Chapter 5

Forward and Backward Security Assurance in Cloud Data Access Provision

The chapter describes a conceptual design that ensures the security of cloud data storage. The design supports efficient addition and revocation of users and ensures that only authorised users could access and modify the outsourced data. These are some of the core properties described as thesis goals in chapter 1. In order to achieve this, some of the concepts and techniques described in chapter 2 of the thesis were used. The conceptual design presented in this chapter provides the core concepts to be applied for cloud based transaction processing systems. The design is validated in section through a proof of concept prototype in section 5.6.

5.1 INTRODUCTION

Cloud storage is utilised in a dynamic collaboration environment, which creates the need for improved access control techniques that anticipates outsourced data from security risk exposures. Users of cloud based enterprises were susceptible to perform numerous abstract operations on low level data objects. This furnished the need for various user trust levels to access multiple cloud based services. Managing authorisations for account spanning services requires an access control model that preserves the security of traditional access control techniques. This includes the provision of different level of abstraction, which allows the administrators to implement high level access policies, which are independent of
low level infrastructures that constitute the cloud services. Though Chapter 4 of the thesis presents a technique to implement different level of abstractions across user access policies, it has become mandatory to ensure the property of forward and reverse security over outsourced cloud data. The chapter defines a Secured Access Provision with Effective Role Management (SAPERM) scheme, which preserves the property of forward and reverse security through effective management of user attribute revocation and delegation activities. The proposed scheme in turn provides an efficient algorithmic step for cloud based enterprises framework establishment. The assumptions made for this chapter are described in section 5.2 and related works in section 5.3. Section 5.4 describes the conceptual design for a dynamic collaboration environment that solves the problem of forward and reverse security. Section 5.5 provides the case study related to SAPERM and access control verification proofs in section 5.6. Experimental implementations and results and discussions were provided in section 5.7 and 5.8. Finally, the conceptual design is discussed and compared to related work and thesis goals in section 5.9 and a summary of SAPERM is given in section 5.10.

5.2 ASSUMPTIONS

When using cloud storage, the Central Administrators (CAs) and System Administrators (SAs) of the untrusted cloud data store were assumed to be honest and curious. This means that the CAs and SAs is curious to learn about user access behaviours and has full rights to manage all the user related activities (role revocation, delegation, etc.), but will honestly
follow any protocol provided by the organisation custom rules. Thus, the CAs and SAs will not actively manipulate any user related data or communications. The same is assumed for any user accessing the untrusted cloud. Since the proposed scheme is applied to larger enterprises it is also assumed that users under several roles work on a session basis to accomplish their tasks in an effective manner. Furthermore, it is assumed that the SAs may attempt to collude with an arbitrary CA to gain knowledge about the unauthorised user access and security threats.

5.3 RELATED WORKS

This section describes and analyses how others have approached the challenge of dynamic collaboration using cloud services. Multiple solutions have been examined, after which an overview of the related works were provided with a comparison to solution properties over thesis goals.

5.3.1 OVERVIEW

In work done by (Zhou et al., 2013), integration of cryptographic techniques with RBAC techniques was made and it uses role keys for data decryption. Further this work presents a hybrid cloud architecture, where the public cloud contains the basic level details and most sensitive information over private cloud. This work separates the property of user delegation to active and passive types, and establishes effective role management through the use of delegation servers and protocols. Most of the larger enterprises make use of the RBAC models for convenient user access provisions. At the current scenario several RBAC models make use
of single system administrator to carry out all the system related activities. RBAC scheme makes an assumption that a single trusted administrator manages all the system related entities. (Zhou et al., 2012) stresses the point that it is highly impractical for the single SA to manage all the system related activities. The author presents a trusted administrative model Adc-RBAC to manage and implement role based access policies for C-RBAC in large scale cloud based systems. This work make use of role-based encryption techniques to assure that only authorised administrators corresponding to a particular role could add or revoke the users to or from a role. The data owner authenticates the trust level of the role before they provide their valid data to the system administrators. The use of cryptographic and trusted access control enforcement techniques guarantees the property of data security at untrusted cloud servers.

An administrative model for large scale RBAC systems with respect to important system security features were presented in (Li and Mao, 2007). This work identifies six design requirements for administrative models for RBAC. With a comparative analysis to existing RBAC models like ARBAC and SARBAC, this work presents an UARBAC model for cloud based systems that has comparative advantages over the existing RBAC techniques. In RBAC models, generally two mappings were made, one from the access policies to roles and the other from the roles to the users. (Takeo Ueda and Ruggiero, 2012), provides a systematic mapping between Roles and Permission and describes the relationship between them. In addition to this an overview about the cryptographic processes associated with role based access control models were provided in this work. In Policy Analysis for Administrative Role Based Access Con-
trol (Sasturkar et al., 2011) provides the solution for key problems like reachability, availability, containment and information flow issues associated with ARBAC models. In (Wen et al., 2009), a three layered access control framework applicable in large scale financial web based systems is given. Web pages, operations and data were the three layers described in this framework. Layers in this framework are associated with the fine-grained and coarse grained access control and other system operations in large scale web based systems. In patterns for session based access control (Fernandez and Pernul, 2006), a clear description about the need for session management in access control models were given. It provides the importance and application of session concept to both the Role Based and Attribute Based Access Control models. This work accomplishes the process of session establishment depending upon users job functions. In a precise manner this work presents a session based RBAC model. In Generalized Temporal Role-Based Access control model (Joshi et al., 2005), a wide range of temporal constraints for RBAC models is discussed. The model allows periodic as well as duration constraints over user roles, user-role assignments and role-permission assignments. This model also imposes cardinality constraints and maximum duration activity constraints over user roles. The generalized temporal Role-Based model applies the property of syntactic structure, which is extended from traditional role based access control models.

5.3.2 COMPARISON OF RELATED WORKS

This section provides a brief summary about two major works that is related to SAPERM scheme described in this chapter and it is depicted
WORK DONE BY ZHOU, LAN, ET AL

A Role Based Encryption (RBE) scheme, which integrates cryptographic techniques with RBAC model were given by (Zhou et al., 2013). RBE scheme presented in this work allows the enforcement of RBAC policies over the encrypted data stored in public cloud environment. This work further presents an RBE based hybrid cloud architecture that insists cloud based organisations to store their data independently over public cloud environment. It allows the users to perform the decryption process by using only the single key associated with the particular user of the system. The efficiency of the system varies accordingly, depending upon the complexity of underlying role hierarchy and user membership associated with the system. The concept of broadcast encryption mechanism given (Delerablée, 2007) was used to manage the problems related to user attribute revocation process in RBE scheme given by this work.

In RBE scheme, the data owners encrypts the data in such a way that only users satisfying the roles specified in the access policy could be able to decrypt and view the data. The role grants permission to the users, who satisfies the role and it can also revoke the permission from the existing users of the role. Even the cloud service provider could not be able to view the data content, if he were not provided with appropriate role associated with the system framework. RBE scheme handles role hierarchies through the inheritance of permissions from other roles of the system. A user is added to the role only after the completion of encryption process done by the data owner. The process of user revocation is performed at any time, so that a revoked user cannot process future access to the
cloud data. This scheme uses only public parameters to manage user-role memberships as well as to perform decryption processes. This work outsources part of the decryption process to cloud. This reduces the complexity of the client side computation process.

RBE scheme provides secured cloud storage through the construction of RBE based hybrid cloud environment. The hybrid cloud architecture given by this work consists of public and private cloud infrastructures. The public cloud contains the actual data in an encrypted form and the private cloud stores sensitive structure information such as role hierarchies and user member information. The process of data sharing and data access could only be done through the public cloud infrastructure, which prevents the leakage of sensitive information. The major contributions of this paper are listed as follows:

1. A Role Based Encryption (RBE) with user revocation facilities, which combines RBAC concepts to cryptographic techniques are given.

2. RBE based hybrid cloud architecture is provided in this work.

3. A practical implementation of RBE scheme over hybrid cloud architecture is given.

4. Analysis of the results provided by RBE in terms of encryption and decryption process operation time is given.

The architecture of RBE scheme consists of six major entities public cloud, private cloud, user, data owner, role manager and administrator. Public cloud is the third party cloud service provider that resides outside
the organisation infrastructure. Private cloud resides inside the organisation and built upon the internal data centres hosted over the organisation, stores sensitive information associated with the system. Role manager is a system entity that manages the relationship between the users and their associated roles. Role manager establishes their intended activities through private cloud infrastructure without any contact with the users of the system. The administrator acts as a certificate authority, generates and stores system parameters over private cloud and issues all the valid credentials to the users of the system. The administrator adds the users to the role through the computation of appropriate role parameters over the private cloud environment. The RBE scheme performs its intended objectives through the execution of steps such as setup, extract, manage role, add user, revoke user, encrypt and decrypt process associated with it. In this manner RBE scheme provides access provision to the users of the system. The RBE scheme described in this section stresses the point that the role acts as a decision making factor, that allows or denies the access to the users of the system and plays a major role in solving attribute revocation problems. But it does not provide any description about the mappings between the users to roles and from roles to access privileges on data objects. This reduces the accuracy of the underlying user access policies. This work further provides an RBE based hybrid cloud architecture, where most of the explanation is provided to the functionalities of the hybrid cloud architecture but it does not give any information regarding standards and protocols used to construct the hybrid cloud architecture. This work even does not provide any information accompanying the process of data migration between the public and private cloud in-
frastructures. In RBE scheme, a single role manager is responsible for managing all the role related activities. But this is highly impractical for cloud based larger organisations. The administrator solely relies on private cloud to generate certificates and manages user revocation problems. This assumption is highly vulnerable in case of malicious cloud server operations and security breaches.

**WORK DONE BY WEN, ZHICHAO, ET AL**

(Wen et al., 2009), presents a three layered role based access control framework for large financial web systems. This work explains the importance of sensitive data associated with web based finance systems and stresses the need for a good access control framework over the security protection mechanisms. It also states that the traditional role-based access control slows down the system performance, even though it provides user access depending upon their roles and it hardly meets the requirement of the larger financial systems. The three layered framework (TL-RBAC) implements access control in three layers such as web pages, operations and data. The use of coarse-grained access control over web pages filters anonymous attacks such as web scan and DOS attacks. Further the deployment of fine-grained access control over operation and data layer prevents the user operations and data access beyond the privileges of the users.

TL-RBAC extends the traditional RBAC models their designs and implementations to provide security over larger web based systems in a layered manner. Coarse-grained access control layer filters the illegal operations and hacker attacks, thereby accelerating the process of large scale frequent access control. The use of fine-grained access control binds user
to the data within their privileges. This states that the user would not be able to do any operations beyond their privilege. The implementation of TL-RBAC takes both security and performance in consideration. The first layer performs access control provision through the restriction of system URLs, which is beyond the limit of the particular user access privilege. The second layer is the code level access control and it provides fine-grained access control through the use of the support provided by the coarse grained access control features given by the first level layer. The users were permitted to perform operations, only when the operations associate to the user and the role satisfies the operation. The third layer is finer than the previous two layers and provides data level access control, which binds the data with certain user. This states that even though the user has access rights to perform certain operations over a data, the TL-RBAC scheme checks the validity of the access rights before the process of data access provision.

The implementation of the first layer of the TL-RBAC system, takes the responsibility of centralised access control management. The portal system present at this layer uses Single Sign On (SSO) mechanism, where the successful login of the authorised user at a single portal were recognised by all the other portals associated with it. This eliminates the need for login process in consequent subsystems. The users were assigned with the access privileges once they login into the system. The implementation of second layer invokes the check operation and verifies whether the user has the privilege to perform the requested operation. If the user has right to perform the operation, the check operation function returns the true value else the false statement were provided to the
users of the system and it rejects user access over the particular data. The complexity factors associated with the financial systems creates the need for the implementation of the third layer over the RBAC systems. This layer bind the data with roles, through the invocation of stored procedures present at the database level and checks whether the data belongs to the particular user or not. The performance testing report of the system depicts that TL-RBAC provides better system performances in terms of throughput and time operations. In this manner the process of user access provision is made in TL-RBAC scheme.

Even though the TL-RBAC scheme provides better user access provision with improved system performance, it does not provide any description to performance measure of the system. This reduces the possibility of its application to larger transaction processing systems. This is due to the fact that larger organisations consist of n number of interconnected networks, where the requirement of system performance plays a major role. The use of Single Sign On mechanism over TL-RBAC scheme reduces the level of security features associated with the system and it may lead to network attacks like session hijacking and man in the middle attacks. Though the implementation of coarse grained layer prevents the DOS attacks the use of SSN over second layer causes man in the middle attacks. In a more precise manner, the TL-RBAC scheme does not provide proper implementation details. This makes its application highly difficult over larger organisations.
The TL-RBAC scheme given in this work can provide better solution to RBAC models, but it fails to better suit the need for larger organisations, where lots of security and access protection features are required.

Table 5.1: Comparisons of Works Relating to SAPERM Scheme

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Security</td>
<td>Cloud and User Cloud</td>
<td>Data Confidentiality</td>
<td>+ + +</td>
<td>++</td>
</tr>
<tr>
<td></td>
<td>Cloud</td>
<td>Data Integrity</td>
<td>++</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>Cloud</td>
<td>Data Availability</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td></td>
<td>Cloud and User Cloud</td>
<td>Non-Repudiation</td>
<td>+ + +</td>
<td>++</td>
</tr>
<tr>
<td></td>
<td>Cloud</td>
<td>Accessibility</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>Cloud</td>
<td>Expressiveness</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>Access Control</td>
<td>Cloud User and Cloud</td>
<td>User Addition and Revocation</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td></td>
<td>Cloud</td>
<td>Fine-grained Access Control</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td></td>
<td>Cloud User and Cloud</td>
<td>Access Control Delegation</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>Cloud User and Cloud</td>
<td>Many-Many File Sharing</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>Performance</td>
<td>Cloud</td>
<td>Scalability</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td></td>
<td>Cloud</td>
<td>Flexibility</td>
<td>–</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>Cloud</td>
<td>Durability</td>
<td>+ +</td>
<td>–</td>
</tr>
</tbody>
</table>
5.4 FORWARD AND BACKWARD SECURITY ASSURANCE IN CLOUD ACCESS PROVISION

5.4.1 OVERVIEW

This section describes Secured Access Provision with Effective Role Management (SAPERM) scheme, which is our conceptual design of an access control model for cloud based enterprises. The access control model described in this chapter is most suitable for dynamic collaboration environment. The SAPERM is built upon existing RBAC and ABE techniques and adds support for SCFAP scheme described in chapter 4 of the thesis. The proposed SAPERM scheme in turn provides access control delegation, efficient user revocation and ability to audit the stored data. The proposed SAPERM scheme is presented in a linear manner where the motivation and objectives were described. Section 5.4.2 discusses the motivation and objectives, system design in section 5.4.3 and system entities in section 5.4.4. Section 5.4.5 and 5.4.6 provides description to hybrid cloud construction and SAPERM scheme. Session management and data archiving in SAPERM is discussed in section 5.4.7 and 5.4.8 of this chapter. The contributions of proposed SAPERM scheme consist of two parts. The first part provides an overview of hybrid cloud infrastructure, where the proposed SAPERM scheme could be applied. The second part depicts a clear explanation about the algorithmic steps associated with system framework establishment and user access provision. This work is done as an extension of work done by (Wen et al., 2009) and (Zhou et al., 2013).
5.4.2 MOTIVATION AND OBJECTIVES

The wide spread adoption of cloud services were found to be prevalent in large scale enterprises in comparison to medium and low level organisations. The unique characteristics of the cloud computing, which is discussed in chapter 2 of the thesis makes it feasible for the larger organisations to adopt cloud services. The multi-tenant nature of the cloud services initiates the impact of security threats over outsourced data. It causes several problems related to user attribute management, which includes user role management, user role revocation, delegation, etc. The existing access control scheme acts as a standard and it is not an actual solution to the above mentioned issues. This creates the requirement for effective role management and access control techniques for cloud computing environment. The key objective of the proposed SAPERM scheme is to provide a clear description about appropriate algorithmic steps associated with system framework establishment and user access provision in large scale enterprises. Through this the property of forward and backward security along with better system performance were achieved. The SAPERM scheme further consists of effective session management and data archiving algorithms that provides collision resistance session establishment and data storage facilities to cloud users.
SAPERM uses the concept of ABE and RBAC techniques to manage cloud based enterprises in an efficient way. Combining these cryptographic techniques enables the CSP to distribute user services in an efficient manner. Figure 5.1 gives an overview of the conceptual design of the system. The data owner share their valid contents through cloud services, internal and external users of the system access the shared data contents through their valid cryptographic keys. The CA merges two or more related roles from the role list to a single role set and maps each related role set to an SA. This means that all the roles that come under a particular related role set would be managed by one particular SA. The CA associated with the system allocates the session timing for every user of the system. This allows the users of the system to work on a session basis. The mapping associated with the system are described in Figure 5.2. If any of the user under a particular role were unable to complete
their work with in their session limits, they can make use of the common session time to complete their pending works. Every SA associated with system framework maintains a log history that contains all the user related activities, through which the problem of user attribute revocation is solved. CA monitors the users log history on a regular basis that enables the CA to archive rarely accessed files, thereby improving the scalability of the system. The user log used in the SAPERM scheme is constructed using the concept of User Behaviour analysis table defined in chapter 3 of the thesis.

**Figure 5.2: Mappings Involved in SAPERM scheme**

### 5.4.4 SYSTEM ENTITIES

The following are the entities associated with SAPERM scheme described in this chapter.

Roles represent the set of job functions corresponding to the organization. Example: Tester, Developer, etc. Role set contains set of related roles. The roles associated with the system were arranged in a linear manner through which the related role set is derived. Role Set defines
both user attributes and job functionalities associated with the organization. Example: The roles representing design, testing and maintenance comes under related role set. This is because all the roles mentioned above were related to a common objective of project development.

CA merges two or more related roles into a single role set and maps the role set to the System Administrator (SA). CA in turns maps the set of related users to their respective role sets and allots the session timing for every role in the role set. CA archives the rarely accessed files with the help of the details provided by the user log.

SA in collaboration with CA maps each user to their corresponding user roles. SA further takes care of activities relating to cryptographic key management and user log maintenance. The user log contains information about user activities such as user name, user roles, files accessed by the user, session timings, utilization of common session. User Set contains the set of users of the organization, belonging to a particular role set. The users are also called as data consumers or content receivers. Data Owner shares/uploads the data content into cloud storage.

5.4.5 HYBRID CLOUD CONSTRUCTION

This section presents an overview of hybrid cloud construction, where the proposed SAPERM scheme is applied and it is based upon the framework given by (Zhou et al., 2013). The hybrid cloud architecture described in this chapter consists of both the public and the private cloud, where the public cloud contains most basic level details of the users and all the other sensitive information would be kept in a private cloud environment. Users from inside and outside the organization make use of
the hybrid cloud service facilities. Users inside the organization forms the internal users and out-side users were called as the external users of the system. For an example, the software vendors provide software tools and other requirements to IT industries; in such a case they were in a position to access the organizations cloud to know about the needs of a company. These types of users fall under the category of external users. System users like personals from top, mid and low level of the organisation forms the internal users of the system. In order to have a safe and secured migration process, the data transfer between the public and the private cloud (Hybrid cloud) occurs through the Secure Socket Layer (SSL) protocol. During the time of data migration the SSL protocol is invoked and a negotiation between the two central nodes on the basis of the required security parameters takes place. The temporary session key to compute message authentication code, random key for the symmetric encryption process and the minimum privilege migration tickets are the three security parameters computed and used during the SSL protocol-based negotiation process. The message authentication code (MAC) is generated through a random function of specified range and it is sent to the mail of the users for authentication process. Each and every data migration process includes the activity of MAC generation. This is because each generated MAC could be used only one time during data migration process, thereby improving the security of the hybrid cloud data access. In order to gain access over private cloud data, users established their public cloud connection through which the secure access to private cloud data is provided. In this manner hybrid cloud environment for the application of SAPERM scheme is constructed.
5.4.6 SAPERM SCHEME FOR ENTERPRISE NETWORKS

This algorithmic step accompanying the system framework establishment and user access provision for cloud based enterprises are described in this section.

STEPS ASSOCIATED WITH SYSTEM FRAMEWORK ESTABLISHMENT

SAPERM provides effective algorithmic steps through which efficient system with better performance and security features is achieved. It consists of seven major steps. The first three steps describe the mappings between the users and system administrators, whereas the remaining steps describe the mappings associated with users and their respective roles. The SAPERM scheme and its associated algorithmic steps discussed in this chapter are designed in such a way, so that it better suits the needs of cloud based enterprises.

STEP 1

This step describes the classification of related user role sets from role sets arranged in random manner. A system consisting of n roles is taken and let R be the master role set that contains n number of user roles in some order, such that \( R = \{R_1, R_2, \ldots, R_n\} \). Now let us consider the first i units from R which is \( R_i = \{R_{i1}, R_{i2}, \ldots, R_{in}\} \). Then we take the next j units such that \( R_j = \{R_{21}, R_{22}, \ldots, R_{jn}\} \) and next k units \( R_k = \{R_{31}, R_{32}, \ldots, R_{kn}\} \). Likewise the set R is divided into the possible number of random role sets. In this system we assume that n number of roles is divided into three role sets \( R_i, R_j \) and \( R_k \), such that the random role sets \( R_i, R_j \) and \( R_k \) is modified to the related role sets based upon the relation between the job functionalities that exists between the roles and is described as
follows:

\[ R_i = \{R_{i1}, R_{i2}, R_{i3}, \ldots, R_{i(n-1)}\} \]  
(5.1)

\[ R_j = \{R_{j1}, R_{j2}, R_{j3}, \ldots, R_{j(n-1)}\} \]  
(5.2)

\[ R_k = \{R_{k1}, R_{k2}, R_{k3}, \ldots, R_{k(n-1)}\} \]  
(5.3)

After the modifications to the role sets, the notations \(R_i, R_j, R_k\) remains the same. Only the role inside them gets changed.

**STEP 2**

During this step, all the related role sets framed during step 1 were grouped into a single set as

\[ R_{set} = \{R_i, R_j, R_k\} \]  
(5.4)

The \(R_{set}\) describes the roles associated with the system in a related order.

**STEP 3**

This step assigns a System Administrator for each related role set present at \(R_{set}\). Where the \(SA_{set}\) consists of a set of SAs. The number of SAs depends on the number of related role sets present in \(R_{set}\).

\[ SA_{set} = \{SA_1, SA_2, \ldots, SA_n\} \]  
(5.5)

The system has only three SAs, since the numbers of related role sets were assumed to be three. The \(SA_{set}\) is mapped to the Role set as follows:

\[ SA_1 \rightarrow R_j \]  
(5.6)
This states that for every related role set a SA is assigned. All the activities explained above were done by Central Administrator (CA). Once the SA is assigned to a particular related role set, the SA performs the following activities.

1. Checks the access privilege of the users.
2. Checks for the user rights to perform an operation.
3. Checks whether the data accessed by the user belongs to him or not.

STEP 4

These step groups the users of the system into a related user set depending upon their common identities. Where \( U = \{U_1, U_2, U_3, \ldots, U_n\} \) be the set of users associated with the system. The user sets are further refined to the set of related users \( U_i, U_j, U_k \). Each related user set contains number of users less than the total number of users associated with the system, such that the grouping of related user set to common user set in a relative manner is described as follows:

\[
U = \{U_i, U_j, U_k\}
\]  

(5.9)

STEP 5

In this step CA maps each set of related roles to its corresponding user sets.

\[
R_i \rightarrow U_i
\]  

(5.10)
The mapping between the related role sets to the user sets were described in equations (5.10), (5.11), (5.12). The mapping $R_i \rightarrow U_i$ states that the users under the set $U_i$ were assigned with their respective roles that come under $R_i$. There exists a possibility that users of the system may contain multiple roles associated with them. In order to overcome this issue and provide effective access management the user roles were classified into related role sets. This partition further assists SAs to effectively manage all the user roles that come under the control of particular SA.

**STEP 6**

This step describes the mapping between system administrators to role sets and from the role sets to the user sets. The mappings made between these three entities are described as follows:

$$SA_{set} \rightarrow R_{set} \rightarrow U_{set}$$  \hspace{1cm} (5.13)

The SA mapped to a particular related role set is responsible for managing all the users accompanying that particular related role set.

For example $SA_1 \rightarrow R_1 \rightarrow U_i$ means that $SA_1$ is responsible for managing all the roles and users that comes under the role set $R_1$ and user set $U_i$.

**STEP 7**

This step assigns session timing to the users of the system. Let $s_t = \{st_1, st_2, st_3, ..., st_n\}$, be session timing for n number of roles in $R_{set}$ such
that every role in the $R_{set}$ is mapped with the session time $st$. In our case $R_{set} = \{R_i,R_j,R_k\}$, assigning the $st$ to all the roles in $R_j$, becomes as,

$$R_i = \{(R_1,st_1),(R_3,st_2),(R_4,st_3),..., (R_n,st_{n-1})\} \quad (5.14)$$

A user with role $R_1$ works at the session time $st_1$. If suppose the related role set $R_1$ consists of three roles $R_1,R_3,R_4$ it means that the users under $R_1$ establishes their session at $st_1$ and the users under $R_3$ at $st_2$ and so on. Similarly session timing $st$ is assigned to all the users of the system. All the activities done by the users are saved at user log structure, where its structure is defined in table 5.2.

### Table 5.2: User Log Structure

<table>
<thead>
<tr>
<th>User name</th>
<th>Role name</th>
<th>Active Roles</th>
<th>Session Time</th>
<th>Delegate</th>
<th>Common Session Usage</th>
<th>Files Accessed</th>
</tr>
</thead>
<tbody>
<tr>
<td>User1</td>
<td>$R_1,R_2$</td>
<td>$R_1,R_2$</td>
<td>$st_1$</td>
<td>No</td>
<td>NO</td>
<td>$f_1,f_3$</td>
</tr>
<tr>
<td>User2</td>
<td>$R_3$</td>
<td>Inactive</td>
<td>$st_2$</td>
<td>User4</td>
<td>Yes</td>
<td>$f_1,f_2$</td>
</tr>
<tr>
<td>User3</td>
<td>$R_4$</td>
<td>$R_4$</td>
<td>$st_3$</td>
<td>No</td>
<td>No</td>
<td>$f_1,f_3,f_4$</td>
</tr>
</tbody>
</table>

### STEPS ASSOCIATED WITH USER ACCESS PROVISION

The algorithmic steps associated with user access provision in SAPERM scheme were constructed based upon the work done by (Zhou et al., 2013). The overall step associated with user access provision is described as follows:

1. Startup ( ) - Takes the input security parameter and returns the passkey and pk. Passkey is kept secret by SA and pk is made public.

2. Derive ( ) It takes the input $(passkey,ID)$ and returns the deckey to the users, only if the id is the identity of the user. Each and every time the user gives the $(passkey,ID)$, it adds the user name and the file name they try to access to the user log maintained at the cloud.
server. If suppose it is the identity of a role then the CA will give the roles secret key which is kept secret by the SA.

3. Add user ( ) - This takes \((P_k, s_{kr}, ulog, ID_u)\) the public key, Role secret key and the identity of the user to add a user to the role and updates the User log in the cloud.

4. Revoke user ( ) - Involves the following keys \((P_k, s_{kr}, ulog, ID_u)\) a user of the role id \(ID_u\) and the role secret key \(s_{kr}\) is revoked and then the user log is updated.

5. Inactive ( ) - Involves \((P_k, s_{kr}, ID_u, ID_r)\) makes the user of \(ID_u\) with the role \(ID_r\) inactive and updates the User log in the cloud.

6. Delegate ( ) - Involves \((p_k, ID_{u1}, ID_r, ID_{u2})\) allows the user with \(ID_{u1}\) to represent the user having \(ID_{u2}\) of the role with an identity \(ID_r\). All the functions which are mentioned above are done by the SA.

7. Encrypt ( ) Done by the data owner of message M takes \((p_k, pub_r)\) and outputs the cipher text CP.

8. Decrypt ( ) This step is done when the data user takes \((p_k, pub_r, dec_k, CP)\) as an input and returns cipher text CP to the users.

Summary about the algorithmic steps associated with user access provision is described in table 5.3.
Table 5.3: Summary of SAPERM Phases

<table>
<thead>
<tr>
<th>Phase No</th>
<th>Phase Name</th>
<th>Input</th>
<th>Output</th>
<th>Doneby</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>StartUp()</td>
<td>Security Parameter</td>
<td>Pass Key</td>
<td>SA</td>
</tr>
<tr>
<td>2</td>
<td>Derive()</td>
<td>Passkey, ID</td>
<td>Dec Key</td>
<td>SA</td>
</tr>
<tr>
<td>3</td>
<td>AddUser()</td>
<td>Public key, Role Secret Key, User Identity</td>
<td>Updates User Log</td>
<td>SA, CA</td>
</tr>
<tr>
<td>4</td>
<td>RevokeUser()</td>
<td>Public key, Role Secret Key, User Identity</td>
<td>Updates User Log</td>
<td>SA</td>
</tr>
<tr>
<td>5</td>
<td>Inactive()</td>
<td>Public key, Role Secret Key, User Identity</td>
<td>Updates User Log</td>
<td>CS, CSP</td>
</tr>
<tr>
<td>6</td>
<td>Delegate()</td>
<td>Public key, Role Secret Key, User Identities</td>
<td>Updates User Log</td>
<td>CS, CSP</td>
</tr>
<tr>
<td>7</td>
<td>Encrypt()</td>
<td>User Public key, Role Key</td>
<td>Plaintext</td>
<td>CS, CSP</td>
</tr>
<tr>
<td>8</td>
<td>Decrypt()</td>
<td>Private Key, Role Key, Dec Key</td>
<td>Ciphertext</td>
<td>CS, CSP</td>
</tr>
</tbody>
</table>
5.4.7 SESSION MANAGEMENT IN SAPERM SCHEME

It was mentioned that every user under the particular role establishes their session during their session time. Assume that the role contains large number of users and all the users under the role establish their session during their respective session. Since a large number of users establish their session at the same time resource allocation to the all the users during the same session time remains to be a challenging factor. In order to solve this scheduling task an easier effective algorithm based on Posec method was used.

First step associated with the session management process involves the creation of list containing the set of tasks associated with the users of the system. A task priority is given to every task present in the list. Before the assignment of task, priority users tasks were divided into four categories, which are described as follows:

1. Mandatory: Set of tasks that requires to be performed on regular basis was assigned with a priority M (Mandatory). Example: Users under the developer role must check their project plan regularly to work on the basis of the project plan. These kinds of activities were prioritized as mandatory tasks.

2. Difficult: Tasks that are important to be completed, but takes lots of time to be completed is given a priority D (Difficult). This kind of task is assigned with a priority D, because the completion this task requires the chaining of other tasks. Example: Writing the overall project status report for a company involves the analysis of several set of project like project A, project B, etc. So this activity can be
chained with analysis of activities relating to projA, projB, etc., so that as soon as each project is completed, its status will be updated to the overall project status report.

3. Unique: Tasks that are to be performed additionally to increase the effectiveness of the system were assigned with a priority U (Unique). Example: Duty of the resource allocator is to allocate resources to development team. But he/she can additionally verify whether the allocated resources were sufficient to carry out the task.

4. Everything else: All the other tasks which is left out were given a priority everything else (E).

Depending upon the task category each tasks was assigned with a priority of high, medium or low, which is described in table 5.4. Based upon the criticality of the priority value the process of resource allocation is made to the users of the system.

$$\forall T \exists P \text{such that} (T \rightarrow P)$$

if and only if

$$(T_p, st) \text{ is valid}$$

else

$$\exists T \rightarrow CSt$$

The process of resource allocation would be done only if the session timing and task priority were found to be a valid one. In case of expiry of session timing the process of resource allocation is postponed to the common session timing. Through the use of the session management algorithm defined by SAPERM scheme resources were allocated in an
efficient manner, which allows the users to accomplish their tasks independently without the need for common session establishment. There may be a possibility that only the tasks with low priority and high difficulty could be added to the common session for its establishment. Since each set of related roles consists of the system administrator to manage to all the user related activities the process of session management activities could be established in an effective manner with reduced complexity.

5.4.8 DATA ARCHIVAL IN SAPERM SCHEME

Since the cloud based enterprises performs large number of transactions per second that leads to the generation of vast amount of data. In order to improve the efficiency of the data storage and retrieval process the proposed SAPERM scheme provides effective algorithm for data archival and retrieval processes. The CA monitors the user logs regularly and archives the rarely accessed files. Most rarely accessed files are stored in tapes and the quite rarely accessed files in hard drive and least rarely accessed files in different hard drives. This is done to improve the scalability of the cloud data.

Let T be the set of transactions such that \( T = \{ t_1, t_2, \ldots, t_n \} \). Now lets take the first j units of the transaction T such that \( T_j = \{ t_1, t_2, \ldots, t_j \} \) and \( F = \{ f_1, f_2, \ldots, f_n \} \) be the files accessed by the transaction \( T_j \).

\[
\forall \ t \text{ in the set } T_j \ (t \rightarrow f) \in (t \cap f)
\]

States that for all the transaction in the set \( T_j \) is mapped to the files involved in the transaction. This forms the data set.

Next we count the number of times \( a \in t \) to the transaction. Find n value
for all the files and calculate the sum of n value = \( f_1 + f_2 + f_3 + \ldots + f_n \).

\[
fixleastvalue = \frac{\text{sum of } f_n}{\text{total no of files}} + \frac{1}{2}
\]  

(5.15)

\( \forall \ f \exists \ a \ n \ value \ if \ (f_n \ value < \text{leastvalue}) \Rightarrow \text{remove } f \text{ from F and assign of an identity} \) then

archive f \( \rightarrow \) floppy

(Remaining f will be added to the dataset). else Take the first value off from the dataset pair that f value with the remaining f values. If suppose the dataset has three values \( f_1, f_2, f_3 \) means pair \((f_1, f_2) \rightarrow f_{p1}\), pair \( f_1, f_2, f_3 \) means pair \((f_1, f_2) \rightarrow f_{p2}\), pair \( f_1, f_2, f_3 \) means pair \((f_1, f_2) \rightarrow f_{p3}\). Similarly do the pairings for all the other values in the dataset.

Now calculate the n value by counting the number of times \( f_p \in t. \text{calculate the n value for all the file pairs and find the n value. Least value is calculated using equation 5.15} \)

\[
\forall \ f_p \exists \ a \ n \ value \\
\text{if(value<leastvalue)} \Rightarrow \text{remove } f_p \text{ from F and assign an identity} \\
\text{then archive } f_p \text{ to hard drive}
\]

Remaining fps will be added to the data set. Else add a new value to the pair \( f_p \).

Lets take the first pair value from the data set and compare the first element in the pair with the first elements in the other pairs. If the first element in any pair is found to be similar combine the two pairs and form the triples.

Let \((f_1, f_2) \) and \((f_1, f_3) \in F\)
then perform 
\[(f_1, f_2) \cup (f_1, f_3) \Rightarrow (f_1, f_2, f_3)\]
This forms the file triples \(f_t\). Remove \(f_t\) from the data set and archive \(f_t\) → tape. In this manner the CA achieves the rarely accessed files based on the user log.

**SUMMARY OF SAPERM ALGORITHM**

The proposed SAPERM scheme solves the problem of attribute revocation through its algorithmic step implementations. This is because a clear mapping is made between the system administrators, users and roles associated with the users. It is highly impractical for the revoked users to access the system. The user log maintained by the SA in turns provides advanced access provision features to the cloud based enterprises. In addition to this the server log also provides a clear history about the entire user related to the system.

### 5.5 CASE STUDY

An enterprise system that adopts cloud services is taken into account and it is described in Figure 5.3. It is assumed that the enterprise network taken in this case consists of large number of users related to several departments such as top management, personal management, finance management, project management, etc. The enterprise network considered in this case consists of larger organization structure. It includes large number of user roles (job functionalities) associated with it. Each user was assigned with an access privilege that limits the user access towards the outsourced data. In this case, the personal management was consid-
ered to be a common identity and the roles that relates to the identity of personal management were assumed to be the related roles and it is formed as a role set. Users were further assigned with the access policies in accordance to the roles they correspond to be a part off. There is also a possibility that a single user can have two or more roles associated with them. Since the roles were classified into the related role sets access provision to a single user with multiple roles is easily handled through SAPERM scheme. In the existing enterprise administrative models, a single SA takes care of all the activities like role management, key management, etc. which is inappropriate for larger enterprise systems. These issues were easily solved through the SAPERM scheme as it assigns multiple system administrators to manage related role sets. The user log structure maintained by the SA keeps track of all the user performed activities. This solves the problem of user attribute revocation and delegation through which the property of efficient user access provision were given to all the users of the system. For example, project management may consist of several roles which include tester, developer, etc. There is a possibility that users under the category project management can play both the roles of the tester as well as the developer. So these two roles tester and the developer are grouped to a related role set because it falls under the category of project management. In this manner, related role set were grouped into a single set and assigned with a SA for monitoring purposes. The CA manages all the SAs and monitors the user log to audit all the system related activities. In case of the overload SA can temporarily delegate their responsibilities to the CA associated with the system. In this manner the application of SAPERM scheme reduces the
computational complexity associated with larger enterprises.

Figure 5.3: Organisation Structure

5.6 ACCESS CONTROL PROOFS AND DISCUSSIONS

Let $R$ be the roles involved in the system such that $R = \{R_1, R_2, \ldots, R_n\}$. Consider that the roles in the $R$ is divided into three set of related role $R_i = \{R_1, R_2, \ldots, R_i\}$, $R_j = \{R_1, R_2, \ldots, R_j\}$ and $R_k = \{R_1, R_2, \ldots, R_k\}$. Now the set of relate roles $R_{set} = \{R_i, R_j, R_k\}$ gives the set of related role sets and the $U_{set} = \{U_i, U_j, U_k\}$ be the users of $R_i, R_j, R_k$. The set of System Administrators $SA_{set} = \{SA_1, SA_2, SA_3\}$ be the SAs of $R_i, R_j$ and $R_k$ then the following axioms must hold true.

Axiom 1

This axiom proves the existence of an SA set for every user set such that each user set is mapped to its respective role set. The description is
given as follows:

\[ \forall R_{set} \forall U_{set}[\forall SA_{set} (SA_{set} \in R_{set} \leftrightarrow SA_{set} \in U_{set}) \rightarrow R_{set} = U_{set}] \] (5.16)

For all the roles under \( R_{set} \) and users under \( U_{set} \) the SA belongs to (manages) the Role set \( R_{set} \) and it is mapped to the User set \( U_{set} \) there by number of role sets will be equal to the role set.

**Axiom 2**

This axiom proves the feasibility of mapping between the SA set and user set. The description is given as follows:

\[ \forall R_{set} \forall U_{set} \exists CA \forall SA_{set} (SA_{set} \in CA \leftrightarrow s = R_{set} \cup SA = U_{set}) \] (5.17)

For all the roles in the \( R_{set} \) and users in the \( U_{set} \) then there exists a \( SA_{set} \) that belongs to CA such that each SA will be mapped to the \( R_{set} \). Such that the number of SA is equal to both user and role set.

**Axiom 3**

This axiom proves the correlation and mapping between three system entities such as System administrators, users and Central administrator. The following provides description to this axiom.

\[ \forall R_{set} \exists U_{set} \forall CA [CA \in U_{set} \leftrightarrow \exists SA (SA \in R_{set} \cap CA \in SA) ] \] (5.18)

For all the roles under \( R_{set} \) there exists a CA maps the \( U_{set} \) to the SA such that SA belongs to \( R_{set} \) and CA belongs to SA.

**Axiom 4**

This axiom describes the condition that a particular user role could satisfy for its existence. This is explained as follows:

\[ \exists R_{set} \forall U_{set} - (U_{set} \in R_{set}) \] (5.19)
There exists a role set for all the $U_{set}$ if and only if the $U_{Set}$ belongs to the $R_{set}$.

**Axiom 5**

A descriptive proof to user access delegation was given through this axiom. This is explained as follows:

$$\exists R_{set}[\phi \in R_{set} \cap \forall U_{set}(U_{set} \in R_{set} \rightarrow suc(U_{set} \in R_{set})]$$

(5.20)

There exists an $R_{set}$ if there is any $U_{set}$ is ready to take up the role set in case of there is no $U_{set}$ under the $R_{set}$.

**Axiom 6**

This axiom states that a user set is mapped to the role set only when the user satisfies the set of access policies defined by the particular role set and it is described as follows:

$$\forall R \exists R_{set} \forall U_{set}(U_{set} \in R - set \leftrightarrow U_{set} \in R \cap \phi(R_{set}))$$

(5.21)

For all the roles in R there exists a $R_{set}$ for all the $U_{set}$ if and only if the belongs to the set of R and the policies given by $R_{set}$.

**Axiom 7**

This axiom explains the one to many relationships between the users and their respective roles, which is depicted as follows:

$$\forall R[\forall R_{i} \in R \exists U_{i} \phi(R_{i}, U_{i}) \rightarrow \exists R_{j} \forall R_{j} \forall U_{i} \in R \exists U_{i} \in R_{j} \phi(R_{i}, R_{j})]$$

(5.22)

For each element of $R_{i}$ ($R_{i}$ represents the set of related roles) in a given set R (set of all the roles) we can find a set $U_{i}$ (set of related users) such that a relation holds for $(R_{i}, U_{i})$, then we can find a set $R_{j}$ (another set of related roles) which collects some $U_{i}$ (some users of the related user set.
$U_i$). Such that for each element of $R_i$ in $R$ there is a $U_i$ in $R_j$ such that a relation holds of $R_i$ and $U_i$.

**Axiom 8**

This axiom states that for every user in related role set $R_i$ there exists a related user set $U_i$, such that a relationship holds between the user set and the role set through which the mapping between user set and role set were made. The proofs to these mappings were described as follows:

$$\forall R[\forall R_i \in R \exists U_i \phi(R_i, U_i) \forall R_j \forall R_i \in R \exists U_i \in R_j \phi(R_i, R_j)] \tag{5.23}$$

For each $R_i$ in $R$ there is exactly one $U_i$ in such that $\phi(R_i, U_i)$ holds.

**Axiom 9**

There exists a probability that a user can belong to two or more related user sets, such that the user is mapped to two or more different roles. The proof to these mappings was discussed below.

$$\forall R \exists U_i \exists S_A[S_A \in U_i \leftrightarrow \forall U_k(U_k \in R_k) \rightarrow (U_k \in R_i)] \tag{5.24}$$

For each $R_i$ there exists a $U_i$ for each $S_A$ belongs to $U_i$ will be mapped to $U_k$ if and only if $U_k$ belongs to $R_i$.

**Axiom 10**

This axiom provides proof to the encryption process accompanying SAPERM system and it is described as follows:

$$\forall R \forall U \exists Enc \exists S_A[\forall R_i \in R \exists U_i \in U \phi(R_i, U_i, S_A) \rightarrow \exists DecK \in EncK(\forall R_i \in R \exists U_i \in DecK \phi(R_i, U_i, S_A) \forall U \in DecK \exists R \phi(R_i, U_i, S_A)] \tag{5.25}$$

For each elements in $R$ (set of all roles) and $U$ (set of all users) there exists
an EncK (encryption key) such that SA for all $R_i$ (related role set) belongs to R there exists an $U_i$ (related user set) belongs to U such that a relation holds between $(R_i, U_i, SA)$ if and only if there exists an DecK belongs to Enck for all $R_i$ belongs to R there exist a DecK such that $\pi(R_i, U_i, SA)$ holds true for all U belongs to DecK there exists a who belongs to the relation $(R_i, U_i, SA)$.

**Axiom 11**

This axiom provides description to set of properties that is mandatory for the SA to possess. The description to this proof is given as follows:

$$\forall SA(\forall CA \in SA \phi(CA) \Rightarrow \phi(SA)) \Rightarrow \forall SA \rightarrow \phi(SA) \quad (5.26)$$

For an arbitrary set SA, whenever a property $\phi$ holds for all the element of SA, which belongs to a particular CA then it propagates to SA itself, then one can conclude that the property $\phi$ for the set SA. This states that the properties of SA set were delegated to CA in case of system overload.

**Axiom 12**

A proof to the propagation of properties from roles to users was given in this axiom and it is described as follows:

$$\forall R(\forall R \in U \phi(R) \Rightarrow \phi(R)) \Rightarrow \forall R \rightarrow \phi(U) \forall R(\forall R \in U \phi(R)) \rightarrow \forall R \rightarrow \phi(U) \quad (5.27)$$

For an arbitrary set R, whenever a property $\phi$ holds for all the element of R, which belongs to the set of users U then it propagates to R itself, then one can conclude that the property $\phi$ for the set R. This states that the properties of the user sets were propagated to form the role set.
5.7 EXPERIMENTAL STUDY

The proposed SAPERM system is implemented using an open source cloud computing platform Open Stack, with Ubuntu operating system. Open Stack by default, makes use of Role Based (RBAC) to deal with access controls, where the SAPERM is deployed by taking user roles as an attribute. The system architecture of Open Stack is depicted in Figure 5.4. Nova acts as a computing Fabric controller in Open Stack Cloud. The essential activities associated with instance life cycle management were handled by Nova. This makes Nova a standard platform to manage various computing resources, networks, authorization and scalability needs of Open Stack cloud. Three nodes such as compute, controller and storage were installed with Ubuntu operating system and all three nodes acts as server nodes, which provide services. The servers at compute and controller node were configured with 128 GB RAM (16 × 8 GB DIMM), 2x Intel Xenon processor with E5-2600 V2, CPUs 2 GHz, 8cores, 16 threads and 6 × 300GB disks for RAID6 Configuration. At the storage controller node 12GB RAM (16 × 8 GB DIMM), 2x Intel Xenon processor with E5-2600 V2, CPUs 2 GHz, 8cores, 16 threads with 24 × 1 TB disks were used for storage purposes. Compute node is installed with Nova packages embedded with services. Controller node consists of packages such as Glance, Keystone and Horizon along with it services. A storage node consists of Swift or cinder packages and services. The package enables each node to perform its intended objectives. All the three nodes were connected internally to Open Stack Dashboard with the help of an internal network. Client systems, which are ready to
use cloud services were connected through external network to controller node of the Hybrid cloud architecture. The use of Open Stack dashboard enables one to manage various Open Stack services. The dashboard is accessed through http followed by IP address. The dashboard assists in the process of managing SAPERM activities such as creating key pairs, managing instances and images, attaching volume to instances, manipulate swift containers, etc. The major responsibility of compute node is to manage virtual machine instances through the use of nova packages accompanying it. The controller node plays a major role in SAPERM implementation and it controls most important system activities. At the controller node, multiple system administrators depending on user roles were implemented. Each administrator controls and manages the entire user and system related activities through controller node. The central administrator in turn monitors activities of system administrators beneath him and maintains user log structure for preserving forward and backward security assurances. System Administrators make use of Keystone package present at controller node to perform authentication and authorization mechanisms for all SAPERM users and services. It is done through multiple forms of authentication including standard username and password credentials and AWS-style logins that use public/private key pairs. The identity service enables distribution of Open Stack services and it is authenticated through endpoints. Endpoint denotes the network address where services pay attention to requests. The identity services uses user attributes like roles to group or isolate resources. The glance package determines, registers, and delivers virtual machine images. It is copied as snapshots and immediately stored as the origin for
new instance deployments. The System administrator of SAPERM provides multiple servers quickly and consistently through stored images. It stores images at /var/lib/glance/image directory of local servers file system. The image service supports multiple back end storage services such as swift and amazon S3. The Horizon package present at controller node is an extensible web based application permits SAPERM system administrators to control and provide compute, storage and network resources. In other words the Horizon package acts as a dashboard that allows system administrators to view state of cloud create users, assigning roles, and set user access limits. The Open Stack dashboard runs as an Apache HTTP server through httpd service. The storage node stores user related confidential credentials such as user names, user roles, user cryptographic keys etc. The storage node accomplishes these activities through the use of swift package accompanying it. Swift provides object storage service to cloud data users. Through the use of Open Stack storage service a fully distributed, API-accessible storage platform were constructed and it is directly integrated to SAPERM applications and it is further used for backup, archiving and data retention. It enables users to store petabytes of data through scalable object storage using clusters of standardized servers. Object storage represents a distributed storage system for static data. Objects and files were stored to storage controller by writing it in multiple disks that is spread throughout the data centers. The additions of new servers were made through scaling storage clusters and it is done by SAPERM system administrators. Users under different roles access the system services through Open Stack graphical user interfaces and the system administrator provides user access management
through three different Open Stack nodes. In case of the overload CA handles the responsibilities delegated by the SA. In this manner the proposed SAPERM scheme is implemented over Open Stack Nova.

Figure 5.4: Implementation of SAPERM Over OpenStack Nova Cloud

5.8 RESULTS AND DISCUSSIONS

From the experimental implementation, certain results were found and compared to the existing schemes. Traditional access control models such as ABE and RBAC were taken for comparison with the proposed SAPERM scheme. From the implementation results certain observations were made and the CSI latency (Computational System Instance Latency) associated with system has been found. It is depicted in Figure 5.5. The CSI latency denotes the time interval between the user simulation request and response processes and it is represented in seconds. Using open stack nova, multiple client nodes were created and n number of client request were given to the cloud server from the multiple number
of user nodes. The requests were given to the server in a concurrent manner. The time limit between the user request and cloud server response were clearly observed, through which the CSI latency associated with the system has been calculated.

The CSI latency found during the experimental observation were compared against the process threshold rate. The process threshold rate indicates the amount of workload the cloud server handles per minute. In other words the process threshold indicates the user given requests per minute. The process threshold rate was highly influenced by factors like amount of CPU utilization, memory utilization and average work load that a cloud server can handle. Here a comparison is made between the concurrent user given requests and CSI latency found during the simulation process. This comparison is made to find the value of process threshold limit up to which the SAPERM could be able to provide standard access provision in comparison to existing access control techniques. The concurrent user requests were kept in x limits and the CSI latency calculation forms the Y limits. The CSI latency calculated is represented in time in seconds, whereas the concurrent user requests were represented in numbers.

The user access requests were given in a concurrent manner starting from 1000 in an incremental manner. It has been found that after a process threshold limit value of 4000 concurrent requests the CSI latency associated with the traditional access control techniques had been increased drastically. As the rate of process workload increases to the cloud server, the computational time taken by the cloud server to process the request has increased rapidly through which the CSI latency as-
associated with system deviates in an inconsistent manner. This system inconsistency is caused due to the complexity of underlying access policies and centralized system management process. But the CSI latency associated with the SAPERM scheme has been found to be consistent throughout the observation process even after exceeding the threshold value of 5000 requests. This consistency is caused due to the simplicity of the underlying access policy. It has been found that the SAPERM takes an average latency of 0.15 to 2 seconds to compute 1000 concurrent requests. In case of RBAC and ABE schemes the average time taken to process 1000 requests were found to be 2 to 3 seconds. After processing 7000 user given concurrent request an Outflow of RBAC scheme has occurred. It is ceased and it could not be able to handle further workloads as it was derived from complex system procedures. Whereas the ABE scheme has been ceased after processing 8000 user given concurrent requests. But the SAPERM system remains consistent even after the processing of more than 8000 user given concurrent request. This is due to the reason that the SAPERM scheme handles user given requests in a distributed manner with the help of multiple system administrators. In case of overload the system administrators delegates their responsibilities to Central Administrators. Since the SAPERM scheme assigns multiple SAs to manage related set of roles the time taken for user request computation process has been found to be comparatively.
5.9 COMPARISON TO THESIS GOALS

A comparative analysis of the SAPERM scheme to the thesis goals was made and it is depicted in table 5.4. It has been found that SAPERM scheme satisfies most of the security and access control properties with better performance measures. This is due to the underlying architecture of the SAPERM scheme. A clear mapping is made between all the system entities with effective session management and data archiving algorithms. In addition to this the maintenance of user log structure in turn adds effective security features and data access to the cloud data. This enables the SAPERM scheme to achieve better security and access control properties.
Table 5.4: Comparison of SAPERM with Thesis Goals

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cloud and User</td>
<td>Data Confidentiality</td>
<td>+++</td>
<td>++</td>
<td>+++</td>
</tr>
<tr>
<td></td>
<td>Cloud</td>
<td>Data Integrity</td>
<td>+</td>
<td>−</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>Cloud and User</td>
<td>Data Availability</td>
<td>++</td>
<td>++</td>
<td>+++</td>
</tr>
<tr>
<td></td>
<td>Cloud</td>
<td>Non-Repudiation</td>
<td>+++</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>Cloud and User</td>
<td>Accessibility</td>
<td>++</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>Cloud</td>
<td>Expressiveness</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td></td>
<td>Cloud</td>
<td>Scalability</td>
<td>++</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>Cloud</td>
<td>Flexibility</td>
<td>−</td>
<td>+</td>
<td>+++</td>
</tr>
<tr>
<td></td>
<td>Cloud</td>
<td>Durability</td>
<td>++</td>
<td>−</td>
<td>+</td>
</tr>
</tbody>
</table>

**Comparison of System Assessment Properties**
5.10 SUMMARY

The SAPERM scheme described in this chapter provides effective algorithmic steps, which assists in the process of efficient system framework establishment for cloud based larger transaction processing systems. It is validated using Open Stack Nova, an open source cloud platform. The major advantage of this scheme is that it reduces the computational complexity through its algorithmic step implementations. The SAPERM scheme merges the related roles into a set and assigns the set of users and system administrators to manage the administration processes. A clear mapping is made between the users of the system. In addition to this the CA associated with the SAPERM scheme maintains a clear log history about the user activities. Through the implementation of all these steps the proposed SAPERM scheme provides an efficient framework for larger enterprises and solves the problem of user attribute revocation and delegation processes. This preserves the property of forward and reverse security associated with the system. The SAPERM scheme described in this chapter most suits the needs for hybrid cloud architecture. In future this work could be extended to multi-cloud environment.