Enhanced Fuzzy Particle Swarm Optimization Load Distribution (EFPSO-LD) for DDOS Attacks Detection and Prevention in Healthcare Cloud Systems

A. Peter Soosai Anandaraj[1], G. Indumathi[2]

[1] Dept.of CSE, Ganapathy Chettiar College of Engineering and Technology, India
[2] Dept.of ECE, MepcoSchelenk Engineering College, India
anandsiriya@gmail.com, indupriyanga@gmail.com

Abstract

Distributed Denial of Service (DDoS) is an attack that threats the availability of the healthcare related cloud services. In order to assure the each and every one time accessibility of patient's data, propose a new solution that allows, firstly, the hypervisor to establish credible trust relationships among VMs by considering purpose and personal trust sources and employing vectors to aggregate them. Secondly Enhanced Fuzzy Particle Swarm Optimization (EFPSO) algorithm which guides the hypervisor to determine the optimal loads distribution among VMs in real-time that maximizes DDoS attacks' detection. EFPSO algorithm which allocates incoming client request to available virtual machines depending on the load i.e. VM with least work load is found and then new request is allocated in the attack detection. The proposed EFPSO algorithm gives the hypervisor with the optimal detection load distribution strategy over VMs that maximizes the detection of DDoS attacks under a limited budget of resources. At finally prevention is performed by using Convex Support Vector Machine (CSVM) classifier. Experimental results are measured in terms of attacks' detection, false positives, negatives, and CPU, memory during DDoS attacks.

Keywords: Cloud computing, Distributed Denial of Service (DDoS), Load distribution, Convex Support Vector Machine (CSVM) and attack detection

1 Introduction

Cloud Computing is an evolving paradigm that is growing rapidly and is a modern model that is intended to provide suitable, on-demand, network access to a common group of configurable computing resources "as a service" on the Internet for fulfilling computing demands of the users.

Wan et al. [1] states that "cloud computing is a new computing paradigm that is built on virtualization, distributed computing, utility computing and service-oriented architecture." Mell and Grance [2] have defined cloud computing as "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction." Security has been one of the most challenging issues for the IT executives particularly in cloud implementation. Several studies, including the one by Sangroya et al. [3] quote security as the primary level confront for cloud users.

Most of these questions are specifically related to data and business logic security [4]. As data and business logic is located on a remote cloud server with no transparent control, most security concerns are not similar to their earlier equivalents in non-cloud infrastructures. Distributed Denial of Service (DDoS) is a major threat to Internet based killer applications for non-cloud computing environments, such as independent news web sites, e-business and online games [5].

DDoS attacks are now carried out by botnets. Resent researches [6] have corrected a long held belief that hackers can easily compromise as many computers as they want.

A different flavor of DoS is Distributed DoS, or DDoS, where attackers are a group of machines targeting a particular service [7]. There is a high rise in the number of reported incidents of DDoS, which makes it one of the most important and fatal threat amongst many [8]. To solve this problem

Many Intrusion Detection Systems (IDSs) [9-10] have been advanced to identify intrusions in cloud environments, where most of these systems are developed and improved from traditional detection techniques used in non-cloud environments.

In the recent work, a trust-based maxim in game is designed for hypervisor and DDoS attackers. The timely detection and prevention of DDoS attacks and optimal load balancing in cloud becomes very difficult

*Corresponding Author: A. Peter Soosai Anandaraj, E-mail: anandsiriya@gmail.com
DOI: 10.3966/160792642020032102012

task.

Load balancing optimization techniques of evolutionary and swarm based algorithms which will help to overcome the optimization problems or resource utilization.

In this work, present a novel Enhanced Fuzzy Particle Swarm Optimization (EFPSO) algorithm which allocates incoming client request to available virtual machines depending on the load i.e. VM with least work load is found and then new request is allocated in the attack detection.

Finally IDSs DDOSs attack detection and prevention is performed by using Convex Support Vector Machine (CSVM) classifier.

2 Literature Review

Sattar et al. [11] proposed a new DDoS attack prevention and detection. DDoS attack is occurs when huge amount of data or packets are sent to a server from various computer. Confidence Based Filtering (CBF) packet filtering method is used to reduce the storage needs and increase the processing speed on the server side. Finally use various techniques detecting and preventing the DDoS attack in cloud computing system. To improve availability of resources, it is essential to provide a mechanism to prevent DDoS attacks.

Lonea et al. [12] focused on detecting and analyzing the DDoS attacks in cloud computing environments. Specifically, when the attacks appear, the VM-based IDS will yield alerts, which will be stored into the MYSQL database placed within the Cloud Fusion Unit (CFU) of the front-end server. Proposed solution uses the Dempsters combination rule to fuse evidence from multiple independent sources.

Jamali et al. [13] proposed a framework in which the defense issue is formulated as an optimization problem and employs the Particle Swarm Optimization (PSO) algorithm to optimally solve the attack problem. A DoS attack can be regarded as an attempt of attackers to prevent legal users from gaining a normal network service. The TCP connection management protocol sets a position for a classic DoS attack, namely, the SYN flood attack. In this attack some sources send a large number of TCP SYN segments, without completing the third handshake step to quickly exhaust connection resources of the under attack system.

Wahab et al. [14] proposed a two-fold solution that allows, firstly, the hypervisor to establish credible trust relationships toward guest VMs by considering objective and subjective trust sources and employing Bayesian inference to aggregate them.

On top of the trust model, it design a trust-based maxim in game solution which guides the hypervisor to determine the optimal detection load distribution among VMs in real-time that maximizes DDoS attacks' detection.

Wahab et al. [15] developed a resource-aware maxmin game theoretical model that guides the hypervisor on how the detection load should be optimally distributed among its guest VMs in the real-time.

Experimental results on Amazon Elastic Compute Cloud (EC2) pricing dataset reveal that model increases the probability of detecting distributed attacks, reduces the false positives, and minimizes the resources wasted during the detection process.

Yu et al. [16] proposed a dynamic resource allocation strategy to counter DDoS attacks against individual cloud customers. It establishes a mathematical model to approximate the needs of our resource investment based on queuing theory.

A cloud usually possesses profound resources and has full control and dynamic allocation capability of its resources. Therefore, cloud offers us the potential to overcome DDoS attacks. The individual cloud hosted servers are still vulnerable to DDoS attacks if they stop run in the traditional way.

Zargar et al. [17] explored the scope of the DDoS flooding attack problem and attempts to combat it. It categorizes the DDoS flooding attacks and classifies existing countermeasures based on where and when they prevent, detect, and respond to the DDoS flooding attacks. This primary intention for this work is to stimulate the research community into developing creative, effective, efficient, and comprehensive prevention, detection, and response mechanisms that address the DDoS flooding problem before, during and after an actual attack.

Liu et al. [18] have described to take only execution time into consideration when scheduling the cloud resources, it may occur serious load imbalance problem between VMs in Cloud Computing environments. In addition to solve this problem, the task scheduling model is proposed which optimizes the task execution time in view of both the task running time and the system resource utilization. Based on the model, a PSO based algorithm is proposed and introduces a simple mutation mechanism and a self-adapting inertia weight method by classifying the fitness values. Ramezani et al. [19] proposed a task-based System Load Balancing method using Particle Swarm Optimization (TBSLB-PSO) that achieves system load balancing by only transferring extra tasks from an overloaded VM instead of migrating the entire overloaded VM. To evaluate the proposed method, it extends the cloud simulator (Cloudsim) package and use PSO as its task scheduling model.

Lombardi and Di Pietro [20] proposed a virtualization-supported security architecture whose main purpose is to ensure the integrity of the VMs while being invisible to end users. To this end, an Interceptor entity is deployed into the kernel space of the host system to constantly monitor the VMs' system-call invocations. Thereafter, a Warning

Recorder entity registers the suspicious activities in a Warning Pool whose responsibility is to prioritize the evaluation order of these activities. The Warning Recorder derives checksums for code, data, and files and passes them to the Evaluator entity that inspects the activities and takes the appropriate decision on whether the system's security has been violated or not. In the review work, trust-based maxim in game is designed for hypervisor and DDoS attackers. The timely detection and prevention of DDoS attacks and optimal load balancing in cloud becomes very difficult task.

3 Proposed Methodology

Propose a new solution that allows, firstly, the hypervisor to establish credible trust relationships among VMs by considering personal trust sources and employing classifier to aggregate them. Secondly Enhanced Fuzzy Particle Swarm Optimization (EFPSO) algorithm which guides the hypervisor to determine the optimal detection load distribution among VMs in real-time that maximizes DDoS attacks' and TCP flood attacks detection. At finally prevention is performed by using Convex Support Vector Machine (CSVM) classifier. Attackers distribute their attacks over a set of malicious VMs to minimize the detection probability, while hypervisors distribute the detection load over the set of guest VMs to maximize this minimization.

Graphical formulation of the above-mentioned problem is given in Figure 1.

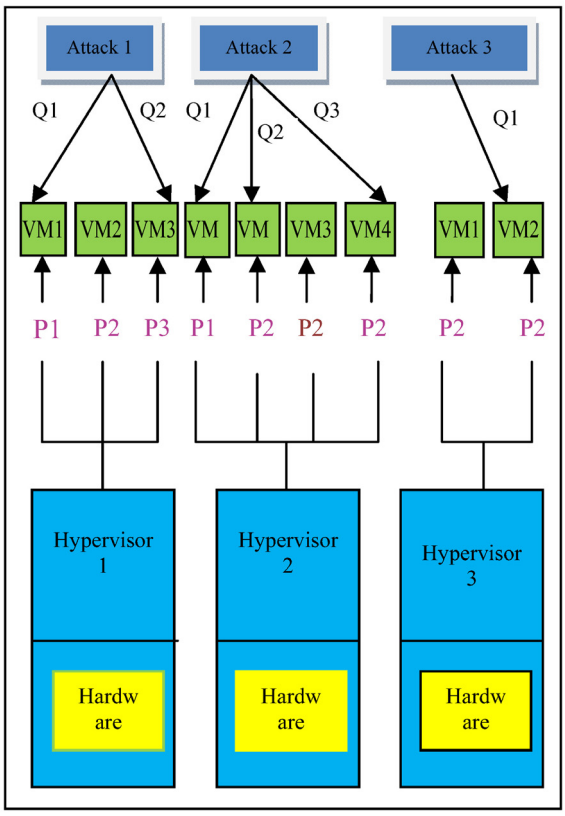

Figure 1. Attack scenario

In the proposed work, hypervisor monitors the VMs' CPU, memory, and network bandwidth consumption directly from the hosting infrastructure and applies the Interquartile Range (IQR) statistical measure to identify any abnormal usage. This constitutes the objective source of trust which is of prime importance in the field of trust and reputation to avoid biased and/or subjective judgments [21].

Cloud Model

Let $HY = (hy_1, ..., hy_n)$ be a finite set of hypervisors, where each hypervisor $hr_i \in H$ hosts a set of virtual machines $VM_1 = (vm_1, ..., vm_1)$. Note that when 'i' is not important or can be induced from the context, also simply use VM instead of VM_i. Each virtual machine $vm_j \in VM$ residing on hi is owned by a client from the set $CL = (Cl_1, ..., Cl_m)$. A hypervisor $hr_i \in H$ is a software agent that stays between the cloud system's hardware and the VMs and whose role is to emulate a set of hardware resources $I = (I_1, I_2, ..., I_n)$ and schedule the access of the VMs to it in order to enable the synchronous running of multiple VMs on a shared cloud infrastructure. A virtual machine is a pair <O, A>, where O represents the underlying Operating System (OS) and A denotes the set of applications running inside 'VM'.

As a first stage, the hypervisor seeks to establish trust relationships toward its guest VMs.

To do so, it first monitors and analyzes the CPU, memory, and network bandwidth utilization of each $vm \in VM$. This allows the hypervisor h to build an initial support vector machine in each VMs trustworthiness denoted as $ISVM_{hy}^{vm}$.

Each recommendation $R_s^{vm} \in [0,1]$ denotes a certain source s's recommendation on the behaviour of a VM based on their previous interactions.

Note that each source s enjoys a fixed number of inquiries it is allowed to make from every (other) hypervisor hy' and is denoted by Inq($s \to hy'$). Initially, all sources enjoy an equal amount of inquiries, where this amount is updated later during the trust establishment process. Now, the hypervisor 'hy' aggregates the results of the monitoring phase with the results of the recommendations phase using the classifier technique to come up with a final SVM $FSVM_{hy}^{vm}$ in each VMs trust worthiness.

4 Optimal Detection Load Distribution Strategy

In this section now proceed with developing the utility functions of both the hypervisor and DDoS attackers and introducing the new security based on

classifier.

The utility of a hypervisor hy quantifies its success in protecting the monitored virtual machines VM, of worth W(vm) each, inversely proportional to hy's in each vm's.

The utility function of hy at time t2+1 that comes after the considered window of time [t1, t2] is computed as follows:

$$U_{t_{2+1}}(hy) = \Sigma_{vm \in VM} \frac{W(vm) \times ADR_{[t_1, t_2]}}{ISVM_{hy}^{vm}} \quad (1)$$

where $W(vm)$ represents the worth of each virtual machine vm (e.g., price, criticality of the applications running on it), $ISVM_{hy}^{vm}$ denotes the SVM of hy in vm's, and $ADR_{[t_1, t_2]}$ is the average detection rate of the IDS agent running on h during the time window $[t_1, t_2]$ and is computed as per Eq. (2).

$$ADR_{[t_1, t_2]}(hy) = 1 - \sum_{x=t_1}^{t_2} \Sigma_{vm \in VM} \frac{(A_x(vm) - B_x(vm))}{t_1 - t_2}$$
$$\text{for each } A_x(vm) > B_x(vm) \quad (2)$$

Distributed over VM with probability $A_x(vm)$ with k attacks, optimal detection load probability distribution vector VM $B_x(vm)$. The results of the $A_x(vm)$ & $B_x(vm)$ us determined via the algorithms such as CSVM and EFPSO.

5 Enhanced Fuzzy Particle Swarm Optimization (EFPSO)

Based on a unique search characteristic of PSO, this study proposes an Enhanced Fuzzy Particle Swarm Optimization (EFPSO), which aims to filter the unnecessary structural analyses in the optimization process.

In the proposed work, the hypervisor monitors the VMs' CPU, memory, and network bandwidth consumption directly from the hosting infrastructure and applies the Interquartile Range (IQR) statistical measure [14]. Each particle monitors the hypervisor directly to record the information for optimal load detection.

The EFPSO considers a fuzzy categorization step into the usual PSO before evaluating the optimal load detection of the particles (hypervisors).

The hypervisor into two sets in each iteration step. EPSO only evaluates the optimal load violation (IQR) for the particles with enhanced objective values $(W(vm))$ compared to their corresponding personal best load distribution results for each VM, and thus reduces the number of constraint evaluations significantly.

Figure 2 illustrates a particle's flying trace in three consecutive iterations as solving a load distribution with VMs' CPU, memory, and network bandwidth consumption constraints.

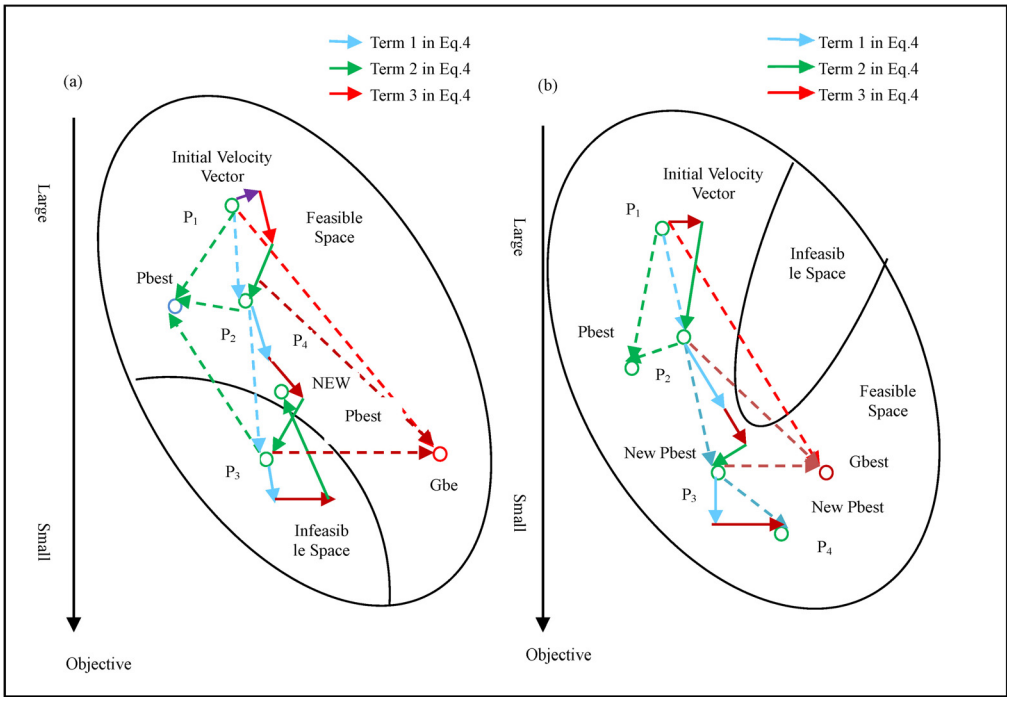

Figure 2. Particle swarm optimization position and velocity update in three iterations with two different constraints

The particle starts in the load distribution space at pos_1, goes through pos_2 and pos_3, and arrives at pos_4 after three iterations.

The arrows with solid lines are the three velocity

components as expressed in Equation (7) and those with dashed lines represent the direction of these velocity components. The particle's search rule depends only on the personal best (Pbest) value of load distribution results, the global best (Gbest) and the initial velocity in the previous iteration according to the procedure of PSO as prescribed in Equations (6) and (7).

As indicated in Figure 2, pos_2 is a position with a deteriorating load distribution value compared to its Pbest and locates in the load distribution space in Figure 2(a) and in the infeasible load distribution space in Figure 2(b).

Based on PSO's search rule, the two hypervisors in Figure 2(a) and Figure 2(b) have the same moving traces from pos_2 to pos_3 while neglecting the stochastic property of the PSO search rule.

The feasible load distribution space check for a hypervisor with a worse distribution space value than its current Pbest thus becomes redundant.

However, upon reaching pos_3, where their load distribution worth values are better than their present Pbest, PSO has to evaluate the load distribution violation to check whether pos_3 can replace the present Pbest.

Therefore, the load distribution violation evaluation depends only on the value of the hypervisors current objective $W(vm)$. From the above statement, a particle in PSO can be spitted into four states.

Figure 3 shows the four possible states of a hypervisor from iteration step k to k + 1. pos_1^{k+1} denotes the new position in the feasible load distribution space with an IQR value worse than the current Pbest. pos_2^{k+1} Indicates the new position in the infeasible load distribution space with an IQR value worse than the current Pbest, pos_3^{k+1} represents.

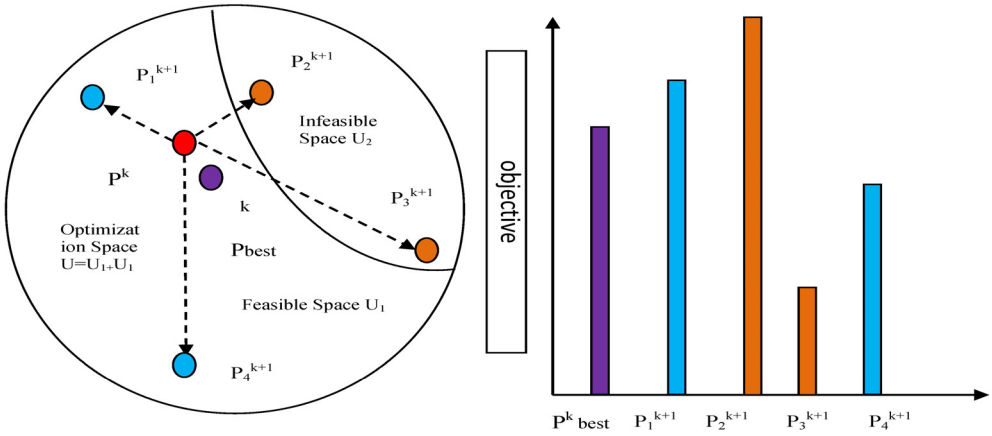

Figure 3. Four possible states of a particle after position update for a constrained problem

The new position in the feasible space with an objective value better than the current Pbest, and pos_4^{k+1} corresponds to the new position in the infeasible load distribution space with an objective value better than the current Pbest. According to the above, PSO only needs to evaluate the particles' load distribution violation condition for pos_3^{k+1} and pos_4^{k+1} before updating the Pbest. The feasibility of load among VM is verified via the use of hypervisor checks for pos_1^{k+1} and pos_2^{k+1} is redundant. Therefore, this study proposes a particle fuzzy categorization step that separates the particles into two sets in each iteration step based on the value of their objective value. Only the potential set, which contains all the particles with better objective values than the Pbest, implements the load distribution violation evaluation. This EFPSO method reduces, consequently, the number of structural analyses in EPSO for structural optimization. Since the particle fuzzy categorization step does not change the search rule of the algorithm, the search ability of the EFPSO depends on the version of PSO. Figure 4 depicts the steps of the proposed EEPSO, which is extended from EPSO [22-23]. The proposed EFPSO anticipates a reduction in the number of structural analyses. This study defines a factor 'R', which measures the efficiency of EPSO:

$$R = \frac{1}{\max_{iter}} \sum_{i=1}^{\max_{iter}} \frac{n_i}{N_p} \quad \textbf{(3)}$$

Where $\max_{itet}$ is the total iteration steps, n_i denotes the number of the particles which require the constraint violation evaluation in the ith iteration step, and N_p refers to the population of the particles. R in Equation (3) thus equals the ratio between the numbers of structural analyses required by.

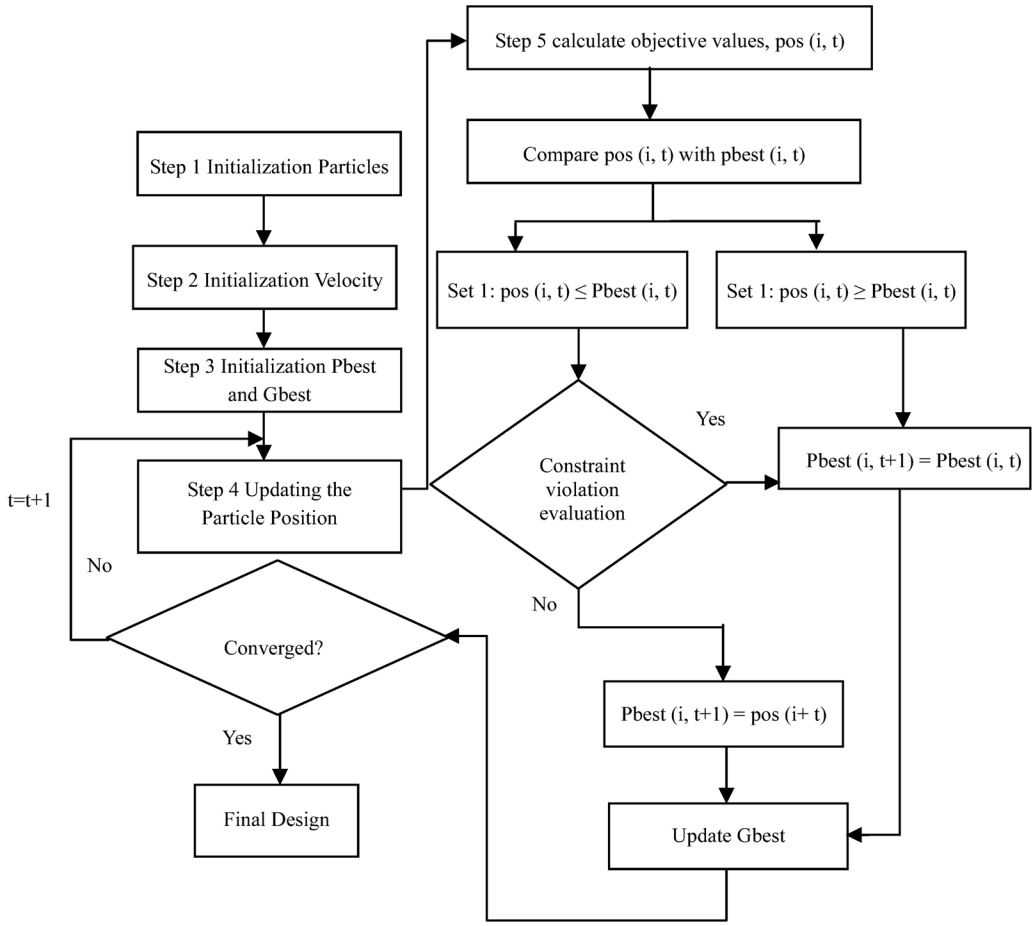

Figure 4. Flowchart representation of proposed EFPSO algorithm

6 Initialization of Feasible Particles

The following steps detail the improved opposition-based initialization strategy [24] which aims to generate the feasible particles:

Generate a particle P via the use of hypervisor randomly in the total space.

Calculate the opposite point of P, denoted by OP, by the following equation:

$$X_{op} = X_{upper} + X_{lower} - X_p \quad \textbf{(4)}$$

Where X_{upper} and X_{lower} denote the vector of the upper and lower bound of design variables, respectively. X_p is the randomly generated design in Step 1 and X_{op} is the opposite position of P in the design space.

If P locates in the infeasible load distribution space, the OP will have a higher possibility of residing in the feasible load distribution space compared with the randomly generated particle.

Evaluate the constraint violation of the two particles P and OP and calculate their objective values, respectively.

If both particles locate in the feasible space, select the one with a better objective IQR value and go back to Step 1. If only one of them is feasible, preserve the feasible particle and go back to Step 1. If both of the particles violate the constraints, go back to Step 1. Repeat the loop until the number of selected particles equals the population of the particles predefined in PSO.

In the equation (4) decision value of X_{op} is determined via the use of the fuzzy parameter. Fuzzify all input VMs' CPU, memory, and network bandwidth consumption ranges into fuzzy membership functions. The equation is changed as the below:

$$X_{op} = X_{fuzzy} - X_p \quad \textbf{(5)}$$

The standard PSO consists of four parameters: the number of particles (hypervisors) NP, the acceleration coefficients ac_1 and ac_2, and the inertia weight *iw*. NP depends on the dimension and complexity of the optimization problem, with a typical range from 10 to 40, or 50 to 100 for some challenging and special problems [25]. The other three parameters, ac_1, ac_2 and *iw*, balance the exploration and exploitation capability of PSO. ac_1 And ac_2 affect the PSO global convergence, while the inertia weight *iw* influences the local search capability. The first is the fixed values for the three parameters, with $ac_1 = ac_2 = 0.8$ and $iw = 0.92$. Equation (6) describes a linear time-

dependent estimation, while Equation (7) shows a quadratic, time-dependent rule to calculate iw:

$$iw_t = iw_{\max} - \frac{iw_{\max} - iw_{\min}}{\max_{iter}} t \quad \textbf{(6)}$$

$$iw_t = iw_{\max} + (iw_{\max} - iw_{\min})[\frac{2t}{\max_{iter}} - (\frac{t}{\max_{iter}})^2] \quad \textbf{(7)}$$

Where $\max_{iter}$ is the maximum number of iterations and t is the current iteration number. The numerical procedure assumes $iw_{\max} = 0.9$ and $iw_{\min} = 0.4$ [26].

7 Convex Support Vector Machine (CSVM)

In this work proposed a novel CSVM algorithm that converges to the Hard Margin SVM solution for detecting attack rate. Consider a training set composed of several numbers of users (Us_a) and corresponding classes $y_a \pm 1$. When the training data is separable, the convex hulls formed by the positive and negative attackers are disjoint. Figure 5 illustrates the geometrical formulation of SVMs [27-28].

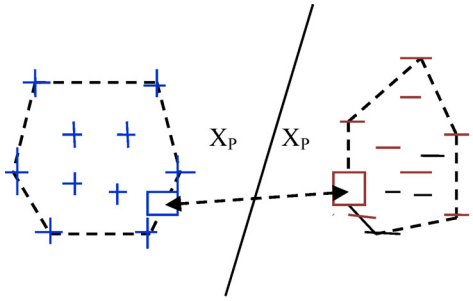

Figure 5. Geometrical interpretation of SVMs

Consider two different types of users Us_p and Us_n belonging to each convex hull. Make them as close as possible without allowing them to leave their respective convex hulls.

The median hyperplane of these two different types of users is the maximum margin separating hyperplane. The users Us_p & Us_n has been formulate as below:

$$Us_p = \Sigma_{a\in p}\alpha_a Us_a \Sigma_{a\in p}\alpha_a = 1 \;\; \alpha_a \geq 0 \quad \textbf{(8)}$$

$$Us_n = \Sigma_{b\in n}\alpha_b Us_b \Sigma_{b\in n}\alpha_b = 1 \;\; \alpha_b \geq 0 \quad \textbf{(9)}$$

Where sets P and N respectively contain the indices of the positive and negative users. The optimal hyperplane is then obtained by solving

$$\min_a \| Us_p - Us_n \|^2 \quad \textbf{(10)}$$

Under the constraints of the parameterization (10). The separating hyperplane is then represented by the following linear discriminant function:

$$y(us) = (Us_p - Us_n)Us + \frac{(Us_n Us_n - Us_p Us_p)}{2} \quad \textbf{(11)}$$

Since Us_p and Us_n are represented as linear combination of the training patterns, both the optimization criterion (10) and the discriminant function (11) can be expressed using dot products between patterns. Sometimes finding the attackers in the current classifier becomes very difficult task.

To solve this problem when a new misclassified sample arrives, Huller attempts to make this new sample become a support vector and removes another support vector from the current classifier via the update of the nearest points of the convex hulls.

The proposed method tries to select the vertices of the convex hulls of the current training cloud users in the off-line step and then retrains the classifier using these selected samples and newly arrived cloud users. The procedure of the CSVM algorithm is shown in Figure 6.

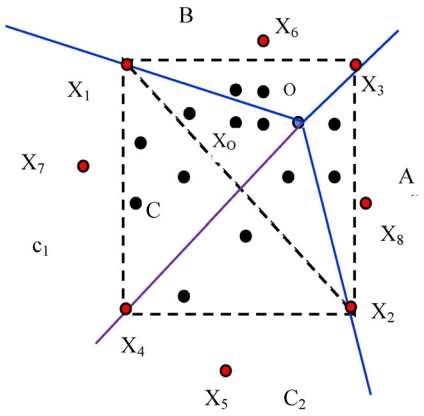

Figure 6. Schematic representation of CSVM algorithm

The convex hull of the set $CH_s = \{Us_a\}_{a=1}^n \subset \mathbb{R}^d$ is defined as:

$$convex(CH_s) = \{\sum_{a=1}^{n} a_a Us_a \,|\, Us_a \in CH_s, \sum_{a=1}^{n} a_a = 1, a_a \geq 0, a = 1, \ldots n\} \quad \textbf{(12)}$$

Given a set CH_s and a point Us, the Euclidean distance between Us and $convex(CH_s)$ can be computed by solving the following quadratic optimization:

$$\min_a \frac{1}{2} a^t Q_a - c^t \;\; a \; s.t. \; e^t \;\; a = 1, a \geq 0 \quad \textbf{(13)}$$

where = [1, 1, ..., 1] T, Q = UsTUsand c = UsTuswith Us = [us1, ..., usn]. Suppose the optimal solution of (13) is α^*. Then the convex hull distance between x and conv(P) is given by

$$dis_c(us, conves(CH_s)) = \sqrt{us^T us - 2c^t a^* + a^{*T} + Qa^*}\ a \quad (14)$$

Consider the problem from the attacker's point of view, the latter's objective is to minimize the hypervisor's maximal probability of detection.

The maximum that the hypervisor can realize is equivalent to the minimum that the attacker.

8 Experimentation Results

In this section, describe the experimental setup and present experimental results by comparing proposed work with a benchmark consisting of the Price based maximin [15], and Bayesian inference [14].

Provide experimental results to test the performance of model and validate the theoretical and numerical results obtained in the previous sections.

To these ends, conduct the experiments using CloudSim [29] in a 64-bit Windows 7 environment on a machine equipped with an Intel Core i7-4790 CPU 3:60 GHz Processor and 16 GB RAM. To build the cloud, create a datacenter whose VMs' configuration is inspired by Amazon EC2 X-large instances [30].

Practically, the created datacenter hosts five physical machines each of which is assigned with a number of VMs varying from 10 to 50 of image size amounting to 10000 MB each.

Every VM is equipped with 5-core CPU of 1000 Millions of Instructions Per Second (MIPS) each. [31]

Each VM has a memory RAM capacity of 16 GB, hard drive storage of 976:5625 GB, and network bandwidth share of 50000 Kbit/s.

In the created datacenter, x86 has been used as a system architecture, Linux as an operating system, and Xen as a Virtual Machine Monitor (VMM).

To populate the trust recommendations regarding VMs, resort to the use of the Epinions data set [32] that has been long used in cloud computing and many other domains for representing trust [33].

The data set comprises 664; 824 ratings given by 49; 290 agents on 139; 738 items.

The Datacenter Properties are represented in Table 1.

Table 1. Datacenter properties

Parameter	Value
Number of physical hosts	5
System architecture	x86
Operating system	Linux
Virtual machine monitor	Xen
Number of VMs	10,20,30, 40 and 50
Number of CPU cores per VM	5
CPU speed per VM	1000 MIPS
RAM memory per VM	16 GB
Hard drive storage per VM	976.56 GB
Network bandwidth share per VM	50000 Kbit/seconds

Figure 7 reveals that CSVM classifier with the EFPSO outperforms in terms of attack detection. The proposed work gives higher attack detection rate of 93.6%, whereas other methods such as price based maximin and Bayesian inference provides only 75.23% and 87.12% for 10 number of VMs.

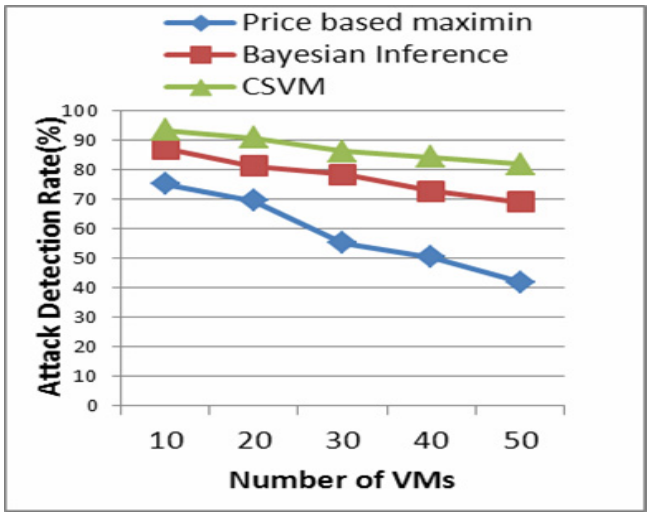

Figure 7. Attack detection rate vs. Number of Virtual Machines (VMs)

Figure 8 measures the false negative percentage that quantifies the percentage of attacks that the system was not able to capture during the detection process. The proposed work gives lesser false negative rate of 6.58%, whereas other methods such as price based maximin and Bayesian inference provides higher of 18.25% and 9.12% for 10 number of VMs.

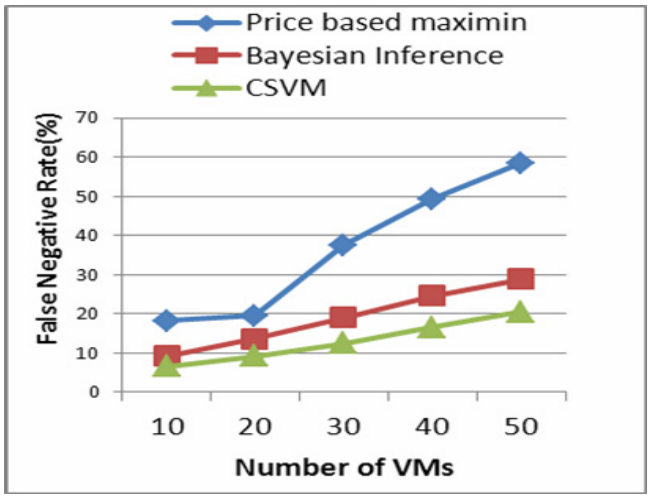

Figure 8. False Negative Rate (FNR) vs. Number of Virtual Machines (VMs)

Figure 9 measures the percentage of false positive incurred by the studied detection methods. The proposed work gives lesser false positive rate of 8.63%, whereas other methods such as price based maximin and Bayesian inference provides higher of 19.53% and 12.89% for 10 number of VMs.

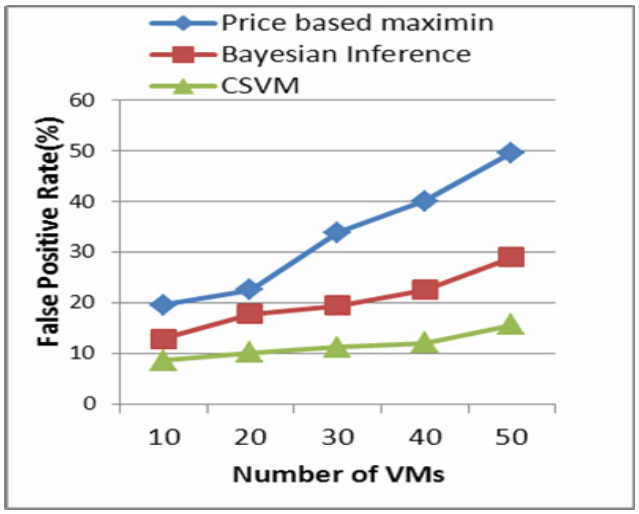

Figure 9. False Positive Rate (FPR) vs. Number of Virtual Machines (VMs)

Figure 10 is introduced to study the effectiveness of the proposed model in minimizing the cloud system's resources consumption when this system faces DDoS attack methods. The proposed work gives lesser CPU usage of 20.15%, whereas other methods such as price based maximin and Bayesian inference provides higher of 43.52% and 35.25% for 10 number of VMs.

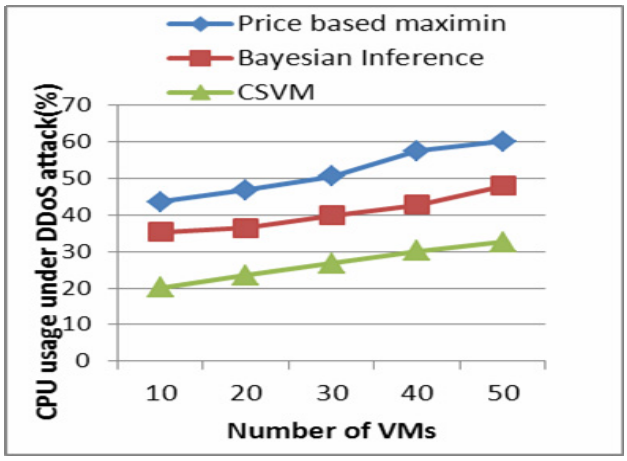

Figure 10. CPU usage vs. Number of Virtual Machines (VMs)

Figure 11 demonstrates that applying CSVM classifier reduces the memory consumption of cloud when compared to models. The proposed work gives lesser memory usage of 12.25%, whereas other methods such as price based maximin and Bayesian inference provides higher of 28.56% and 22.05% for 10 number of VMs.

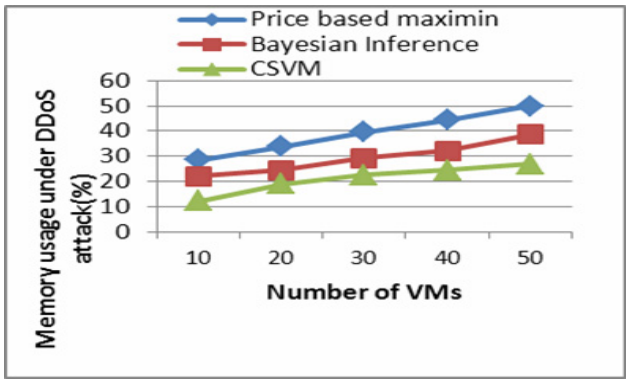

Figure 11. Memory usage vs. Number of Virtual Machines (VMs)

Figure 12 shows that CSVM classifier decreases the network bandwidth's consumption compared to the other two methods. This is due to the effectiveness of CSVM classifier in identifying attacks, which reduces the flux of the malicious traffic on the datacenter's network. The proposed work gives lesser bandwidth usage of 15.56%, whereas other methods such as price based maximin and Bayesian inference provides higher of 24.36% and 21.05% for 10 number of VMs.

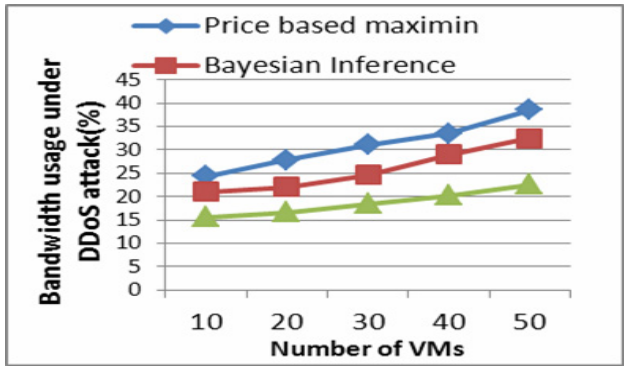

Figure 12. Bandwidth usage vs. Number of Virtual Machines (VMs)

Figure 13 measures the execution time of the three studied models with the variation in the number of VMs. The proposed work gives lesser execution time of 2.15 seconds, whereas other methods such as price based maximin and Bayesian inference provides higher of 0.96 seconds and 1.35 seconds for 10 number of VMs.

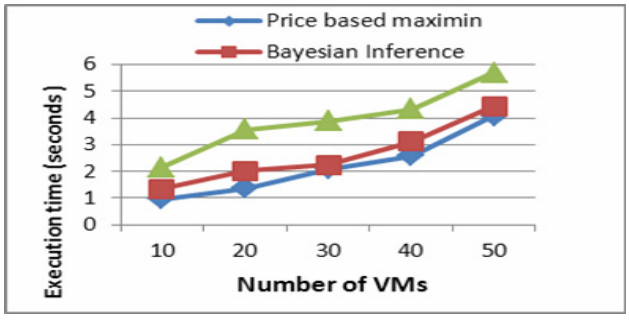

Figure 13. Execution time vs. Number of Virtual Machines (VMs)

9 Conclusion and Future Work

In this work, Enhanced Fuzzy Particle Swarm Optimization (EFPSO) algorithm is proposed which guides the hypervisor to determine the optimal detection load distribution among VMs in real-time that maximizes DDoS attacks' detection.

In the proposed work, the hypervisor monitors the VMs' CPU, memory, and network bandwidth consumption directly from the hosting infrastructure. At finally prevention is performed by using Convex Support Vector Machine (CSVM) classifier. The proposed work gives higher attack detection rate of 93.6%, whereas other methods such as price based maximin and Bayesian inference provides only 75.23% and 87.12% for 10 number of VMs.

Conduct the experiments using Cloud Sim in a 64-bit Windows 7 environment. The detection methods results are measured using the metrics like detection rate metrics like attack detection, false positive and false negative percentages. Moreover, CSVM solution proves to be able to minimize the cloud system's CPU consumption, memory consumption, and network bandwidth consumption under DDoS scenarios. Spoofing attack detection in the CC environment is left as scope of future work.

References

[1] Z. Wan, J. E. Liu, R. H. Deng, HASBE: A Hierarchical Attribute-based Solution for Flexible and scalable Access Control in Cloud Computing, *IEEE Transactions on Information Forensics and Security*, Vol. 7, No. 2, pp. 743-754, April, 2012.

[2] P. Mell, T. Grance, *The NIST Definition of Cloud Computing*, Report No. SP 800-145, September, 2011.

[3] A. Sangroya, S. Kumar, J. Dhok, V. Varma, Towards Analyzing Data Security Risks in Cloud Computing Environments, *International Conference on Information Systems, Technology and Management*, Bangkok, Thailand, 2010, pp. 255-265.

[4] L. M. Kaufman, Data Security in the World of Cloud Computing, *IEEE Security & Privacy*, Vol. 7, No. 4, pp. 61-64, July-August, 2009.

[5] T. Peng, C. Leckie, K. Ramamohanarao, Survey of Network Based Defence Mechanisms Countering the DoS and DDoS Problems, *ACM Computing Survey*, Vol. 39, No. 1, pp. 1-42, April, 2007.

[6] M. A. Rajab, J. Zarfoss, F. Monrose, A. Terzis, My Botnet Is Bigger Than Yours (Maybe, Better Than Yours): Why Size Estimates Remain Challenging, *Proceedings of the FIRST Conference on First Workshop Hot Topics in Understanding Botnets*, Cambridge, MA, USA, 2007, pp.1-8.

[7] J. Mirkovic, P. Reiher, Taxonomy of DDoS Attack and DDoS Defence Mechanisms, *ACM SIGCOMM Computer Communication Review*, Vol. 34, No.2, pp. 39-53, April, 2004.

[8] S. Mansfield-Devine, The Growth and Evolution of DDoS, *Network Security*, Vol. 2015, No. 10, pp. 13-20, October, 2015.

[9] U. Tupakula, V. Varadharajan, N. Akku, Intrusion Detection Techniques for Infrastructure as a Service Cloud, *IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*, Sydney, Australia, 2011, pp. 744-751.

[10] J. Nikolai, Y. Wang, Hypervisor-based Cloud Intrusion Detection System, *International Conference on Computing, Networking and Communications*, Honolulu, HI, USA, 2014, pp. 989-993.

[11] I. Sattar, M. Shahid, Y. Abbas, A Review of Techniques to Detect and Prevent Distributed Denial of Service (DDoS) Attack in Cloud Computing Environment, *International Journal of Computer Applications*, Vol.115, No. 8, pp. 23-27, April, 2015.

[12] A. M. Lonea, D. E. Popescu, H. Tianfield, Detecting DDoS Attacks in Cloud Computing Environment, *International Journal of Computers Communications & Control*, Vol. 8, No. 1, pp. 70-78, February, 2013.

[13] S. Jamali, G. Shaker, Defense against SYN Flooding DoS Attacks by Employing PSO Algorithm, *Computers & Mathematics with Applications*, Vol. 63, No.1, pp. 214-221, January, 2012.

[14] O. A. Wahab, J. Bentahar, H. Otrok, A. Mourad, Optimal Load Distribution for the Detection of VM-based DDoS Attacks in the Cloud, *IEEE Transactions on Services Computing*, Vol. 17, No. 1, pp.1-1, April, 2017.

[15] O. A. Wahab, J. Bentahar, H. Otrok, A. Mourad, How to Distribute the Detection Load among Virtual Machines to Maximize the Detection of Distributed Attacks in the Cloud, *IEEE International Conference on Services Computing (SCC)*, San Francisco, CA, USA, 2016, pp. 316-323.

[16] S. Yu, Y. Tian, S. Guo, D. O. Wu, Can We Beat DDoS Attacks in Clouds, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 25, No. 9, pp. 2245-2254, September, 2014.

[17] S. T. Zargar, J. Joshi, D. Tipper, A Survey of Defense Mechanisms against Distributed Denial of Service (DDoS) Flooding Attacks, *IEEE Communications Surveys & Tutorials*, Vol. 15, No. 4, pp. 2046-2069, November, 2013.

[18] Z. Liu, X. Wang, A PSO-based Algorithm for Load Balancing in Virtual Machines of Cloud Computing Environment, *International Conference in Swarm Intelligence*, Shenzhen, China, 2012, pp. 142-147.

[19] F. Ramezani, J. Lu, F. K. Hussain, Task-based System Load Balancing in Cloud Computing Using Particle Swarm Optimization, *International Journal of Parallel Programming,* Vol. 42, No. 5, pp. 739-754, October, 2014.

[20] F. Lombardi, R. Di Pietro, Secure Virtualization for cloud Computing, *Journal of Network and Computer Applications*, Vol. 34, No. 4, pp. 1113-1122, July, 2011.

[21] O. A. Wahab, J. Bentahar, H. Otrok, A. Mourad, A Survey on Trust and Reputation Models for Web Services: Single, Composite, and Communities, *Decision Support Systems*, Vol.

74, pp. 121-134, June, 2015.

[22] R. E. Perez, K. Behdinan, Particle Swarm Approach for Structural Design Optimization, *Computers & Structures*, Vol. 85, No. 19, pp. 1579-1588, October, 2007.

[23] H. Cao, X. Qian, Z. Chen, H. Zhu, Enhanced Particle Swarm Optimization for Size and Shape Optimization of Truss Structures, *Engineering Optimization*, Vol. 49, No. 11, pp. 1939-1956, November, 2017.

[24] H. Jabeen, Z. Jalil, A. R. Baig, Opposition Based Initialization in Particle Swarm Optimization (O-PSO), *Proceedings of the Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, Québec, Canada, 2009, pp. 2047-2052.

[25] V. Plevris, M. Papadrakakis, A Hybrid Particle Swarm—Gradient Algorithm for Global Structural Optimization, *Computer-Aided Civil and Infrastructure Engineering*, Vol. 26, No.1, pp. 48-68, January, 2011.

[26] L. J. Li, Z. B. Huang, F. Liu, Q. H. Wu, A Heuristic Particle Swarm Optimizer for Optimization of Pin Connected Structures, *Computers & Structures*, Vol. 85, No. 7, pp. 340-349, April, 2007.

[27] K. P. Lin, M. S Chen, On the Design and Analysis of the Privacy-preserving SVM Classifier, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 23, No.11, pp. 1704-1717, November, 2011.

[28] A. Alkan, M. Günay, Identification of EMG Signals Using Discriminant Analysis and SVM Classifier, *Expert Systems with Applications*, Vol. 39, No. 1, pp. 44-47, January, 2012.

[29] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, R. Buyya, Cloudsim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms, *Software: Practice and Experience*, Vol. 41, No. 1, pp. 23-50, January, 2011.

[30] Amazon Web Services, *Amazon EC2*, https://aws.amazon.com/ec2/details/.

[31] *SPEC Java Virtual Machine Benchmark 2008*, http://www.spec.org/jvm2008/.

[32] J. Leskovec, *Epinions Social Network*, https://snap.stanford.edu/data/soc-Epinions1.html.

[33] S. Deng, L. Huang, G. Xu, Social Network-based Service Recommendation with Trust Enhancement, *Expert Systems with Applications*, Vol. 41, No. 18, pp. 8075-8084, December, 2014.

Biographies

A. Peter Soosai Anandaraj received his bachelor's degree (B.E-Bachelor of Computer Science and Engineering) from Raja College of engineering and Technology, Madurai, and affiliated to Anna University, Chennai, in 2010 and then Completed his Master Degree in computer science and engineering from Pandian Saraswathi Yadav Engineering College, Sivagangai, affiliated to Anna University, Chennai, in 2012. He is currently working as an Asst Prof in Ganapathy Chettiar College of Engineering and Technology, Paramakudi. He is also pursuing Ph.D. in Information Communication Engineering, Anna University, and Chennai. He has 5 years of teaching experience in Computer Science and Engineering. His current research interests include the area of Network and its Security

G. Indumathi obtained Bachelor's degree in Electronics and Communication Engineering and Master's Degree in Communication Systems both from Madurai Kamaraj University, Madurai, India in 1990 and 1997 respectively. She received doctoral degree from Anna University, Chennai for her research work in the domain of Information and Communication engg in the year 2012 presently she is working as a Professor in the department of Electronics and Communication Engineering in Mepco Schelenk Engineering College, Sivakasi. She has published many International Journals and National Journals. Her current research interests are in the areas of Wireless and Network Security and Digital signal Processing. She is a Life member in Indian Society for Technical Education and Fellow in Institution of Electronics and Telecommunications Engineers.