Policy Management Engine (PME): A Policy-Based Schema to Classify and Manage Sensitive Data in Cloud Storages

Faraz Fatemi Moghaddam*,†, Philipp Wieder*, Ramin Yahyapour*,†
*Gesellschaft fur wissenschaftliche Datenverarbeitung mbH Göttingen (GWDG), Göttingen, Germany
†Institute of Informatics, Georg-August-Universität Göttingen, Göttingen, Germany
Email: {faraz.fatemi-moghaddam, ramin.yahyapour, philipp.wieder}@gwdg.de

***Abstract*— One of the most challenging issues regarding to the information policy concerns in cloud computing environments is to provide an appropriate level of security for the stored data in cloud storages. In fact, each individual cloud customer needs to be granted reliable security level(s) based on defined details in SLA. The main aim of this paper is to propose multi-level policy-based schema to classify and manage data in cloud storages based on the sensitivity and confidentiality for enhancement of reliability in cloud computing environments. Furthermore, an efficient algorithm has been introduced to ensure the accuracy and authenticity of applying and managing defined policies according to the capabilities of the cloud providers and requirements of cloud customers. The most important characteristic of this model is syntactic and semantic analysis of requested policies by validity engine to provide reliable mapping between security mechanism and requested policies. Moreover, Policy Match Gate and Policy Checkpoint have been introduced to ensure about the policy application process for all stored data based on defied policies in Security Level Certificate.**

Keywords—Cloud Computing; Policy Management; Security; Policy Application; SLA; Security Level Certificate.

I. INTRODUCTION

Cloud computing technology is an unprecedented paradigm that uses the concepts of virtualization, processing power, storage, connectivity, and sharing to provide pool of resources, store and share them between various devices via a broad network (*i.e.* Internet) to offer on-demand services to end users in compliance with the concepts of isolation, security, distribution, and elasticity [1]. Despite the considerable benefits of cloud-based environments [2], there are some information policy concerns such as security, privacy and access control that have affected the reliability of this newfound technology [3]. One of the most challenging issues regarding to the information policy concerns is to provide an appropriate level of security for the stored data in cloud storages. In fact, each individual cloud customer needs to be granted reliable security level(s) based on defined details in SLA [4].

These security levels might be common for all customers or independent based on the data sensitivity. Applying a single security level for all stored data is not efficient and takes considerable processing power to manipulate sensitive and also non-sensitive data. On the other hand, managing multiple security levels is the most challenging concern in multi-level policy models and needs an appropriate and efficient algorithm. The most popular approach to express high-level security constraints is based on the usage of metadata and languages for the specification of security policies [5]. The main aim of this paper is to propose multi-level policy-based schema to classify and manage data in cloud storages based on the sensitivity and confidentiality for enhancement of reliability in cloud computing environments. Furthermore, an efficient algorithm has been introduced to ensure the accuracy and authenticity of applying and managing defined policies according to the capabilities of the cloud providers and requirements of cloud customers.

II. RELATED WORKS

There are various policy-related classification algorithms that are based on language and metadata for specification of security policies. One of the defined frameworks that includes policy model, definition language, sets of policy technologies and a policy architecture meta model is proposed by IETF [6] to represent, manage, share and reuse policies and policy information in a vendor-independent, interoperable and scalable manner. IETF model mostly focuses on network policies to control IP-Security and Quality of Service (QoS) [7]. One of the most challenging efforts in this model is to map IETF models into specified implementation schema [8] such as Web Based Enterprise Management (WBEM) [9] or Directory Enabled Network (DEN) [10].

Ponder is an object oriented-based policy management framework that was proposed by Imperial College [11] and includes general architecture, policy deployment model and various extensions for access control schema and QoS management. This model allows users to define events, constraints, constants and other reusable elements that can be part of many policies and allows the instantiation of typed policy specification to support parameterization of policies. The main drawback of Ponder is the lack of generality [8] by using several basic policy types and compositing policy types each with various syntax.

KAoS is an ontology language uses the Web Ontology Language (OWL) [12] to allow users to define policies to grant predictability and controllability of agents and distributed systems such as grid computing and multi-agent systems [13]. Four main types of policies are defined in KAoS: *Positive-Authorization*, *Negative-Authorization*, *Positive-Obligation* and *Negative-Obligation* and similar to IETF, each policy is associated with management properties. KAoS provides two sets of services: Domain Services (that enable the hierarchical grouping of users and computational entities to administrate policies easier) and Policy Services (that is based on specification, conflict resolution, management and policy enforcement) [14].

In 2002, HP labs proposed a policy framework (Rei) [15] to provide an independent domain policy specification based on deontic constructs and F-OWL reasoned [16] 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 behavior of the policy) and Meta-Meta Policy (to allow the setting of precedencies based on default meta policies).

Di Modica and Tomarchio (2015) [17] suggested an approach 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 [19]. 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).

In overall, the main concern of described models is the discovery, interoperability and compatibility of security requirements based on characteristics of current distributed networks and cloud-based environments. Hence, this paper uses the concepts of security ontology to define reliable and interoperable security policies in virtualized infrastructure and to grant syntactic and also semantic matching of security policies.

III. Proposed Model

The architecture of our proposed model is based on Protection Ontology and have been described in two phases: Establishment of Security Rings (ESR) and Policy Management System (PMS). Protection ontology [19] is a defined object oriented framework that includes potential security terms (*i.e.* protocols, mechanisms and algorithms) and appropriate relations between them to provide an efficient and reliable policy management framework. This ontology is based on two super classes (Policy Matrix and Policy Packages) and three levels of security includes Protocol Level (Access Control, Cryptography, Key Management, Transport, Authentication and Signature Protocols), Mechanism Level (Several security mechanisms for each protocol that are defined to provide appropriate relation between the highest level and the lowest level of architecture) and Algorithm Level (The lowest level of the protection ontology that includes different security algorithms). Fig. 1 shows an example of protection ontology in details. In fact, the main aim of this ontology is to categorize potential security terms of cloud provider (based on capabilities) and establish appropriate connections between them for a reliable policy management model.

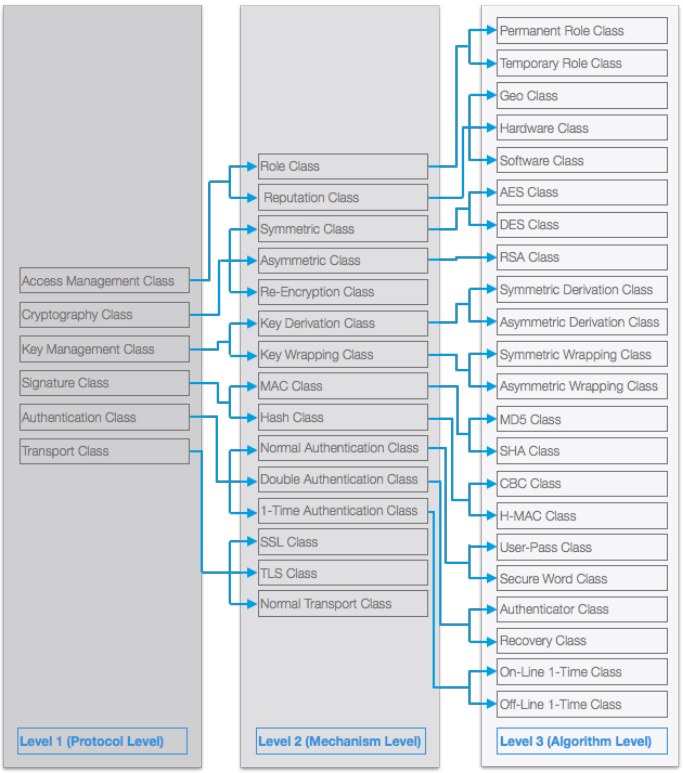

Fig. 1. Structure of Protection Ontology

To have a better understating of the model, the following terms are defined:

- Cloud Provider (CP): A service provider that offers customers cloud-based services such as storage. In our suggested model, CP offers security models as a service to customers and guarantees the availability and reliability of offered terms based on defined details in SLA.
- 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 that use cloud-based services that are offered by CC according to the internal contracts. (e.g. Students or Lecturers in a University).

As was described, the proposed model is based on two phases and the overview of these phases have been shown in figure 2.

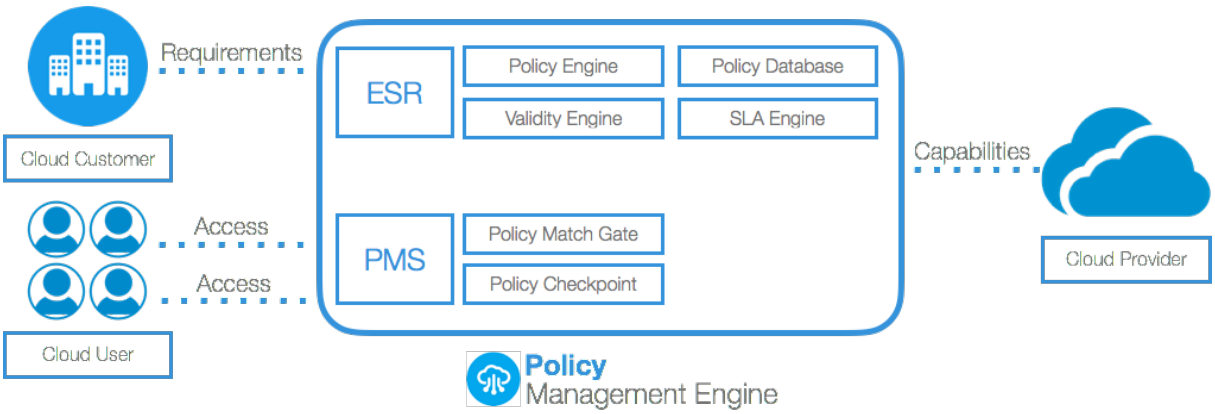

Fig. 2. Overview of Policy Management Engine

A. Establishment of Security Rings (ESR)

The first phase of our model is providing security as a service model for cloud customers based on protection ontology to create dedicated security rings (levels) according to their requirements and the capabilities of the cloud provider. In this phase, cloud customers select desired security mechanisms and set appropriate controls and policies regarding to the available and offered mechanisms of service provider. Fig. 3. shows the process of establishment of security rings in details.

In this model, Policy Database component is updated based on the potential security policies according to the capabilities of the service provider (*i.e.* new or revoked mechanisms). Typically, these updates have been done in the lowest level of protection ontology (Algorithm Level) by creation or elimination of new classes and affect relevant classes in upper levels based on inheritance concepts. Finally, the policy matrix super class is updated and one policy matrix object is created by Policy Engine component for cloud customers regarding to the last updates.

The created object is accessible to cloud customer by API and the customer selects appropriate security mechanisms in each protocol based on requirements. This selection may be one or more in each protocol and is prioritized by the customer if the selection in each mechanism is more than one algorithm.

The selected object is sent to Validity Engine component for syntactic and semantic analysis of the selected policy object. The process of validity analysis is performed in two steps according to syntactic and semantic checking of policy matrix object (Algorithm 1) to provide an un-certified policy package object based on WS-Policy [18].

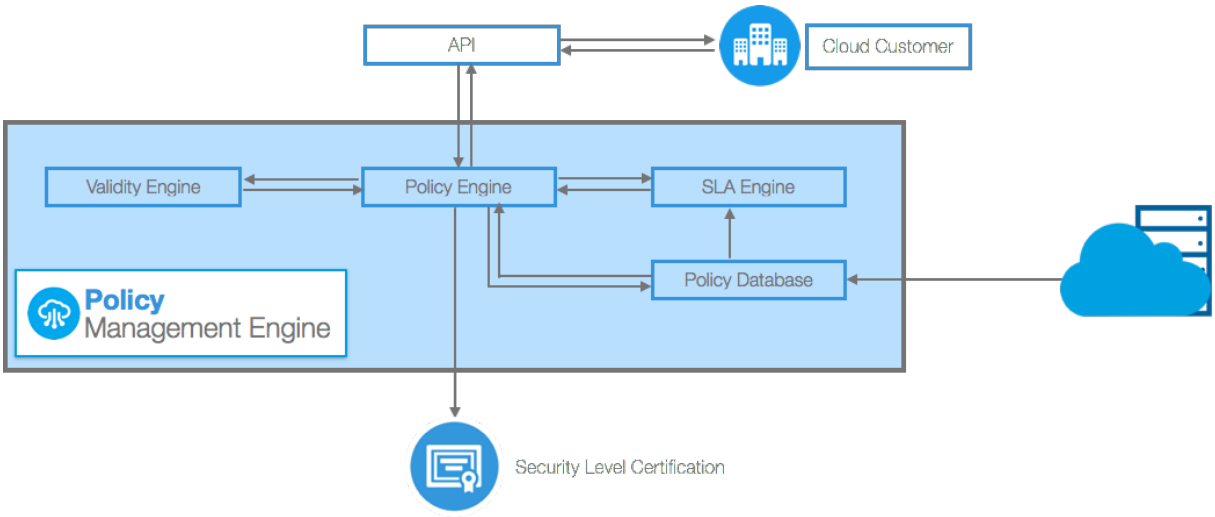

Fig. 3. Establishment of Security Rings

Algorithm 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

$for\ (int\ x\ =\ 1\ ;\ x\ <=\ M\ ;\ ++x)\ \{$
$for\ (int\ y\ =\ 1\ ;\ y\ <=\ N\ ;\ ++y)\ \{$
$for\ (int\ z\ =\ 1\ ;\ z\ <=\ K\ ;\ ++z)\ \{syntemp_{xy} = syntemp_{xy} + pm\_object.x.y.z;\}$
$if\ (syntemp_{xy} == 1\ ||\ syntemp_{xy} == 0) then\ break;$
$else\ for\ (int\ z\ =\ 1\ ;\ z\ \leq\ K\ ;\ ++z)\ \{SynPeriority(pm\_object.x.y, 1,2, \dots, syntemp_{xy});\}\}\}$

//where M,N and K are the number of rows, columns and leafs in policy matrix respectively.

//In the first syntactic analysis, all capable security algorithms in each security mechanism are analyzed. If the user does not apply for any of algorithms in a security mechanism, the first analysis is stopped. Also, if the user selects only one security algorithm, the selected algorithm achieves the highest priority. 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 prioritization problem is solved in [19].

2

$1st\_Periority(pm\_obejct);$
$if\ (valtemp == true)\ \{$
$Policy\_Package\ pp\_object\ =\ new\ Policy\_Package(User_u, req)$
$for\ (int\ x\ =\ 1\ ;\ x\ <=\ M\ ;\ ++x)\ \{$
$for\ (int\ y\ =\ 1\ ;\ y\ <=\ N\ ;\ ++y)\ \{$
$for\ (int\ z\ =\ 1\ ;\ z\ <=\ K\ ;\ ++z)\ \{$
$if\ (pm_{object}.x.y.z == true)\ then\ pp\_object.sp_x = p_{xyz};\}\}\}$
$esle\ \{2nd\_Periority(pm\_obejct);$
$if\ (valtemp == true)\ \{$
$Policy\_Package\ pp\_object\ =\ new\ Policy\_Package(User_u, req)$
$for\ (int\ x\ =\ 1\ ;\ x\ <=\ M\ ;\ ++x)\ \{$
$for\ (int\ y\ =\ 1\ ;\ y\ <=\ N\ ;\ ++y)\ \{$
$for\ (int\ z\ =\ 1\ ;\ z\ <=\ K\ ;\ ++z)\ \{$
$if\ (pm_{object}.x.y.z == true)\ then\ pp\_object.sp_x = p_{xyz};\}\}$
$else\ \{\ 3rd\_Periority(pm\_obejct);$
$if\ (valtemp == true)\ \{$
$Policy\_Package\ pp\_object\ =\ new\ Policy\_Package(User_u, req)$
$for\ (int\ x\ =\ 1\ ;\ x\ <=\ M\ ;\ ++x)\ \{$
$for\ (int\ y\ =\ 1\ ;\ y\ <=\ N\ ;\ ++y)\ \{$
$for\ (int\ z\ =\ 1\ ;\ z\ <=\ K\ ;\ ++z)\ \{$
$if\ (pm_{object}.x.y.z == true)\ then\ pp\_object.sp_x = p_{xyz};\}\}\}$
$else\ Send\ (pm\_object, VE, PE);\}\}\}$

//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. 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. 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. After the solving conflicts in each priority the policy package object is created from policy package super class.

The policy engine published an un-certified policy document based on the policy package object that was created by validity engine. This document is sent to SLA engine to calculate and provide the billing details and service agreement. After generating the SLA document, policy engine sends this document and the un-certified policy document to cloud customer for final approval. The Security Level Certification (SLC) is generated after the approval to create a dedicated security ring for cloud customer. Example 1 is a generated security level certification based on WS-Policy and protection ontology.

```
<wsp:Policy>
   <wsp:ExactlyOne>
      <wsp:all>
         <security:AccessManagementProtocol rdf:ID="AccessManagementRequirement">
            <security:RoleMechanism rdf:ID="RoleClassRequirement">
               <security:PermanentAlgorithm rdf:resource="PermanentRoleClass"/>
            </security:RoleMechanism>
            <security:ReputationMechanism rdf:ID=ReputationClassRequirement">
               <security:GeoAlgorithm rdf:resource="GeoClass"/>
            </security:ReoutationMechanism>
         </security:AccessManagementProtocol>
         <security:CryptographyProtocol rdf:ID="CryptographyRequirement">
            <security:SymmetricMechanism rdf:ID="SymmetricRequirement">
               <security:AESClass rdf:resource="AESClass", key:"256"/>
            </security:SymmetricMechanism>
            <security:ReEncryptionMechanism rdf:ID=REEncryptionRequirement">
               <security:ManualReEncryption rdf:resource="ManualReClass"/>
            </security:ReEncryptionMechanism>
         </security:CryptographyProtocol>
         <security:SignatureProtocol rdf:ID="SignatureRequirement">
            <security:HashMechanism rdf:ID="HashClassRequirement">
               <security:CBCAlgorithm rdf:resource="CBCClass"/>
            </security: HashMechanism>
         </security: SignatureProtocol>
         <security:KeyManagementProtocol rdf:ID="KeyManagementRequirement">
            <security:KeyWrappingMechanism rdf:ID="KeyWrappingClassRequirement">
               <security:SymmetricAlgorithm rdf:resource="SymmetricWrClass"/>
            </security: KeyWrappingMechanism>
            <security:KeyDerivationMechanism rdf:ID=KeyDerivationClassRequirement">
               <security:SymmetricAlgorithm rdf:resource="SymmetricDerClass"/>
            </security: KeyDerivationMechanism>
         </security:KeyManagementProtocol>
         <security:AuthenticationProtocol rdf:ID="AuthenticationRequirement">
            <security:DoubleMechanism rdf:ID="DoubleClassRequirement">
               <security:AuthenticatorAlgorithm rdf:resource="AuthenticatorClass"/>
            </security: DoubleMechanism>
         </security: AuthenticationProtocol>
         <security:TransportProtocol rdf:ID="TransportRequirement">
            <security:TLSMechanism rdf:ID="TLSClassRequirement"/>
         </security: TransportProtocol>
      </wsp:all>
   </wsp:ExactlyOne>
</wsp:Policy>
```

In this example, cloud customer requests a permanent role class (*e.g.* Manager, Employee, *etc.*) and geo class (*e.g.* IP address from US) as access management protocol, AES-256 [20] for encryption associated with manual re-encryption as cryptography protocol, CBC hash function as signature protocol, symmetric key derivation and key wrapping as key management protocol, second-password authenticator as authentication class, and TLS as transport class.

B. Policy Management System (PMS)

The second phase of the suggested model is to manage established security rings and to assure applied security mechanisms based on defined security rings. Policy Match Gate is a defined component to schedule applying policies 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 donated 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 Files (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.
- SLC Index (SI): SLC index is a defined index ($1 \leq SI \leq 4$) that prioritize the scheduling process according to the chosen security mechanisms in SLC (the value of 1 is the highest 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 all security mechanism in SLC. Table 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 2 shows the example of calculating the final value of SI).

TABLE I. 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	

TABLE II. AN EXAMPLE FOR CALCULATING THE FINAL VALUE OF SI

	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 ($\delta \in \{1,2, \dots N\}$ and N represents total number of dedicated security levels)		

The main objective of PMS is how to schedule item of 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) 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.

Algorithm 2
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 $Sort\ (F1, SI, QD)$;
// 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 is the second in the case of same SI values.

2 $D(F_1)$
$\{for\ (int\ x = 1\ ;\ x \leq M\ ;\ ++x)\ \{$
$for\ (int\ y = 1\ ;\ y \leq I\ ;\ ++y)\ \{$
$if\ \left((CVM_y \neq OVM_y)\ \&\&((OVM_y - CVM_y) > W(f_x))\right)\ Ass(f_x, VM_y);$
$f_x.ass = true$
$break;\}$
$if\ (f_x.ass = false)\ re\_add\ (f_x, F_1);\}\}$
// The process of scheduling is 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. (M represent the number of files in the high priority queue)

3 $Sort\ (F_2, SI, QD)$;
// The second priority queue ($Pf_x = 2$) is sorted based on the value of SI and QoS-Driven parameters same as F_1.

4 $D(F_2)$
// The second priority set is distributed in VMs based on the optimal and current utilization of each virtual machine. However, if a task is added to the first priority set the un-assigned tasks in F_2 will be assigned after processing F_1 again.

5 $Sort\ (F_3, SI, QD)$;
// The low priority queue ($Pf_x = 3$) is sorted based on the value of SI and QoS-Driven parameters same as F_1.

6 $D(F_3)$
// The low priority set is distributed in VMs based on the optimal and current utilization of each virtual machine. However, if a task is added to the first and second priority sets the un-assigned tasks in F_3 will be assigned after processing F_1 and F_2 again.

TABLE III. COMPARISON BETWEEN PROPOSED SCHEMA (PME) AND OTHER POLICY MANAGEMENT MODELS

	IETF [6]	Ponder [11]	KaoS [12]	Rei [15]	Di Modica [17]	PME
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
SLC	No	No	No	No	No	Yes
PMS	No	No	No	No	No	Yes
Policy Match Gate	No	No	No	No	No	Yes
Validity Check	Yes	Yes	No	Yes	Yes	Yes

Accordingly, PPU tries to classify the queue to 3 main priority categories:

$$
\begin{aligned}
&for\ (int\ x\ =\ 1\ ;\ x\ <=\ J\ ;\ ++x)\ \{\\
&\quad \alpha = SLC\ (f_x);\\
&\quad switch\ (\alpha)\{\\
&\qquad case\ (1 \le \alpha < 2): Pf_x = 1;\\
&\qquad case\ (2 \le \alpha < 3): Pf_x = 2;\\
&\qquad case\ (3 \le \alpha \le 4): Pf_x = 3;\ \}
\end{aligned}
$$

The scheduling process in each category is based on QoS-Driven in cloud-based environments [21] 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 2).

The second component in PMS is policy checkpoint 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 etc. After this validation process, a confirmation is sent to cloud customer about the successful policy application to data.

IV. EVALUATION AND CONSLUSION

Table 3. shows the advantages of the proposed schema in comparison with similar policy management models. The most important characteristic of this model is syntactic and semantic analysis of requested policies by validity engine to provide reliable mapping between security mechanism and requested policies according to the capabilities of cloud provider and requirements of the cloud customers. Furthermore, two other components were introduced to ensure about the policy application process for all stored data based on defied policies in each SLCs.

ACKNOWLEDGMENT

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

REFERENCES

[1] 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.

[2] F. Fatemi Moghaddam, M. Ahmadi, S. Sarvari, M. Eslami, and A. Golkar, "Cloud Computing Challenges and Opportunities: A Survey," in *Proc. of 1st International Conference on Telematics and Future Generation Networks (IEEE TAFGEN),* 2015, pp. 34–38.

[3] H. Takabi, J.B. Joshi, and G.J. Ahn, "Security and Privacy Challenges in Cloud Computing Environments," *IEEE Security & Privacy*, vol. 8, no. 6, pp. 24–31, 2010.

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

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

[6] Y.Snirand, Y.Rambergand, J.Strassnerand, R.Cohenand, and B. Moore, "Policy Quality of Service (QoS) Information Model," *Technical report, IETF*, 2003.

[7] B. Moore, E. Ellesson, J. Strassner, and A. Westerinen, "Policy Core Information Model," *Version I Specification*, 2001.

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

[9] IETF. Specification for the Representation of CIM in XML, Version 2.2. *Technical Report, IETF*, 2007.

[10] J. Strassner, "Mapping the Policy Core Information Model to a Directory," *Technical Report, OASIS*, 2001.

[11] N. Damianou, "A Policy Framework for Management of Distributed Systems," *Imperial College*, 2002.

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

[13] A. Uszok, J. Bradshaw, R. Jeffers, M. Johnson, A. Tate, J. Dalton, and S. Aitken, "KAoS Policies for Web Services," *W3C*, 2003.

[14] A. Uszok, J. Bradshaw, R. Jeffers, N. Suri, P. Hayes, M. Breedy, L. Bunch, M. Johnson, S. Kulkarni, and J. Lott, "KAoS Policy and Domain Services: Toward a Description Logic Approach to Policy Representation, De-confliction, and Enforcement," *Policy*, 2003.

[15] L. Kagal, "Rei: A Policy Language for the Me-Centric Project," *Technical Report, HP Labs*, 2002.

[16] Y. Zou, T. Finin, and H. Chen, "F-OWL: An Inference Engine for the Semantic Web. In Formal Approaches to Agent-Based Systems," *Lecture Notes in Computer Science. Springer*, vol. 3228, pp. 238-248, 2004.

[17] G. Di Modica and O. Tomarchio, "Matchmaking Semantic Security Policies in Heterogeneous Clouds," *Future Generation Computer Systems*, vol. 55, pp. 176–185, March 2015.

[18] W3C, Web Services Policy 1.5 - Framework, W3C Recommendation, September 2007. Available at: http://www.w3.org/TR/ws-policy/.

[19] F. Fatemi Moghaddam, R. Yahyapour and P. Wieder, "Policy Engine as a Service (PEaaS): An Approach to a Reliable Policy Management Framework in Cloud Computing Environments," (In Press).

[20] J. Daemen, and V. Rijmen, "AES Proposal: Rijndael". *National Institute of Standards and Technology*, pp. 1-10. April 2001.

[21] X. Wu, M. Deng, R. Zhang and S. Shengyuan, "A Task Scheduling Algorithm Based on QoS-Driven in Cloud Computing," *Procedia Computer Science*, Vol. 17, pp. 1162–1171.