential resources for re-encryption level.
- *Credential*: Describes certificate details of re-encryption policy.

According to POBRES ontology, the process of policy generation for a specific security level is described as follows:

Step. 1. A cloud customer applies for a re-encryption policy in policy generation engine. An object is created from POBRES class (RP_i):

$$RP_i \;\; POBRES = POBRES(CC_n, V)$$

where CC_n and V are cloud customer ID and version of re-encryption policy respectively.

Step. 2. Re-encryption main policies are selected by cloud customer according to the defined capabilities of service provider: $\{CP_1, CP_2, \dots, CP_j\}$ where CP is a re-encryption capability that is offered by cloud provider.

Step. 3. Sub-policies for each policy are defined by cloud customer based on requirements (*e.g.* time units for time-based re-encryption, error numbers for error-based re-encryption and authorized user IDs for manual-based re-encryption):

$$for \left(i = 0 \; to \sum_{\alpha=0}^{R} (rdf\!: HLSP\_Policy)_{\alpha} \right)$$
$$\left\{ HLSP_i = \left(HLSP_i \;\; \| \; add\left(SP_{\mu}\right) \right) \right\}$$

where $HLSP$, SP and R are high level sub-policy, defined values of sub-policies and total number of main policies respectively.

Step. 4. The re-encryption policy is linked to encryption mechanisms based on resources of cloud customers:

$$RP_i.\, LinkCrypto(Enc, K_s, K_l);$$

where Enc, K_s and K_l are encryption algorithm, key size and key location respectively.

Step. 5. The expiration time of re-encryption policy (ET) is settled and finally the certificate ($Cert$) is generated by service provider according to WS-Policy and POBRES ontology. The following elements are allocated to re-encryption policy set:

$$RP_i = \left\{ \begin{matrix} CC_n, V, h(C_i), h(E \oplus C_i), (CP_1, h(SP_I), h(SP_{II}), \dots), \dots, \\ , \dots, (CP_R, h(SP_I), h(SP_{II}), \dots), (Enc, K_s, h(\, K_{loc})) \end{matrix} \right\}$$

Step. 6. The re-encryption policy set and certificate are sent to re-encryption policy database.

$$Send\ (RP_i\ , Cert_i);$$

Table 6.1. The Notations of Proposed Model

Notations	Description	Notations	Description
RP_i	Re-Encryption Policy Object	CC_n	Cloud Customer ID
V	Version of Re-Encryption Policy	$HLSP$	High Level Sub Policies
SP	Sub Policy Value	R	Total Number of Main Policies
Enc	Encryption Algorithm	K_s	Key Size for Encryption Algorithm
K_l	Key Location	ET	Expiration Time of Re-Encryption Policy
$Cert$	Certificate	C_i	Certificate ID
$h()$	One-Way Hash Function	$\oplus$	Exclusive-Or Function
$\Vert$	String Concatenation Operation	x	Total Number of Resources Assigned to a Re-Encryption Policy
RES	Set of Resources	$RTASK$	Re-Encryption Task Class
TID	Re-Encryption Task ID	TST	Re-Encryption Task Status
CI	Classification Index for Task Object		

6.3.2. Policy Database

All of the generated POBRES re-encryption policies, sub-policies and certificates are stored in Policy Database. Each policy is assigned to set of resources based on cloud customers' requirements.

$$RES_\pi(CC_n, RP_i) = \{Rs_1, Rs_2, \dots, Rs_x\};$$

6.3.3. Policy Check Point (PCP)

To manage policies and establish an efficient communication between defined policies and re-encryption tasks, Policy Check Point (PCP) is introduced. PCP uses a policy task class to create re-encryption tasks management according to the characteristics of requests.

$$RTASK\ \ RT_u = RTASK(TID, TST, RP_i, RES_\pi)$$

Where $RTASK, TID$ and TST are re-encryption task class, task ID and task status respectively. The main job of PCP is to call re-encryption tasks based on characteristics of defined policies as follows:

1. To schedule and call time-based policies according to the re-encryption timeline in policy database.

$$if\begin{pmatrix} \left(RP_i.h(E\oplus C_i) = Cert_i.h(HLSPExpire\oplus C_i)\right)\ and \\ \left(h(C_i) = h(Cert_i.ID)\right)\ and \\ (\exists\ SP \in CP_\gamma | h(SP) = h(HLSPTime)) \end{pmatrix}$$
$$then\ RTASK\ \ RT_u = RTASK(TID_\delta, TST_\delta, RP_i, RES_\pi)$$

2. To check associated polices with revoked users.

$$if\begin{pmatrix} \left(User_\beta.Status = Revoked\right)\ and \\ \left(h(C_i) = h(Cert_i.ID)\right)\ and \\ \left(\exists\ SP \in CP_\gamma \middle| h(SP) = h(HLSPRevoke)\right)\ and \\ \left(RP_i.h(E\oplus C_i) = Cert_i.h(HLSPExpire\oplus C_i)\right) \end{pmatrix}$$
$$then\ RTASK\ \ RT_u = RTASK(TID_\delta, TST_\delta, RP_i, RES_\pi)$$

3. To organize occurred errors and associated policies according to error details.

$$if\begin{pmatrix} \left(h(ErrorID) = h(HLSPError)\right)\ and \\ \left(h(C_i) = h(Cert_i.ID)\right)\ and \\ \left(\exists\ SP \in CP_\gamma \middle| h(SP) = h(HLSPError)\right)\ and \\ \left(RP_i.h(E\oplus C_i) = Cert_i.h(HLSPExpire\oplus C_i)\right) \end{pmatrix}$$
$$then\ RTASK\ \ RT_u = RTASK(TID_\delta, TST_\delta, RP_i, RES_\pi)$$

4. To authorize manual re-encryption requests based on authentication process and approved access.

$$if\begin{pmatrix} \left(User_\beta.SSO = Approved\right)\ and \\ \left(h(C_i) = h(Cert_i.ID)\right)\ and \\ \left(\exists\ SP \in CP_\gamma \middle| h(SP) = h(HLSPManual)\right)\ and \\ \left(RP_i.h(E\oplus C_i) = Cert_i.h(HLSPExpire\oplus C_i)\right) \end{pmatrix}$$
$$then\ RTASK\ \ RT_u = RTASK(TID_\delta, TST_\delta, RP_i, RES_\pi)$$

5. To schedule re-encryption tasks according to the loss of keys.

$$if \begin{pmatrix} (\exists\ Rs\ \in RES_{\pi}\mid Rs.Key.status = lost)\ and \\ \big(h(C_i) = h(Cert_i.ID)\big)\ and \\ (RP_i.Enc.K_s.Status = lost)\ and \\ \big(RP_i.h(E \oplus C_i) = Cert_i.h(HLSPExpire \oplus C_i)\big) \end{pmatrix}$$

$$then\ RTASK\ \ RT_u = RTASK(TID_{\delta}, TST_{\delta}, RP_i, RES_{\pi})$$

According to each scenario, policy check point creates and object from re-encryption policy task class and sends to Policy Engine to process the re-encryption request.

6.3.4. Policy Engine

Policy Engine uses a scheduler to manage re-encryption tasks based on their characteristics. In fact, re-encryption policies are scheduled, applied and updated according to created policy task objects from PCP.

Let $T = \{RT_1, RT_2, \dots, RT_U\}$ represents all re-encryption task objects that are received from PCP and each object, RT_u $(u \in \{1,2,\dots,U\})$ is associated with $(TID_{\delta}, CI_{\delta})$ where CI is classification index of re-encryption object. Algorithm 6.1 shows the scheduling process in details:

Algorithm 6.1. Re-Encryption Task Management

Input: $T = \{RT_1, RT_2, \dots, RT_U\}$: the set of re-encryption task objects received form PCP

Output: Scheduled task queue.

```
1. Task Classification
   Re-encryption tasks are classified based on the characteristics.
   $for \left(i = 0\ to \sum_{\alpha=0}^{U} RT_u\right)$
   $\{if\ (RT_u.CI = Time)\ then\ Classify(RT_u, TimeSet)$
   $if\ (RT_u.CI = Error)\ then\ Classify(RT_u, ErrorSet)$
   $if\ (RT_u.CI = Revoke)\ then\ Classify(RT_u, RevokeSet)$
   $if\ (RT_u.CI = Manual)\ then\ Classify(RT_u, ManualSet)$
   $if\ (RT_u.CI = Key)\ then\ Classify(RT_u, KeySet)\}$
2. Sort Phase
   Each set is sorted based on the characteristics of re-encryption policy. Time-based policies are
   sorted regarding to the settled time, Error-based policies are sorted according to the prioritize index
   of error numbers, rest of policies are sorted based on the priority index of security level according
   to encryption policy.
   $Sort(TimeSet, RT_u.RP_i.CP_{time}.SP, Deadline);$
   $Sort\big(ErrorSet, RT_u.RP_i.CP_{Priority}.SP, ErrorPriority\big);$
   $Sort(RevokeSet, RT_u.RP_i.CP_{Revoke}.SP, PriorityIndex);$
```

$Sort(ManualSet, RT_u.RP_i.CP_{Manual}.SP, PriorityIndex)$;

$Sort\big(KeySet, RT_u.RP_i.CP_{key}.SP, PrioirtyIndex\big)$;

3. **Scheduling and Application**

   Each set is assigned to associated VM in cloud server to schedule according to the priority of the re-encryption task. Furthermore, if one of associated VMs is capable to process re-encryption tasks from other sets due to less number of requests, scheduler assigns other re-encryption tasks to the VM with lower workload.

4. **Task Status Update**

   Whenever a re-encryption policy applies to the related resource, the status of the task is changes as follows: $RT_u.TST_\delta = done$

5. **Update Phase**

   All of the task objects with $done$ value are send to policy check point for confirmation and updating the version, certificate and cryptography details as follows:

   $Update(RP_i.V)$

   $Update\big(RP_i.h(C_i)\big)$

   $Update\big(RP_i, h(E \oplus C_i)\big)$

   $Update\big(RP_i.Enc.h(K_{loc})\big)$

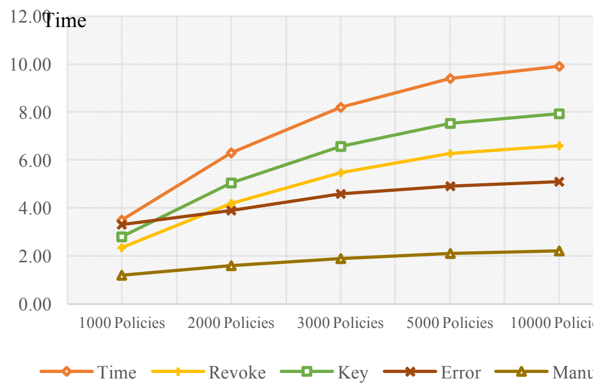
6.4. Discussion

In order to incarnate the superiorities of this schema in cloud-based environments, we give a security and performance analysis of the proposed model in this section. Accordingly, several scenarios are described as follows:

6.4.1. Case Study 1 (Performance of Policy Check Point)

In the first case study, the performance of policy check point was evaluated. The aim of the first case study was to ensure about the creation of re-encryption tasks according to the number of defined policies. Accordingly, 1000 re-encryption policies (200 for each capability) were defined in the first step and the performance of PCP in analyzing and processing policies were examined. This experiment was replicated by increasing number of defined policies to 10000 in policy database. Figure 5.3 shows the results in details. According to Fig 6.3A, total processing time in PCP for each capability was examined and the results showed the increase rate of processing time was considerably declined after 3000 numbers of policies and it proved the model is scalable by dramatic increase of re-encryption policies.

Fig 6.3B shows the total process time of all policy types in PCP. According to the results, the total processing time was increased significantly between 1000 to 3000 defined policies and after that the total processing time was increased slightly. Finally, Fig 6.3C classified object creations from re-encryption policy task class according to the type of capability. The results showed more than 25% of re-encryption requests were related to time-based re-encryption policies and less than 10% of re-encryption requests were manual-based. It helped to classify re-encryption resources (VMs) to task sets more efficient and based on approximate estimation of re-encryption task object numbers.



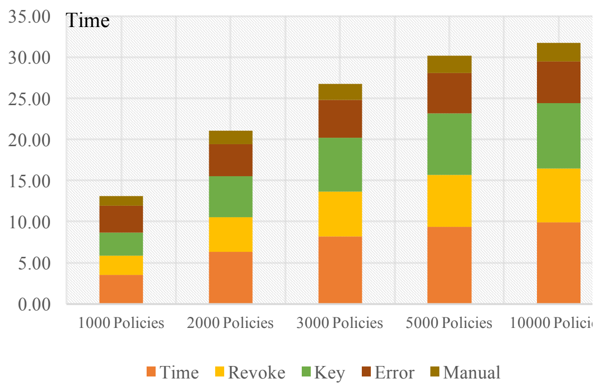


Fig. 6.3A. Total Process Time in PCP for Each Capability by Increasing Number of Policies in Policy Database

Fig. 6.3B. Total Process Time in PCP by Increasing Number of Policies in Policy Database

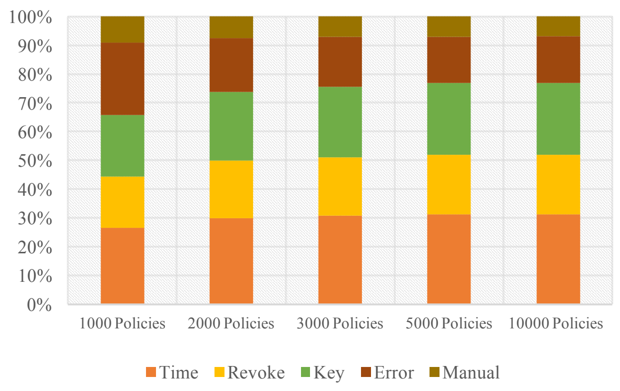


Fig. 6.3C. Classification of Re-Encryption Requests According to Capability Type of Defined Policies

Fig. 6.3. Performance of Policy Check Point by Increasing Number of Policies in Policy Database

6.4.2. Case Study 2

In this scenario, the performance of policy engine and policy application process were evaluated. Thus, four types of workloads were defined: Static Workload, Dramatic Changing Workload, Predictable Workload and Random Workload. Also, the number of re-encryption tasks were increased from 50 to 3000.

Figure 6.4 shows the performance of policy engine in different types of workloads and capability types. According to the results, the highest increase rate was for dramatic re-encryption task workload due to error-based or key-based tasks. Moreover, predictable workloads for time-based tasks increased slightly with approximate static slope. Finally, manual-based tasks with random workload had an unpredictable and unarranged behavior by increasing the number of re-encryption tasks from 50 to 3000.

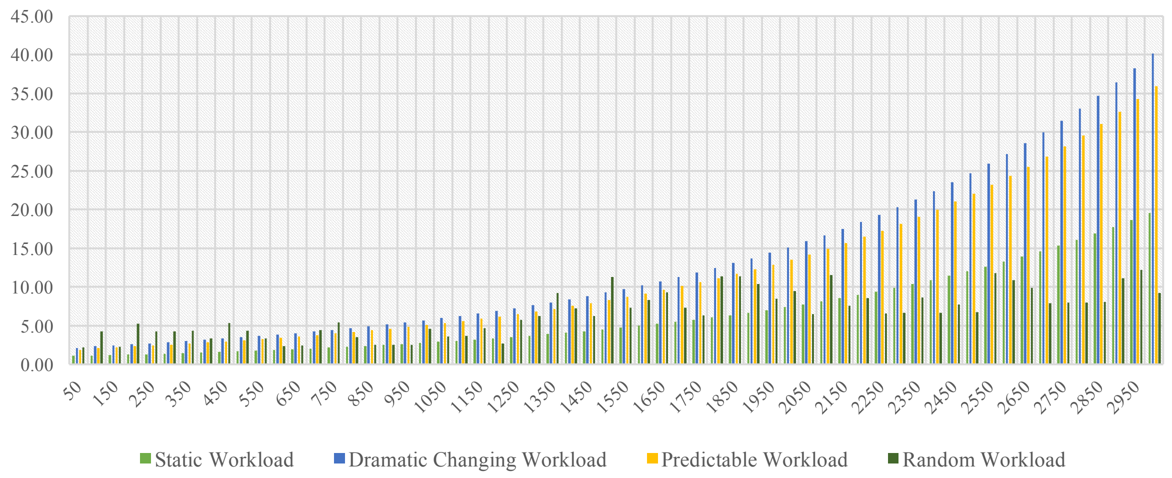


Fig. 6.4. Performance of Policy Engine in Different Workloads by Increasing Number of Re-Encryption Tasks

Table 6.2. Security Analysis of Proposed Model

Security Concern	Classification	Solution Method
Security of Policy Levels	Policy Definition	Using Version, Certificate and Expiration Time in POBRES Ontology $V, h(C_i), h(E \oplus C_i),$
Security of Policy Certificate	Policy Definition	Using Exclusive or and Hash Function for Ensuring the Security of Certificate $h(C_i), h(E \oplus C_i),$
Security of Applied Polices	Policy Checking	Checking the Certificate in Each Step of Policy Management $\left(RP_i.h(E \oplus C_i) = Cert_i.h(HLSP \oplus C_i)\right)$
Security of Capabilities	Policy Checking	Checking Capabilities in Policy Object and POBRES Certificate $(\exists\, SP \in CP_\gamma \mid h(SP) = h(HLSP))$
Security of Task Management	Policy Application	Task Classification Based on Characteristics.
Prioritize Re-Encryption Tasks	Policy Application	Sorting Re-Encryption Tasks based on Priority $Sort(TimeSet, RT_u.RP_i.CP_{time}.SP, Deadline);$ $Sort\left(ErrorSet, RT_u.RP_i.CP_{Priority}.SP, ErrorPriority\right);$ $Sort(RevokeSet, RT_u.RP_i.CP_{Revoke}.SP, PriorityIndex);$ $Sort(ManualSet, RT_u.RP_i.CP_{Manual}.SP, PriorityIndex);$ $Sort\left(KeySet, RT_u.RP_i.CP_{key}.SP, PrioirtyIndex\right);$

Security of Scheduled Tasks	Policy Application	All of the task objects with $done$ value are send to policy check point for confirmation and updating the version, certificate and cryptography details as follows: $Update(RP_i.V)$ $Update(RP_i.h(C_i))$ $Update(RP_i, h(E \oplus C_i))$ $Update(RP_i.Enc.h(K_{loc}))$

6.4.3. Security Analysis

The security of a policy-based re-encryption schema is granted in 3 main parts:

- Security of Defined Policies: All of the defined policies needs to be securely stored in database.
- Security of Re-Encryption Tasks: All of the re-encryption policies needs to be applied and managed reliably.
- Security of Re-Encryption Task Management: The re-encryption policies needs to be applied based on the priority and importance of policies and sensitivity of resources.

Table 6.2 shows the security concerns and proposed solutions of the proposed model in details.

6.4.4. Competitive Analysis

Table 5.3. shows the competitive analysis of the proposed model according to the objectives of a policy-based re-encryption schema in cloud-based environments. According to the results, the proposed model is able to provide a multi-level re-encryption process by policy definition in different re-encryption scenarios, policy management and policy application by different components.

Table 5.3. Competitive Analysis of Proposed Model

Objectives	HABE	TPRS	KP-ABE	3REM	POBRES
Policy-Based Re-Encryption	No	No	Yes	No	Yes
Multi-Level Ontology	No	No	No	Yes	Yes
Policy Mapping	No	No	No	No	Yes
Re-Encryption Task Management	Yes	Yes	Yes	No	Yes
Security Level Certificate	No	No	No	No	Yes
Policy Check Point	No	No	No	No	Yes
Language-Based	Yes	Yes	Yes	No	Yes
Cloud-Based	Yes	Yes	Yes	Yes	Yes
Re-Encryption Task Classification	No	Yes	No	No	Yes

6.5. Conclusion

As described in previous chapters, managing the process of re-encryption is a challenging issue that involves many limitations such as time management, resource confidentiality, and level of access. Therefore, a policy-based re-encryption model was presented in this chapter to define, establish and manage re-encryption policies in cloud computing environments. To achieve this purpose, a multi-level ontology was defined based on WS-Policy.

This ontology helps re-encryption components to provide reliable re-encryption policy definitions, generation and management in clouds. The results of comprehensive performance and security evaluation of proposed model shows POBRES increases the reliability of re-encryption processes in cloud storages considerably and provides an efficient policy management for re-encryption tasks.

As described before, protecting data according to defined policies is one the most important aims of our policy management system. In addition to the described reliable re-encryption model, an effective user revocation schema is presented in the next chapter according to our policy management system to ensure the process of data protection based on defined policies by cloud customers and sensitivity of data.

Chapter 7

7. POLICY-BASED USER REVOCATION SCHEMA

7.1. Introduction

One of the major risks in modern distributed networks is to guarantee the privacy and security of resources after the process of user revocation by admins or owners. Typically, all of the associated authentication, access management and data protection (*e.g.* Encryption) processes are affected after a user is removed from accessing to specific cloud resource by the owner or admin [104]. Hence, an efficient process should be provided to receive and manage user revocation requests and to review and update the security and privacy of associated resources.

In modern cloud computing, all of the security and privacy requirements of cloud customers are defined as structural policies [37]. These policies are classified to several security protocols such as authentication, access control, cryptography, transport and key management [41]. The problem of managing user revocation requests in policy-based cloud computing is the main focus of this work. In fact, each of revocation requests should be mapped to defined policies of associated resources in the request for evaluation of the user revocation process and updating defined security policies.

Accordingly, an effective user revocation model is presented in this chapter for mapping revocation requests to defined policies of associated resources to guarantee the authenticity of security and privacy policies in cloud-based resources.

7.2. Related Works

Most of current user revocation models is based on Attribute-Based Signature (ABS). In fact, the most challenging issue in encryption-based user revocation models is to re-encrypt data and manage associated keys after a user is revoked from cloud services [87]. One of these revocable ABS models was proposed by Escala *et al.* [88] proved to be adaptive secure in the standard model. This schema assigns a randomly selected identity to each user in addition to the attributes associated with an external entity to keep a secret verification key and a list of revoked user identities. Also, the verification key is used to trace a signature to the signer. However, this model conflicts with the unlink-ability and anonymity properties of ABS schemes.

Using an external party as a mediator to manage instantaneous user revocation [89] or structural timestamps associated with the attribute private key [90] are two extra features that was proposed for ABS-based user revocation models. The main drawback of these features is the potential overhead and performance impact due to the lack of immediate user revocation process. To decrease these overheads, Attributed-Based Group Signature (ABGS) schemas are proposed to provide anonymity for users in a group and generate a signature on behalf of the group [91]. The validation process can only verify the correctness of the signature, and whether it is produced by a valid user in the group. However, this scheme relies on the group manager to link a signature to a signer before the signature is revoked.

Panda [92] is a public auditing mechanism for shared data that was proposed with an efficient user revocation mechanism. The idea of proxy re-signatures was used in this model to allow the cloud to resign blocks on behalf of existing users during user revocation, so that existing users do not need to download and re-sign blocks by themselves. Also, Panda uses a public verifier to audit the integrity of shared data without retrieving the entire data from the cloud, even if the cloud re-signs a part of shared data. This main drawback of this model is the considerable processing power for manipulating sensitive and also non-sensitive data after the process of user revocation. In fact, the whole associated resources need to be updated to ensure the security of cloud after a user is removed from accessing.

To decrease the processing power and provide an efficient user revocation model, each revocation request needs to be processed according to the defined security policies. In fact, security and privacy policies (*i.e.* encryption, signature, access control and authentication) for each associated cloud node specifies whether the additional manipulations and processes are needed or not [93]. Accordingly, this chapter introduces a policy-based user revocation schema to process revocation requests based on defined policies and capabilities of services provider.

7.3. Problem Description

As described in previous section, the main aim of this model is to provide an efficient and reliable user revocation process to ensure the security and privacy of associated cloud-based resources [100]. Assume that there are N physical or virtualized cloud servers in data center, denoted as $\{S_1, S_2, \dots, S_N\}$, and each of servers (S_n) is associated with a policy set $PS(S_n) = \{PS_1, PS_2, \dots, PS_M\}$ where PS_m is a defined access policy. Given X registered users $\{U_1, U_2, \dots, U_X\}$ that have been revoked from accessing to one or several cloud servers and each

$U_x: x \in (1,2,\dots,X)$ is associated with revocation set $\{U_x, \{S_1,\dots,S_r\}\}$ where $\{S_1,\dots,S_r\} \subset \{S_1,S_2,\dots,S_N\}$.

Table 7.1. The Notations of Proposed Model

Notations	Description
S_n	Cloud Node
$P(S_n)$	Set of Policies for for n
PS_m	Defined Policy
U_x	User to be Revoked
$Policy_{Type}$	Policy Type (Cryptography, Access, Authentication)
T_r	Time-Stamp of Received Request
T_r'	Time-Stamp of Cloud Node
$h(\,)$	Hash Function
TS	Temporary Suspension
$RF(S_n)$	Revocation Flag for Cloud Node
RFS	Revocation Flag Status

Each cloud server has some access policies including cryptography mechanisms, role or reputation access controls and additional authentication policies. The proposed user revocation model tries to map the capabilities of the revoked user to defined policies of associated resource for eliminating the access pass of revoked user and for managing and updating access policies in associated resource according to revocation request.

7.4.Policy-Based User Revocation Model

There are several policy-based security models such as [100], [38] and [43] that have been proposed to multi-level security schema for classification of cloud-based resources to several security rings. A policy-based classification uses security protocols such as cryptography, access control and authentication to provide security levels based on capabilities of service provider and requirements of cloud users.

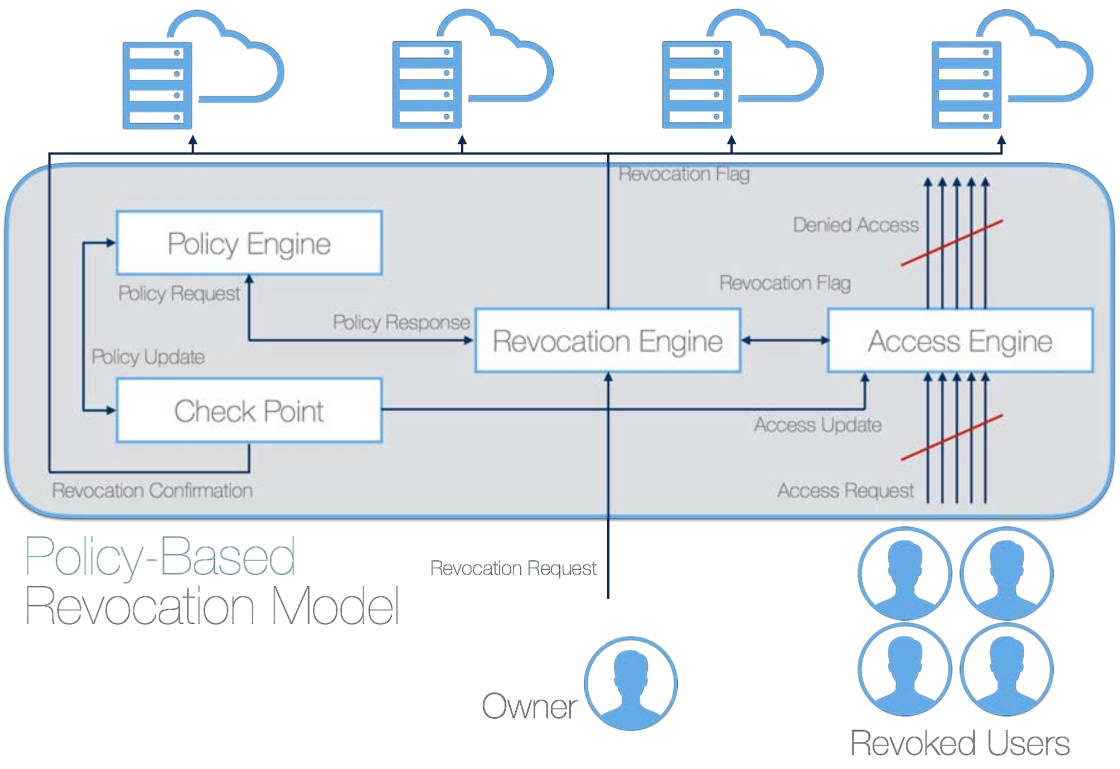


Fig. 7.1. Overview of Policy-Based Revocation Model

As described, each cloud node S_n is associated with a policy set $PS(S_n)$ includes cryptography (encryption, re-encryption and key management policies) , authentication (normal, double and one-time policies) and access control (role-based or reputation-based) policies. An effective user revocation model is based on associated policies of a cloud node and capabilities of cloud user. The proposed model is based on four main components and Figure 7.1 shows the overview of the model.

7.4.1. Policy Engine

The main duty of Policy Engine is to define security policies for each cloud node according to the requirements and capabilities of cloud provider. Each cloud node has a policy set $PS(S_n) = \{PS_1, PS_2, \dots, PS_M\}$ where PS_m is a defined access policy that is classified to 3 categories: Cryptography, Access Control and Authentication. Each policy is defined as follows:

$$PS_m = \{Policy_{Type}, Policy, SubPolicy_1, SubPolicy_2, \dots .\}$$

where $Policy_{Type} \in \{Crypto, Access, Authentication\}$ and each $SubPolicy$ is a defined property for main policy (*e.g.* key size for encryption policy, defined roles for role-based access policy or authenticator models for two-factor authentication policies).

7.4.2. Revocation Engine

This engine receives the revocation requests from the administrator or resource owner and processes them based on defined policies of each cloud node. Each revocation request is defined as $\{U_x, \{S_1, \dots, S_r\}, h(T_r)\}$ where U_x is the target user to be revoked, $\{S_1, \dots, S_r\} \subset \{S_1, S_2, \dots, S_N\}$ are the target cloud nodes that need to be protected from the revoked user and T_r is the timestamp of received request.

Processing access requests of users is done by *Access Engine.* Whenever a user revocation request is received by revocation engine, a Boolean property "Temporary Suspension" is activated for the user. By this activation, all of access requests from U_x (whether is related to revoked nodes or not) are transferred to revocation engine to be processed. In fact, revocation engine evaluates access requests from the suspended user until the process of revocation and policy update is completed. Furthermore, a revocation flag is generated from each associated node based on node time stamp. Algorithm 7.1 shows the process of user revocation in details.

Algorithm 7.1. User Revocation Process

Input: $\{U_x, \{S_1, \dots, S_r\}, h(T_r)\}$: User Revocation Request.
Output: U_x is revoked and $Update_{Policies}\{S_1, S_2 \dots, S_r\}$.

1. **Revocation Flag of U_x**

   $\forall\ S_n \in \{S_1, S_2, \dots, S_r\}: RF(S_n) = \left\{U_x, RFS_{S_n}, h(T_r), h(T_r')\right\}$

   A revocation Flag in generated for all of the associated nodes from revocation request where the default status value of the flag is "Under Process".

2. **Temporary Suspension of U_x**

   $TS(U_x) = true;$

   $\forall\ AccessRequest(U_x, S_n):$

   $Send(AccessEngine, RevocationEngine)$

   The user is temporary suspended until the process user revocation and policy update is finished. Before that, all of the access requests from U_x is process in revocation engine instead of access engine.

3. **Policy Application**

   $\forall\ S_n \in \{S_1, S_2, \dots, S_r\}: Policy_{Request}(S_n, PolicyEngine)$

   All of the associated policies from related cloud nodes are requested from Policy Engine.

4. **Policy Classification**

   $\forall\ PS_m \in S_n: Classify\ (PS_m, Policy_{Type})$

   $Switch\ \left(PS_m(Policy_{Type})\right)$

   $Case\ Crypto: Send(PS_m, CheckPoint);$

   $Case\ Access: Send(PS_m, PolicyEngine);$

   $Case\ Authentication: Send(PS_m, AcceesEngine);$

All of the policies are classified by policy type for policy update process. If there is an encryption policy, *Check Point* receives it for applying to cloud nodes. For Access and Authentication policies, *Policy Engine* and *Access Engine* are responsible respectively for policy update policies.

5. **Cryptography Policy Update in Check Point**

   Encryption policies are evaluated in this stage for each cloud node related to the revoked user. These policies are classified to re-encryption, key management and key-generation policies that are processed to be applied in cloud nodes. After this step, the confirmation is sent to cloud nodes, policies are updated in policy engine and the revocation flag status is updated.

6. **Access Policy Update in Policy Engine**

   All policies related to role-based, reputation-based or temporary-based access controls are updated in policy engine. If the user is revoked from the whole system, it affects the whole policy but if the user is revoked from several nodes, the policy engine updates access sub-policies of each cloud node, sends the confirmation to the associated cloud nodes and updated the revocation flag status.

7. **Authentication Policy Update in Access Engine**

   Authentication policies are mostly related to additional identity details such as second or one-time passwords for specific cloud node. The duty of access engine is to check authentication policies and to eliminate these additional password generation processes for associated cloud nodes. After this update, the confirmation is sent to cloud nodes, authentication policies and revocation flag status is updated in policy engine and access engine respectively.

8. **Revocation Flag Update**

   $\forall\, RF(S_n) \in RF\colon if\ \big((h(T_r) = h(T_r')) \wedge (RFS_{S_n} = true)\big)$

   $then\ Success\big(RF(S_n)\big)$

   If the timestamp is still valid and the status of revocation flag is true then the process of revocation is done successfully and all of the associated policies are updated.

9. **Elimination of Temporary Suspension**

   $\forall U_x \in \{U_1, U_2, \dots, U_X\}\colon if\ \big((RF(S_1) = Success) \wedge (RF(S_2) = Success) \wedge \dots \wedge (RF(S_n) = Success)\big)$

   $then\ TS(U_x) = false;$

   The temporary suspension of revoked user is eliminated if all of the revocation flags of all associated cloud nodes are successfully done. After the elimination of TS, all access requests are transferred to access engine again.

7.4.3. Access Engine

All of the normal access requests are sent to access engine for processing and mapping the requests to the associated cloud nodes according to the defined polices. In the proposed model, the process of revocation management is separated from access control. In fact, the access engine can use every developed access management models (*e.g.* role-based, attribute-based, *etc.*) without any limitation. This helps to decrease the processing power for access requests during revocation management.

Whenever the *Temporary Suspension* flag is activated for a user due to the revocation request, all requests from that user are sent to revocation engine. By this transfer, the access management scheduler is not affected for other users by processing suspended requests and checking timestamps and other access flags. A stand-alone access management scheduler in revocation engine processes suspended requests and checks extra properties according to status of TS flag.

7.4.4. Check Point

The main duty of *Check Point* is to process encryption, re-encryption and key-management policies. As described in section 7.2, one of the most challenging issues in encryption-based revocation models is the potential overhead and performance impact. In the proposed model, all security policies in affected cloud nodes are classified to three parts: Encryption, Authentication and Access Management.

This classification helps to decrease the overhead in *Check Point*. In fact, all key management (*e.g.* symmetric or asymmetric) and re-encryption policies (*e.g.* by request, by revocation or periodical) are processed in *Check Point* based on the revocation flag status and revocation confirmations are sent to cloud nodes.

7.5. Evaluation

In order to incarnate the superiorities of this schema in cloud-based environments, we give a security and performance analysis of the proposed model in this section. Accordingly, several scenarios are described as follows:

7.5.1. Performance Analysis

In the first discussion round the performance and scalability of the proposed user revocation model are evaluated. Hence, the number of revocation requests, number of associated cloud nodes with revocation requests and the number of both users and associated nodes are increased. Figure 7.2 shows the performance analysis results of policy-based revocation model in comparison with ABS-based and ABGS-based revocation models.

The number of revocation requests is increased in figure 7.2A and each request is associated with one node. In the first few requests the processing time for ABS and ABGS models are less than policy-based. However, the performance of proposed model is better after the first 100 requests. This difference is even more considerable after 1000 requests and proves the better performance of policy-based revocation in large number of requests.

Figure 7.2B shows the results of increasing the total number of associated nodes in revocation requests. Same as previous experiment, ABS and ABGS models have better results in less that 50 nodes but the total processing time in policy-based model is considerable less than other models by increasing the number of associated nodes.

Finally, in the third experiment the total processing time is evaluated in both increase of revoked users and associated nodes. As expected, the processing time in policy-based revocation model is significantly less than other models after few requests and it proves the efficiency and scalability of policy-based user revocation.

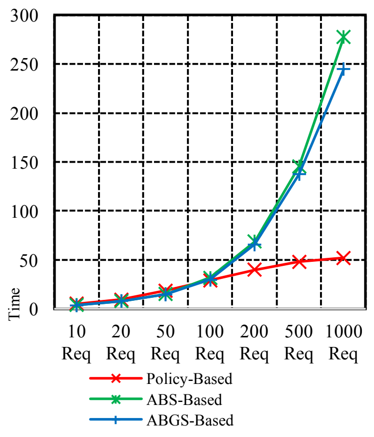


Fig 7.2A Increase The Number of Revoked Users

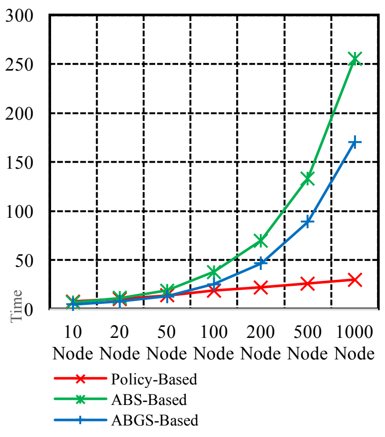


Fig 7.2B Increase The Number of Associated Nodes

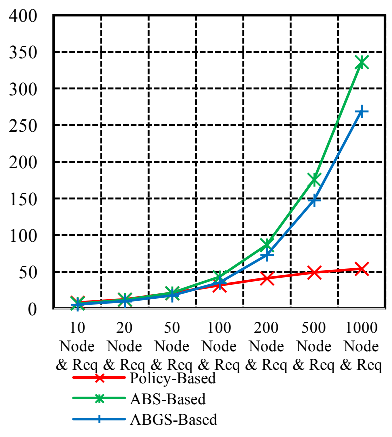


Fig 7.2C Increase Both Users and Associated Nodes

Fig. 7.2. Performance and Scalability Analysis of Policy-Based Revocation Model by Processing Time

7.5.2. Security Analysis

The security of our policy-based revocation model is granted in three main parts: users, associated nodes and managing policies. In fact, after submission a user revocation request, the proposed model should guarantee the privacy of associated nodes and updates and applies the defined policies for them. Table 7.2 shows the security analysis of proposed model according to defined objectives and possible treats. The analysis shows the suggested user revocation model can resist against possible treats by various methods and solutions during access request, policy mapping, policy application and policy update.

Table 7.2. Security Analysis of Proposed Model

Security Concern	Classification	Solution Method
Access Requests after Revocation	Access	Using **Temporary Suspension Flag** to transfer access requests of affected users from *Access Engine* to *Revocation Engine*.
Requests on the Run-Time	Access	Using **Time-Stamps** in both sides for checking the access requests.
Security of Time-Stamps	Access	Using **Hash-Function** in both Time Stamps.
Security of Associated Nodes	Policy	Calling **Policies** for each associated node of a user revocation request from *Policy Engine.*
Security and Reliability of Policies	Policy	**Classification of Policies** to three main parts based on Priority and Type by Revocation Engine.

Integrity of All Defined Policies	Application	Using **Revocation Flag** to check all policies are applied or not.
Encryption and Re-Encryption	Application	Using *Check Point* to schedule re-encryption methods based on defined policies and to check whether all encryption policies are applied or not.
Reliable Policy Update	Application	Checking *Policy Engine* **Database** to ensure are necessary policies are updated according to the user revocation request.
Security of Nodes	Application	Using **Revocation Flag** to check all policies are applied or not. Also, sending **Revocation Confirmation** Flag to associated cloud nodes after successful Policy Application.

7.5.3. Competitive Analysis

Table 7.3 shows the competitive analysis of the suggested schema according to the objectives of a policy-based user revocation model in cloud-based environments. According to the results this policy-based schema is able to provide a multi-level user revocation algorithm for mapping associated nodes in requests to defined security policies of each node. Same as ABS-based models, the proposed model uses encryption and also re-encryption but in separate stand-alone component.

Moreover, access and revocation managements are separated in policy-based model and access requests from normal users are also separated according to **Temporary Suspension** flag. In addition, associated policies are classified in the model according the type and priority to ensure each policy is applied efficiently. Finally, User Revocation and Revocation Confirmation flags are used to ensure the reliability and success of user revocation process for each associated node.

Table 7.3. Competitive Analysis of Proposed Model

Objectives	ABS	E-ABS	AGBS	Panda	Proposed Model
Policy-Based	No	No	No	No	Yes
Encryption & Re-Encryption	Yes	Yes	Yes	Yes	Yes
Separate Access Management	No	No	No	Yes	Yes
Group Access Management	No	No	Yes	No	Yes
Temporary Suspension	No	No	No	No	Yes
Revocation Flag	No	Yes	No	Yes	Yes
Policy Classification	No	No	No	No	Yes
3 Level Revocation	No	No	No	No	Yes
Cloud-Based	Yes	Yes	Yes	Yes	Yes

7.6. Conclusion

According to the importance of privacy in cloud-based environments and due to the lack of efficient user revocation process in clouds, a policy-based user revocation model was presented in this chapter to ensure the security of associated cloud nodes after a user is revoked from a part of whole system.

To decrease the processing power and provide an efficient user revocation model, each revocation request was processed according to the defined security policies. In fact, security and privacy policies (*i.e.* encryption, signature, access control and authentication) for each associated cloud node specifies whether the additional manipulations and processes are needed or not.

Accordingly, four main components are defined to define and manage security policies, to separate access and revocation management processes, and to apply encryption and re-encryption policies after user revocation. This model was evaluated with performance, security and competitive analysis. According to the results, the reliability and efficiency of the suggested schema was assured for managing user revocation requests in cloud-based environments. Also, the analysis showed the suggested user revocation model can resist against possible treats by various methods and solutions during access request, policy mapping, policy application and policy update.

Chapter 8

8. Conclusion and Future Works

8.1. Introduction

To enhance the quality on managing security policies in cloud-based environments and to provide efficient, secure and reliable matching between security requirements of customers and capabilities of service providers, a multi-layered policy engine was introduced in this thesis. A well-established policy framework was defined to generate security policies which are compliant to requirements and capabilities. Moreover, a federated policy management schema is introduced based on the policy definition framework and multi-level policy application to create and manage virtual clusters with identical or common security levels. The model was evaluated according to performance and security parameters and proved that this multi-layered policy engine enhances the reliability and efficiency of managing security polices in cloud computing environments during policy definition, policy application and policy mapping procedures. In this chapter, an overall discussion is presented regarding to the policy managing in clouds and the proposed model.

8.2. Overall Discussion

In this thesis an efficient and reliable policy generation and management schema was presented for Multi Security Level Cloud Computing. Accordingly, a structural multi-level ontology was introduced in the third chapter to define and manage security levels according to the capabilities of service provider, constraints and requirements of cloud customers.

CSON and MLO were the proposed ontologies based on multi-level and multi-layered classification. These ontologies provide a special cloud-based security structure to establish standard and dedicated security levels in a cloud computing environment through syntactic and semantic analysis of the security requirements and capabilities. The structure and foundation of policies use inherited super classes to map requests and offers, as well as to establish appropriate communications between the Policy Engine, the Policy Database and the SLA Engine. Furthermore, RAE as a ring analysis component justifies the security levels and evaluates the process of mapping syntactically and semantically. SLC as the output of this model defines standard or dedicated security levels for various cloud customers.

In the forth chapter a policy application and mapping schema was proposed based semantic policy clustering to classify nodes with same or common security policies in an aggregate virtualized cluster for federating defined policies according to their characteristics. By this federation, the processing time for each policy mapping was reduced due to the elimination of gratuitous and avoidable matching jobs.

We defined a well-established policy framework to define security policies which are compliant to requirements and capabilities. Moreover, a federated policy management schema was introduced based on the policy definition framework and policy clustering to create and manage virtual clusters with identical or common security levels.

The next three chapters presented different policy-based scenarios based on authentication, data protection (*i.e.* re-encryption) and user revocation to examine policy-engine in various use cases.

In the fifth chapter, a policy-based user authentication model was presented to provide a reliable identity management mechanism for establishing multiple access policies in different virtualized nodes and mapping access requests to defined policies regarding to capabilities of cloud servers and requirements of resources. Accordingly, a structural policy language and policy engine were introduced for policy generation, application and management. Moreover, a policy match framework was described for managing identities and mapping access requests to virtualized resources based on defined policies.

The proposed model has been able to provide a multi-level authentication process by policy definition in different cloud servers and dual authenticator in both match gate and cloud server to ensure about the security and efficiency of this process. Furthermore, the proposed model was significantly scalable in comparison with most of authentication models to have a better performance in face with large number of access requests and defined policies. Finally, this schema uses semantic analysis of access requests with an efficient authentication task management and token-based identity management to ensure about the security and reliability of this challenging issue in cloud computing environments.

A policy-based re-encryption model was presented in the sixth chapter to define, establish and manage re-encryption policies in cloud computing environments. To achieve this purpose, a multi-level ontology was defined based on WS-Policy. This ontology helps re-encryption components to provide reliable re-encryption policy definitions, generation and management in clouds.

The results of comprehensive performance and security evaluation of proposed model shows our proposed re-encryption schema increases the reliability of re-encryption processes in cloud storages considerably and provides an efficient policy management for re-encryption tasks.

In the seventh chapter a policy-based user revocation schema was presented to decrease the processing power and provide an efficient user revocation model by processing each revocation request according to the defined security policies. In fact, security and privacy policies (i.e. encryption, signature, access control and authentication) for each associated cloud node specifies whether the additional manipulations and processes are needed or not.

The presented model used four main components to define and manage security policies, to separate access and revocation management processes, and to apply encryption and re-encryption policies after user revocation. This model was evaluated with performance, security and competitive analysis. According to the results, the reliability and efficiency of the suggested schema was assured for managing user revocation requests in cloud-based environments.

8.3. Constraints

The results of the proposed policy management schema proved that this model is scalable enough to use in real data centers for defining, applying and managing security mechanisms in cloud-based environments based on the capabilities of service provider and requirements of cloud customers. However, the most challenging issue is to enhance the process of mapping requirements to capabilities according to the possible conflicts (i.e. Ring Analysis Engine in Chapter 3). According to the results, 78% of the requirements were matched to the most suitable security algorithms offered in CSRT. The minimum value of the Perfect match is related to the access management protocol due to the variety of algorithms and the concepts of reputation and role-based access control models. To enhance the process of matching, an optimized algorithm is needed specially in access management protocol for efficient elimination of conflicted algorithms in CSDS super class. We expect the rate of perfect matches is increased by applying a semi-supervised algorithm in Ring Analysis Engine.

8.4.Future Perspective

To enhance the process of managing policy and security levels, two main paths were specified as the future works:

- Optimization of Dedicated Security Levels: In the first path, an optimized algorithm needs to be developed to enhance the process of mapping requirements and capabilities for non-professional customers to avoid conflicts between security algorithms during the SLC generation. In fact, the process of resolving conflicts by RAE should be optimized to increase the rate of perfect matches between requirements and capabilities.

- Nomination Process of Virtual Clusters: In the second path, each virtual cluster with common security policies introduce the most appropriate node to work as the nominated note, manage access requests, and approve the identity of the requester. In fact, all nodes in the associated cluster approve the access request without re-checking by each node or the policy engine, if and only if the nominated node approved the request before.

Chapter 9

9. Bibliography

[1] S. Azodolmolky, P. Wieder, and R. Yahyapour, "Cloud Computing Networking: Challenges and Opportunities for Innovations," *IEEE Commun. Mag.*, vol. 51, no. 7, pp. 54–62, 2013.

[2] F. Shahzad, "State-of-the-art Survey on Cloud Computing Security Challenges, Approaches and Solutions," *Procedia Comput. Sci.*, vol. 37, pp. 357–362, 2014.

[3] M. D. Assunção, R. N. Calheiros, S. Bianchi, M. A. S. Netto, and R. Buyya, "Big Data Computing and Clouds: Trends and Future Directions," *J. Parallel Distrib. Comput.*, vol. 79–80, pp. 3–15, May 2015.

[4] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A View of Cloud Computing," *Commun. ACM Mag.*, vol. 53, no. 4, pp. 50–58, Apr. 2010.

[5] M. A. Khan, "A Survey of Security Issues for Cloud Computing," *J. Netw. Comput. Appl.*, vol. 71, pp. 11–29, Aug. 2016.

[6] M. Metheny, "Security and Privacy in Public Cloud Computing," in *Federal Cloud Computing*, Elsevier, 2017, pp. 79–115.

[7] F. Fatemi Moghaddam, M. Ahmadi, S. Sarvari, M. Eslami, and A. Golkar, "Cloud

Computing Challenges and Opportunities: A Survey," in *1st International Conference on Telematics and Future Generation Networks (TAFGEN),* 2015, pp. 34–38.

[8] S. A. Hussain, M. Fatima, A. Saeed, I. Raza, and R. K. Shahzad, "Multilevel Classification of Security Concerns in Cloud Computing," *Appl. Comput. Informatics*, vol. 13, no. 1, pp. 57–65, Jan. 2017.

[9] D. Zissis and D. Lekkas, "Addressing Cloud Computing Security Issues," *Futur. Gener. Comput. Syst.*, vol. 28, no. 3, pp. 583–592, 2012.

[10] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility," *Futur. Gener. Comput. Syst.*, vol. 25, no. 6, pp. 599–616, 2009.

[11] F. Fatemi Moghaddam, M. Baradaran Rohani, M. Ahmadi, T. Khodadadi, and K. Madadipouya, "Cloud Computing : Vision, Architecture and Characteristics," in *IEEE 6th Control and System Graduate Research Colloquium (ICSGRC)*, 2015, pp. 1–6.

[12] C. A. B. de Carvalho, R. M. de C. Andrade, M. F. de Castro, E. F. Coutinho, and N. Agoulmine, "State of the Art and Challenges of Security SLA for Cloud Computing," *Comput. Electr. Eng.*, vol. 59, pp. 141–152, 2017.

[13] L. Coppolino, S. D'Antonio, G. Mazzeo, and L. Romano, "Cloud security: Emerging Threats and Current Solutions," *Comput. Electr. Eng.*, vol. 59, pp. 126–140, Apr. 2017.

[14] T. Phan, J. Han, J.-G. Schneider, T. Ebringer, and T. Rogers, "A Survey of Policy-Based Management Approaches for Service Oriented Systems," in *19th Australian Conference on Software Engineering (aswec 2008)*, 2008, pp. 392–401.

[15] S. A. de Chaves, C. B. Westphall, and F. R. Lamin, "SLA Perspective in Security Management for Cloud Computing," in *2010 Sixth International Conference on Networking and Services*, 2010, pp. 212–217.

[16] U. Khalid, A. Ghafoor, M. Irum, and M. A. Shibli, "Cloud Based Secure and Privacy Enhanced Authentication &amp; Authorization Protocol," *Procedia Comput. Sci.*, vol. 22, pp. 680–688, 2013.

[17] S. Lakshminarayanan, "Interoperable Security Standards for Web Services," *IT Prof.*, vol. 12, no. 5, pp. 42–47, Sep. 2010.

[18] M. Tao, K. Ota, and M. Dong, "Ontology-Based Data Semantic Management and Application in IoT and Cloud-Enabled Smart Homes," *Futur. Gener. Comput. Syst.*, Nov. 2016.

[19] H. B. Rahmouni, K. Munir, M. C. Mont, and T. Solomonides, "Semantic Generation of Clouds Privacy Policies," Springer, Cham, 2015, pp. 15–30.

[20] C. Choi, J. Choi, and P. Kim, "Ontology-Based Access Control Model for Security Policy Reasoning in Cloud Computing," *J. Supercomput.*, vol. 67, no. 3, pp. 711–722, Mar. 2014.

[21] "OWL - Semantic Web Standards." [Online]. Available: https://www.w3.org/OWL/. [Accessed: 16-Jan-2017].

[22] R. García-Castro and A. Gómez-Pérez, "Interoperability Results for Semantic Web Technologies using OWL as the Interchange Language," *Web Semant. Sci. Serv. Agents World Wide Web*, vol. 8, no. 4, pp. 278–291, Nov. 2010.

[23] A. Khurat, B. Suntisrivaraporn, and D. Gollmann, "Privacy Policies Verification in Composite Services Using OWL," *Comput. Secur.*, Mar. 2017.

[24] I.-C. Hsu, Y. K. Tzeng, and D.-C. Huang, "OWL-L: An OWL-Based Language for Web Resources Links," *Comput. Stand. Interfaces*, vol. 31, no. 4, pp. 846–855, Jun. 2009.

[25] B. Amel and M. Ramedane, "From OWL-S to Timed Automata Network: Operational Semantic," *Procedia Comput. Sci.*, vol. 83, pp. 409–416, 2016.

[26] Y. Zou, T. Finin, and H. Chen, "F-OWL: An Inference Engine for Semantic Web," Springer, Berlin, Heidelberg, 2004, pp. 238–248.

[27] A. Uszok, J. M. Bradshaw, M. Johnson, R. Jeffers, A. Tate, J. Dalton, and S. Aitken, "KAoS Policy Management for Semantic Web Services," *IEEE Intell. Syst.*, vol. 19, no. 4, pp. 32–41, Jul. 2004.

[28] A. Uszok, J. M. Bradshaw, J. Lott, M. Breedy, L. Bunch, P. Feltovich, M. Johnson, and H. Jung, "New Developments in Ontology-Based Policy Management: Increasing the Practicality and Comprehensiveness of KAoS," in *2008 IEEE Workshop on Policies for Distributed Systems and Networks*, 2008, pp. 145–152.

[29] N. Damianou, N. Dulay, E. Lupu, and M. Sloman, "The Ponder Policy Specification Language," Springer, Berlin, Heidelberg, 2001, pp. 18–38.

[30] N. Dulay, E. Lupu, M. Sloman, and N. Damianou, "A Policy Deployment Model for the Ponder Language," in *2001 IEEE/IFIP International Symposium on Integrated Network Management Proceedings. Integrated Network Management VII. Integrated Management Strategies for the New Millennium (Cat. No.01EX470)*, pp. 529–543.

[31] N. Damianou, N. Dulay, E. Lupu, and M. Sloman, "Managing Security in Object-Based Distributed Systems Using Ponder."

[32] S. Bajaj, D. Box, D. Chappell, F. Curbera, G. Daniels, P. Hallam-Baker, M. Hondo, C. Kaler, D. Langworthy, and A. Malhotra, "Web Services Policy Framework (WS-Policy)," *Specif. IBM, BEA, Microsoft, SAP AG, Sonic Software, VeriSign*, 2004.

[33] S. Anderson, J. Bohren, T. Boubez, M. Chanliau, G. Della-Libera, B. Dixon, P. Garg, M. Gudgin, P. Hallam-Baker, and M. Hondo, "Web Services Trust Language (WS-Trust)." May, 2004.

[34] G. Della-Libera, M. Gudgin, P. Hallam-Baker, M. Hondo, H. Granqvist, C. Kaler, H. Maruyama, M. McIntosh, A. Nadalin, and N. Nagaratnam, "Web Services Security Policy Language (WS-SecurityPolicy)," *Public Draft Specif. (Juli 2005)*, 2002.

[35] S. Anderson, J. Bohren, T. Boubez, M. Chanliau, G. Della-Libera, B. Dixon, P. Garg, E. Gravengaard, M. Gudgin, and S. Hada, "Web Services Secure Conversation Language (WS-SecureConversation)," *Actional Corp. Syst. Inc./Computer Assoc. Int. Inc./IBM Corp.*, vol. 7, p. 35, 2005.

[36] Bill Parducci and Hal Lockhart, “eXtensible Access Control Markup Language (XACML) Version 3.0,” 2013.

[37] M. B. Chhetri, B. Q. Vo, and R. Kowalczyk, “Policy-Based Management of QoS in Service Aggregations,” in *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, 2010, pp. 593–595.

[38] G. Di Modica and O. Tomarchio, “Matchmaking Semantic Security Policies in Heterogeneous Clouds,” *Futur. Gener. Comput. Syst.*, vol. 55, pp. 176–185, Mar. 2015.

[39] K. Hashmi, E. Najmi, Z. Malik, B. Medjahed, A. Alhosban, and A. Rezgui, “Automated Negotiation Using Semantic Rules,” in *2014 IEEE International Conference on Services Computing*, 2014, pp. 536–543.

[40] N. Sriharee, T. Senivongse, K. Verma, and A. Sheth, “On Using WS-Policy, Ontology, and Rule Reasoning to Discover Web Services,” Springer Berlin Heidelberg, 2004, pp. 246–255.

[41] G. Di Modica and O. Tomarchio, “Semantic Security Policy Matching in Service Oriented Architectures,” in *2011 IEEE World Congress on Services*, 2011, pp. 399–405.

[42] Y.-J. Hu, W.-N. Wu, and D.-R. Cheng, “Towards Law-Aware Semantic Cloud Policies with Exceptions for Data Integration and Protection,” in *Proceedings of the 2nd International Conference on Web Intelligence, Mining and Semantics - WIMS ’12*, 2012, p. 1.

[43] F. Fatemi Moghaddam, P. Wieder, and R. Yahyapour, “Policy Engine as a Service (PEaaS): An Approach to a Reliable Policy Management Framework in Cloud Computing Environments,” in *IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud)*, 2016, pp. 137–144.

[44] S. Veloudis and I. Paraskakis, “Ontological Templates for Modelling Security Policies in Cloud Environments,” in *Proceedings of the 20th Pan-Hellenic Conference on Informatics - PCI ’16*, 2016, pp. 1–6.

[45] Janet Daly, Marie-Claire Forgue, and Yasuyuki Hirakawa, “World Wide Web Consortium Issues RDF and OWL Recommendations,” 2004.

[46] F. Pereniguez, R. Marin-Lopez, G. Kambourakis, S. Gritzalis, and A. F. Gomez, “PrivaKERB: A User Privacy Framework for Kerberos,” *Comput. Secur.*, vol. 30, no. 6, pp. 446–463, 2011.

[47] A. Xiaoguang and L. Xiaofan, “Research on Detection Mechanism of Cloud Security Policy Collision,” in *2016 International Conference on Robots & Intelligent System (ICRIS)*, 2016, pp. 341–344.

[48] R. Moreno-Vozmediano, E. Huedo, I. M. Llorente, R. S. Montero, P. Massonet, M. Villari, G. Merlino, A. Celesti, A. Levin, L. Schour, C. Vázquez, J. Melis, S. Spahr, and D. Whigham, “BEACON: A Cloud Network Federation Framework,” Springer, Cham, 2016, pp. 325–337.

[49] L. Coppolino, S. D'Antonio, G. Mazzeo, and L. Romano, "Cloud Security: Emerging Threats and Current Solutions," *Comput. Electr. Eng.*, Mar. 2016.

[50] P. Massonet, S. Dupont, A. Michot, A. Levin, and M. Villari, "Enforcement of Global Security Policies in Federated Cloud Networks with Virtual Network Functions," in *2016 IEEE 15th International Symposium on Network Computing and Applications (NCA)*, 2016, pp. 81–84.

[51] J. Modic, R. Trapero, A. Taha, J. Luna, M. Stopar, and N. Suri, "Novel Efficient Techniques for Real-Time Cloud Security Assessment," *Comput. Secur.*, vol. 62, pp. 1–18, Sep. 2016.

[52] M. Ramachandran and V. Chang, "Towards Performance Evaluation of Cloud Service Providers for Cloud Data Security," *Int. J. Inf. Manage.*, vol. 36, no. 4, pp. 618–625, Aug. 2016.

[53] Y. A. Younis, K. Kifayat, and M. Merabti, "An Access Control Model for Cloud Computing," *J. Inf. Secur. Appl.*, vol. 19, no. 1, pp. 45–60, Feb. 2014.

[54] C. P. Cahill, A. Hal Lockhart, B. Systems Michael Beach, B. Rick Randall, H. Tim Alsop, C. Limited Nick Ragouzis, E. John Hughes, A. Origin Paul Madsen, E. Irving Reid, H.-P. Paula Austel, I. Maryann Hondo, I. Michael McIntosh, I. Tony Nadalin, I. Scott Cantor, R. Metz, N. Prateek Mishra, N. C. Peter Davis, N. Frederick Hirsch, N. John Kemp, N. Charles Knouse, O. Steve Anderson, O. John Linn, R. Security Rob Philpott, R. Security Jahan Moreh, and G. Whitehead, "Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0," 2010.

[55] D. Hardt, "The OAuth 2.0 Authorization Framework," 2012.

[56] D. Recordon and D. Reed, "OpenID 2.0: : A Platform for User-Centric Identity Management," in *Proceedings of the second ACM workshop on Digital identity management - DIM '06*, 2006, p. 11.

[57] Jen-Ho Yang, Ya-Fen Chang, and Chih-Cheng Huang, "A User Authentication Scheme on Multi-Server Environments for Cloud Computing," in *2013 9th International Conference on Information, Communications & Signal Processing*, 2013, pp. 1–4.

[58] G. Dólera Tormo, F. Gómez Mármol, and G. Martínez Pérez, "Towards the Integration of Reputation Management in OpenID," *Comput. Stand. Interfaces*, vol. 36, no. 3, pp. 438–453, 2014.

[59] R. L. |Cantor. S. S. W. K. Morgan, "Federated Security: The Shibboleth Approach.," *Educ. Q.*, vol. 27, no. 4, pp. 12–17, 2004.

[60] S. Suoranta, K. Manzoor, A. Tontti, J. Ruuskanen, and T. Aura, "Logout in Single Sign-On Systems: Problems and Solutions," *J. Inf. Secur. Appl.*, vol. 19, no. 1, pp. 61–77, 2014.

[61] C. Neuman, S. Hartman, T. Yu, and K. Raeburn, "The Kerberos Network Authentication Service (V5)," 2005.

[62] F. Pereñíguez-García, R. Marín-López, G. Kambourakis, A. Ruiz-Martínez, S.

Gritzalis, and A. F. Skarmeta-Gómez, “KAMU: Providing Advanced User Privacy in Kerberos Multi-Domain Scenarios,” *Int. J. Inf. Secur.*, vol. 12, no. 6, pp. 505–525, Nov. 2013.

[63] R. Marín-López, F. Pereñíguez, G. López, and A. Pérez-Méndez, “Providing EAP-based Kerberos Pre-Authentication and Advanced Authorization for Network Federations,” *Comput. Stand. Interfaces*, vol. 33, no. 5, pp. 494–504, 2011.

[64] A. Moralis, V. Pouli, S. Papavassiliou, and V. Maglaris, “A Kerberos Security Architecture for Web Services based Instrumentation Grids,” *Futur. Gener. Comput. Syst.*, vol. 25, no. 7, pp. 804–818, 2009.

[65] J. R. Vollbrecht, B. Aboba, L. J. Blunk, H. Levkowetz, and J. Carlson, “Extensible Authentication Protocol (EAP),” 2004.

[66] Y. Ohba, B. Patil, D. Forsberg, H. Tschofenig, and A. E. Yegin, “Protocol for Carrying Authentication for Network Access (PANA),” 2008.

[67] A. Pérez-Méndez, F. Pereñíguez-García, R. Marín-López, and G. López-Millán, “Out-of-Band Federated Authentication for Kerberos based on PANA,” *Comput. Commun.*, vol. 36, no. 14, pp. 1527–1538, 2013.

[68] M. A. P. Leandro, T. J. Nascimento, D. R. Dos Santos, C. M. Westphall, and C. B. Westphall, “Multi-Tenancy Authorization System with Federated Identity for Cloud-Based Environments Using Shibboleth,” in *The Eleventh International Conference on Networks (ICN)*, 2012, pp. 88–93.

[69] E. Ghazizadeh, M. Zamani, J. Ab Manan, and A. Pashang, “A Survey on Security Issues of Federated Identity in the Cloud Computing,” in *4th IEEE International Conference on Cloud Computing Technology and Science Proceedings*, 2012, pp. 532–565.

[70] P. Madsen, Y. Koga, and K. Takahashi, “Federated Identity Management for Protecting Users from ID Theft,” in *Proceedings of the 2005 workshop on Digital identity management - DIM ’05*, 2005, p. 77.

[71] Z. A. Khattak, S. Sulaiman, and J.-L. A. Manan, “A Study on Threat Model for Federated Identities in Federated Identity Management System,” in *2010 International Symposium on Information Technology*, 2010, pp. 618–623.

[72] A. Bhargav-Spantzel, A. C. Squicciarini, and E. Bertino, “Establishing and Protecting Digital Identity in Federation Systems,” *J. Comput. Secur.*, vol. 14, no. 3, pp. 269–300, 2006.

[73] Zubair Ahmad, J.-L. Ab Manan, and S. Sulaiman, “User Requirement Model for Federated Identities Threats,” in *International Conference on Advanced Computer Theory and Engineering(ICACTE)*, 2010, pp. V6-317-V6-321.

[74] S. Kalra and S. K. Sood, “Secure Authentication Scheme for IoT and Cloud Servers,” *Pervasive Mob. Comput.*, vol. 24, pp. 210–223, 2015.

[75] S. Ma, “Identity-Based Encryption with Outsourced Equality Test in Cloud Computing,” *Inf. Sci. (Ny).*, vol. 328, pp. 389–402, 2016.

[76] S. Kumari, X. Li, F. Wu, A. K. Das, K.-K. R. Choo, and J. Shen, "Design of a Provably Secure Biometrics-based Multi-Cloud-Server Authentication Scheme," *Futur. Gener. Comput. Syst.*, vol. 68, pp. 320–330, 2017.

[77] X. Lu and Y. Xu, "An User Behavior Credibility Authentication Model in Cloud Computing Environment," in *Proceedings of 2nd International Conference on Information Technology and Electronic Commerce*, 2014, pp. 271–275.

[78] J. H. Yang and P. Y. Lin, "An ID-Based User Authentication Scheme for Cloud Computing," in *2014 Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, 2014, pp. 98–101.

[79] S. Speiser, "Semantic Annotations for WS-Policy," in *2010 IEEE International Conference on Web Services*, 2010, pp. 449–456.

[80] M. Darwish, A. Ouda, and L. F. Capretz, "A Cloud-based Secure Authentication (CSA) Protocol Suite for Defense Against Denial of Service (DoS) Attacks," *J. Inf. Secur. Appl.*, vol. 20, pp. 90–98, 2015.

[81] M. Ahmadi, F. Fatemi Moghaddam, A. J. Jam, S. Gholizadeh, and M. Eslami, "A 3-Level Re-Encryption Model to Ensure Data Protection in Cloud Computing Environments," in *IEEE Conference on Systems, Process and Control (ICSPC)*, 2014, pp. 36–40.

[82] G. Wang, Q. Liu, and J. Wu, "Hierarchical Attribute-based Encryption for Fine-Grained Access Control in Cloud Storage Services," in *Proceedings of the 17th ACM conference on Computer and communications security - CCS '10*, 2010, p. 735.

[83] Qin Liu, C. C. Tan, Jie Wu, and Guojun Wang, "Reliable Re-Encryption in Unreliable Clouds," in *2011 IEEE Global Telecommunications Conference - GLOBECOM 2011*, 2011, pp. 1–5.

[84] N. Park, J. Kwak, S. Kim, D. Won, and H. Kim, "WIPI Mobile Platform with Secure Service for Mobile RFID Network Environment," Springer, Berlin, Heidelberg, 2006, pp. 741–748.

[85] J.-M. Do, Y.-J. Song, and N. Park, "Attribute Based Proxy Re-Encryption for Data Confidentiality in Cloud Computing Environments," in *2011 First ACIS/JNU International Conference on Computers, Networks, Systems and Industrial Engineering*, 2011, pp. 248–251.

[86] M. Blaze, G. Bleumer, and M. Strauss, "Divertible Protocols and Atomic Proxy Cryptography," Springer, Berlin, Heidelberg, 1998, pp. 127–144.

[87] N. Balani and S. Ruj, "Temporal Access Control with User Revocation for Cloud Data," in *2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications*, 2014, pp. 336–343.

[88] A. Escala, J. Herranz, and P. Morillo, "Revocable Attribute-Based Signatures with Adaptive Security in the Standard Model," Springer Berlin Heidelberg, 2011, pp. 224–241.

[89] Dan Cao, Xiaofeng Wang, Baokang Zhao, Jinshu Su, and Qiaolin Hu, "Mediated

Attribute-Based Signature Scheme Supporting Key Revocation," 2012.

[90] Y. Lian, L. Xu, and X. Huang, "Attribute-Based Signatures with Efficient Revocation," in *2013 5th International Conference on Intelligent Networking and Collaborative Systems*, 2013, pp. 573–577.

[91] J. Camenisch, "Efficient and Generalized Group Signatures," Springer, Berlin, Heidelberg, 1997, pp. 465–479.

[92] B. Wang, B. Li, and H. Li, "Panda: Public Auditing for Shared Data with Efficient User Revocation in the Cloud," *IEEE Trans. Serv. Comput.*, vol. 8, no. 1, pp. 92–106, Jan. 2015.

[93] F. Fatemi Moghaddam, P. Wieder, and R. Yahyapour, "Federated Policy Management Engine for Reliable Cloud Computing," in *IEEE International Conference on Ubiquitous and Future Networks (ICUFN 2017)*, 2017.

[94] L. Kagal, "Rei: A Policy Language for the Me-Centric Project," 2002.

[95] F. Fatemi Moghaddam, P. Wieder, and R. Yahyapour, "Federated Policy Management Engine for Reliable Cloud Computing," in *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*, 2017, pp. 910–915.

[96] Z. Wu, "Multi-Cloud Policy Enforcement Through Semantic Modeling and Mapping," in *Proceedings of the 2015 IEEE 9th International Conference on Semantic Computing (IEEE ICSC 2015)*, 2015, pp. 448–451.

[97] A. Pérez Méndez, R. Marín López, and G. López Millán, "Providing Efficient SSO to Cloud Service Access in AAA-Based Identity Federations," *Futur. Gener. Comput. Syst.*, vol. 58, pp. 13–28, 2016.

[98] Z. Liu, H. Yan, and Z. Li, "Server-Aided Anonymous Attribute-Based Authentication in Cloud Computing," *Futur. Gener. Comput. Syst.*, vol. 52, pp. 61–66, 2015.

[99] Q. Liu, G. Wang, X. Liu, T. Peng, and J. Wu, "Achieving Reliable and Secure Services in Cloud Computing Environments," *Comput. Electr. Eng.*, vol. 59, pp. 153–164, Apr. 2017.

[100] F. Fatemi Moghaddam, P. Wieder, and R. Yahyapour, "Policy Management Engine (PME) - A Policy-Based Schema to Classify and Manage Sensitive Data in Cloud Storages," *J. Inf. Secur. Appl.*, vol. 36, pp. 11–19, 2017.

[101] M. Hajivali, F. Fatemi Moghaddam, M. T. Alrashdan, and A. Z. M. Alothmani, "Applying an Agent-Based User Authentication and Access Control Model for Cloud Servers," in *International Conference on ICT Convergence (ICTC)*, 2013, pp. 807–812.

[102] K. M. Sim, "Agent-Based Cloud Computing," *IEEE Trans. Serv. Comput.*, vol. 5, no. 4, pp. 564–577, 2012.

[103] H. Kandil and A. Atwan, "Mobile Agents' Authentication Using a Proposed Light Kerberos System," in *2014 9th International Conference on Informatics and Systems*, 2014, p. CNs-39-CNs-45.

[104] D. S. Kasunde and A. A. Manjrekar, “Verification of Multi-Owner Shared Data with Collusion Resistant User Revocation in Cloud,” in *2016 International Conference on Computational Techniques in Information and Communication Technologies (ICCTICT)*, 2016, pp. 182–185.

[105] X. Wu, M. Deng, R. Zhang, B. Zeng, and S. Zhou, “A Task Scheduling Algorithm based on QoS-Driven in Cloud Computing,” *Procedia Comput. Sci.*, vol. 17, pp. 1162–1169, 2013.

10. APPENDICES

10.1. List of Associated Publications

#	Name	Publication Name	Status
1	**Faraz Fatemi Moghaddam**, Ramin Yahyapour, Philipp Wieder: Policy Engine as a Service (PEaaS): An Approach to a Reliable Policy Management Framework in Cloud Computing Environments	IEEE FiCloud’16, Vienna, Austria	Published
2	**Faraz Fatemi Moghaddam,** Philipp Wieder, and Ramin Yahyapour: POBRES: Policy-Based Re-Encryption Schema for Secure Resource Managment in Clouds	17th IEEE EuroCon. Ohrid, Macedonia	Published
3	**Faraz Fatemi Moghaddam**, Philipp Wieder, and Ramin Yahyapour: Federated Policy Management Engine for Reliable Cloud Computing	IEEE ICUFN’17, Milan, Italy	Published
4	**Faraz Fatemi Moghaddam**, Philipp Wieder, and Ramin Yahyapour: Token-Based Policy Management (TBPM): A Reliable Data Classification and Access Management Schema in Clouds	51st IEEE ICCST’17, Madrid, Spain	Accepted
5	**Faraz Fatemi Moghaddam**, Philipp Wieder, and Ramin Yahyapour: Policy Management Engine (PME): A Policy-Based Schema to Classify and Manage Sensitive Data in Cloud Storages	Elsevier Journal of Information Security and Applications	Published
6	**Faraz Fatemi Moghaddam**, Philipp Wieder, and Ramin Yahyapour: A Multi-Level Policy Engine to Manage Identities and Control Accesses in Cloud Computing Environment	Elsevier Future Generation Computer Systems	Submitted
7	**Faraz Fatemi Moghaddam**, Philipp Wieder, and Ramin Yahyapour: A Multi-Layered Policy Generation and Management Engine for Semantic Policy Mapping in Clouds	Elsevier Digital Communications and Network	Submitted
8	**Faraz Fatemi Moghaddam**, Philipp Wieder, and Ramin Yahyapour: A Structural Policy Management Engine based on Policy Mapping and Ring Analysis for Clouds	Elsevier Journal of Information Security and Applications	Submitted
9	**Faraz Fatemi Moghaddam**, Philipp Wieder, and Ramin Yahyapour: A Multi-Layered Access Policy Engine for Reliable Cloud Computing	IEEE NOF’17, London, GB	Accepted
10	**Faraz Fatemi Moghaddam**, Philipp Wieder, and Ramin Yahyapour: A Policy-Based Identity Management Schema for Managing Accesses in Clouds	IEEE NOF’17, London, GB	Accepted
11	**Faraz Fatemi Moghaddam**, Philipp Wieder, and Ramin Yahyapour: An Effective User Revocation for Policy-Based Access Control Schema in Clouds	IEEE CloudNet’17, Prague, Czech Republic	Accepted

10.2. Example of Generated Security Level Certificate (SLC)

```
<wsp:Policy>
…namespace definition…
   <wsp:SLC rdf:ID=”A543D3527B3”>

      <wsp:Header>
         <security:Scope>
            <security:hasOwner rdf:ownerID=”A3453”/>
            <security:hasTarget rdf:target=”cu-ns#owner|cu-ns3#regEmail#@gwdg.de”/>
         </security:Scope>
         <security:Objective>
            <security:haspurpose rdf:purpose=”confidentiality#integrity#authentication”/>
            <security:reqRequirement rdf:requestID=”T353S387”/>
         </security:Objective>
         <security:Essential>
            <security:hasValue rdf:SLAresourceID=”sla-ns#owner#slcID”/>
            <security:reqResource rdf:resource=”VC-ns#VM-ns#Pr-ns#DS-ns”/>
         </security:Essential>
      </wsp:Header>

      <wsp:Body>
         <security:AccessManagementProtocol rdf:ID=”AccessManagementRequirement”>
            <security:RoleMechanism rdf:ID=”RoleClassRequirement”>
               <security:PermanentAlgorithm rdf:resource="PermanentRoleClass”>
                  <rdf:HLSP_Role rdf:HLSP_Role_1=”Lecturer” rdf:HLSP_Role_2=”Student”
               rdf:HLSP_Role_3=”Professor” rdf:HLSP_Role_4=”Staff” rdf:HLSP_Role_5=”Admin”/>
               </security:PermanentAlgorithm>
            </security:RoleMechanism>
            <security:ReputationMechanism rdf:ID=ReputationClassRequirement”>
               <security:GeoAlgorithm rdf:resource="GeoClass”>
                  <rdf:HLSP_Geo rdf:HLSP_Geo_1=”Germany” rdf:HLSP_Geo_2=”World”/>
               </security:GeoAlgorithm>
            </security:ReoutationMechanism>
         </security:AccessManagementProtocol>

         <security:CryptographyProtocol rdf:ID=”CryptographyRequirement”>
            <security:SymmetricMechanism rdf:ID=”SymmetricRequirement”>
               <security:AESClass rdf:resource="AESClass”>
                  <rdf:HLSP_AES rdf:HLSP_AES_KeySize=”256” rdf:HLSP_AES_Mode="CBC”/>
               </security:AESClass>
            </security:SymmetricMechanism>
            <security:ReEncryptionMechanism rdf:ID=REEncryptionRequirement”>
               <security:ManualReEncryption rdf:resource="ManualReClass”>
                  <rdf:HLSP_MRE rdf:HLSP_MER_1=”Admin” rdf:HLSP_MER_2=”Owner”/>
               </security:ManualReEncryption>
            </security:ReEncryptionMechanism>
         </security:CryptographyProtocol>
         <security:SignatureProtocol rdf:ID=”SignatureRequirement”>
            …
         </security: SignatureProtocol>
            <security:HashMechanism rdf:ID=”HashRequirement”>
               <security:SHA3Class rdf:resource="SHA3Class”>
                  <rdf:HLSP_SHA3 rdf:HLSP_SHA3_Size=”384” rdf:HLSP_SHA3_BLK=”832”/>
               </security:SHA3Class>
            </security:HashMechanism>
        </security:SignatureProtocol>

         <security:KeyManagementProtocol rdf:ID=”KeyManagementRequirement”>
             <security:SymmetricMechanism rdf:ID=”SymmetricRequirement”>
               <security:KMCClass rdf:resource="KMCClass”>
                  <rdf:HLSP_KMC rdf:HLSP_KMC_Version=”137”/>
               </security:KMCClass>
            </security:SymmetricMechanism>
         </security:KeyManagementProtocol>

         <security:AuthenticationProtocol rdf:ID=”AuthenticationRequirement”>
             <security:DoubleMechanism rdf:ID=”DoubleAuthRequirement”>
               <security:AuthenticatorClass rdf:resource="AuthenticatorClass”>
```

```
                  <rdf:HLSP_Authenticator rdf:HLSP_Authenticator_Version="APP"/>
               </security:AuthenticatorClass>
            </security:DoubleMechanism>
         </security: AuthenticationProtocol>

         <security:TransportProtocol rdf:ID="TransportRequirement">
             <security:EncryptedMechanism rdf:ID="EncryptedRequirement">
               <security:TLSClass rdf:resource="TLSClass">
                  <rdf:HLSP_TLS rdf:HLSP_TLS_Version="1.2"/>
               </security:TLSClass>
            </security:EncryptedMechanism>
         </security: TransportProtocol>

      </wsp:Body>

   </wsp:SLC>
</wsp:Policy>
```

10.3. Policy Management System (Priority-Based Policy Application based on SLC)

Priority-based policy application schema was the former suggested model for applying defined policies according to a valid priority index based on SLC. The main aim of this model was to manage established security rings and to assure applied security mechanisms based on defined security rings.

Policy Match Gate is a structural component to schedule policy application tasks and to manage resources for this policy mapping by managing PMS virtual cluster. This cluster includes special VMs to apply defined policies to data and a *Pre-Processing Unit* (PPU) to distribute requests on VMs. Hence, several components and variables are defined as follows:

- Virtual Machines (VM): Given I virtual machines that are hosted by PMS virtual cluster and denoted as $\{VM_1, VM_2, \dots, VM_I\}$ where the current and optimal CPU utilization of VM_i $(i = 1,2,\dots I)$ is CVM_i and OVM_i.

- Applied Policy Data (F): Given J files waiting to be processed for applying security polices based on defied levels of SLC where the $j-th$ file corresponds to W_j that represents the size of F_j.

Table C.1. An Example of SI Value in Cryptography Protocol

	Mechanism	Algorithm	Key-Size	SI	Selection
1	Symmetric	AES	256	1.0	
2	Re-Encryption	Manual	-	1.0	√
3	Symmetric	AES	192	1.3	
4	Symmetric	AES	128	1.7	
5	Symmetric	3DES	256	1.7	√
6	Asymmetric	RSA	2048	1.7	
7	Symmetric	3DES	128	2.0	
8	Asymmetric	RSA	1024	2.0	
9	Symmetric	DES	256	2.2	
10	Symmetric	DES	192	2.5	
11	Re-Encryption	Time-Based	-	2.5	
12	Asymmetric	RSA	512	3.2	
13	Symmetric	DES	128	3.2	
14	Asymmetric	Diffie-Hellman	2048	3.5	
15	Asymmetric	Diffie-Hellman	1024	3.8	
			Total Value of SI : 1.35		

- SLC Index (SI): SLC index is a defined index $(1 \le SI \le 4)$ for prioritizing the scheduling process according to the chosen security mechanisms in SLC (the value of 1 is the highest priority and the value of 4 is the lowest priority). In fact, this index is calculated based on the confidentiality of the security level regarding to the all security mechanism in SLC. Table C.1 shows an example of how to define SI for cryptography protocol. The final value of SI is calculated according to the average value of SI in each protocol (Table C.2).

The main objective of PMS is how to schedule item of set $F = \{f_1, f_2, ... f_J\}$ where the $j-th$ file is associated with SLC_δ ($\delta \in \{1,2, ... N\}$ and N represents the total number of dedicated security levels) to I virtual machines by classifying them to several load groups based on SI as a priority index. In fact, PMS uses a priority-based scheduling schema based on a defined prioritization algorithm. Accordingly, PPU tries to classify the queue to 3 main priority categories:

```
$for\ (int\ x = 1\ ;\ x <= J\ ;\ ++x)\{$
  $\alpha = SLC\ (f_x);$
  $switch\ (\alpha)\{$
    $case\ (1 \leq \alpha < 2): Pf_x = 1;$
    $case\ (2 \leq \alpha < 3): Pf_x = 2;$
    $case\ (3 \leq \alpha \leq 4): Pf_x = 3;\ \}$
```

Table C.2. An Example for the final value of SI Based on Calculated Value in Each Protocol

	Protocol	SI
1	Cryptography	1.35
2	Authentication	2.10
3	Key Management	2.25
4	Transport	1.70
5	Signature	2.25
6	Access Control	1.50
	Final Value of SI for SLC_δ : 1.85 ($\delta \in \{1,2, ... N\}$ and N represents the total number of dedicated security levels)	

The scheduling process in each category is based on QoS-Driven in cloud-based environments [105] to use SI associated with other task attributes such as user privilege, expectation, task length and the pending time in queue to compute the priority and sort tasks by the priority. In fact, the scheduler sorts the high priority level based on the value of SI associated with QoS-Driven parameters and distributes them to VMs based on the value of CVM_i and OVM_i in each VM (Algorithm C.1).

If the first step, The high priority queue ($Pf_x = 1$) is sorted based on the value of SI and QoS-Driven parameters. If fact, SI is the first priority for comparison and QD (QoS-Driven Parameter) is the second in the case of same SI values (Step 2). The process of scheduling is started based on the optimal and current utilization of each virtual machine. In the first round, all sorted tasks in the first category are distributed in VMs according to the value of CVM_i and OVM_i. If a first priority task is not assigned on any VMs due to the utilization value, it will be re-added to the high priority queue. (Step 3).

The second priority queue ($Pf_x = 2$) is sorted in the next step same as the previous queue according to the value of SI following by the value of QD (Step 4 and 5). Furthermore, the same procedure is applied for the lowest priority queue ($Pf_x = 3$) according the current and optimal value of associated VMs (Step 6 and 7).

The second component in PMS is *Policy Check Point* to ensure about applied policies based on defined SLC. After applying policies to each file, the checkpoint component reviews the applied polices to confirm the validity security terms such as access management and authentication headers, encryption algorithms and keys, re-encryption procedures and associated algorithms. After this validation process, a confirmation is sent to cloud customer about the successful policy application to data.

Algorithm C.1.
Policy Management Scheduling System Based on SLC Index

Input: set $F = \{f_1, f_2, \dots f_J\}$ where the $j-th$ file is associated with SLC_δ ($\delta \in \{1,2,\dots N\}$ and N represents the total number of dedicated security levels)
Output: $D(F)$; Applying Defined Policies to all items in set F.

```
1: $\forall f_x \in F: Pf_x = 1$
   $\{AddTask\ (f_x, Queue_1)\}$
2: $Sort\ (Queue_1, SI(f_x));$
   $if\ (\exists (f_m, f_n) \in Queue_1: SI(f_m) = SI(f_n))\ then, Sort((f_m, f_n), QD(f_m), QD(f_n));$
3: $\forall f_x \in Queue_1$
   $if\ \left(\exists VM_k \in PMS: \left((CVM_k \neq OVM_k) \wedge \left((OVM_k - CVM_k) \geq W(f_x)\right)\right)\right)\ then,$
   $Assign(f_x, VM_k);$
   $else, ReAdd\ (f_x, Queue_1);$
4: $Sort\ (Queue_2, SI(f_x));$
   $if\ (\exists (f_m, f_n) \in Queue_2: SI(f_m) = SI(f_n))\ then, Sort((f_m, f_n), QD(f_m), QD(f_n));$
5: $\forall f_x \in Queue_2$
   $if\ \left(\exists VM_k \in PMS: \left((CVM_k \neq OVM_k) \wedge \left((OVM_k - CVM_k) \geq W(f_x)\right)\right)\right)\ then,$
   $Assign(f_x, VM_k);$
   $else, ReAdd\ (f_x, Queue_2);$
6: $Sort\ (Queue_3, SI(f_x));$
   $if\ (\exists (f_m, f_n) \in Queue_3: SI(f_m) = SI(f_n))\ then, Sort((f_m, f_n), QD(f_m), QD(f_n));$
7: $\forall f_x \in Queue_3$
   $if\ \left(\exists VM_k \in PMS: \left((CVM_k \neq OVM_k) \wedge \left((OVM_k - CVM_k) \geq W(f_x)\right)\right)\right)\ then,$
   $Assign(f_x, VM_k);$
   $else, ReAdd\ (f_x, Queue_3);$
```

10.4. Performance Evaluation of Priority-Based Policy Application System

A simulated environment was designed based on PMS virtual cluster, as well as, 3 defined SLC with calculated SI (*i.e.* high secure, medium secure and low secure ring). In this experiment, the scalability of the scheduler was examined by increasing the number of policy application tasks associated with defined SLCs.

Figure D.1 shows the result of this experiment in details. As expected, the total processing time in tasks associated with high secure SLC is considerable higher than other tasks due to the complexity of high secure algorithms such as encryption key sizes and access control policy headers. On the other hand, the total processing time difference by increasing 100 tasks is significantly reduced after 1600 tasks and proves that the algorithm is scalable enough in by different types of workloads.

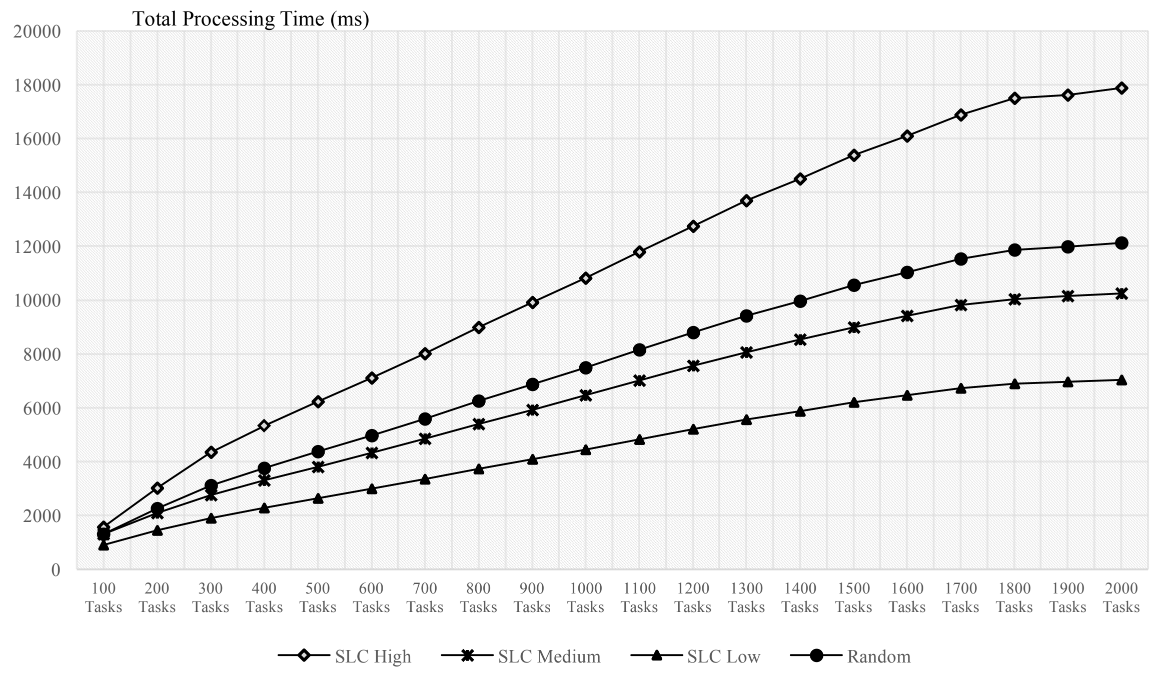


Figure. D.1. Performance Evaluation of Policy Management System by Increasing Number of Policy Application Tasks

10.5. Policy Matrix Super Class

Policy Matrix Super Class (PMSC) is the former version of CSRT that was defined based on Protection Ontology [43]. Regarding to this ontology, PMSC is capable of directing various security algorithms and classifying them based on the security mechanisms in the first level and security protocols in the second level of classification. If fact, the cloud service provider only needs to update the algorithm level based on new or revoked features and capabilities, and to categorize them according to the first and second level of protection ontology.

The process of mapping policies is done based on defined resources in each protocol, mechanism or algorithm. Hence, PMSC is created by establishment of appropriate mapping between security terms and semantic concepts. Table E.1 shows an example of this establishment in details. Each security term is mapped to a semantic resource based on a protocol (row), a mechanism (column) and an algorithm (leaf).

Table E.1
Mapping Between Security Terms and Semantic Concepts Based on PMSC

Security Term	Semantic Resource	Protocol	Mechanism	Algorithm	Row	Column	Leaf
Permanent Role Access Control	`PermanentRoleClass`	Access	Role	Permanent	1	1	1
Temporary Role Access Control	`TemporaryRoleClass`	Access	Role	Temporary	1	1	2
Geographical Access Control (IP-Based)	`GeoClass`	Access	Repudiation	Geo	1	2	1
Hardware Access Control (MAC)	`HardwareAccessClass`	Access	Repudiation	Hardware	1	2	2
Software Access Control (OS, Browser)	`SoftwareAccessClass`	Access	Repudiation	Software	1	2	3
AES Encryption	`AESClass`	Cryptography	Symmetric	AES	2	1	1
DES Encryption	`DESClass`	Cryptography	Symmetric	DES	2	1	2
RSA Encryption	`RSAClass`	Cryptography	Asymmetric	RSA	2	2	1
Manual Re-Encryption	`ManualREClass`	Cryptography	Re-Encrypt	Manual	2	3	1
Time-Based Re-Encryption	`TimeREClass`	Cryptography	Re-Encrypt	Periodically	2	3	2
Symmetric Key Wrapping	`SymmetricWrClass`	Key Manage	Wrapping	Symmetric	3	1	1
Symmetric Key Wrapping	`AsymmetricWrClass`	Key Manage	Wrapping	Asymmetric	3	1	2
Symmetric Key Derivation	`SymmetricDerClass`	Key Manage	Derivation	Symmetric	3	2	1
Asymmetric Key Derivation	`AsymmetricDerClass`	Key Manage	Derivation	Asymmetric	3	2	2
MD5 Signature Algorithm	`MD5Class`	Signature	MAC	MD5	4	1	1
SHA Signature Algorithm	`SHAClass`	Signature	MAC	SHA	4	1	2
CBC Signature Algorithm	`CBCClass`	Signature	Hash	CBC	4	2	1
H-MAC Signature Algorithm	`HMACClass`	Signature	Hash	H-MAC	4	2	2
Asymmetric Digital Signature Algorithm	`DSSClass`	Signature	Digital	DSS	4	3	1
User-Pass Authentication	`UserPassClass`	Authentication	Normal	User-Pass	5	1	1
User-Pass and Secure Word	`SecureWordClass`	Authentication	Normal	S-Word	5	1	2
Using Authenticator Component	`AuthenticatorClass`	Authentication	Double	Auth-App	5	2	1
Using Recovery Email/Number	`RecoveryClass`	Authentication	Double	Recovery	5	2	2
Online One-Time Password	`OnlineOneTimeClass`	Authentication	Double	Online	5	3	1
Offline One-Time Password	`OfflineOneTimeClass`	Authentication	Double	Offline	5	3	2
SSL Connection	`SSLClass`	Transport	SSL	-	6	1	-
TLS Connection	`TLSClass`	Transport	TLS	-	6	2	-
Normal Connection	`NormalConClass`	Transport	Normal	-	6	3	-

10.6. Syntactic and Semantic Analysis of Policy Matrix Object

In the first syntactic analysis, all capable security algorithms in each security protocol are analyzed. If the user does not apply for any of algorithms in a security mechanism, the first analysis is stopped (Step 2). Also, if the user selects only one security algorithm, the selected algorithm achieves the highest priority (Step 3). However, if the user selects more than one algorithm in one security mechanism column, the selected items should be prioritized based on the capabilities and requirements with the highest priority of 1 (perfect match), low priorities of 2 and 3 (close and possible match) and no-match priority with value of 0.

The process of semantic analysis is performed in 3 steps. In the first steps all of the security algorithms with value of 1 are considered and analyzed. The simplest scenario is happened when there are one or more than one un-conflicted algorithms with value of 1 in each security protocols (Step 5).

The second step of semantic analysis uses three main functions to match all desired policies with un-conflicted security terms: Elimination, Substitution and Finalization. The first method eliminates one of the conflicted terms without any significant effect to users' security requirements and the second on tries to substitute one of the conflicted terms with other priorities. If all of conflicted terms are modified and the current policy matrix object does not meet any confliction in all algorithms with value of 1, the object is finalized and other priority values (*i.e.* 2 and 3) are changed to 0 (Step 6). Else, the next semantic analysis round is called. The process of semantic analysis needs 3 rounds to check conflicted security terms with same functions.

If the confliction is solved by each of these modification functions, the finalization function of each round transfers the object to the previous round (Step 7). After the solving conflicts in each priority the policy package object is created from policy package super class (Step 8).

Algorithm F.1
Syntactic and Semantic Analysis of Policy Matrix Object

Input: pm_{object}: a policy matrix object form policy matrix super class that is created by policy engine and modified with desired inputs by cloud customer.

Output: $\begin{cases} pp_{object} & pm_{object}.val = true \\ Send\ (pm_{object}, VE, PE) & pm_{object}.val = falsa \end{cases}$

A Policy Package Object is created if the syntactic and semantic analysis is confirmed in validity engine, else, Policy Object is sent to PE.

```
1:  $\forall\ Protocol : Protocol\ \in pm_{object}$
    $Protocol.Check\ (Protocol)$
2:  $if\ (\nexists\ Algorithm \in Protocol)\ then\ Protocol.Next(Protocol)$
3:  $if\ (\exists!\ Algorithm\ \in Protocol)\ then,$
    $\begin{Bmatrix} SetPriority(Algorithm, 1) \\ Protocol.Next\ (Protocol) \end{Bmatrix}$
4:  $if\ (\exists\ Algorithm\ \in Protocol : Algorithm.count > 1)\ then,$
```

```
        $\begin{Bmatrix} SetPriority(Algorithm, 1,2,3,0) \\ Protocol.Next\ (Protocol) \end{Bmatrix}$
5:      $CheckPeriority(pm_{obejct}, 1)\{$
        $Policy_{Package}\ pp_{object} =\ new\ Policy_{Package}(CC_u, req)$
        $for\ (\forall\ Algorithm\ \in pm_{object} : Algorithm.Value = 1)$
           $SetPolicy(pp_{object}, Algorithm)\}$
6:      $CheckPeriority(pm_{obejct}, 2)\{$
        $Policy_{Package}\ pp_{object} =\ new\ Policy_{Package}(CC_u, req)$
        $for\ (\forall\ Algorithm\ \in pm_{object} : Algorithm.Value = 2 \vee 3)($
           $Substitute\ (Algorithm, pm_{object}) \vee Eliminate\ (Algorithm, pm_{object});$
           $Finalize\ (\forall\ Algorithm\ \in pm_{object} : Algorithm.Finalize);$
           $SetPolicy(pp_{object}, Algorithm))\}$
7:      $if\ (pm_{objcet}.CheckConflict = false)\ then, \{$
           $CheckPeriority(pm_{obejct}, 3)\{$
           $Policy_{Package}\ pp_{object} =\ new\ Policy_{Package}(CC_u, req)$
           $for\ (\forall\ Algorithm\ \in pm_{object} : Algorithm.Value = 2 \vee 3)($
              $Substitute\ (Algorithm, pm_{object}) \vee Eliminate\ (Algorithm, pm_{object});$
              $Finalize\ (\forall\ Algorithm\ \in pm_{object} : Algorithm.Finalize);$
              $SetPolicy(pp_{object}, Algorithm))\}$
          $Repeat\ Step(7);\}$
        $else\ (pm_{object}.val = true);$
8:      $if\ (pm_{object}.val = true)\ then, send\ (pp_{object})$
        $else\ \left(send\ (pm_{object}, VE, PE)\right);$
```

10.7. Policy-Based Access Control Framework (PAF) based on Protection Ontology

Policy-Based Access Control Framework (PAF) is the proposed model that uses *Protection Ontology* [43] to create dedicated security levels (rings) for various cloud customers based on different requirements. Based on this ontology, security terms are classified into 3 levels: protocols, mechanisms and algorithms.

Access management and user authentication are the basic protocols of our model to map between cloud users, cloud customers and cloud providers and other protocols such as cryptography and transport have not been considered in PAF. *Protection Ontology* uses WS-Policy and policy matrix super class to establish security policies for each protocol according to the capabilities of service provider and requirements of cloud customers. As an example, consider a cloud customer (*e.g.* A Hospital) subscribes cloud-based services for users (*e.g.* doctors, nurses, patient, office staff, *etc.*) and create 3 dedicated security rings (levels) in the cloud based on the sensitivity of stored data.

```
<wsp:Policy>
   <wsp:ExactlyOne>
      <wsp:all>
         <security:AccessManagementProtocol rdf:ID="AccessManagementRequirement">
            <security:RoleMechanism rdf:ID="RoleClassRequirement">
               <security:PermanentAlgorithm              rdf:resource="PermanentRoleClass"
SP="#PATIENT#DOCTOR"/>
            </security:RoleMechanism>
            <security:ReputationMechanism rdf:ID=ReputationClassRequirement">
               <security:GeoAlgorithm rdf:resource="GeoClass" SP="#Germany"/>
            </security:ReoutationMechanism>
         </security:AccessManagementProtocol>
         <security:AuthenticationProtocol rdf:ID="AuthenticationRequirement">
            <security:DoubleMechanism rdf:ID="DoubleClassRequirement">
               <security:AuthenticatorAlgorithm rdf:resource="AuthenticatorClass"/>
            </security: DoubleMechanism>
         </security: AuthenticationProtocol>
      </wsp:all>
   </wsp:ExactlyOne>
</wsp:Policy>
```

The above-mentioned example is the access control and user authentication protocols in one of the dedicated rings regarding to the requirements of cloud customer. Accordingly, this security ring uses a permanent role based access control algorithm and geo-based reputation algorithm (*e.g.* IP addresses inside Germany) for access management protocol and an authenticator (*i.e.* second password generator) for double user authentication protocol. The aim of our research is how to map access requests from cloud users to defined access management and user authentication protocols.

The architecture of PAF is based on 3 main components: Policy Database, Policy Mapping and Checkpoint (Fig. G.1). After generation of various security rings by cloud customer based on protection ontology, each data of the cloud customer is assigned to one of the dedicated rings in associated with sub-policies (e.g. a patient's lab result is assigned to the dedicated ring in example 1 with "Doctor" and "Patient" as the sub-policies of permanent role access mechanism and "Germany" as the sub-policy of reputation mechanism). In fact, values of the established mechanisms in each policy are defined as sub-policies and stored in policy database. The structure of storing policies in the database is based on policy/sub-policy architecture. This means, each data has one record that shows a security level ID as the main policy and several properties as sub-policies.

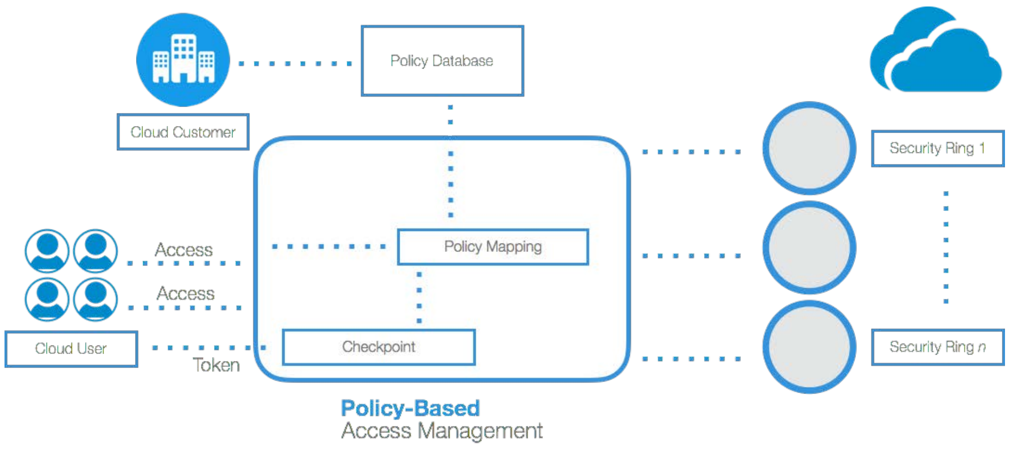


Fig. G.1 Architecture of PAF

PAF uses the policy mapping component to map between defined policies and access request. Typically, cloud users log into their accounts with the simple user-password authentication method. By this login process, the lowest security ring of the cloud customer will be available.

Indeed, the simplest authentication process makes assigned data in the lowest security level of cloud customer available if there is not any defined sub-policy for specific data in the lowest security ring.

Checkpoint component is responsible to generate and manage session tokens for a flexible and scalable access control process. After a basic login, a session token is generated by the checkpoint component to provide accesses for the cloud user. For the sub-policies in the lowest level or policy/sub-policies in the the higher levels, checkpoint will update the session token for current and further accesses before session is expired. Figure G.2 and algorithm G.1 show the process of policy-based accessing in details.

According to the algorithm, a session token object is created by the access token class in the first step and is updated based on every mapping processes between requested data and the main policy and associated sub-policies that is assigned to the requested data.

For instance, if a cloud user tries to access a particular document in the high security ring, the main policy and associated sub-policies that were checked before and were stored in the session token, are not mapped again and only the un-mapped policy and sub-policies are checked by the policy mapping and check point component.

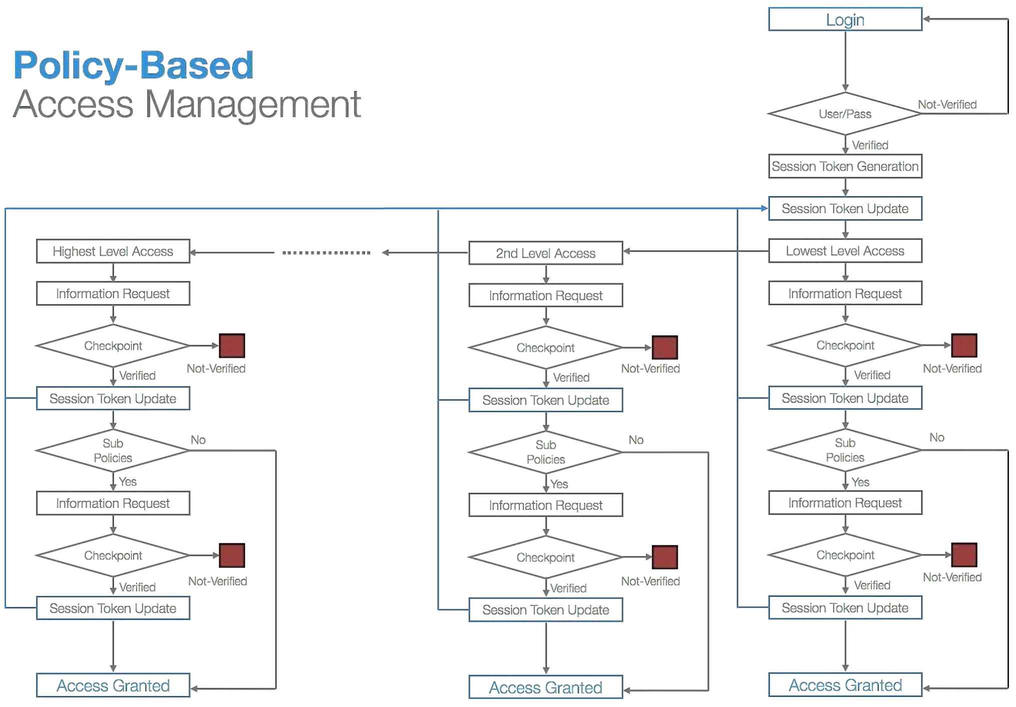


Fig. G.2. Algorithm of Accessing to Stored Data in Clouds based on Policies and Sub-Policies

Algorithm G.1.
The Process of Access Token Generation

```
1:     $Login\ (U_i, P_i);$
1.1.   $if\ ((U_i == Username_i)\&\&(P_i == Password_i))\ then\ \{$
         $Token\ T_i = CreateToken(UID_i, CC_j);$
         $for\ (int\ x = 0; x \le CC_j.L_1.count(sp); ++x)\{$
           $if\ (CC_j.L_1.sp_x\ != true)\ then\ \{$
              $AccessGrant(UID_i, CC_j.L_1.sp_x);$
              $T_i.Update(CC_j.L_1.sp_x);\}$
           $else\ (inforeq(UID_i, CC_j.L_1.sp_x));\}\}$
       $else\ (inforeq(UID_i, CC_j.L_1));\}$
```

// U, CC, L and sp are cloud user, cloud customer, security level and sub-policy respectively.
// In the first step, the cloud user uses the username and password to login and if both of them are matched, an access token object is created from the access token class based on the cloud user ID and cloud customer ID. After that, all of the sub-policies in the lowest level of cloud customer rings are checked and all records without specific sub-policies are available to the cloud user and the token object is updated. Moreover, if the user needs a particular data with defined sub-policies in the lowest level, additional info needs to be provided (*e.g.* second pass, IP address, etc.) based on the established sub-policies.

```
2.     $AccessReq(UID_i, D_n);$
2.1.   $RequestSubPolicy\ RSPtemp = CheckSP(D_n);$
2.2.   $for\ (int\ x = 0; x \le RSPtemp.Count(SP);\ ++x)\{$
         $for\ (int\ y = 0; y \le T_i.GSP.Count(SP);\ ++y)\{$
           $if\ (RSPtemp_x == T_i.TGSP_y)\ then\ RSPtemp_x = true;\}\}$
2.3.   $for\ (int\ x = 0; x \le RSPtemp.Count(SP);\ ++x)\{$
         $if\ (RSPtemp_x == false)\ then\ \{$
```

```
          $inforeqtemp = inforeq(UID_i, CC_j.L_{RSPtemp}.RSPtemp_x);$
            $if\ (inforeqtemp == true) then\ \{$
              $T_i.Update(CC_j.L_{RSPtemp}.RSPtemp_x);$
              $RSPtemp_x = true; \}\}\}$
2.4.   $for\ (int\ x = 0; x \le RSPtemp.Count(SP);\ ++x)\{$
          $if\ (RSPtemp_x == true)\ then\ temptotal = temptotal + 1;$
2.5    $if\ (temptotal == RSPtemp.Count(SP)) then\ AccessGrant(UID_i, D_n);$
       // $D$ and $GSP$ are the requested data by cloud user and the previous granted accesses in session token respectively.
       // When a cloud user tries to access a particular data in higher security levels, the main policy and associated sub-policies are retrieved for mapping process. If the main policy and some of the associated sub-policies are mapped before and cached in the session token, there is no need to retrieve them again and to request for additional info from cloud user. Otherwise, the requested policy and sub-policies should be mapped with additional info that are provided by cloud user and the session token are updated by each new mapping process. Finally, the access to that particular data is granted if the main policy and all of the associated policies are checked and mapped by policy mapping component.
3.     $if\ ((Login\ (U_i, P_i)))\ \|\ (T_i.timeexpired == true))\ then\ delete(T_i);$
       // The session is deleted if the cloud user logouts from the system of the session expired time is reached.
```