# CBAC4C: Conflict-based VM Isolation Control For Cloud Computing

M.T. Dlamini[1,4], JHP Eloff[1,2], HS Venter[1], MM Eloff[3],
[1]ICSA Research Group, Department of Computer Science, University of Pretoria, South Africa
[2]Cyber-security & Big Data Science RG, Department of Computer Science, University of Pretoria, South Africa
[3]Institute for Corporate Citizenship, University of South Africa, Pretoria, South Africa
[4]Information and Cyber Security, Defence and Security Cluster, CSIR, Pretoria, South Africa
{[1,4]moses.dlamini; [1,2]jan.eloff; [1]hein.venter}@up.ac.za
[3]eloffmm@unisa.ac.za

ABSTRACT

For businesses to benefit from the many opportunities of Cloud computing, they must first address a number of security challenges, such as the potential leakage of confidential data to unintended third parties. An inter-VM attack, also known as cross-VM attack is one threat through which Cloud-hosted confidential data could be leaked to unintended third parties. An inter-VM attack exploits vulnerabilities between co-resident guest VMs that share the same Cloud infrastructure. In an attempt to stop such an attack, this paper uses the principles of logical analysis to model a solution that provides physical separation of VMs belonging to conflicting tenants based on their levels of conflict. The derived mathematical model is founded on scientific principles and implemented using four conflict-aware VM placement algorithms. The resultant algorithms consider a tenant's risk appetite and cost implications. The model offers guidance to VM placement and is validated using a proof-of-concept. A Cloud simulation tool was used to test and evaluate the effectiveness and efficiency of the model. The findings reflect that the introduction of the proposed model introduced a time lag in the time it took to place VM instances. On top of this, it was also discovered that the number and size of the VM instances has an effect on the VM placement performance. The findings further illustrate that the conflict tolerance level of a VM has a direct impact on the time it took to place.

Keywords

Cloud computing, mathematical modelling, conflict-aware virtual machine placement, data leakage

## 1. INTRODUCTION

Shared public Cloud infrastructures open up a subtle data leakage threat which happens on the virtualization layer. This threat allows tenants to gain unauthorised access to confidential data from other co-resident tenants. For example, Perez-Botero, Szefer and Lee (2013) reflect on how malicious attackers could practically exploit vulnerabilities to the memory management unit to compromise the confidentiality of data belonging to co-resident guest VMs. Cloud computing uses virtualization and multi-tenancy to place tenants' Virtual Machines (VMs) on a shared physical infrastructure. Virtualization abstracts and multiplexes the shared physical infrastructure of a Cloud Service Provider (CSP) and hides the complexity of its underlying pool of virtualized resources. Multi-tenancy, on the other hand, allows different tenants to share the pool of virtualized resources provided by the CSP (Auer et al. 2019; Goyal et al. 2019). Both these techniques are realised by the concept of a VM instance.

Multi-tenancy and virtualization make it possible for VMs of co-located tenants to be instantiated on the same physical infrastructure. Given this scenario, it means that tenants' confidential data within VMs sharing the same physical infrastructure is only logically separated. If VMs are co-located on the same physical Cloud infrastructure, they can reveal confidential data (Lefray et al. 2015; Xing et al. 2019; Bazm et al. 2019). It would only take another tenant, who might be a competitor, to launch an inter-VM attack targeted at its rival to gain access to their confidential data. For this reason, the ever-increasing demand of Cloud computing services places VM placement at the heart of CSPs' decision making process (Goyal et al. 2019).

An inter-VM attack can be carried out by a malicious guest VM in order to compromise the confidentiality, integrity or availability of virtualized resources of other co-resident guest VMs. This

attack exploits vulnerabilities on the physical separation between co-resident guest VMs that share the same physical infrastructure (Ahmad and Bakht 2019; Bazm et al. 2019). To broaden the attack surface, malicious tenants can compromise the hypervisor to escalate their access privileges and gain administrator access to a wide range of VMs. Given that a hypervisor runs at the highest privilege level and mediates all user access to the shared pool of virtualize resources, it becomes a single point of failure. This makes it a critical component from a security perspective and a prime target for attackers. A compromise on a hypervisor puts all VM instances that it monitors in danger (Ahmad and Bakht 2019). Exploiting a hypervisor would allow an attacker to perform administrative tasks of creating, starting, stopping and destroying guest VMs as they please. For example, some research has already demonstrated how a malicious user's guest VM could exploit vulnerabilities to trick a hypervisor to issue a command that could destroy another co-resident guest VM (Almutairy and Al-Shqeerat 2019). Consequently, some researchers are proposing solutions that either aim to remove or harden the hypervisor (Barrowclough and Asif 2018; Nanavati 2019).

This paper derives and proposes a mathematical Conflict-Based VM Isolation Control for Cloud Computing (CBAC4C) model. The CBAC4C model is aimed at improving the physical separation between VMs in order to contain data leakage threats posed by inter-VM attacks. The physical separation focuses on the physical hosts, clusters and data centres. The rest of the paper is structured as follows: Section II discusses related work. Section III highlights system requirements, problem context and assumptions, and provides a theoretical formulation of the problem. Section IV derives and presents a mathematical formulation of the problem. The proposed CBAC4C architecture and algorithms thereof are presented in Section V. Section VI briefly discusses some screenshots of our proof-of-concept prototype. Section VIII concludes the paper and provides direction for future work.

## 2. RELATED WORK

A number of research efforts (Mashayekhy et al. 2014; Ristenpart et al. 2009; Kamran 2018; Yadav, Bharti and Raw 2018) have made attempts to address the problem of VM placement (in our context placement includes both initial placement and subsequent migration) on the Cloud from different view-points. In support, Bartok and Mann (2015) and Ferdaus et al. (2017) assert that the VM placement problem in Cloud computing has been extensively studied with greater emphasis on server resource utilization, energy consumption and quality of service. Han et al. (2013) and (2014) argues that the VM placement problem can be categorised based on how it balances workloads, reduces power consumption and improves security. Some existing research efforts argue that the VM placement problem can be categorised based on performance issues and maximum utilisation of resources (Calcavecchia et al. 2012; Azar et al. 2014). Some research work (Kamran 2018; Doung-Ba et al 2018) takes a business view to focus on VM placement that aims to minimize costs or maximize returns for CSPs. The work of Filho et al. (2018) provides a comprehensive classification of literature focusing on the different aspects of VM placement problem which also include a pointer to information security.

This paper puts more emphasis on existing work that attempts to address the VM placement problem from the view of providing security guarantees. For example, Nanavati (2019) incorporates stronger VM isolation in a trusted computing base of a hypervisor. This work mediates and restricts access to shared Cloud infrastructure but does not cover the shared cache memory. The short-coming of Nanavati's work comes in its failure to provide a greater degree VM separation. This is due to the lack of support from the underlying shared hardware which makes it computationally expensive to provide a greater degree of VM separation (Nanavati 2019). Similar to Nanavati's work, our research also does not extend the separation of VMs to the shared cache, but it learns and uses Nanavati (2019) as a baseline to provide an improvement on the degree of VM separation.

Mashayekhy et al. (2014) propose a data protection framework for federated Cloud infrastructure. The propose framework therein incorporates a VM placement and migration strategy. Mashayekhy et al.' strategy considers data protection requirements in terms of restricting VM co-residence and co-location in what they refer to as 'trust restrictions' and 'disclosure restrictions' respectively. Mashayekhy et al. (2014) considers the separation of tenants' VMs based on conflict of interest as a level of data protection on top of encryption. It is argued therein that the strategy subsequently reduces the need to encrypt all data (Mashayekhy et al. 2014). Encrypting all Cloud-bound data has proven to increase computational costs and is also time consuming (Mosola et al. 2017). Other work (Zhang et al. 2013; Zhang et al. 2014) as cited in Mashayekhy et al. (2014) have made plausible attempts to address the cost and time implications of encryption in the Cloud.

Moreover, Tobin et al. (2018) report a widely used encryption standard which has a gaping backdoor that allows law enforcement agencies uninterrupted access to Cloud-hosted data. The work of Abelson et al. (2015) reports on more calls for regulations mandating Cloud Service Providers amongst others to design backdoors on their crypto systems. Such backdoors are argued to give law enforcement agencies 'exceptional access' to intercept suspected tenant data for analysis. This compromises the integrity of cryptographic standards and poses one of the greatest risks, as malicious entities can also use such 'exceptional access' backdoors for stealing data.

Compromising the integrity of encryption standards indicates that CSPs cannot be trusted to prevent Cloud-hosted data from being leaked to unauthorized third parties. A good example is the United States of America vs Paige A. Thompson case, where a disgruntled former employee of Amazon - a Cloud computing company decided to leak confidential data of a tenant (financial institution) and host it in GitHub (Martini and Theiler 2019). Such cases which can be easily prevented by our proposal continue to erode consumer trust on Cloud services. The untrustworthiness of CSPs necessitates a tenant-side encryption mechanism to protect data from disgruntled and malicious CSP employees. This requires tenants to encrypt their Cloud-bound data on their premises before it is uploaded on the Cloud servers. Mosola et al. (2017) posits that tenants must use their own personalised encryption algorithms and keys that cannot be shared with the CSP.

Kumar et al. (2014) make use of the Bell-LaPadula confidentiality model to detect data leakage on the Cloud. This work aims to identify in real-time those who are responsible for data leakage. Kumar et al. (2014) reflects on the computational cost, complexity and time lag introduced by robust cryptographic algorithms. Therefore, they make use of a lightweight AES-128 encryption algorithm, SHA-512 hashing algorithm and water-marking to provide twice as much security with half the computational time required (Kumar et al. 2014). It is for the above reasons that the current paper also considers the separation of tenants' VMs based on conflict of interest as extra layer of protection running on top of encryption to provide defence-in-depth. The idea is not necessarily to replace encryption, but to provide an extra layer of data protection on top of it to prevent data leakage. Encryption in itself does not prevent data leakage threats, it only ensures that if data is leaked, attackers would not be able to decrypt and use it.

There are scanty research efforts that investigate data leakage threats in the Cloud (Ristenpart et al. 2009; Zhang 2012; Tsai et al. 2011; Si et al. 2013). Existing efforts aim to strengthen the physical isolation (also referred to as separation) layer by employing the popular Chinese Wall Model (CWM) (Brewer and Nash 1989; Tsai et al. 2011). For example, Tsai et al. (2011) developed a Chinese Wall Central Management System (CWCMS) that manages the deployment of VMs based on the CWM. However, the CWM was designed in the time of centralised architectures where most of the data was hosted in a centralised location. The CWM cannot cope with the complex environments presented by

Cloud infrastructures. Consider for example scalability problems in that (in the worst case) a Cloud infrastructure is created in which all VMs of tenants are hosted on their own physical nodes. Such an approach is not scalable.

Si et al. (2013) proposed a Security awareness VM Placement Scheme (SVMPS). This scheme considers two conflict-of-interest relations, namely "Aggressive Conflict of Interest Relation" (ACIR) and "Aggressive In Ally with Relation" (AIAR). Both relations are based on Brewer and Nash's Chinese Wall security model. Using these, Si et al. (2013) were able to formulate and enforce physical isolation rules for placing and migrating VMs owned by conflicting users on the Cloud. Unlike Ristenpart et al. (2009), Sailer et al. (2005) does not discuss the issue of optimal utilisation of resources in the Cloud infrastructure. Furthermore, Si et al.' solution does not provide tenants with any visibility with regard to the placement of their VM in the Cloud infrastructure. This is one aspect that this current paper tries to address with its ability to generate GPS coordinates and show the location for tenants' VM placements at any point in time.

The work of Wang et al. (2012) and Wang et al. (2013) provides tenants with a mechanism to verify CSP's physical VM isolation – an issue that they argue has not yet been fully investigated. This is due to the complex nature of conflicts that arise between entities, more especially smaller ones which may not be in a position to disclose their conflicts for the sake of losing business. Such complexity is made worse, if for example some of the tenants choose to spoof their identity and intentionally register incorrect line of business. The current paper also assumes trustworthy tenants and specifically focuses on addressing an inter-VM attack which happens within the Cloud infrastructure where potentially conflicting tenants' VM may co-reside. Though, it may be the best thing to model our solution to cater for all possible attack vectors, it is not feasible to do so in one paper.

Wang et al. (2013) and Sailer et al. (2005) adopted and enforced the CWM to address the problem of insecure information flow as an attempt to prevent accidental leakage of confidential information to unintended parties. Similar to this work, Ristenpart et al. (2009); Wang et al. (2013); and Wu et al. (2010) resolves conflict-of-interest issues in Cloud computing at the IaaS layer. Unfortunately these efforts do not consider the potential different degrees of conflict of interest that arises in the Cloud.

Han et al. (2013) make use of game theory to determine a VM placement policy that minimizes an attacker's possibility of co-locating with their targets. This work also maintains a workload balance and lower power consumption. Han et al. (2014) propose a VM placement policy that allocates a new VM to a physical node with most VMs. This is an improvement on their previous work (Han et al. 2013). They argue that this approach increases the difficulty for an attacker to achieve co-residence with their target. Han et al. (2017) extends (2013) and (2014) with a mathematical formulation (Jeihoonian, Zanjani and Gendreau 2020) of the solution to mitigate the threat of co-resident attacks whilst satisfying constraints in workload balance and power consumption. In an attempt to prevent an attacker from starting too many VM instances, and contrary to our work, Han et al.'s efforts attempt to bundle all VMs of a user in one physical host. This goes against the second goal of load balancing and bundling all VMs in one physical host might have far reaching consequences with respect to hardware failures.

Miao, Wang and Wu (2018) proposes a technique that minimizes co-residence of conflicting VM on the same host. Miao et al. (2018) is consistent with our work because it also considers different degrees of conflict in their *ConflictDegree (cd)* matrix, where they claim that $cd_{ij}$ expresses the degree of conflict between tenant *i* and tenant *j*. However, it is not clear how the different degrees of conflict correlate to the placement if their isolation is only focused on the physical host without considering the cluster and data centre levels. Miao et al. (2018) also does not show exactly how the isolation of partially conflicting

tenants is to be handled. Without this, their work also becomes a dichotomy in that there is either a conflict or no conflict, yet this is not the true reflection of what happens in reality. By far Miao et al. (2018) is closer to the approach taken in this paper, though they also consider load balancing and cost of migration.

Lefray et al. (2015) proposes a fine-grained VM allocation mechanism to prevent information leakage through micro-architectural covert channels. This work allows tenants to specify isolation properties and acceptable risk that fits their risk profile before placement can be done. Though the work of Lefray et al. (2015) does not consider conflict of interest, their proposed model is similar to the current paper in that it gives tenants a chance to choose VM placement based on their chosen isolation properties and risk profile.

The current paper extends and takes the work of Ratsoma et al. (2015) as a baseline to incorporate jurisdictional issues which allows potential tenants to make an informed decision on the location of their VM placements. The current work also includes the associated cost and risk of placement or migration of tenant VMs. This paper also extends the work in Dlamini et al. (2014) by providing the actual model, VM algorithms with the implementation thereof.

The overview of related work has highlighted a number of shortcomings of existing approaches. For example, none of the covered literature discusses the need for solutions that consider varying degrees of conflict, improved scalability and visibility of VM placements. There is also a need for mechanisms that present a well-structured IaaS approach for the placement of VMs on the Cloud. The one major shortcoming of existing approaches is that none of the work focuses on providing value for the tenants. All the work seems to be focused on providing value for the CSPs. Tenants need to have an input into how their VMs are placed and migrated on the Cloud. Furthermore, existing work seems to be taking one view of the problem by placing more emphasis on the placement of a new VM. This leaves a huge gap for research efforts that takes a holistic approach to cover initial VM placement and subsequent migration of an existing VM.

The main contribution of this paper is as follows:

(1) It introduces a novel tenant-oriented conflict-aware VM placement approach that considers a holistic view (i.e. both initial VM placement and migration) based on tenants' different degrees of conflict of interest, i.e. direct (very high), high, medium, low and no conflict.

(2) It provides physical Separation (S) of conflicting tenants' VMs according to the potential data leakage Risk (R) posed and Cost (C) in order to mitigate and defend against inter-VM attacks.

(3) It abstracts, defines and models the Cloud infrastructure into Location, Data Centre, Cluster, Physical Node and VM in order to provide for jurisdictional and regulatory compliance issues associated with the VM placement and migration problem. The idea is to ensure that the final decision to place or migrate VMs is informed by the tenants as well as regulatory requirements and jurisdictional mandates.

(4) It provides transparency and visibility of VM placement or migration and ensures that all such efforts are consented and approved by the tenants. Users of the Cloud have always complained that they do not have control over where their data sits on the Cloud at any particular point in time. Even though tenant data might be hosted in VMs on the Cloud infrastructure, this does not absolve them of their responsibility to account for those Cloud-hosted datasets. Hence, this paper partly places VM placement and migration back to the hands of tenants.

(5) It formulates the VM placement problem as a prioritized best-fit and multi-objective bin packing heuristic. A prioritized best-fit and multi-objective bin packing heuristic makes use of a priority list of potential VM placement addresses. These heuristics are used to determine not only the best-fit VM placement options but one that is prioritized according to the needs of the tenant.

The next section outlines the requirements of the proposed solution. These are mainly based on the gaps in literature as identified above and the main contribution.

## 3. REQUIREMENTS, PROBLEM CONTEXT AND ASSUMPTIONS FOR CBAC4C

The CBAC4C model as proposed in this paper is based on the CWM policy where each tenant belongs to only one Conflict-of-Interest (CoI) class. A CoI is a grouping of tenants from the same functional business domain. Furthermore, it is assumed that the data sets of a tenant could be assigned to and stored on one or more VMs. Based on these assumptions and the current body of knowledge, the requirements for CBAC4C are as follows:

**Requirement 1**: Focus on Infrastructure as a Service (IaaS).

**Requirement 2**: Provide VM isolation based on Conflict-of-Interest (CoI) classes and Conflict Tolerance Levels (CTL).

**Requirement 3:** Manage "different degrees of conflict" between tenants.

**Requirement 4**: Minimise the risk of data leakage caused by inter-VM attacks.

**Requirement 5**: Provide visibility with regard to the placement of VMs.

Below, the authors contextualize the problem and provides some assumptions. The CBAC4C is modelled to conform to and operate in a hierarchical Cloud IaaS architecture as illustrated in Figure 1 below.

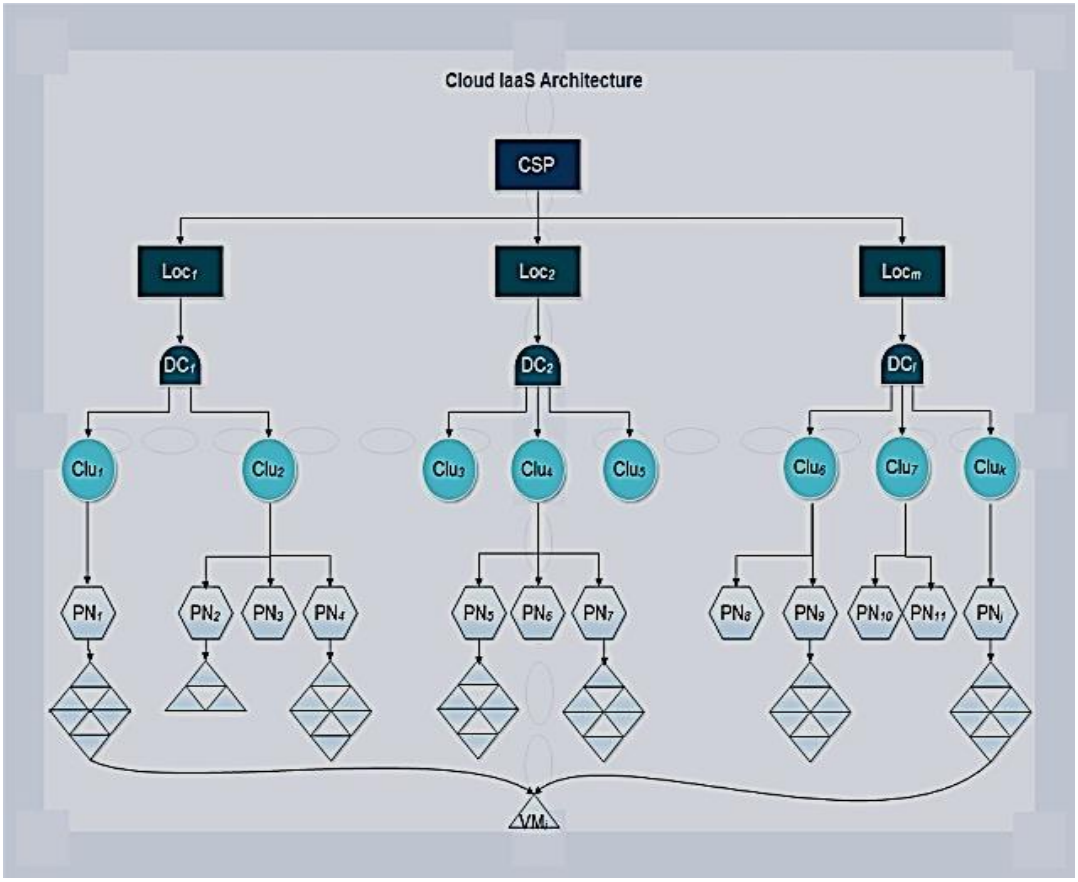


**Fig 1** Cloud IaaS Architecture

The CBAC4C model is based on the context of Figure 1 and following assumption i.e.

$$VM \subseteq PN \subseteq Clu \subseteq DC \subseteq Loc$$

This basically mean that a VM is contained in a Physical Node (PN) which is contained in a Cluster (Clu) that is contained in a Data Centre (DC) which is found at a specific geographical Location (Loc). However, the relationships between each of these are not necessarily of a one-to-one type.

## 4. MATHETICAL PROBLEM FORMULATION

The CBAC4C is modelled based on a relationship between a tenant *te*, which operates within a Functional Business Domain (FBD) and has a CoI with its competitors. The relationship is modelled as follows: a tenant *te* is a member of a FBD; a FBD is associated with a specific CoI class. For example, an oil company *Oil-A* belongs to the Petroleum FBD. Hence, *Oil-A* - a direct competitor of company *Oil-B* belong to the same FBD and CoI class, which means they are in direct conflict with one another. Consequently, requests by *Oil-A* and *Oil-B* for VM placement or migration must be handled in such a way that the risk of inter-VM attacks is minimised. This is modelled in such a way that no one tenant will have knowledge of other tenants. Only the Cloud Service Provider has access to all information about each of the tenants.

For a VM placement, a tenant ($te_i$) provides four inputs:

(1) A tenant identification ($te_i\_ID$)
(2) A Conflict Tolerance Level (*CTL*)
(3) A *FBD* which is expressed in terms of a CoI class

(4) Size of a requested VM (*sizeOfVM*) that is directly proportional to the resource capacity constraint – this can also be deduced from the data that the tenant wants to host.

A $te_i\_ID$ is a unique identifier for each tenant. The *CTL* determines how much conflict a tenant $te_i$ can tolerate for being hosted on the same infrastructure with a conflicting tenant $te_j$. The *FBD* of $te_i$ and $te_j$ is the same if they are in direct conflict. The size of a requested VM refers to the storage resource capacity for example in terms of megabytes (MB). Each VM, PN, Clu, DC and Loc at time *t* is subject to resource capacity constraints, i.e. $c_i(t)$, $c_j(t)$, $c_k(t)$, $c_l(t)$ and $c_m(t)$ respectively. The model targets the IaaS of Cloud computing, and this paper then puts an emphasis on storages as compared to CPU, networking and memory.

### 4.1 Sphere of Conflict and Non-Conflict

Each tenant $te_i$ has its own Sphere-of-Conflict ($SoC_i$) and Sphere-of-non-Conflict ($SonC_i$) sets (Loock and Eloff 2005; Loock 2012). The Sphere-of-Conflict set is denoted as:

$$SoC_i=\{te_1,\ldots. te_n\}$$

where; $te_1$, …..$te_n$ is a set of conflicting tenants to $te_i$ that uses the same CSP.

For each $VM_i \in te_i$ in the set $SoC_i = \{te_1,\ldots. te_n\}$ is associated with an address $Addr_i$ in the following form:

$$Loc_m.DC_l.Clu_k.PN_j.VM_i.$$

A placement matrix is the list of all addresses for all VMs in the set $SoC_i$ that are hosted in the CSP. This set could be further refined by applying a *CTL* to eliminate some addresses from the union set $SoC_i$. A tenant $te_i$ may be in conflict with any other tenant $te_j$ due to it being in the same *FBD*, in other words the same or related *CoI* class.

The proposed CBAC4C model is designed to be flexible and allow CSPs to co-host a tenant's VMs with other conflicting tenants' for specific requests. This is referred to as non-optimal placement because it does not guarantee adequate physical separation between the conflicting tenant VMs. Tenants could, however, opt for non-optimal placement of their VMs for various reasons, including affordability and hosting of public data without any confidentiality or secrecy clauses. Tenants could also opt to not host their VMs with any conflicting tenants. This is called optimal placement which implies that tenants can only co-host their VMs with non-conflicting tenants to ensure adequate physical separation. The set of all non-conflicting tenants is called a Sphere-of-non-Conflict.

The Sphere-of-non-Conflict set is denoted as:

$$SonC_i=\{te_o,\ldots,te_z\}$$

where; $te_o,\ldots,te_z$ is a set of non-conflicting tenants to $te_i$ that uses the same CSP.

For each $VM_i \in te_i$ in the set $SonC_i = \{te_o,\ldots. te_z\}$ is associated with an address $Addr_i$ in the following form:

$$Loc_m.DC_l.Clu_k.PN_j.VM_i.$$

A placement matrix is the list of all addresses for all VMs in the set $SonC_i$ that is hosted in the CSP.

### 4.2 Conflict-of-Interest Relationship

A *CoI* relation is defined as neither an equivalence nor a binary relationship, i.e.:

- Not reflexive: if ($te_i \neg CoI\ te_i$)
  $\Rightarrow$ $te_i$ cannot be in conflict with itself.
- Symmetric: if ($te_i\ CoI\ te_j$) the reverse is also true i.e. ( $te_j\ CoI\ te_i$)
  $\Rightarrow$ that the *CoI* relation is mutually inclusive. If $te_i$ is in conflict with $te_j$ then $te_j$ is in conflict with $te_i$.
- Not transitive: if ($te_i\ CoI\ te_j\ CoI\ te_k$)
  $\Rightarrow \neg$ ($te_i\ CoI\ te_k$). If $te_i$ is in conflict with $te_j$ who is in conflict with $te_k$, it does not imply that $te_i$ is in conflict with $te_k$.

Therefore, this is neither an equivalence nor a binary relationship.

### 4.3 Risk, Physical Security and Cost

The proposed model recognizes that tenants can have different degrees of conflict with each other. For this reason, the risk (*R*) of confidential data leakage posed by inter-VM attacks also varies depending on the degree of conflict. For example, co-residence of VMs that belong to two competing tenants that are in direct conflict of interest with one another, poses the highest risk, *R*.

A physical separation, *S* is defined for implementing *CTL*. This indicates how much isolation a tenant requires between its VMs and that of its co-resident competitors. The separation is implemented in terms of co-reside in the same PN or reside in a different PN within the same Clu; in the same Clu but different DC; in the same DC but different Loc.

This is followed by a cost value, *C* for implementing a *CTL*. C is not necessarily given in monetary value like the United States Dollar or South African Rand. It is just a metaphorical figure used to enable the modelling. For each $SoC_i$ a cut-off point $r_i$ is defined beyond which conflict of interest is to be regarded by a tenant $te_i$ as minimal or insignificant. This defines a starting point of $SonC_i$ for $te_i$. Ideally, these could be the only tenants that $te_i$ would prefer to be co-resident with. However, CSPs are likely to charge a high cost, *C* for non-conflict placement of tenants' VMs and relative low *C* for flexible co-resident placement with conflicting tenants.

### 4.4 Utility Function *U*

CBAC4C introduces a global utility function, the dependent variable *U* for combining *S*, *R* and *C*. *U* computes the overall physical separation of tenants' VMs using a conflict-aware VM placement algorithms. *U* uses the *SoC* and *SonC* sets as derived from a tenant's inputs of *CTL* and *FBD*. The output is a set (i.e. either *SoC* or *SonC*) of all possible VM placements.

The CBAC4C model delivers a reduced set of VM placements using a best-fit algorithm which is also embedded in the placement algorithms. The result is used by a CSP to identify all potential nodes where a tenant's VM could be placed whilst preserving confidentiality and managing potential placement conflicts in a transparent manner. Tenants can then make informed decisions on where their VMs could be placed by the CSP and at what cost, risk and security. Final placement also requires a tenant's approval before it can be done. This feature gives tenants the opportunity to decline placements that do not comply with their own requirements.

### 4.5 The Objective Function

The main objective from a potential tenant's perspective is to minimise *R* and *C* while maximising *S*, in other words to follow a weighted sum multi-objective optimisation approach (Branke et al. 2008;

Fox et al. 2019; Gunantara and Ai 2018). This approach is used to simultaneously quantify trade-offs in satisfying different conflicting objectives (i.e. minimise *R* and *C* whilst maximising *S*) with weights that are proportional to their relative importance. The aim is to maximise *U* of physically separating tenants' VMs on the CSP's infrastructure. The main goal is to address the data leakage threats posed by inter-VM attacks on the Cloud; subject to a number of constraints, i.e. resource capacity, and *CTL*, which indirectly implies cost and risk appetite.

In this context, the VM placement problem is herein framed as a nested multi-dimensional variable vector-size bin-packing problem (Hatzopoulos et al. 2013; Wu et al. 2014). A nested multi-dimensional variable vector-size bin-packing problem is formulated as a vector of Locs to DCs to Clus to PNs and VMs bins.

The proposed solution has the following binary decision variables $x_{ij}(t)$, $y_{jk}(t)$, $z_{kl}(t)$ and $w_{lm}(t)$. $x_{ij}(t)$ denotes that $VM_i$ is placed on $PN_j(t)$; $y_{jk}(t)$ denotes that $PN_j$ is placed in $Clu_k(t)$, $z_{kl}(t)$ denotes that $Clu_k$ is place in a $DC_l(t)$; and $w_{lm}(t)$ denotes that $DC_l$ is placed in $Loc_m(t)$ at a particular time *t*. Each decision variable is associated with three coefficient weight vectors, i.e.

$\alpha_{i1}=\{\alpha_{11}, \alpha_{21}, \alpha_{31}, \alpha_{41}\}$ $\forall i = \{1,..,4\}$ relates to the Security (S) vector.

$\beta_{j2}=\{\beta_{12}, \beta_{22}, \beta_{32}, \beta_{42}\}$ $\forall j = \{1,..,4\}$ relates to the Cost (C) vector.

$\gamma_{k3}=\{\gamma_{13}, \gamma_{23}, \gamma_{33}, \gamma_{43}\}$ $\forall k = \{1,..,4\}$ relates to the Risk (R) vector.

$\alpha_{i1}=\{\alpha_{11}, \alpha_{21}, \alpha_{31}, \alpha_{41}\}$, $\beta_{j2}=\{\beta_{12}, \beta_{22}, \beta_{32}, \beta_{42}\}$ and $\gamma_{k3}=\{\gamma_{13}, \gamma_{23}, \gamma_{33}, \gamma_{43}\}$ are the coefficient weights for each binary decision variable $x_{ij}(t)$, $y_{jk}(t)$, $z_{kl}(t)$ and $w_{lm}(t)$ respectively. For example, $x_{ij}(t)$ has a coefficient weight vector $\{\alpha_{11}, \beta_{12}, \gamma_{13}\}$ where $\alpha_{11}$ is the coefficient weight in terms of physical separation security; $\beta_{12}$ is the coefficient weight in terms of cost and $\gamma_{13}$ is the coefficient weight in terms of risk posed on a $VM_i$ being placed in $PN_j$. $y_{jk}(t)$ has a coefficient weight vector $\{\alpha_{21}, \beta_{22}, \gamma_{23}\}$ where $\alpha_{21}$ is the coefficient weight in terms of physical separation security; $\beta_{22}$ is the coefficient weight in terms of cost and $\gamma_{23}$ is the coefficient weight in terms of risk posed on a $PN_j$ being placed in $Clu_k$. Furthermore, $z_{kl}(t)$ has a coefficient weight vector $\{\alpha_{31}, \beta_{32}, \gamma_{33}\}$ where $\alpha_{31}$ is the coefficient weight in terms of physical separation security; $\beta_{32}$ is the coefficient weight in terms of cost and $\gamma_{33}$ is the coefficient weight in terms of risk posed on a $Clu_k$ being placed in $DC_l$. The same applies for $w_{lm}(t)$'s coefficient vector $DC_l$ being placed in $Loc_m$.

Hence, the mathematical problem formulation (multi-objective function) (Jangiti, Vijayakumar and Subramaniyaswamy 2020) is as follows:

$$\boldsymbol{Max}\ U(S,C,R) = \sum_{i=1}^{m} \begin{pmatrix} \alpha_{11} \\ \beta_{12} \\ \gamma_{13} \end{pmatrix} x_{ij} + \sum_{j=1}^{n} \begin{pmatrix} \alpha_{21} \\ \beta_{22} \\ \gamma_{23} \end{pmatrix} y_{jk} + \sum_{k=1}^{o} \begin{pmatrix} \alpha_{31} \\ \beta_{32} \\ \gamma_{33} \end{pmatrix} z_{kl} + \sum_{l=1}^{p} \begin{pmatrix} \alpha_{41} \\ \beta_{42} \\ \gamma_{43} \end{pmatrix} w_{lm} \qquad (1)$$

subject to the following constraints:

$$\sum_{i=1}^{m} \sum_{j=1}^{n} \alpha_{ij}.x_{ij}(t) \leq CTL_i(t) \qquad (2)$$

$$\sum_{i=1}^{m} \sum_{j=1}^{n} \beta_{ij}.x_{ij}(t) \leq CTL_i(t) \qquad (3)$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n}\gamma_{ij}.x_{ij}(t) \le CTL_i(t) \quad (4)$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n}s_{ij}.x_{ij}(t) \le c_j(t)$$

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad \forall i \in \{1,\ldots,m\} \quad (5)$$

$$\sum_{j=1}^{n}\sum_{k=1}^{o}s_{jk}.y_{jk}(t) \le c_k(t)$$

$$\sum_{j=1}^{n} y_{jk} = 1 \qquad \forall j \in \{1,\ldots,n\} \quad (6)$$

$$\sum_{k=1}^{o}\sum_{l=1}^{p}s_{kl}.z_{kl}(t) \le c_l(t)$$

$$\sum_{k=1}^{o} z_{kl} = 1 \qquad \forall k \in \{1,\ldots,o\} \quad (7)$$

$$\sum_{k=1}^{o}\sum_{l=1}^{p}s_{lm}.w_{lm}(t) \le c_m(t)$$

$$\sum_{k=1}^{o} w_{lm} = 1 \qquad \forall l \in \{1,\ldots p\} \quad (8)$$

$$x_{ij}(t) \in \{0,1\} \qquad \forall i \in \{1,\ldots,m\},\ \forall j \in \{1,\ldots,n\} \quad (9)$$

$$y_{jk}(t) \in \{0,1\} \qquad \forall j \in \{1,\ldots,n\},\ \forall k \in \{1,\ldots,o\} \quad (10)$$

$$z_{kl}(t) \in \{0,1\} \qquad \forall k \in \{1,\ldots,o\},\ \forall l \in \{1,\ldots,p\} \quad (11)$$

$$w_{lm}(t) ) \in \{0,1\} \qquad \forall l \in \{1,\ldots,p\},\ \forall m \in \{1,\ldots,q\} \quad (12)$$

The multi-objective function (1) maximises $U$ of placing each $VM_i$ in a Physical Node, $PN_j$ ($x_{ij}$); a $PN_j$ in a Cluster, $Clu_k$ ($y_{jk}$); and a $Clu_k$ in a Data Centre, $DC_l$ ($z_{kl}$); and a $DC_l$ in a Location, $Loc_m$ ($w_{lm}$) of a CSP. This ensures that placement of tenant VMs is done based on their conflict tolerance levels (i.e. constraints (2) to (4)), and resource capacity (i.e. constraints (5) – (8)). Resource capacity constraints ensure that $VM_i$ capacity must not exceed the capacity $c_j(t)$ of $PN_j$, $c_k(t)$ of $Clu_k$, $c_l(t)$ of $DC_l$ and $c_m(t)$ of $Loc_m$. The binary decision variables $x_{ij}(t)$, $y_{jk}(t)$, $z_{kl}(t)$ and $w_{lm}(t)$ denote that $VM_i$ is placed on $PN_j$, which is placed in $Clu_k$ within a $DC_l$ that is also placed at $Loc_m$ at time $t$. Constraints (9) – (12) ensure that the decision variables $x_{ij}(t)$, $y_{jk}(t)$, $z_{kl}(t)$ and $w_{lm}(t)$ can either take the value one, which indicates placement, and otherwise zero for all VMs to be placed in PNs, for all PNs to be placed in Clus, for all Clus to be placed in DCs and for all DCs to be placed in Locs.

## 5. CBAC4C Conflict-Aware VM Placement Architecture

Illustrated in Figure 2 is an architectural diagram that shows how the mathematical CBAC4C model has been implemented and how VM placement is done in our architecture. It starts off with a prospective tenant requesting to host their data in a VM from a CSP. Depending on the size of the dataset, the CSP takes the request, creates a VM and allocates the necessary resources as specified by the prospective tenant. The CSP then uses the CBAC4C conflict-aware placement solution to select an appropriate PN within an appropriate Clu held in an appropriate DC within an appropriate Loc which then ensures optimal or non-optimal physical separation of VMs from conflicting tenants.

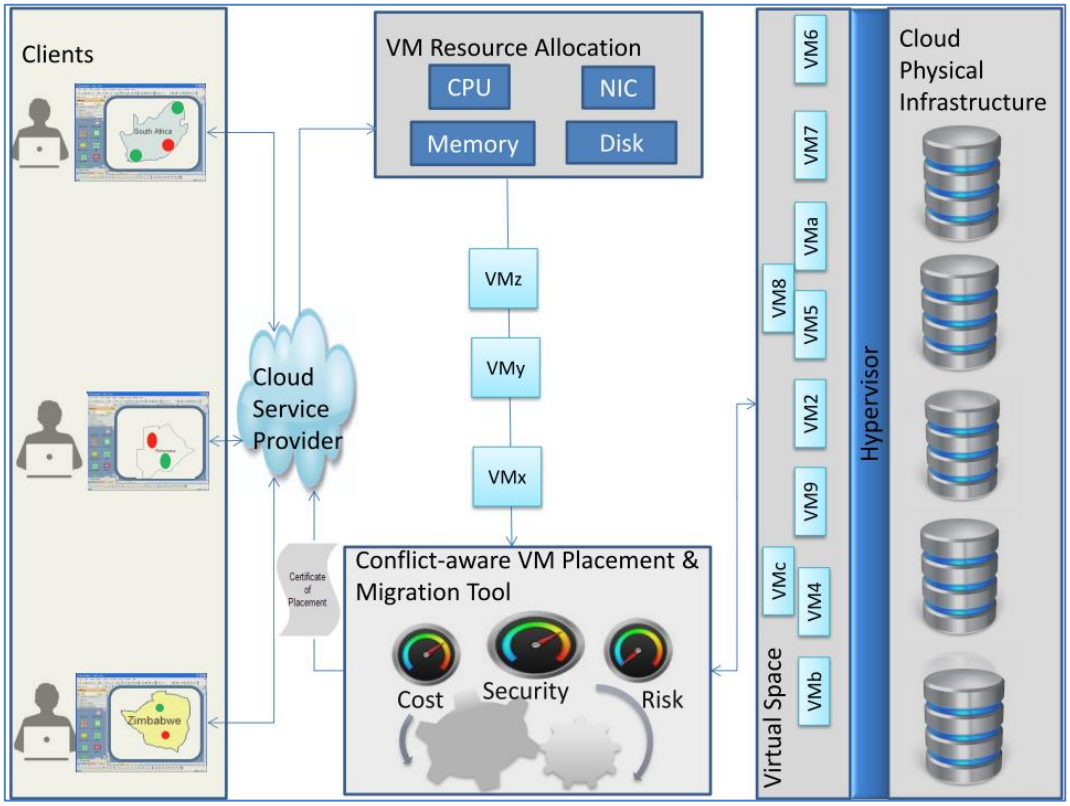


**Fig 2** CBAC4C Conflict-aware VM Placement Architecture

The conflict-aware VM placement algorithms within this solution also makes use of the best-fit heuristic algorithm (Lin and Kernighan 1978; Burke, Kendall and Whitwell 2009) to choose the best fitting placement. The choice of best-fit heuristic algorithm is made based on the fact that it is known to offer a near-optimal solution in a reasonable time (Jangiti et al. 2020). In the context of this research, this algorithm ensures that VMs are placed where they fit best to avoid underutilization of resources. However, placing a VM using the best-fit heuristic algorithm might mean that there is not much room for growth.

A placement matrix is created from which a tenant is allowed to choose their best location based on the guidance of the solution. A tenant may choose their VM placement or migration based on jurisdiction and data protection regulatory compliance mandates. For example, it may be illegal for some tenants to host their VMs in certain jurisdictions. This is where the solution offers some guidance and ensures that the final decision to place or migrate VMs is informed by a tenant's regulatory requirements and jurisdictional mandates. In order to guarantee tenant-CSP confidentiality, the placement matrix does not reveal or leak any data about the identity of other co-resident tenants' VMs. Such information is only visible to the CSP and not in any way revealed to the existing tenants or prospective ones. On the approved tenant's choice, the solution creates a GPS point to mark the geographical location of the

newly created VM on a cartographical map. A certificate of placement with the full address and allocated resources of the VM is issued and sent back to the tenant. The cartographical map on the tenant's side gets updated automatically with the new location of the new VM. It must be explicitly stated that tenants can only see the location of their own VMs on the cartographic map and not those of others.

The architecture in Figure 2 is implemented using the following four algorithms – the first two algorithms are for initial VM placement and the last two are for migrating existing VMs. Algorithm 1 below considers a new tenant whose VM is to be placed for the first time with a requirement to be only placed with non-conflicting tenants.

**Algorithm 1 Optimal VM Placement Algorithm**

**Require:**
$te_{id}$, *FBD, CTL, sizeOfVM, p_Type*
**Ensure:**
*SonC*{$te_o$,... $te_z$}
**if** (*p_Type* = *optimal*) **then**
  *SonC*{} ← {$te_o$,… $te_z$}
  **for all** $te_i \in$ *SonC*{$te_o$,… $te_z$} **do**
    allNCAddresses{} ← {$te_o$,… $te_z$}
    $i \leftarrow 0$
    **for all** $addr_i \in$ allNCAddresses{$te_o$,… $te_z$}**do**
      **if** ($addr_i$) $\geq$ *sizeOfVM* **then**
        sufNCAddresses{} ← $addr_i$
      **end if**
      $i \leftarrow i + 1$
      **for all** $addr_i \in$ sufNCAddresses{$te_o$,… $te_{z-n}$} **do**
        **Find** bestFitAddress **(**$addr_i$**)**
        **Place** $vm_{ix}$ in bestFitAddress **(**$addr_i$**)**
        **Return** $vm_{ix}$. $addr_i$
        **Update** AllocPlaceMatrix ($vm_{ix}$.$addr_i$)
        **Update** visibility.Map ($vm_{ix}$.$addr_i$)
      **end for**
    **end for**
  **end for**
**end if**

Algorithm 1: Placing a New VM in an optimal manner

For this algorithm a tenant $te_i$ makes a request to host its data on a newly allocated VM. The CSP checks the tenant's ID ($te_{id}$), and gets the *FBD*, *CTL* and size of the VM of the tenant. Based on the *CTL*, and the type of placement, which is optimal in this case, the CSP then provides a list of all existing tenants that are not in conflict with the prospective tenant. This is referred to as a *SonC* set. The requesting tenant cannot see this list. This is a set of all tenants that are not in conflict with the prospective tenant. The next step is to locate and list all addresses of VMs owned by each of these tenants. From this list of addresses (i.e. Loc.DC.Clu.PN), Algorithm 1 determines if the host (i.e. PN) has sufficient space to host the tenant's VM.

The solution then applies a best-fit algorithm (in the function bestFitAddress ($addr_i$)) to further determine a set of addresses to best place the new VM. These are addresses with space that is at best

the same as the size of the prospective VM to be placed. Once the best potential host is found, the algorithm places the VM on the approval of the prospective tenant. Our solution then sends the address of the new VM placement to the tenant.

It then updates the placement matrix (in the AllocPlaceMatrix ($vm_{ix}.addr_i$) function). This is basically a list or matrix of all the tenant's VM placements within the CSP's infrastructure; and cartographic map. The cartographic map reads from the placement matrix to then visualise (using the visibility.Map ($vm_{ix}.addr_i$) function) and display the information with respect to geographical location. The best-fit algorithm, placement matrix and cartographic map are applied in a similar manner across the rest of the algorithms.

Algorithm 2 considers a new tenant whose VM is to be placed for the first time – with a requirement to co-reside with conflicting tenants of a specified *CTL*. Tenant $te_i$ requests placement of its new VM and specifies a certain *CTL*. The specified *CTL* is less restrictive and allows some level of flexibility for the tenant to co-host with some conflicting tenants. The CSP checks the tenant's *ID* ($te_{id}$) (note that $te_{id}$ is used in place of $te_i\_ID$), and gets the *FBD*, *CTL* and size of the VM of the tenant.

**Algorithm 2 Non-optimal VM Placement Algorithm**

**Require:**
$te_{id}$, *FBD, CTL, sizeOfVM, p_Type*
**Ensure:**
*SoC*{$te_1$,... $te_n$}
**if** (*p_Type* = *non-optimal*) **then**
    *SoC*{} ← {$te_1$,… $te_n$}
    **for all** $te_i \in$ *SoC*{$te_1$,… $te_n$} **do**
        allCAddresses{} ← {$te_1$,… $te_n$}
        $i \leftarrow 1$
        **for all** $addr_i \in$ allCAddresses{$te_1$,… $te_n$}**do**
            **if** ($addr_i$) $\geq$ *sizeOfVM* **then**
                sufCAddresses{} ← $addr_i$
            **end if**
            $i \leftarrow i + 1$
            **for all** $addr_i \in$ sufCAddresses{$te_1$,… $te_{n-p}$} **do**
                **Find** bestFitAddress **(**$addr_i$**)**
                **Place** $vm_{ix}$ in bestFitAddress **(**$addr_i$**)**
                **Return** $vm_{ix}$. $addr_i$
                **Update** AllocPlaceMatrix ($vm_{ix}.addr_i$)
                **Update** visibility.Map ($vm_{ix}.addr_i$)
            **end for**
        **end for**
    **end for**
**end if**

Algorithm 2: Placing a New VM in a non-optimal manner

Based on the *CTL*, the CSP then provide a list of all existing tenants that are within the specified *CTL* with the prospective tenant. This is called a *SoC* set; a set containing all tenants that are in conflict with $te_i$. From this set, Algorithm 2 then chooses all addresses of tenants that are within the *CTL* range specified by $te_i$ and subsequently reduces the list to only those addresses with sufficient storage space. This algorithm helps the prospective tenant to choose the one address that best fits the new VM. On the

tenant's approval, Algorithm 2 next places the new VM on the best potential host, sends the address of the new VM placement to the tenant, and updates the placement matrix and cartographic map.

For brevity and readability, the next two algorithms have been deliberately left out. Due to their similarity to the first two algorithms, and to avoid duplication, it suffices just to explain the differences. The first two algorithms are used to perform an initial VM placement, yet the next two i.e. algorithm 3 and 4 are used to migrate an existing tenant's VM from one existing host to another. Algorithm 3 considers an existing tenant whose VM is to be migrated from one PN to another PN. The VM is to be migrated to a new PN where it can co-reside with non-conflicting tenants only. This algorithm handles such issues in a similar manner to Algorithm 1. The only difference is that Algorithm 3 can only be used by a CSP to migrate an existing VM from one address to another instead of a new placement. This implies that when the VM has been migrated, the old entry must be removed and the placement matrix be updated with the new address. Otherwise, all steps are carried out exactly as in algorithm 1. There are numerous reasons that could necessitate this, such as load balancing, changing tenant requirements and operational cost optimisation purposes. Finally, algorithm 4 considers an existing tenant whose VM is to be migrated from one PN to another – with a requirement to co-reside with conflicting tenants of a specified $CTL_{i+1}$ that is different from the initial $CTL_i$ in the new PN. This algorithm is similar to Algorithm 2. However, Algorithm 4 can be used by a CSP to migrate an existing VM from one existing address to another. Similar to algorithm 3, algorithm 4 has an extra step of replacing the old address from which the VM is with the new address.

The scenarios raised by these algorithms demonstrate the dynamic nature of today's organisations and the potential conflict involved. For example, a conflicting tenant today could be a non-conflicting tenant tomorrow. This could be due to a drop in the sensitivity of the data within an existing VM. VM with confidential data could be re-classified at some point later to a public classification. For example, a VM that holds blueprints of new products are mostly confidential and highly sensitive prior to the products' release date. Once the products have been released, such blueprints somehow lose their sensitivity and their risk changes to a lower level. Each of these scenarios demonstrate the scalability of CBAC4C in handling different tenants' requirements.

Overall, these algorithms are used for making initial VM placement and migration decisions. It must be noted though that placement and migration of VMs could also be initiated by the CSP for load-balancing purposes and other reasons. However, this must be done in a privacy-preserving, transparent and visible manner to the implicated tenants but should still conform to the requirements as spelt out by the respective tenants. The privacy element indicate that tenants are only able to see their own VM placement and migration.

The next section briefly discusses a proof-of-concept on how the authors implemented the proposed CBAC4C architecture and algorithms to prove our concept.

## 6. PROOF-OF-CONCEPT

The proof-of-concept demonstrates how each of the proposed algorithms was implemented to achieve the proposed conflict-aware VM placement solution. Figure 3 depicts a UML class model for the proof-of-concept classes. The model comprises of CSP, Tenant, Conflict_of_Interest, VM, Placement, Address, Position and GPS classes. The CSP uses each of these classes to effectively place and migrate a Tenant's VM on the Cloud.

A Tenant belonging to a specific *CoI* class has a VM associated with it. The VM is placed or migrated based on the Tenant's requirements (optimal or non-optimal based on their *CTL*) using the Placement class. The Placement class returns an Address of placement or migration that shows the global

positioning of a Tenant VM. A tenant can use this cartographical map to trace and track their VM placement at any particular point in time.

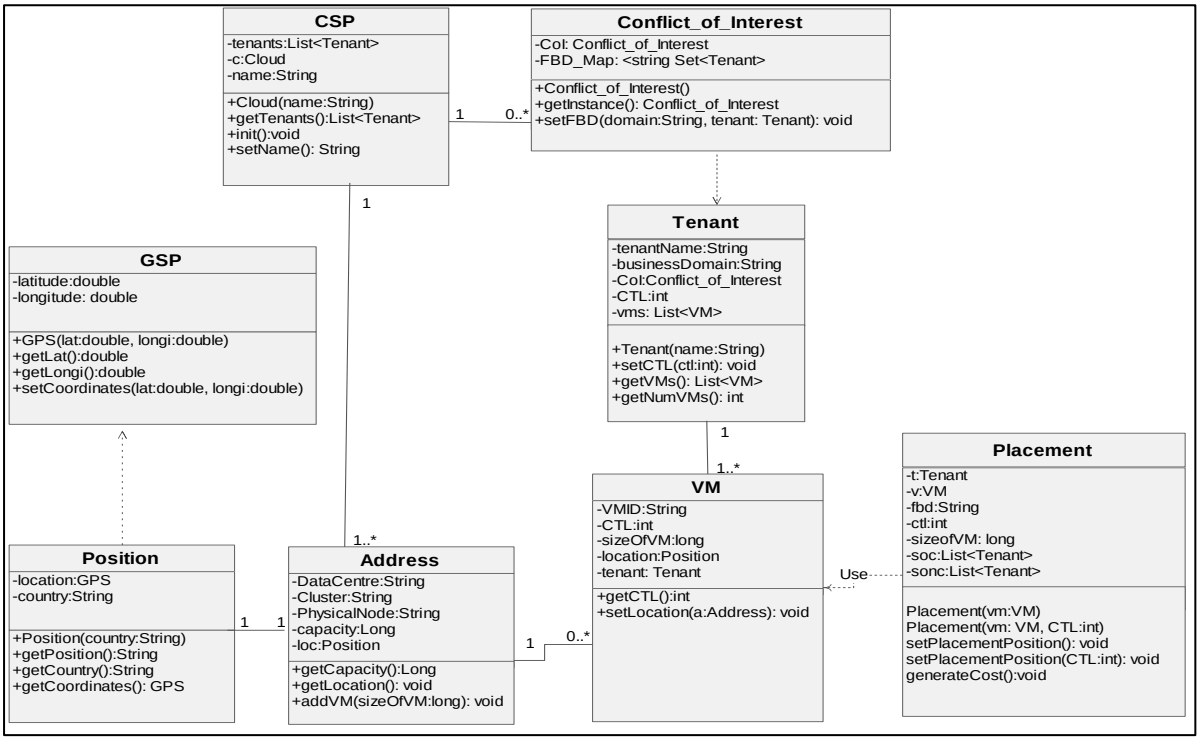


**Fig 3** UML Class Diagram for the Implementation of the Proposed Architecture

## 7. EVALUATION AND DISCUSSION OF CLOUDSIM EXPERIMENTAL RESULTS

A number of experiments were conducted to test and evaluate the effectiveness and efficiency of the algorithms of the proposed conflict-aware VM placement model. The authors defined a set of four VM instances, i.e. small, medium, large and extra-large for CloudSim – a cloud simulation tool (Barbierato et al. 2019). This is similar to the naming convention in Amazon Web Services IaaS. Table 1 shows the corresponding number of VM instances used in each of the three experiments on CloudSim, with a CTL of 0 for all VM placements.

**Table 1** VM Instances in CloudSim

| **Experiment No.** | **VM Instance Classes** | | | |
|---|---|---|---|---|
| | **Small** | **Medium** | **Large** | **Extra Large** |
| 1 | 10 | 15 | 30 | 50 |
| 2 | 15 | 30 | 50 | 75 |
| 3 | 30 | 75 | 115 | 150 |

The specification of the different VM instances is shown in Table 2.

**Table 2** Specification of VM Instances in CloudSim

| | **Specification of VM Instance Classes in CloudSim** | | | |
|---|---|---|---|---|
| | **Small** | **Medium** | **Large** | **Extra Large** |

| Storage (GB) | 2 | 5 | 10 | 30 |
|---|---|---|---|---|
| RAM (MB) | 128 | 512 | 768 | 1024 |
| No. of CPU | 1 | 1 | 1 | 1 |
| No. of vCPU | 1 | 1 | 1 | 1 |

The results of running these experiments are shown in Figure 4. The results show the time it took to place each of the VM instances in CloudSim. For all VM instances, the CTL was kept at zero, which means they were placed wherever there was sufficient space. Figure 4 reflects on the results of the three experiments.

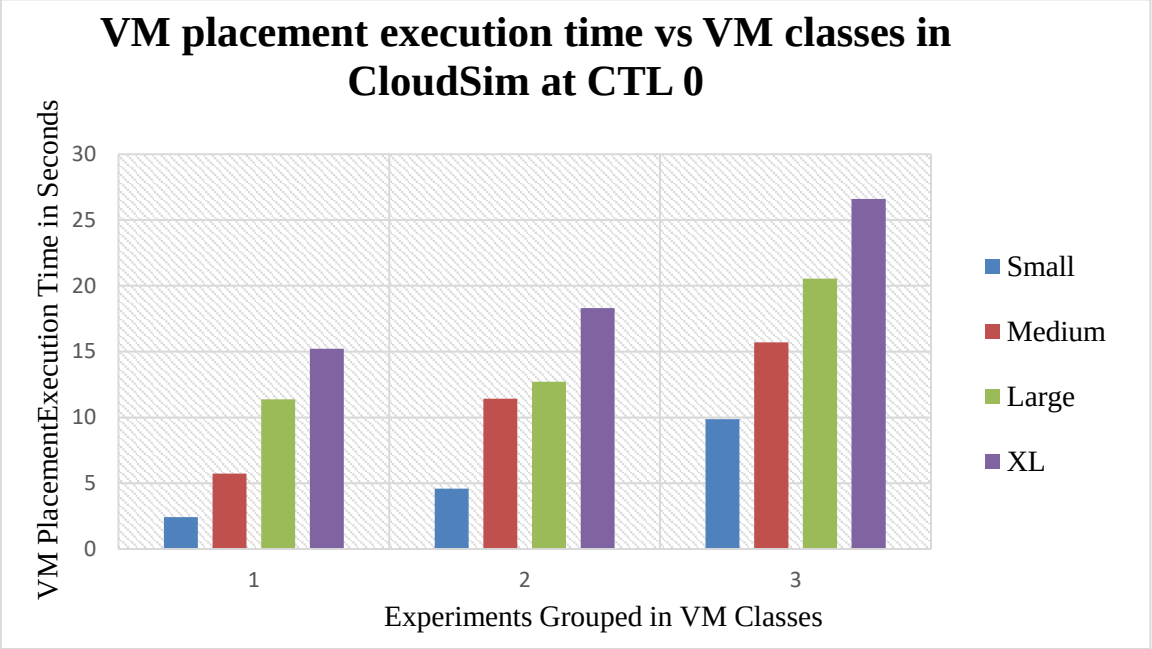


**Fig 4** VM Placement Performance across the Different VM Classes in CloudSim

Given the constant CTL in all three experiments, the results reflect a linear increase across the different types of VM instances. Furthermore, small VM instances appeared to take less time to be placed, compared to the other classes of instances. The author consequently deduced that the number and size of the VM instances had an effect on the VM placement performance. At this point it is important to note that the increase in placement times related to the number and size of VM instances. However, in Experiment 2, a deviation occurred from the normal linear increase. This is where it took about 13 seconds to place 50 large instances, which is almost the same time it took to place 30 medium instances in the same experiment. This is one of the rare occurrences which might be attributed to some external factor that could not be determined.

Figures 5, 6 and 7 reflect on the results of re-running the experiment with different CTLs ranging from zero to four and an increasing number of VMs. This is done as a continuation from the previous three experiments where the CTL was constant and set to be zero.

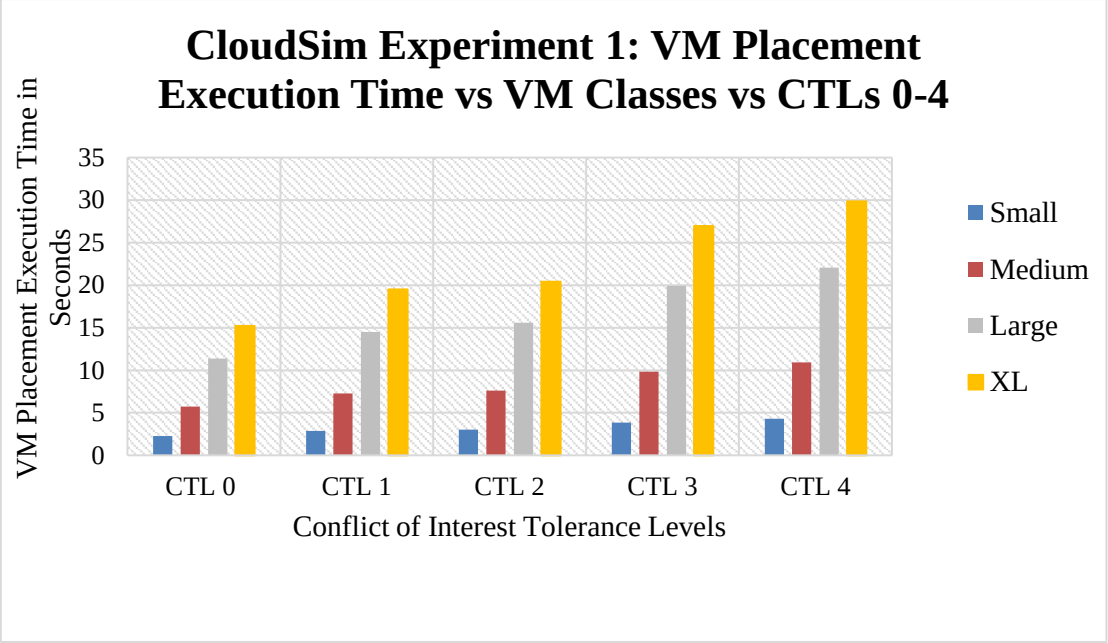


**Fig 5** Experiment 1: Results of VM Placement Execution Time against Different CTLs in CloudSim

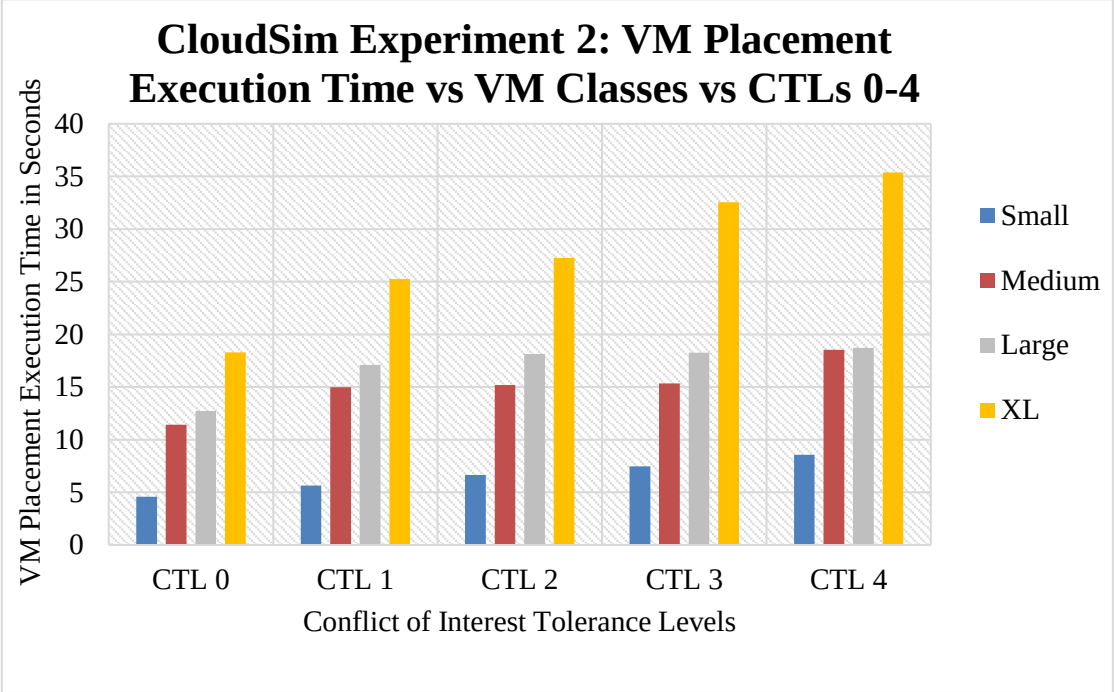


**Fig 6** Experiment 2: Results of VM Placement Execution Time against Different CTLs in CloudSim

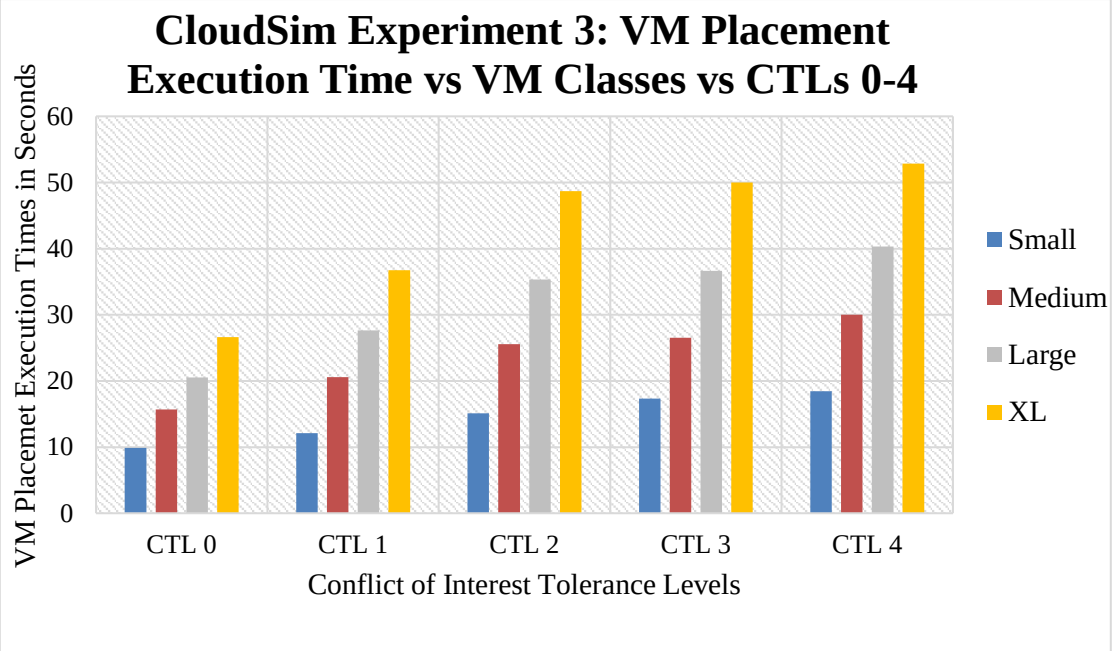


**Fig 7** Experiment 3: Results of VM Placement Execution Time against Different CTLs in CloudSim

From these three figures it can be deduced that the VM placement execution time increased linearly with a linear increase in CTL. For an example, an extra-large VM instance of CTL zero took approximately half the time to place, compared to an exact same size with a CTL of 4. This means that it took more time to place VMs of high CTL than those of a lower one. In general, the results illustrate that the CTL of a VM had a direct impact on the time it took to place a VM in CloudSim. The higher the CTL of the VM instance to be placed, the more time it took to do the actual placement.

Furthermore, it follows that the bigger the size of the VM instances that were to be placed, the more time it took to do the VM placement. Therefore, the authors conclude that the introduction of the proposed CBAC4C model introduced a time lag in the time it took to place VM instances in CloudSim. Based on the results, the authors argues that it was better to allocate small VMs of zero CTL than large ones of higher CTLs. Furthermore, the author deduced that distributing tenants' data in many small VMs, even with higher CTLs, would also improve the resilience of the cloud, more especially when other hardware drives fail. The tenants would still be able to retrieve parts of their data elsewhere. However, this approach might have a negative impact on data retrieval. Retrieving data from several distributed small VMs might introduce time delays. This could be more of a challenge if the data needs to be assembled in a sequential order at the tenant's side. An inherent challenge to having many small VMs distributed across the cloud infrastructure is that with more tenants coming on board, it would become very difficult to find non-conflicting slots. However, this problem can be addressed by a conflict-aware load balancing. Since conflict-aware load balancing falls outside the scope of this research, it is recommended as future work.

In concluding this section, the simulated conflict-aware VM placement provides good guidance on what to expect in a real cloud environment. However, cloud simulations cannot be relied upon for a true reflection of what could happen on the real cloud infrastructure. Therefore, it is important that future research also evaluates the proposed model on a real cloud infrastructure. The next section concludes the paper and points out a direction for future work.

## 8. CONCLUSION

The VM placement problem has direct impact on security, costs, performance and energy consumption and other computational aspects. This paper focused on the security of Cloud-hosted data which may be leaked to unintended parties through an inter-VM attack. This paper reports on the formulation of a CBAC4C model to prevent confidential data leakage threats posed by inter-VM attacks and manage conflict of interest between tenants on the Cloud. The CBAC4C model as the main contribution of this paper uses varying degrees of conflict, the construct of Sphere-of-Conflict and Sphere-of-non-Conflict to provide for the physical separation of VMs belonging to conflicting tenants. Unlike most VM placement algorithms, this work algorithms consider a tenant's risk appetite and cost implication of VM placement. CSPs can use our model, algorithms and architecture to make informed VM placement decisions that factor in their tenants' security profile – balanced against cost constraints and risk appetite. CSPs can also use the proof-of-concept class diagram to validate the CBAC4C model before they can implement our algorithms in their operational environments. In order to strengthen the scientific foundation of this paper, it consists of the experimentation on a simulated cloud environment. Future work is still required on a real cloud infrastructure to get more detailed empirical performance analysis of our proposed CBAC4C model's algorithms in comparison with other VM placement algorithms. Future work will also include formal proofs of the model and resultant algorithms. Whilst the current paper has focused on addressing inter-VM attacks, from trusted tenants who are assumed to provide the correct identity and line of business, future work will extend the attack vector to include malicious actors using spoofed identities or incorrect line of business information to defeat the current model.


## ACKNOWLEDGEMENTS

The support of the University of Pretoria's Cyber-security and Big Data Science Research, the ICSA Research Group, the Institute for Corporate Citizenship of the University of South Africa and CSIR is acknowledged. Opinions expressed and conclusions arrived at are solely those of the authors and should not be attributed to neither the University of Pretoria nor the Institute for Corporate Citizenship of the University of South Africa nor CSIR.


## REFERENCES


Abelson, H., Anderson, R., Bellovin, S.M., Benaloh, J., Blaze, M., Whitfield, D., Gilmore, J., Green, M., Landau, S., Nuemann, P.G., Rivest , R.L., Schiller, J.I., Schneier, B., Specter, M. and Weitzner, D.J. (2015). Keys Under Doormats: Mandating Insecurity by Requiring Government Access to all Data and Communications. http://dspace.mit.edu/bitstream/handle/1721.1/97690/MIT-CSAIL-TR-2015-026.pdf?sequence=8. Accessed: 15 May 2020.

Ahmad, I. and Bakht, H. (2019). Security Challenges from Abuse of Cloud Service Threat, *International Journal of Computing and Digital Systems*, 8(1). DOI: http://dx.doi.org/10.12785/ijcds/080103

Auer, U., Hauck, R.-J., Schlar, U. and Driesen, V. (2019). Exchanging shared containers and adapting tenants in multi-tenancy database systems, *SAP SE, Germany*, USA Patent Application, Pub Number.: US 2019/0129991 A1, May 2, 2019.

Azar, Y., Kamara, S., Menache, I., Raykova, M. and Shepard, B. (2014). Co-Location-Resistant Clouds. *In* Proceedings of the 6th edition of the ACM Workshop on Cloud Computing Security *(CCSW '14). ACM*, New York, NY, USA, 9-20. DOI=10.1145/2664168.2664179

Barbierato, E., Gribaudo, M., Iacono, M. and Jakobik, A. (2019). Exploiting CloudSim in a ultiformalism modeling approach for cloud based systems, *Journal of Simulation Modeling Practice and Theory*, 93, 133 – 147. DOI: https://doi.org/10.1016/jsimpat.2018.09.018

Barrowclough, J.P. and Asif, R. (2018). Security Cloud Hypervisors: A Survey of the Threats, Vulnerabilities, and Countermeasures, *Security and Communication Networks, Wiley Hindawi*, 2018, 1 - 20. DOI: 10.1155/2018/168198

Bartók, D. and Mann, Z.A. (2015). A branch-and-bound approach to virtual machine placement, *3rd Scientific Symposium, Operating the Cloud*, Hasso Plattner Institute, Potsdam, Germany, November 3rd, 2015.

Bazm, M.-M., Lacoste, M., Südholt, M. and Menaud, J.-M. (2019). Isolation in Cloud computing infrastructures: new security challenges, *Annals of Telecommunications*, 74(3-4), 197 - 209. DOI: https://doi.org/10.1007/s12243-019-00703-z

Branke, J., Deb, K., Miettinen, K. Slowinski, R. (2008 Eds). Multiobjective Optimization – Interactive and Evolutionary Approaches, *Lecture Notes in Computer Science (LNCS) 5252, Springer-Verlag Berlin Heidelberg, Springer*, Germany. ISBN: 978-3-540-88907-6

Brewer, D.F.C. and Nash, M. (1989). The Chinese Wall Security Policy. *IEEE Symposium on Security and Privacy*, 206.

Burke, E.K., Kendall, G. and Whitwell, G. (2009). A simulated annealing enhancement of the best-fit heuristic for the orthogonal stock-cutting problem, *INFORMS Journal on Computing*, 21(3); 505-51.

Calcavecchia, N.M., Biran, O., Hadad, E. and Moatti, Y. (2012). VM Placement Strategies for Cloud Scenarios. *IEEE 5th International Conference on Cloud Computing (CLOUD)*, 24-29 June 2012; 852-859, DOI: 10.1109/CLOUD.2012.113

Dlamini, M. T., Eloff, J. H. P. and Eloff, M. M. (2014). CBAC4C: Conflict Based Allocation Control for Cloud, *The 9th International Conference for Internet Technology and Secured Transactions (ICITST-2014)*, London, UK, 2014, 447-448. DOI: 10.1109/ICITST.2014.7038854

Doung-Ba, T. Tran, T. Nguyen and B. Bose, (2018). A Dynamic Virtual Machine Placement and Migration Scheme for Data Centers, *IEEE Transactions on Services Computing*, 1 – 14. DOI: 10.1109/TSC.2018.2818208

Ferdaus, M.H., Murshed, M., Calheiros, R.N. and Buyya, R. (2017). An Algorithm for network and data-aware placement of multi-tier applications in Cloud data centers, *Journal of Network and Computer Applications*, 98(2017), 65 – 83. DOI: 10.1016/l.jnca.2017.09.009

Filho, M.C.S., Monteiro, C.C., Inacio, P.R.M. and Freire, M.M. (2018). Approaches for optimizing virtual machine placement and migration in Cloud environments: A survey, *Journal of Parallel and Distributed Computing*, 111(2018), 222 – 250. DOI: http://dx.doi.org/10.1016/j.jpdc.2017.08.010

Fox, A.D., Corne, D.W., Adame, C.G.M., Polton, J.A., Henry, L-A. and Roberts, J.M. (2019). An Efficient Multi-Objective Optimization Method for Use in the Design of Marine Protected Area Networks, *Fontiers in Marine Science*, 6, DOI: https://doi.org/10.3389/fmars.2019.00017

Garcia, A.G., Espert, I.B. and Garcia, V.H. (2014). SLA-driven dynamic Cloud resource management, *Future Generation Computer Systems Journal*, 31, 1 -11. DOI: 10.1016/j.future.2013.10.005

Garg, S.K., Toosi, A.N., Gopalaiyengar, S.K. and Buyya, R. (2014). SLA-based virtual machine management for heterogeneous workloads in a Cloud datacenter, *Journal of Network and Computer Applkications* 45, 108 – 120.

Goyal, N., Pandey, A.K., Gupta, S.K. and Pandey, R. (2019). Suppleness of Multi-Tenancy in Cloud Computing: Advantages, Privacy Issues and Risk Factors. *Proceedings of the International Conference on Sustainable Computing in Science, technology and Management (SUSCOM)*, Amity University Rajasthan, Jaipar – India, February 26-28, 2019. DOI: http:/dx.doi.org/10.2139/ssrn.3358249

Gunantara, N. and Ai, Q. (2018). A Review of Multi-objective Optimization: Methods and Its Applications, *Cogent Engineering*, 5(1), DIO: 10.1080/23311916.2018.1502242

Han, Y., Alpcan, T., Chan, J. and Leckie, C. (2013). Security Games for Virtual Machine Allocation in Cloud Computing, Decision and Game Theory for Security, *Lecture Notes in Computer Science Series, 2013; 99 – 118: Springer International Publishing*. DOI: 10.1007/978-3-319-02786-9_7

Han, Y., Alpcan, T., Chan, J. and Leckie, C. (2014). Virtual Machine Allocation Policies against Co-resident Attacks in Cloud Computing, *Proceeding IEEE Conference on Communications (ICC 2014)*, Sydney, NSW, Australia, 786 – 792. DOI: 10.1109/ICC.2014.6883415

Hatzopoulos, D., Koutsopoulos, I., Koutitas, G. and van Heddeghem, W. (2013). Dynamic virtual machine allocation in Cloud server facility systems with renewable energy sources, *2013 IEEE International Conference in Communications (ICC 2013)*, , 9-13 June 2013; 4217-4221. DOI: 10.1109/ICC.2013.6655225

Jangiti, S., Vijayakumar, V. and Subramaniyaswamy, V. (2020). Hybrid Best-Fit Heuristic for Energy Efficient Virtual machine Placement in the Cloud Data Centers, *EAI Endorsed Transactions on Energy Web*, Ghent, 7(26), 1 – 7. DOI: 10.4108/eai.13-7-2018.162689

Jeihoonian, M., Zanjani, M.K. and Gendreau, M. (2020). Dynamic reverse supply chain network design under uncertaunty: Mathematical modeling and solution algorithm, *International Transactions in Operational Research (ITOR)*, DOI: https://doi.org/10.1111/itor.12865

Kamran, B.N. (2018). QoS-aware VM placement and migration for hybrid Cloud infrastructure, *The Journal of Supercomputing*, 74(9), 4623 – 4646. DOI: https://doi.org/10.1007/s11227-017-2071-1

Karger, D. and Onak, K. (2007). Polynomial Approximation Schemes for Smoothed and Random Instances of Multidimensional Packing Problems. *In SODA* 2007; 7; 1207-1216.

Kumar, N. , Katta, V., Mishra, H. and Garg, H. (2014). Detection of Data Leakage in Cloud Computing Environment, *6th Internationa Conference on Coputational Intelligence and Communication Networks, IEEE Computer Society*, 803 – 807.

Lefray, A., Caron, E., Rouzaud-Cornabas, J. and Toinard, C. (2015). Microarchitecture-Aware Virtual Machine Placement under Information Leakage Constraints, *8th IEEE International Conference on Cloud Computing (IEEE Cloud 2015), IEEE Computer Society*, http://www.cloudcomputing.org/2015/, June 27 – July 2 2015, New York, United States, 588 – 595. DOI: 10.1109/CLOUD.2015.84

Lin, S. and Kernighan, B.W. (1978). An effective heuristic algorithm for the traveling-salesman problem, *Operations research*, 21(2): 498 – 516.

Loock, M. (2012). CBAC – A model for conflict-based access control, Thesis, *University of Pretoria*, South Africa.

Loock, M. and Eloff, J.H.P. (2005). A New Access Control Model based on the Chinese wall Security Policy Model,*Proceedings of the Information Security South Africa Conference (ISSA 2005)*, Sandton, South Africa, edited by HS Venter, L. Labuschagne, MM Eloff. ISBM: 1-86854-625-X

Martini, J. and Theiler, M.A. (2019). United States of America vs Paige A. Thompson, *United States District Court for the Westen District of Washington at Seattle*, Case No. MJ19-0344, United States Courthouse, Seattle, Washington

Mashayekhy, L. Nejad, M.M. and Grosu, D.A. (2014). Framework for Data Protection in Cloud Federation, *3rd International Conference on Parallel Distribution*, Minneapolis, 283-290. DOI: 10.1109/ICPP.2014.37

Miao, F., Wang, L. and Wu, Z. (2018). A VM Placement Based Approach to Proactively Mitigate Co-Resident Attacks in Cloud, *2018 IEEE Symposium on Computers and Communications (ISCC)*, 285 – 291. DOI: 10.1109/ISCC.2018.8538543

Moffitt, M.D. (2015). Multi-dimensional physical arrangement technique using bin-packing with per-branch combination tries, Patent US 8918750 B1. www.google.com/patents/US8918750#backward-citations. Accessed: 15 May 2020.

Mosola, N.N., Dlamini, M.T., Blackledge, J.M., Eloff, J.H.P. and Venter, H.S. (2017). Chaos-based Encryption Keys and Neural Key-store for Cloud-bound Data Confidentiality, *Proceeding of the 2017 Southern Africa Telecommunications and Networks Applications Conference (SATNAC 2017)*, Barcelona, Spain, Royal Caribbean International, September 3-10, 2017, 168 – 173.

Nanavati, M.S. (2019). Operator, Number Please: Mediating Access to Shared Resources for Efficience and Isolation, Thesis, *The University of British Columbia, Vancouver*, January 2019

Nejad, M.M., Mashayekhy, L. and Grosu, D. (2013). A Family of Truthful Greedy Mechanisms for Dynamic Virtual Machine Provisioning and Allocation in Clouds, *2013 IEEE Sixth International Conference on Cloud Computing (CLOUD)*, Santa Clara, CA, USA, 188-195. DOI: 10.1109/CLOUD.2013.14

Perez-Botero, D., Szefer, J. and Lee, R.B. (2013). Characterizing Hypervisor Vulnerabilities in Cloud Computing Servers, *In 2013 Proceedings of the Workshop on Security in Cloud Computing (SCC 2013)*, Hangzhou, China, 8 May 2013, 3 - 10. DOI: 10.1145/2484402.2484406

Ristenpart, T., Tromer, E., Shacham, H. and Savage, S. (2009). Hey you, get off of my Cloud: Exploring Information Leakage in Third-Party Compute Clouds, *CCS'09, Proceedings of the 16th ACM Conference on Computer and Communication Security*, Chicago, Illinois, USA, November 9-13, 2009, 199 - 212. DOI: 10.1145/1653662.1653687

Sailer, R., Jaeger, T., Valdez, E., C'aceres, R., Perez, R., Berger, S., Griffin, J.L. and van Doon, L. (2012). Building a MAC-based security architecture for the Xenopensource hypervisor. *Proceedings of the Annual Computer Security Applications Conference*, Tucson, AZ, USA. DOI: 10.1109/CSAC.2005.13

Si, Y., Wang, J., Feng, T. and Jianqiang, Z. (2013). A Security-awareness Virtual Machine Placement Scheme based on Chinese Wall Policy in Cloud Computing, *2013 IEEE 10th International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing, 13-15 Nov 2013*, Zhangjiajie, China, *The Scientific World Journal*, DOI: 10.1109/HPCC.and.EUC.2013.152

Stillwell, M., Vivien, F. and Casanova, H. (2012). Virtual Machine Resource Allocation for Service Hosting on Heterogeneous Distributed Platforms. *In* Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium *(IPDPS '12). IEEE Computer Society*, Washington, DC, USA, 2012; 786-797; DOI: 10.1109/IPDPS.2012.75

Szefer, J. and Lee, R.B. (2011). A Case for Hardware Protection of Guest VMs from Compromised Hypervisors in Cloud Computing, *in Proceeding of the Second International Workshop on Security and Privacy in Cloud Computing (SPCC 2011)*, Kyoto, Japan, June 2011; DOI: 10.1109/ICDCSW.2011.51

Tobin, P. (2018). On the Application of PSpice for Localised Cloud Security, Thesis, *Dublin Institute of Technology*, http://arrow.dit.ie/engdoc/111

Tsai, T., Chen, Y., Huang, H., Huang, P. and Chou, K. (2011).A Practical Chinese Wall Security Model in Cloud Computing. *The 13th Asia-Pacific Network Operations and Management Symposium (APNOMS): Managing Clouds, Smart Networks and Services*, Taipei, Taiwan, 1-4.

Wang, Z., Sun, K., Jajodia, S. and Jing, J. (2012). Disk Storage Isolation and Verification in Cloud, *Globecom – Communication and Information System Security Symposium*, Anaheim, CA, USA, 3-7 December 2012; 771-776. DOI: 101109/GLOCOM.2012.6503206

Wu, H., Ren, S., Timm, S., Garzoglio, G. and Noh, S.-Y. (2014). Overhead-Aware-Best-Fit (OABF) Resource Allocation Algorithm for Minimizing VM Launching Overhead, *4th International Workshop on Network-aware Data Management (in Cooperation with ACM SIGHPC; in conjunction with SC14: IEEE/ACM International Conference for High Performance Computing, Networking, Storage and Analysis), 7th Workshop on Many-Task Computing Clouds, Grids and Supercomputers (MTAGS 2014), Co-located with Supercomputing/SC* 2014.

Wu, R., Ahn, G-J., Hu, H. and Singhal, M. (2010). Information Flow Control in Cloud Computing, *6th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2010)*, 12 May 2011, 1 – 7.

Xing, L., Levitin, G. and Xiang, Y. (2019). Defending N-version Programming Service Components against Co-resident Attacks in IoT Cloud Systems, in *IEEE Transactions on Services Computing*. DOI: 10.1109/TSC.2019.2904958

Yadav, A.K., Bharti, R.K. and Raw, R.S. (2018). Security Solution to Prevent Data Leakage Over Multitenant Cloud Infrastructure, *International Journal of Pure and Applied Mathematics*, 118(7), 2018, 269 – 276.

Zaman, S. and Grosu, D. (2013). A Combinatorial Auction-Based Mechanism for Dynamic VM Provisioning and Allocation in Clouds, *IEEE Transactions on Cloud Computing*, July-December 2013, 1(2); 129 – 141. DOI: 10.1109/TCC.2013.9

Zhang, S. (2012). Deep-diving into an easily-overlooked Threat: Inter-VM Attacks. Whitepaper, *provided by Kansas State University, TechRepublic/US2012*. http://www.techrepublic.com/resource-library/whitepapers/deep-diving-into-an-easily-overlooked-threat-inter-vm-attacks/. Accessed: 16 August 2020.

Zhang, X., Liu, C., Nepal, S., Pandey, S. and Chen, J. (2013). A privacy leakage upper bound constraint-based approach for cost-effective privacy preserving of intermediate data sets in Cloud, *IEEE Transactions on Parallel and Distributed Systems*, 24(6), 1192 – 1202. DOI: 10.1109/TPDS.2012.238

Zhang, X., Yang, L.T., Liu, C. and Chen, J. (2014). A scalable two-phase top-down specialization approach for data anonymization using mapreduce on Cloud, *IEEE Transactions on Parallel and Distributed Systems*, 25(2), 363 -373. DOI: 10.1109/TPDS.2013.48