CERIAS Tech Report 2014-6

Risk-Aware Virtual Resource Management for Access Control-Based Cloud Datacenters

by Abdulrahman Almutairi, Arif Ghafoor
Center for Education and Research
Information Assurance and Security
Purdue University, West Lafayette, IN 47907-2086

RISK-AWARE VIRTUAL RESOURCE MANAGEMENT FOR ACCESS CONTROL-BASED CLOUD DATACENTERS

Abdulrahman Almutairi, *Member, IEEE,* and Arif Ghafoor, *Fellow, IEEE*

Abstract

Multitenancy and virtualization features of cloud computing enhance resource utilization and lower the cloud provider total cost of hosting customers data for big data applications. However, the cloud computing has many security challenges that are exacerbated by virtual resource sharing. In particular, sharing of resources among potentially untrusted tenants can result in an increased risk of information leakage due to vulnerability of virtual resources causing side channel attacks or VM escape. For the big data applications, an access control policy such as RBAC can be used to control the data sharing among cloud customers. However, an unintelligent cloud resources management mechanism can significantly increase the risk of data leakage among roles. The goal of this paper is to develop efficient risk-aware virtual resource assignment mechanisms for Cloud's multitenant environment. The objective is to minimize of risk of information leakage due to Cloud virtual resources vulnerability. Such an assignment problem is shown to be NP-Complete. We present several scheduling heuristics including a scalable solution and compare their relative performance.

Index Terms

Cloud Computing, RBAC, Risk Assessment, Vulnerability

1 Backgraound: Vulnerability and Data Leakage in Multitenat Could Datacenter

The cloud computing Platform-as-a-Service (PaaS) paradigm provides the application developers a platform to deploy big data applications in the Cloud. Some of these applications can be found in the areas of healthcare, e-government, science, and businesses [1]. The PaaS cloud providers can host the customer datastore on-premises and outsource the computation using virtual resources from multiple IaaS cloud providers as shown in Figure 1(b). The virtual resources can be hosted by multitenant public cloud providers such as Amazon EC2. The sheer size of big data poses serious security challenges for these applications. The backend datastore can use an access control mechanism to isolate and enforce a controlled sharing of data [2]. However, when the data is brought from backend datastore to application logic, it can be leaked through the virtual resource vulnerabilities. In a multitenat environment, untrusted tenants can exploit these vulnerabilities which can lead to an increased risk of data leakage. In this research we focus on the virtual resources vulnerabilities that can cause data leakage such as side channel attacks and virtual machine escape [3] [4]. These problems draw the attention to design secure virtual resources isolation e.g. Trusted Virtual Domain [5], secure hypervisor [6], and Chinese Wall policies [7]. However, the isolation could be improved at the cost of lowering the resource utilization. We propose an intelligent virtual resource allocation techniques that assign resources to cloud customer data centric applications. These techniques have low complexity and try to minimize the imposed risk. We assume a role-based access control (RBAC) [8] mechanism is used for the datacenter protection. However, the approach is generic and can be applied to any security policy including the discretionary access control or mandatory access control.

In [9] we have proposed a distributed access control architecture for Cloud virtual resources. A key component of this architecture is the Virtual Resource Manager (VRM) which allocates virtual resources to cloud customers based on an access control policy that is enforced by Access Control Module (ACM) as shown

• *A. Almutairi and A. Ghafoor are with the Department of Electrical and Computer Engineering, Purdue University, West Lafayette, IN,*

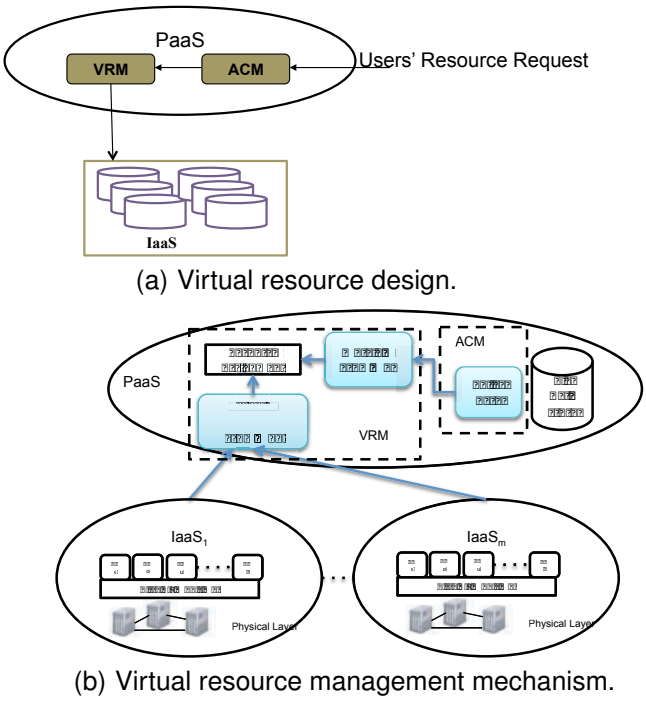


Fig. 1. The virtual resource management architecture

in Figure 1(a). In general, these resources are allocated to satisfy some SLA requirements for each cloud customer and to minimize the cost of provisioning for the PaaS cloud providers. As notice in Figure 1(b), VRM consists of workload estimation, resource vulnerability estimation, and resource assignment components. The workload estimation component estimates the sharing of data among different roles of the RBAC policy. The resource vulnerability estimation component of VRM uses security analysis tools [10] to estimate the virtual resource vulnerability. Subsequently, this component can be used to characterize the virtual resource to different vulnerability security measurement named as. High, Medium and Low. The resource assignment component uses the workload and vulnerability estimations to assign the virtual resources to cloud customers applications with a goal to minimize the total risk of data leakage. In following sections we describe this goal and provide a detailed design of all these components.

The risk of data leakage depends on the access control policy and virtual resources vulnerability. The risk is defined in ISO 27005 as "*The potential that a given threat will exploit vulnerability of an asset or group of assets and thereby cause harm to the organization*". Based on this definition we can formulate the risk due to data leakage for an application in a datacenter as

$$Risk = Assets \quad Vulnerability \quad Threat \tag{1}$$

In this paper we assume the assets of a role (cloud customer) is the number of the data objects that it has access to and which are stored in PaaS . The vulnerability is the probability of data leakage as a results of virtual resources vulnerability. To capture the worst case scenario for the risk assessment, we assume that the threat is equal to one for all roles (cloud customers). In other words, due to resource vulnerability, each cloud customer poses a threat in term of accessing the data objects of other customers and vice versa.

The summery of research presented in this paper is as follows. First, we propose a workload approximation model based on a given RBAC policy and characterization of cloud datacenter. Second, based on this characterization, we present a risk-aware assignment problem. Finally, we present assignment heuristics for virtual resources allocation and compare their performance.

2 Role-Based Access Control Policy Model for ACM

An RBAC access control policy for a datacenter defines permissions to access data objects for its roles [8]. Such assignment is formally defined as follow.

Definition 2.1: Given an RBAC policy $\mathcal{P}$ for a big datacenter with R is the set of roles and with O being the set of data objects, the permission-to-role assignment PA can be represented as a directed bipartite graph

$G(V, E)$, where $V = R \cup O$ s.t. $R \cap O = \phi$. The edges $e_{ij} \in E$ in G represents the existence of role-to-permission assignment $(r_i \times o_j) \in PA$ in the RBAC policy $\mathcal{P}$, where $r_i \in R$ and $o_j \in O$.

The out-degree of a role vertex represents the cardinality of the role and the in-degree of a data object vertex represents the degree of sharing of that object among roles. In Figure 2(a), an RBAC policy with $|R| = 4$ and $|O| = 20$ is represented as bipartite graph model. We can notice that the cardinality of role r_1 is $out\text{-}degree(r_1) = 11$. Also, the degree of sharing of data object o_{20} is $in\text{-}degree(o_{20}) = 4$.

The resource assignment component of VRM requires the cardinality of shared data objects among roles, as discussed later in Section 6. For big datacenters, computation of these cardinalities from the bipartite graph is a daunting task. We propose an alternative representation of RBAC by accumulating all data objects that are accessed by the same roles into a set of common partitions. The cardinalities of these partitions represent a *spectral representation model* of RBAC policy, called $\mathcal{W}$. This model has two advantages over the bipartite graph model. First, it can be used to characterize an RBAC policy in terms of its *datacenter sensitivity* using a single parameter. Based on the degree of sharing among roles, the datacenter sensitivity can be high, medium, or low as elaborated later in Section 4. In addition, the *spectral model* allows resource assignment based on a given percentage of datacenter. Varying this percentage can lead to variable complexity of the assignment algorithm. The *spectral model* of RBAC is formally defined as follow.

Definition 2.2: (RBAC spectral model). Given a bipartite graph representation of RBAC policy $G(V, E)$, let $P(R)$ be the power set of R excluding the null set ϕ. The spectral representation of RBAC is the vector $\mathcal{W}$, indexed by $P(R)$ and lexicographically ordered. Formally, let $p \in P(R)$, then $w_p \in \mathcal{W}$ is defined as follows:
$w_p = \mid \{\forall r_j \in p, and \quad \forall t_k \in T, \quad \exists e_{jk} \in E(G)\} \mid$

We map bipartite graph to $\mathcal{W}$ by assigning each data object to a single partition representing all the roles accessing this data object and not a subset of these roles. It can be noticed that the resulting partitions are independent in the sense that there are no common objects among them. The cardinalities of these partitions are represented as $w_p \in \mathcal{W}$. In essence, $\mathcal{W}$ can be viewed as a lattice (i.e. binary n-cube) with n levels, where n is the number of roles and each $w_p \in \mathcal{W}$ is a vertex in the lattice. For example nodes at level one of the lattice represent the size of partitions corresponding to unshared data objects belonging to individual roles. The level two nodes represent the cardinalities of data partitions that are accessed by two roles in R. Similarly, the nodes at level n contains data objects shared by all the roles. The nodes of $\mathcal{W}$ can be indexed using some role IDs, as shown in Figure 2(b). It can be observed that indices of W vector are subsets of $P(R)$ and its elements are the cardinalities of partitions that can be accessed by all the roles in these subsets. Note, $\sum_{\forall w_p \in \mathcal{W}} w_p$ equals to the total size of datacenter. In following we present an example of mapping bipartite graph to spectral model.

Example 2.1: In Figure 2(a) , an access control policy with $|R| = 4$ and $|O| = 20$ is shown as a bipartite graph. The spectra model is shown as lattice in Figure 2(b). Notice that $w_{\{1,4\}} = |\{o_9, o_{10}\}| = 2$ because o_9 and o_{10} are accessed by both r_1 and r_4.

3 Datacenter Workload Estimation for RBAC Policy

As $\mathcal{W}$ is used for our RBAC policy, we need to specify the cardinality of each of its elements. For a big datacenter specifying the exact value of these cardinalities is a challenging problem. One practical approach is to use the cardinality estimation techniques [11]. For example, in the transactional workload model, we can use the selectivity estimation of query processing a big datacenter to get the size estimation of a given query [11] whereby query (or a collection of queries) can represent a role. In case role mining is used to design an RBAC policy, the role mining techniques such as multi-assignment clustering [12] can provide an estimate of $\mathcal{W}$. In this paper we assume the access of data objects in a datacenter can follow a Zipfian distribution, an assumption which is supported by in Yahoo's cloud benchmark [13]. According to this distribution, some objects are shared by a large number of roles (queries) while most of the objects are shared among a fewer number of roles (queries). The Zipfian distribution is given as follows:

$$f(\alpha; s, N) = \frac{\alpha^{-s}}{\sum_{i=1}^{N} i^{-s}} \tag{2}$$

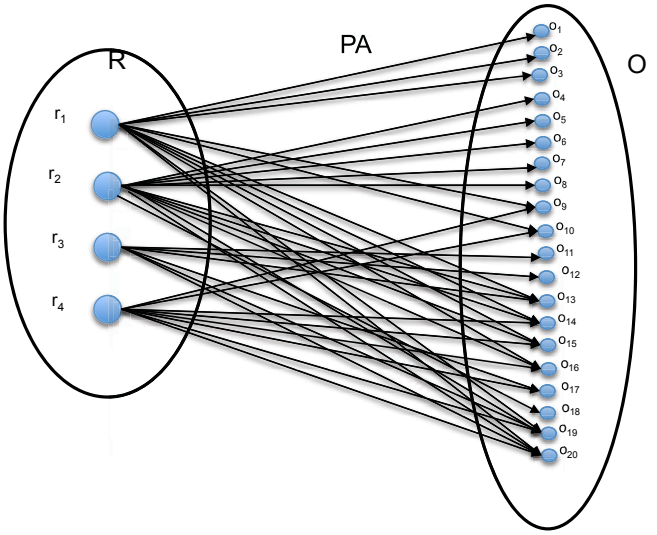
Where:

N : the maximum rank

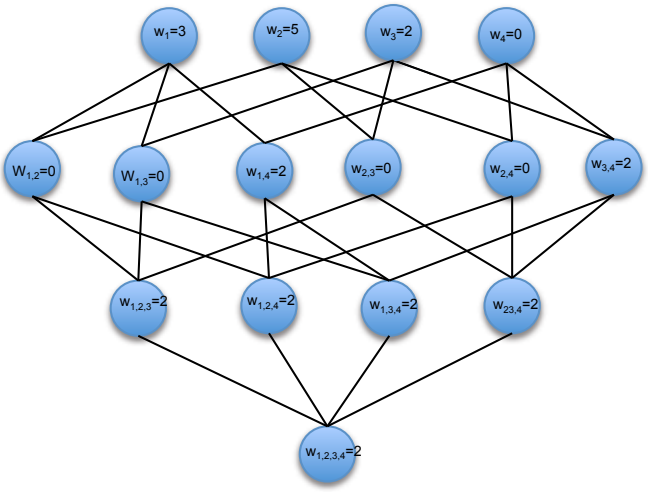
α : the selected rank

s : scalar parameter to control the shape of distribution

According to this distribution if scalar parameter $s = 1$, then the probability that data object is assigned to a single role (belongs to rank $\alpha = 1$) doubles the probability that data object is assigned to two roles (belongs



(a) RBAC permission assignment.



(b) Spectral lattice representation.

Fig. 2. Example of RBAC representation.

to rank $= 2$). As the scalar parameter s increases the value of Zipfian density function for the rank one of the distribution becomes higher. Accordingly, the number of data objects assigned only to a individual role becomes larger as shown in Figure 3. Note, the value of s should be greater than or equal to 1.

For the evaluation of the proposed heuristics in Section 7 , we propose a two-step algorithm (Algorithm 1) for generating RBAC-based workload using Zipfian distribution. In the first step, data objects are classified into n buckets where each bucket represents the number of total data objects assigned to a lattice level of Figure 2(b). For example, data objects in bucket one are exclusively accessed by only one role. On the other hand, data objects in bucket l n are shared by all the roles. The number of data objects in each bucket follows Zipfian distribution. In the second step, we assign data objects of bucket i to randomly selected partitions at level i of the lattice. Note, the number of partitions at level i is $\binom{n}{i}$. The following of the Zipfian distribution in the assignment of data objects to buckets and subsequently to partitions ensures that the two-steps algorithm generates a heterogenenous workload.

4 Characterizing Sensitivity of Datacenter Using Spectral Model

Based on the statistical property of access control workload as discussed in pervious section, we propose a data sensitivity based classification of cloud datacenters. The sensitivity classification is dependent on the level of sharing of data objects among the roles. In particular, we define the sensitivity of a datacenter as the average degree of sharing among its data objects. For example in Figure 2, the datacenter average degree of sharing equals to $\frac{10 \quad 1+4 \quad 2+8 \quad 3+2 \quad 4}{20} = 2\ 5$. In case the degree of sharing on average is low, we term the datacenter to

Algorithm 1: Workload generation algorrithm

Input: Number of data objects $\mathcal{O}$, number of roles n, constant$s$.
Output: spectral representation of RBAC $\mathcal{W}$.

Let $\mathcal{B} =$ B_1 B_n bucket array;
foreach $i = 1$ $mathcal\ O$ **do**
    = zipf(n,s);
    B $= B$ $+ 1$;
foreach $i = 1$ n **do**
    foreach $j = 1$ B_i **do**
        = zipf($\binom{n}{i}$,s);
        map to random partition in level i call it p;
        w_p = w_p +1;
    add w_p to $\mathcal{W}$
return $\mathcal{W}$

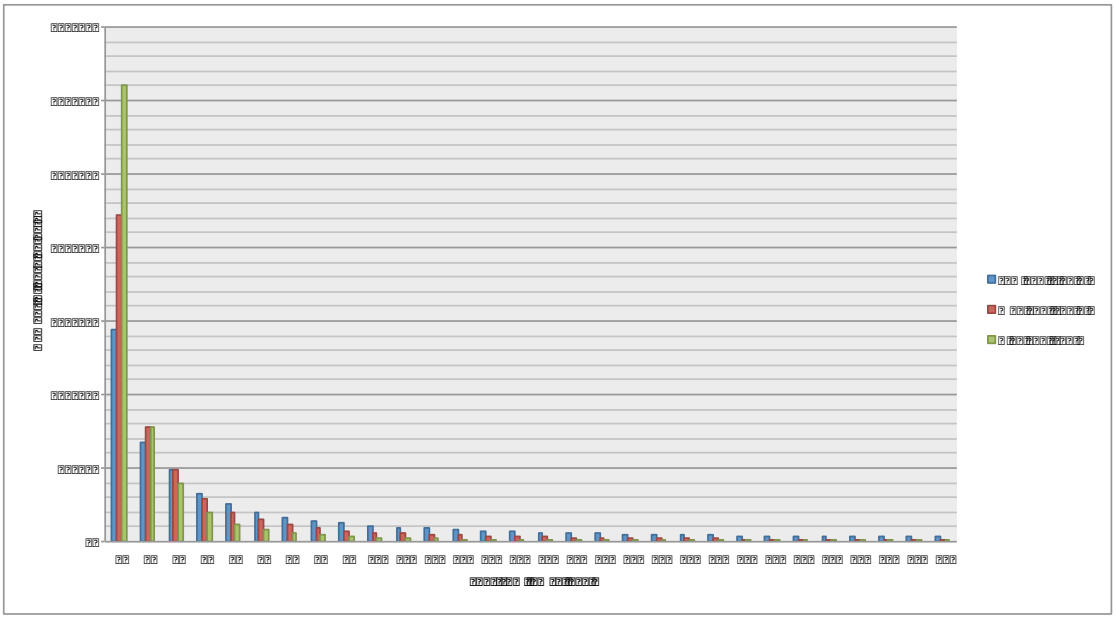


Fig. 3. data objects distribution for High, Medium, and Low sensitive datacenters

have a high sensitivty. On the contrary, if extensive sharing of data objects among roles is present, we term this datacenter to have a low sensitivity. The medium sensitivity class falls in the middle. It can be notice the sharing of data objects and classification of a datacenter can be modeled using the Zipfian distribution. The key parameter to characterize the sensitivity of datacenter is the scalar parameter s of Zipfian density function shown in equation 2. As shown in Figure 3, the smaller the value of s the more data objects are uniformly distributed in the $\mathcal{W}$ vector of the spectral model of RBAC. In the following example, we illustrate how Zipfian parameter s can be used to classify the sensitivity of datacenter.

Example 4.1: For datacenter with $T = 0\ 5\ \ 10^6$ data objects, suppose we have three RBAC policies ($P_1\ P_2\ P_3$) each with $n = 30$ roles. Figure 3 shows histogram of object across the spectral lattice. Depending on the Zipfian distribution, three classes of datacenters, namely; High sensitive (HSD), Medium sensitive (MSD), and Low sensitive (LSD), can be identified with respect to policies P_1, P_2, and P_3. For example, HSD can has a large value of s (s 2) since the sharing of data objects among P_1 roles is very small. On other hand, the LSD has a small value of s ($1\ 5 > s$ 1) depicting the case of extensive sharing of data objects among roles of policy P_3. The MSD has the value of $s$which falls in the middle ($2 > s$ $1\ 5$). Note, the number of data objects at level one in HSD is double the number of data objects at level one in LSD.

Note, the size of spectral representation vector, $\mathcal{W}$, increases exponentially with the number of role; since $\mathcal{W}\ = 2^n$ 1. For large datacenter, generating the total workload (i.e. capturing the cardinalities of all elements in $\mathcal{W}$) substantially increases the complexity of the virtual resources assignment for the given RBAC. Therefore, we propose an approximation strategy for workload characterization. The strategy is based on considering a smaller percentage of total size of datacenter. Let such percentage be denoted as D. In particular, D identifies

the cutoff level k in the lattice of $P(R)$ which can be used for developing low complexity heuristic for VRM. The following lemma describes the process of determining this cutoff k.

Lemma 4.1: Given $\mathcal{W}$ (spectral representation of RBAC), s (scalar parameter of Zipfian distribution), and D (the percentage of data considered), the cutoff level k in the lattice $P(R)$ is defined as
$k = min_{0 \leq k \leq n} H_{k,s} \geq \frac{D}{100} \times H_{n,s}$
Where:
$H_{k,s} = \sum_{i=1}^{k} i^{-s}$ is k^{th} generalized harmonic number.

Proof: Given the Zipfain density function $f(k, s, n)$, we find minimum k such that:
$\sum_{i=1}^{k} f(k, s, n) \geq \frac{D}{100}$
$\sum_{i=1}^{k} \frac{i^{-s}}{H_{n,s}} \geq \frac{D}{100}$
$\sum_{i=1}^{k} i^{-s} \geq \frac{D}{100} \times H_{n,s}$
$H_{k,s} \geq \frac{D}{100} \times H_{n,s}$ □

For a given D of a datacenter the spectral vector $\mathcal{W}$ needs to be truncated accordingly as detailed in Lemma 4.1. The truncated vector, denoted as $\mathcal{W}'$, is defined as follow.

Definition 4.1: (Truncated spectral vector$\mathcal{W}'$): We define $\mathcal{W}'$ to consist of all the partitions of $\mathcal{W}$ starting from level one up to and including the cutoff level k. The vector $\mathcal{W}'$ is formally defined as:
$\mathcal{W}' = \{w_p | (w_p \in \mathcal{W}) \wedge |p| \leq k\}$
Note, different sensitivity classes of datacenter yield different cutoff levels for the same percentage D. Accordingly, the size of $\mathcal{W}'$ varies. The example below illustrates how the D and the sensitivity classes can affect the value of k.

Example 4.2: For $\mathcal{W}$ of Example 4.1, for $D = 70\%$; the cutoff (k) is 2 for HSD and for LSD the cutoff is 8. For $D = 95\%$ the cutoff for MSD is 18 as compared to 9 and 24 for HSD and LSD, respectively.

In Section 8 we provide a trade off between the accuracy of risk assessment based on the total workload, represented by $\mathcal{W}$, vs. approximate workload, represented by $\mathcal{W}'$.

5 Heterogeneous Virtual Resource Vulnerability Characterization

For risk-aware resource assignment, in addition to workload characterization with respect to RBAC policy, the VRM also estimates the software vulnerability for a virtual resource e.g. virtual machine (VM) in our case. The estimator uses the security configuration of VMs to qualitatively classify VMs into multiple classes. The classification is based on the Common Vulnerability Scoring System (CVSS) metric scores. CVSS score uses an interval scale of 0-10 to measure the severity of vulnerabilities [14], [15]. To represent the probability of data leakage we convert the 0-10 scale to a scale from zero to one. Based on CVSS scores, we assume four discrete classes of VMs, namely highly secure, medium secure, low secure, and unscecure VMs. Although, we select four classes to model the vulnerability with respect to heterogenous virtual resources, our solution can be generalized to an arbitrary number of heterogeneous classes.

In addition to probability of leakage within a virtual resource, the VRM also needs to consider the leakage across virtual resources (VMs) within the same IaaS. VRM estimates the vulnerability of each IaaS cloud providers . Different remote cloud provider can deploy different security configuration and virtualization software (e.g. hypervisor) with varying level of vulnerabilities. Similar to the VMs classification, the VRM estimator also classifies the vulnerabilities of remote cloud providers into multiple classes. The vulnerability measurement within VMs are independent from the across VMs. For example, a highly secure cloud provider can host unsecure VMs. We assume that the probability of leakage between two VMs belonging to different remote Clouds is negligible. Subsequently, we convert these qualitative measures into probability of data leakage. We assume that probability of leakage within any VM is higher than probability of leakage across any two VMs. This is due to the fact that the size of trusted code base across VMs is generally smaller as compared to commercial OS used in a given VM . The trusted code base represents the software stack shared in the multitenant environment where within VM the shared software stack (e.g. OS and middleware) is larger as compared to shared software across VMs (e.g. the hypervisor).

Definition 5.1: (The cloud virtual resources model). Given $VM_1, VM_2, ..., VM_m$ as the suite of m virtual machines available to VRM , let d_{ij} represents the the probability of data leakage between VM_i and VM_j estimated using vulnerability estimation component as shown in Figure 1. Accordingly, the Cloud virtual resources can be modeled as a fully connected undirected graph $H(V, E)$ where vertices V represent the set of VMs and the weights on the edges represent d_{ij}.

6 Risk-aware Resource Assignment

Based on the RBAC workload characterization and aforementioned virtual resource vulnerability model, we now formally define the risk-aware assignment problem (RAP)

Definition 6.1: (The risk-aware assignment problem (RAP)). Given the spectral representation $\mathcal{W}$ of RBAC policy and the adjacency matrix $\mathcal{D}$ the adjacency matrix representing the virtual resources probability of leakage (based on $H(V, E)$), the risk-aware assignment problem (RAP) is to minimize the total risk of data leakage by assigning access control roles to the virtual resources.

The cost function of total risk is defined as follow:

$$\min R_t = \min \sum_{w_p \in \mathcal{W}'} w_p \times \sum_{j=1}^{n} Threat_{p,j} \times \max_{1 \leq l,q \leq m} \{d_{lq} \times I_{jq} \times I_{pl}\} \tag{3}$$

Where:
$I_{pq} = I_{jq} \quad \forall p \subset j$
$\sum_{q=1}^{n} I_{jq} = 1 \quad \forall j \in R$

$$I_{jq} = \begin{cases} 1 & \text{if } r_j \text{ is assigned to } vm_q \\ 0 & \text{Otherwise} \end{cases}$$

$$Threat_{i,j} = \begin{cases} 1 & \text{if } r_j \in w_i.p \\ 0 & \text{Otherwise} \end{cases}$$

Theorem 6.1: RSP problem is NP-complete

Proof: We construct a decision formulation of RASP problem as follow:
For a given $\mathcal{C}$, is there such $\mathcal{I}$ that $\sum_{w_p \in \mathcal{W}'} w_p \times \sum_{j=1}^{n} Threat_{p,j} \times \max_{1 \leq l,q \leq m} \{d_{lq} \times I_{jq} \times I_{pl}\} \leq \mathcal{C}$
First, given $\mathcal{I}$ and $\mathcal{C}$ we can check if the RASP decision problem satisfies the bound in polynomial time. Accordingly, the decision formulation of RASP is NP. Next, we show that decision RASP problem is NP-complete, by shown that Travel Salesman Problem (TSP) can be reduce to RASP. We formulate the TSP as for a given set V with a distance metric $d(a, b)$, $a, b \in V$, and a number $\mathcal{C}$, is there such an ordering of elements of $V : (v_1, v_2, \ldots v_n, v_1)$ such that $\sum_{i=1}^{n} d(v_i, v_{i+1}) + d(v_n, v_1) \leq \mathcal{C}$

Lets $\mathcal{W}$ is spectral representation of RBAC policy with n roles. For $\forall w_p \in \mathcal{W}$, we define w_p as follows:

$$w_p = \begin{cases} 1 & \text{if } p = \{r_i, r_j\} \text{ and } |r_i - r_j| mod\ n = 1 \\ 0 & \text{Otherwise} \end{cases}$$

Assume $L = \{VM_1, VM_2, \ldots, VM_n\} = V$,then assume that assignment of w_p to VM_j as if theTSP visiting city v_j. Therefore, if we find a optimal order of visits in TSP, we can find the optima assignment of partitions to VMs in RASP.

□

7 Assignment Heuristics

In this section, we propose two heuristics for solving RAP which can be deployed by the resource assignment component in VRM in Figure 1(b). The first heuristic called Sharing-based Heuristic (SBH) uses a best fit greedy strategy. In SBH, each role is assigned to the best available VM, in terms of its probability of leakage, such that any increase in the total risk is kept at minimum. This heuristic has a high complexity since it finds the local optimal assignment at each step. Alternatively, we proposes a low complexity scalable heuristic, called Partition-based Heuristic (PBH) , that uses a top down clustering based approach. In each step, the PBH algorithm divides the roles based on the highest risk partition. Below we formally describe the algorithms and prove their complexities. We then provide a detailed simulation based performance evaluation of the algorithms. As mentioned earlier, due to the exponential complexity of $\mathcal{W}$, both SBH and PBH use $\mathcal{W}'$. Both heuristics use a two phases approach. In phase one, a heuristic optimizes the assignment based on partitions in $\mathcal{W}'$. In phase two, all the remaining partitions ($\mathcal{W} - \mathcal{W}'$) are assigned automatically with their associated roles.

Prior to presenting the heuristics, we approximate the size of $\mathcal{W}'$ in term of cutoff k, by the following lemma.

Lemma 7.1: $|\mathcal{W}'| \leq (n+1)^k$

Proof:
$|\mathcal{W}'| = \sum_{i=1}^{k} \binom{n}{i}$
$\sum_{i=1}^{k} \binom{n}{i} \leq \sum_{i=0}^{\infty} n^i$
$since (n+1)^k = \sum_{i=0}^{\infty} n^i$
$\implies |\mathcal{W}'| \leq (n+1)^k$

□

7.1 Sharing Based Heuristic (SBH)

SBH follows the best fit approach where we initially select a role with the largest number of data objects. This role is assigned to a VM with the least probability of leakage. We then select the role that has the highest data sharing with the perviously assigned roles and allocate it to a VM such that any incremental in the total risk is kept at minimum. This step is repeated until all the roles are assigned. Notice, the first m roles assignments are made to distinct VMs as the probability of leakage across VMs is always less than probability of leakage within a VM. Therefore, the performance of SBH depends on the initial m role assignments. The remaining $n-m$ roles are co-allocated with the already assigned roles such that the total increase in leakage risk is kept low.

Formally the SBH is shown as Algorithm 2. In Lines 1-3, we perform the initial role assignment to a VM. The assigned roles are stored in list A. The unassigned roles are kept in the list F. C is a cost matrix that stores the cost of assignment of each unassigned role to each VM. In Lines 4-6 , each iteration of outer loop assigns a role from list F with the minimum value in C to a VM and updates the assignment matrix $\mathcal{I}$ which includes assignment of all the partitions so far. The inner loop updates the cost matrix C by removing the entry of last assigned role and updating the entry of C as a result of the new assignment. Finally, we return the assignment matrix at the end of algorithm.

Lemma 7.2: The complexity of SBH is $O(n^2 \times m \times |\mathcal{W}'|)$

Proof: The complexity of computing $C_{i,j}$ in the algorithm is $n \times |\mathcal{W}'|)$ since we need to find the sharing between role r_i and all other roles in A. This cost computation needs to be repeated m times for each role in F. Therefore, the total complexity of SBH is $O(n^2 \times m \times |\mathcal{W}'|)$.

□

Algorithm 2: SBH

Input: A spectral representation of RBAC $\mathcal{W}'$, vulnerability matrix $\mathcal{D}$.
Output: An assignment matrix $\mathcal{I}$ of roles to VMs.

1. Find initial vm which is vm_j with minimum d_{jj};
2. Find initial Role which is r_i with largest sensitive data p_i;
3. $\mathcal{I}(r_i, vm_j) = 1$;
4. Let $A = \{r_i\}$ be the set of assigned roles;
5. Let $F = R - \{r_i\}$ be the set of free roles;
6. Let $C_{i,j}$ be the risk of assign r_i to vm_j;
7. **foreach** $r_i \in F$ **do**
8. | **foreach** $j = 1, \ldots, m$ **do**
9. | | Compute $C_{i,j}$;
10. | Let $\hat{C_{kl}}$ be minimum $C_{i,j}$;
11. | $\mathcal{I}(r_k, vm_l) = 1$;
12. | $A = A \cup \{r_k\}$;
13. | $F = F - \{r_k\}$;
14. return $\mathcal{I}$;

7.2 Partition Based Heuristic (PBH): A Scalable Approach

In this heuristic, a scalable top down clustering approach is used. Initially, all the roles are assumed to be in one cluster. We then find the highest attackable partition in $\mathcal{W}'$. The attackability of partition is defined as the size of the partition multiplied by the number threats for that partition. The number of threats for a partition equals the total number of roles minus the partition's level in the lattice of $P(R)$. Note, as the size and number of threat increases the partition become more attackable. We begin with division of the root cluster and split it into two clusters. The first cluster contains the roles that are associated with highly attackable partition. The remaining roles are stored in the second cluster. By splitting the roles into two clusters based on the highly attackable partition, we eliminate the possibility of VM co-allocating the roles with highly attackable data objects with other roles that pose threats to them. The last step is repeated until the number of cluster equals the number of VMs. Subsequently, the cluster with highest policy risk is assigned to the least vulnerable VM. In essence, this greedy approach of dividing the roles based on the high attackable partition favors the m top attackable partitions over the rest of the partitions.

Algorithm 3, formally represents the PBH heuristic. In Line 1-2, The $\mathcal{W}'$ is sorted and saved in temporary vector P. The initial cluster list has one cluster which is the set of all roles. In Line 4- 11, we loop over P and divide a cluster into two new clusters if the partition p intersects with any cluster in C. We continue the loop in Line 4-11 until the number of clusters equal to number of VMs till we are done with all the partitions in

P. Then, we sort the VMs indexes in l and computes the policy based risk for each cluster. The policy based risk is the policy assets that are in threat before multiplying it with probability of leakage. The clusters are sorted and the roles of highest risk cluster assigned to VM with low probability of leakage, as illustrated in Line 15-17, The final assignment is stored in assignment matrix $\mathcal{I}$.

Lemma 7.3: The complexity of PBH is $O(|\mathcal{W}'| \times \log|\mathcal{W}'| + n \times |\mathcal{W}'|)$

Proof: In PBH we need to sort $\mathcal{W}'$ that gives the complexity of $|\mathcal{W}'| \times \log|\mathcal{W}|'$. Then to find the policy risk for a given cluster we need to compute the attackability for each partition in the cluster with complexity $n \times |\mathcal{W}'|)$. The assignment of cluster to VM can be computed in $O(n \times m)$ and m is always less than $|\mathcal{W}'|$. □

Algorithm 3: PBH

Input: A spectral representation of RBAC $\mathcal{W}'$, vulnerability matrix $\mathcal{D}$.
Output: An assignment matrix $\mathcal{I}$ of roles to VMs.

1. Sort $\mathcal{W}'$ starting with largest w_i to smallest w_j;
2. Let P holds indexes of the sorted $\mathcal{W}'$;
3. Let $C = \{R\}$ be the initial cluster;
4. **foreach** $p_i \in P$ **do**
5. **foreach** $c_j \in C$ **do**
6. **if** $p_i \cap c_j \neq \phi$ **then**
7. $C = C - c_j$;
8. $C = C \cup (p_i \cap c_j)$;
9. $C = C \cup \{c_j - p_i\}$;
10. **if** $|C| \geq m$ **then**
11. break;
12. Let $L = \{l_1, \ldots, l_m\}$ be the sorted VMs based on d_{jj} from smalls to largest;
13. Compute the intra risk for each $c_i \in C$;
14. Let $\hat{C}$ be the sorted C based clusters risk. ;
15. **foreach** $i = 1, \ldots, m$ **do**
16. **foreach** $r_k \in \hat{c}_i$ **do**
17. $\mathcal{I}(r_k, vm_{l_i}) = 1$;
18. return $\mathcal{I}$;

Note, due to the its subquadratic complexity in terms of number of role the PBH is a scalable heuristic.[1]

7.3 Performance Evaluation Metrics for Resource Assignment

We introduce two metrics to compare the performance of the proposed SBH and PBH. The formal definitions of these metrics are presented in this section and the experiment results are given in next section. The data leakage risk is the main metric that needs to be minimized. The risk metrics has two sub-metrics which we use to evaluate the proposed heuristics. The first risk metric is computed from the total workload which represents all the partitions in $\mathcal{W}$. This risk metric represents the total risk and denoted as R_t. In other words R_t represents the potential risk for the full datacenter. The second sub-metric of risk is based on the all partition of $\mathcal{W}'$ and is used to study the performance of heuristics and compare their effectiveness. This sub metric is called partial risk R_p. Note, R_p is risk resulting from phase two of the proposed heuristics. Formally, the risk metrics are given as:

$$R_t = \sum_{w_p \in \mathcal{W}} Risk(w_p) \tag{4}$$

$$R_p = \sum_{w_p \in \mathcal{W}'} Risk(w_p) \tag{5}$$

Note, the difference between total risk and partial risk represents an error introduce by the heuristic as a result of its lack of knowledge of all partitions. We define the relative error as follow .

Definition 7.1: The relative error of an assignment algorithm is
$E = \frac{(R_t - R_p)}{R_p} = (\frac{R_t}{R_p} - 1)$

1. http://en.wikipedia.org/wiki/Scalability

8 Performance Evaluation

We evaluate our heuristics and study their performance for different statistical workload approximation for RBAC and data sensitivity classification. Our workload generation algorithm uses the same statistical model used in Yahoo Cloud Serving Benchmark [13] where the selectivity of data objects is based on the Zipfian distribution. The datacenter sensitivity is carried with these aforementioned settings, named High sensitive ($s = 2$), Medium sensitive ($s = 1.5$), and Low sensitive ($s = 1$). Different data percentages (D) of datacenter are considered in our experiment which varies from 70% to 95% of total size of datacenter. which is assumed to be $500k$ of data objects. We simulate RBAC policies with 120, 150, and 200 roles.

In our experiment, we implement four classes of VM vulnerability named; High secure (with probability of leakage $d_{i,i} = 0.2$), Medium secure ($d_{i,i} = 0.45$), Low secure ($d_{i,i} = 0.6$), and unsecured ($d_{i,i} = 0.8$). Based on the survey in [10], we assume that 1% of VMs are unsecured, 22% are highly secured, 46% are medium secured VMs, and 33% are low secured VMs. Accordingly, we classify the security of IaaS into three categories. The first category is the highly secure IaaS cloud provider which we assume the probability of data leakage among its VMs is extremely small ($d_{i,j} = 0.001$). The moderately secure IaaS cloud provider constitutes the second category which has $di, j = 0.045$. The last category is the low secure IaaS, with $d_{i,j} = 0.1$.

In Table 1 we show the total risk R_t resulting from SBH assignment of 120 roles onto 40 VMs. R_t of the HSD is noted to be higher than the MSD, and LSD. This is because most of data objects in HSD are highly attackable does can increase the number of potential threats as compared to other type of datacenters.

In general, we can notice that R_t decreases as the value of D increases from 70% to 90%. This decrease in R_t is because for the large value of D, the VRM has more knowledge about $\mathcal{W}$. In other words, as $(\mathcal{W} - \mathcal{W}')$ gets smaller it causes a decrease in uncertainty in assignment decision. However in some cases R_t increases when the data percentage is increased from 90% to 95% as depicted in Low sensitivity column in Table 1. The reason is both heuristics try to minimize the partial risk R_p associated with $\mathcal{W}'$, and in some cases such minimization can result an increase of R_t.

TABLE 1
R_t for n = 120 and m = 40

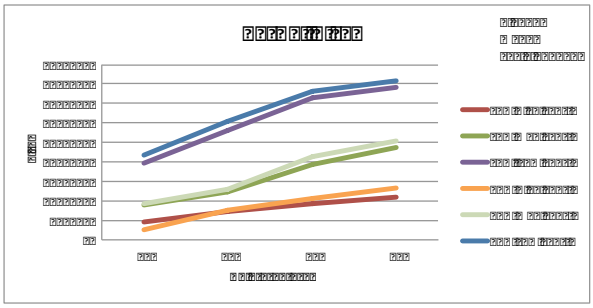
	Low		Medium		High	
	SBH	PBH	SBH	PBH	SBH	PBH
70	3360231	3606591	2445374	2452659	1518554	2139564
80	3291886	3522308	2244157	2401973	1465960	1512723
90	3215064	3470285	2154764	2309507	1231830	1405526
95	3344730	3470285	2217724	2302854	1224026	1406141

TABLE 2
R_t for n = 150 and m = 50

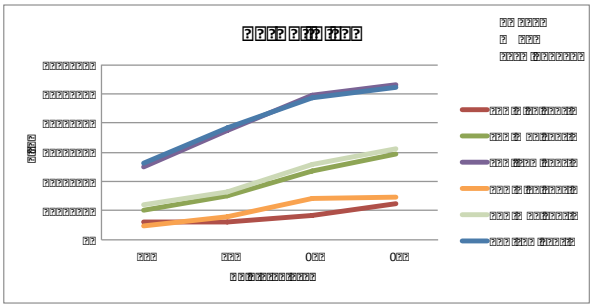
	Low		Medium		High	
	SBH	PBH	SBH	PBH	SBH	PBH
70	4121669	4225904	2932017	3263288	1767732	2658996
80	4078425	4196071	2785004	2828237	1756312	2234971
90	4068345	4196071	2663637	2782716	1558556	1761015
95	4038736	4201984	2647521	2782716	1468684	1692846

TABLE 3
R_t for n = 200 and m = 70

	Low		Medium		High	
	SBH	PBH	SBH	PBH	SBH	PBH
70	5685296	5402101	3865387	4362990	2228901	3627837
80	5562226	5352938	3686288	3613758	1973325	2837346
90	5580674	5351282	3356819	3446952	1692089	2355639
95	5475651	5350779	3355768	3446952	1754023	1970093



(a) R_p for SBH and PBH for n =150 .



(b) R_p for SBH and PBH for n =200 .

Fig. 4. SBH & PBH performance

In Figures 5 we present the performance of SBH and PBH by computing the relative error E. The general behavior is that the relative error decreases as the D increases. The reason being that the cutoff level k increases proportionally with D resulting in a reduction of E. Further, we can notice that the relative error (E) is high for HSD as compared to LSD. The reason is that for HSD the cutoff k is smaller which results in ignoring data objects in some risky partitions among roles associated with level $\frac{n}{2}$ of the lattice in Figure 2(b). For the case of PBH, E is high for HSD when $D = 70\%$. This is due to the fact that the number of partitions that are highly attackable is few. Accordingly, PBH may not be an attractive choice for HDS with low value of D. However, as can be notice from Tables 1 and 2 the relative error for PBH with respect to SBH is within 10% for both LSD and MSD cases. Being a scalable algorithm PBH can therefore offer a viable choice for these cases.

9 Conclusion

In this paper, we have presented a risk-aware cloud virtual resources assignment for big datacenter employing RBAC policy. We have defined a formal definition of spectral representation of access control policy and proposed statistical workload characterization and the data percentage approximation for cloud datacenters. We have proposed a sensitivity based characterization for large datacenter. The virtual resources vulnerability probability using CVSS score is presented to model the heterogeneity of cloud virtual resources security configurations. We have proposed two heuristics algorithms for scheduling to solve the assignment problem with simulation based performance evaluation. The key finding of our work is the PBH being a scalable algorithm can provide a viable heuristic for scheduling cloud resource for low sensitivity and medium sensitivity datacenters. However, for high sensitivity datacenter, SBH performance is relatively better in terms of reducing the overall risk.

References

[1] G.-H. Kim, S. Trimi, and J.-H. Chung, "Big-data applications in the government sector," *Communications of the ACM*, vol. 57, no. 3, pp. 78–85, 2014.

[2] J. Alcaraz Calero, N. Edwards, J. Kirschnick, L. Wilcock, and M. Wray, "Toward a multi-tenancy authorization system for cloud services," *Security Privacy, IEEE*, vol. 8, no. 6, pp. 48–55, 2010.

[3] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds," in *Proceedings of the 16th ACM conference on Computer and communications security*. ACM, 2009, pp. 199–212.

[4] M. Pearce, S. Zeadally, and R. Hunt, "Virtualization: Issues, security threats, and solutions," *ACM Computing Surveys (CSUR)*, vol. 45, no. 2, p. 17, 2013.

[5] L. Catuogno, A. Dmitrienko, K. Eriksson, D. Kuhlmann, G. Ramunno, A.-R. Sadeghi, S. Schulz, M. Schunter, M. Winandy, and J. Zhan, "Trusted virtual domains–design, implementation and lessons learned," in *Trusted Systems*. Springer, 2010, pp. 156–179.

[6] U. Steinberg and B. Kauer, "Nova: a microhypervisor-based secure virtualization architecture," in *Proceedings of the 5th European conference on Computer systems*. ACM, 2010, pp. 209–222.

[7] S. Berger, R. Cáceres, K. Goldman, D. Pendarakis, R. Perez, J. R. Rao, E. Rom, R. Sailer, W. Schildhauer, D. Srinivasan *et al.*, "Security for the cloud infrastructure: Trusted virtual data center implementation," *IBM Journal of Research and Development*, vol. 53, no. 4, pp. 6–1, 2009.

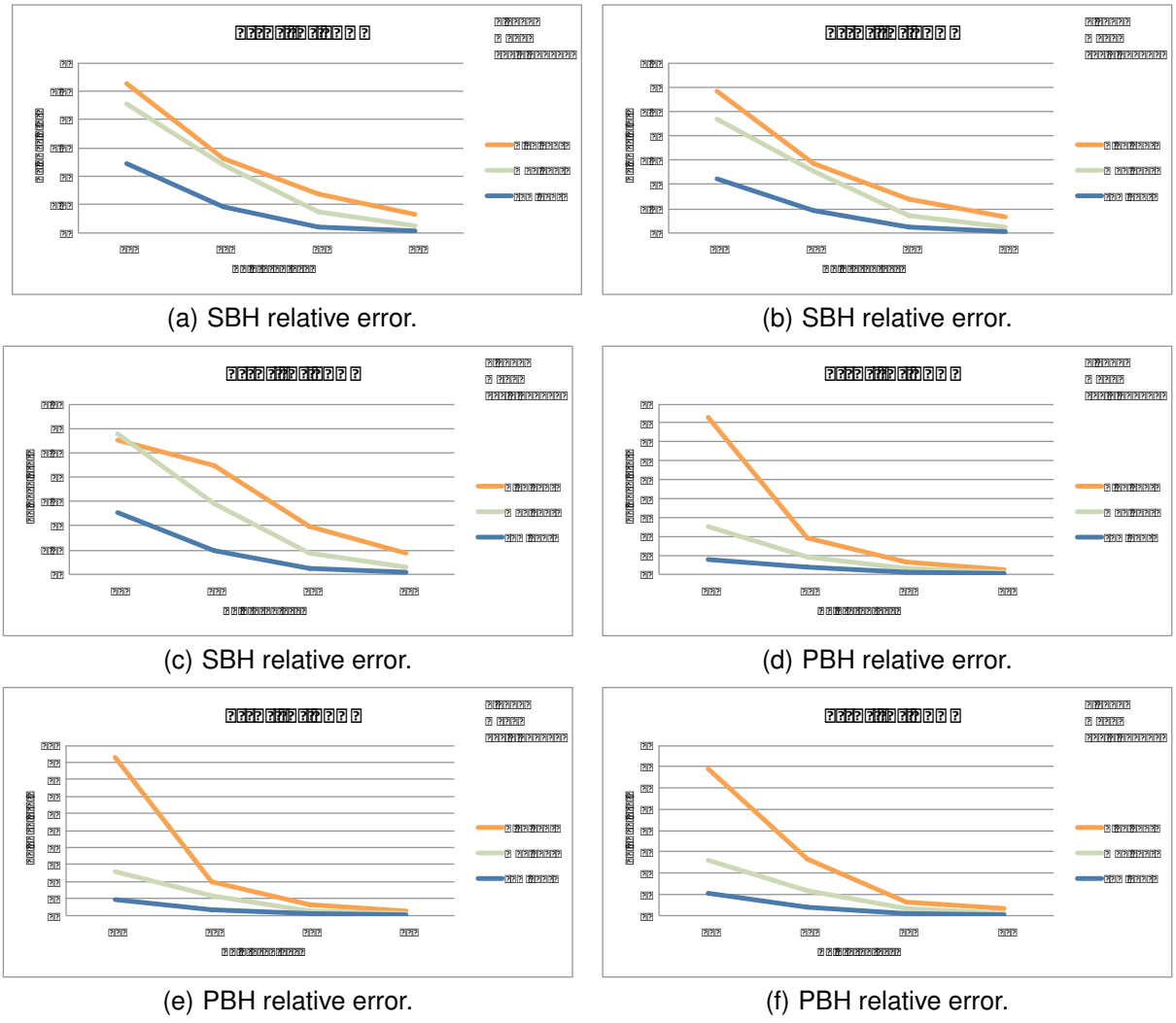
[8] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli, "Proposed nist standard for role-based access control," *ACM Transactions on Information and System Security (TISSEC)*, vol. 4, no. 3, pp. 224–274, 2001.

[9] A. Almutairi, M. Sarfraz, S. Basalamah, W. Aref, and A. Ghafoor, "A distributed access control architecture for cloud computing," *Software, IEEE*, vol. 29, no. 2, pp. 36–44, 2012.

[10] M. Balduzzi, J. Zaddach, D. Balzarotti, E. Kirda, and S. Loureiro, "A security analysis of amazon's elastic compute cloud service," in *Proceedings of the 27th Annual ACM Symposium on Applied Computing*. ACM, 2012, pp. 1427–1434.

[11] H. Zhang, I. F. Ilyas, and K. Salem, "Psalm: Cardinality estimation in the presence of fine-grained access controls," in *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on*. IEEE, 2009, pp. 505–516.

[12] M. Frank, A. P. Streich, D. Basin, and J. M. Buhmann, "Multi-assignment clustering for boolean data," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 459–489, 2012.



(a) SBH relative error.

(b) SBH relative error.

(c) SBH relative error.

(d) PBH relative error.

(e) PBH relative error.

(f) PBH relative error.

Fig. 5. Relative Error

[13] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with ycsb," in *Proceedings of the 1st ACM symposium on Cloud computing*. ACM, 2010, pp. 143–154.

[14] P. Mell, K. Scarfone, and S. Romanosky, "A complete guide to the common vulnerability scoring system version 2.0," in *Published by FIRST-Forum of Incident Response and Security Teams*, 2007, pp. 1–23.

[15] H. Holm, M. Ekstedt, and D. Andersson, "Empirical analysis of system-level vulnerability metrics through actual attacks," *Dependable and Secure Computing, IEEE Transactions on*, vol. 9, no. 6, pp. 825–837, 2012.