Secure Authentication and Load Balancing of Distributed Edge Datacenters

Deepak Puthal*, Rajiv Ranjan†, Ashish Nanda*, Priyadarsi Nanda*, Prem Prakash Jayaraman‡, and Albert Y. Zomaya¶

*University of Technology Sydney, NSW, Australia

†Newcastle University, Newcastle upon Tyne, United Kingdom

‡Swinburne University of Technology, VIC, Australia

¶The University of Sydney, NSW, Australia

Abstract—Edge computing is an emerging research area to incorporate cloud computing into edge network devices. An Edge datacenter, also referred to as EDC, processes data streams and user requests in real-time and is therefore used to decrease the latency and congestion in the network. EDC is usually setup as a distributed system and is accordingly placed between the cloud datacenter and the data source. These EDCs work as an intermediate layer in the fog hierarchy between IoT and Cloud datacenter. EDC's are aided by load balancers, responsible for distributing the workload amongst multiple EDC, in order to optimize resource utilization and response time. The load balancers make sure that the workload is equally divided amongst the available EDCs to avoid over loading of some EDCs while other remain idle as this directly impacts the user response and real-time event detection. Given the fact that EDCs are deployed in remote environments, the need for secure authentication is of major importance. In this paper we propose a novel load balancing technique that enables EDC authentication as well as identification of idle EDCs for better load balancing. The proposed load balancing technique is also compared with existing approaches and proves to be more efficient in locating EDC's with less workload. In addition to the improved efficiency, the proposed scheme also strengthens the security of the network by incorporating destination EDC authentication.

Key words— Edge computing, Fog computing, Edge datacenter, Cloud, Security, Authentication, Load balancing.

1. Introduction

An overlapping of the features of cloud along with additional attributes, such as location awareness and EDC deployment, is referred to as fog computing. When distributed geographically in large numbers, EDC's can provide mobile, low latency data transparency to achieve real-time requests and responses [1]. As a popular choice in providing scalable computation, cloud computing can process large amounts of data (referred as big data), provide storage and provision resources based on the user requirements. Fog computing proposes the migration of cloud resources over to EDCs which are then deployed across the network [7]. The fog computing has various proposed architecture that link it with the edge deployment. A block diagram of the three architectural layers of fog computing is portrayed in Figure 1. The model begins with the bottom layer compromising of various terminal devices, such as wireless sensors and smart devices, that are responsible for the transmission of data onto the upper layers. The second layer of the model comprises mainly of highly intelligent devices, such as the routers, switches and gateways that aid the network. Some architecture models are known to divide the middle layer, Edge Layer, into its two components; the edge device and the edge datacenter, however in a fog computing architecture these two component layers are combined into a single edge layer. The topmost layer (also the third layer) comprises of several high-end servers, known as fog servers, and acts as a cloud datacenter. These cloud datacenters, when deployed, also contain user response facilities and occupy the topmost layer of the fog architecture. The fog computing is defined as a combination of the above-stated three layers as portrayed in Figure 2 along with its comprehensive architecture and various modules.

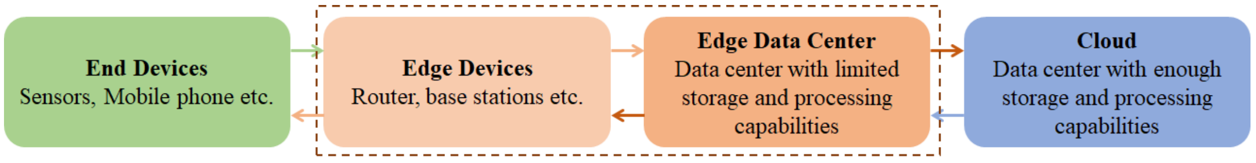

End Devices
Sensors, Mobile phone etc.
Edge Devices
Router, base stations etc.
Edge Data Center
Data center with limited storage and processing capabilities
Cloud
Data center with enough storage and processing capabilities

Figure 1. Fog hierarchy with edge datacenter deployment.

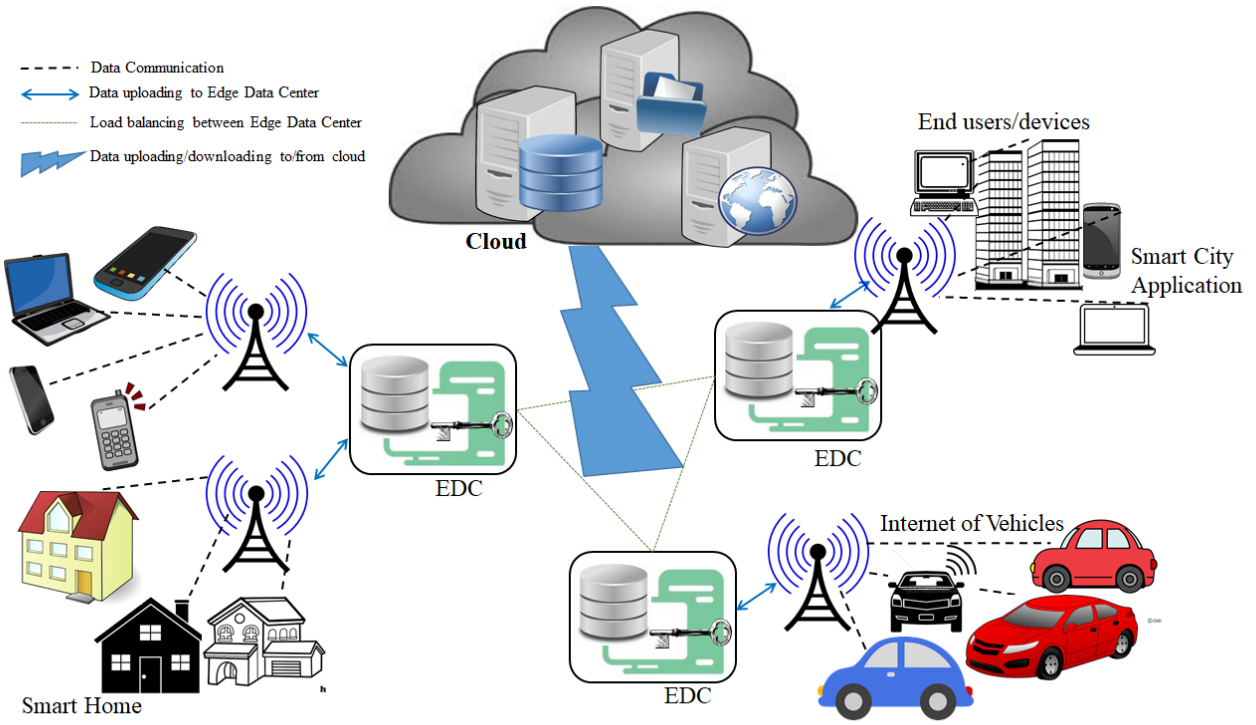

Data Communication
Data uploading to Edge Data Center
Load balancing between Edge Data Center
Data uploading/downloading to/from cloud
Cloud
End users/devices
Smart City Application
EDC
EDC
Internet of Vehicles
EDC
Smart Home

Figure 2. An all-inclusive architecture of edge datacenter deployment with complete data flow from IoT to cloud datacenter.

As computing environments achieve great advancements, the EDC service's availability in fog computing has also improved raising a lot of attention towards the load balancing problem faced by EDCs. Various research models have been proposed in order to solve the load balancing problem, however they fail to adequately address the concerns regarding EDC authentication. Given that EDC deployment is usually in remote unattended scenarios, authentication is an important step before any load balancing takes place. In addition, as network structure for an EDC deployment is distributed, the load balancing also works in a distributed scenario and is classified into dynamic load balancing and static load balancing [2].

In the static load balancing technique, the performance function is minimized by providing a set of teaks to the specific EDCs. This can be achieved using either deterministic means or probabilistic means. According to the deterministic balancing technique, EDC-I is responsible for allocation of tasks to EDC-J each time it is required. However, in a probabilistic balancing technique, the allocation of tasks done by EDC-I to EDC-K is with a probability x and similarly for EDC-L is with a probability y. A major drawback of static load balancing is due to the fact that it does not take into account the status of the destination EDC when deciding the load balancing. The dynamic load balancing takes a more real-time approach by considering the current load over individual EDC and accordingly suggest a destination EDC. This enables the tasks to be assigned dynamically from an overloaded EDC to an under-loaded or idle EDC. Compared to the static approach, the dynamic approach is much difficult to implement, however it provides a better solution towards achieving a sustainable solution to load balancing. Given the above benefits, this paper considers the dynamic load balancing technique in the proposed solution

There are many solutions available to implement authentication over network systems, however none of them are suitable for the authentication of EDC. Due to the hostile nature of EDC deployment, authentication become a major issue to identify destination EDC before final assignment of incoming tasks. Existing authentication techniques authenticate the network edges [12] [20], whereas proposed authentication technique getting destinations load information at the time of authentication. Which brings novel properties to build sustainable load balancing with proper authentications. In this paper we propose a secure authentication method to select the EDC for load balancing. Below is a summary of the main contributions made by our proposed approach:

- Our proposed approach uses a centralized cloud datacenter to put forward an adaptive EDC authentication technique which once initiated by the enables the EDCs to authenticate each other using the cloud credentials.
- The approach also considers the real-time load on each of the destination EDCs to achieve dynamic load balancing. This information is collected and also shared during the above-mentioned authentication process minimizing any additional communication that would have taken place if the information was not shared.
- The proposed approach combines the above stated dynamic load balancing technique along with the authentication and apply it to the EDCs. The proposed scheme is also evaluated for its performance to validates its efficiency and stability.

The remainder of the paper is organized as following. Section 2 discusses related works. Section 3 describes the proposed solution for the secure and sustainable load balancing of the EDCs. Section 4 presents the formal security analysis and verification of our model. Section 5 evaluates the performance and efficiency of the proposed solution through extensive experimentation. Section 6 covers the conclusions and potential future directions.

2. Related Works and Problem Analysis

A brief background study of related works and subsequently problem analysis of the proposed method is described in this section. This gives a clear understanding of the research problem and a motivation to our proposed method.

2.1 Related Works

IoT smart sensing devices, including mobile users, dump their tasks to the nearest EDC [1]. As the source devices in this scenario are mobile, the loads on different EDCs may vary depending upon their location in the network. This can result a disbalance of task distribution as some EDCs in high activity area may be overloaded whereas as others in a low activity area may be idling with very less work load. There exist several works that proposed distinctive strategies to address load balancing issues. When considering the load balancing problem in EDC as an optimization problem, Jia et al. [5] proposes a scalable algorithm that is able to redirect tasks amongst a given set of EDCs in the network, thereby minimizing the average response time. Willebeek-LeMair and Reeves [6] in 1993 proposed a tool for distributed systems that provided basic load balancing, this initial tool have been made efficient towards different scenarios and applications by the contributions of various researchers. Tong et al. [7] also proposed a similar technique that handles peak load while satisfying the requirements made by remote program executions. Various other authors took the concept of cloud server and hosted it in network edges in order to design edge-computing architecture along with proposed algorithms that maintain the load balancing efficiently. In order to achieve geographical load balancing, the workloads are routed dynamically which overall help minimize energy consumption [8]. Zhang et al. [8] proposed an algorithm to solve the challenging load balancing problem optimally and efficiently by discovering the entire design space of strategic bidding. Tripathi et al. [17] proposed a non-cooperative distributed load balancing algorithm game to minimize the operating cost and obtain the structure of Nash equilibrium. This method provides the approximate global optimal solution in terms of the cost and it also ensures fairness among the users. Wu et al. [18] presented a decentralized system that detects short-term overload situations in a timely fashion and optimally distributes the overload to remote data centers by overall response times. This method autonomously handles them using geographical load balancing and admission control to minimize the resulting performance degradation. Paya and Marinescu [19] introduce an energy-aware operation model used for load balancing and application by focusing on cloud scaling. The basic philosophy of this method is defining an energy-optimal operation regime and attempting to maximize the number of servers operating in this regime. Fan et al. [22] proposed a novel load balancing approach for mobile cloudlet or edges. This approach follows balls-and-bins theory to handle extreme incoming loads and finally experimented to validate their work.

The need for security in an EDC deployment is discussed in [9] based upon a study and classification of the current security issues that impact EDCs. Various existing schemes and techniques present their own way of dealing with the authentication problem relevant to the network scenario. The authors Butun et al. [10] propose a cloud-centric multi-level authentication scheme that addresses constraints such as scalability, time and effectiveness. Their proposed scheme is aimed towards connected devices in the Internet of Things by providing an authentication technique for public safety. He and Zeadally [11] analyze the overall architecture associated with security requirements and propose an efficient authentication technique, with a focus on healthcare technology, that works for body area network. Another authentication scheme proposed by He et al. [12] uses anonymous authentication for a similar wireless body area network. Taking in account the current cyber threat scenarios, system identification protection can be achieved by developing a security perimeter [13]. Jan et al. [20] proposed a lightweight payload-based mutual authentication scheme for a cluster-based sensor networks. This method initially elects the cluster heads, authenticates, and allows communication with neighboring nodes and subsequently the cluster head authenticates the nearby nodes for cluster formation. This remains a challenging task for the EDC, as EDCs are deployed in an open access network. There is a need for a new authentication scheme for EDC load balancing, a need addressed by this paper.

2.2 Problem analysis

Edge computing and fog computing is still an open research problem where lots of researchers are focusing to bring cloud resources to network edges [16] [21]. This will facilitate processing of the data in edge datacenters for emergency data processing and quick responses. EDCs are distributed in nature and deployed with network edges for quick response on user queries. As EDCs are deployed in the network edges, they do not have enough resources and computing power like cloud [25]. As a result, EDCs easily get overloaded with incoming tasks. Another prominent issue with the cyber threats is of hostile/open deployment [9]. So, not selecting a fraudulent or malicious EDC for load balancing is still an open challenge. Finally, we conclude that the following two issues are really important for edge computing and subsequently develop a novel method to avoid the issues.

1. Securely authenticate and select the recipient EDC to share loads.
2. Efficient load balancing method to balance the load between EDCs.

3. Proposed Method

As stated previously and based on a recent literature review, an architecture that can authenticate the edge datacenter before allocating tasks does not exist. Hence, the authors propose a novel architecture that not only authenticates but also collects the current load information on the EDCs before assigning it any tasks. The following subsections send light over the complete procedure opted for the proposed load balancing technique, this includes a discussion of the secure authentication process for the EDCs followed by the sustainable load balancing technique. The notation used in the proposed method description is listed in the Table 1.

Table 1: Notation

Acronym	**Description**
EDC	Edge datacenters
Cloud	Cloud datacenters
K_c	Cloud shared key
K_i/K_j	Key of EDC-I and EDC-J
E_i	ID of ith EDC
K_i	Secret key of ith EDC
PrK_i	Private key of ith EDC
PuK_i	Public key of ith EDC
$\|$	Concatenation operation
ACK	Acknowledgement

3.1 Secure Authentication

As per the fog computing architecture, all the data must be stored and processed in the cloud. In such a network EDCs are intermediate datacenters that aid the network and reduce latency for user requests. The cloud deployment always takes place in a secure environment and is considered as fully trusted, whereas EDCs are considered as partially trusted as they are deployed at the network edges. For this reason, we consider the cloud for the initiation of the authentication process. This process begins with the assigning of an initial ID (Ei) linked to the key (Ki) and the common shared key (Kc) for each EDC during the EDC deployment $(Cloud \rightarrow EDCs\ \{E_i||K_i||K_c\})$. To store the secret information provided by the cloud and the rekeying process, EDC's use trusted modules such as the Trusted Platform Module (TPM) [3]. Once initialized, individual EDCs will begin the authentication process to verify other EDCs in the neighborhood. This step prevents malicious EDCs to join the load balancing in the future. The individual keys exchange for EDCs is initialized and maintain by cloud data center or fog server.

Let us consider EDC-I as the edge datacenter that initiates the authentication process. It will combine its own ID with the associated key, it will then encrypt using the shared key initiated by the cloud $\left(E_{K_c}(E_i \parallel K_i)\right)$. EDC-I will then broadcast the generated packets to all available EDCs in the locality. One an EDC receives the authentication request packet, it uses the cloud shared key $\left(D_{K_c}(E_i \parallel K_i)\right)$ to decrypt it. Given that the cloud shared key remains constant for all EDCs, it can be used for both encryption and decryption process. A common shared key (Kc) is, however, initiated by the cloud and provided to individual EDCs and is also trusted by other EDCs. When the destination EDC (EDC-J) receives the source ID and the key associated with it, a check is performed with the cloud to verify the authenticity of the source EDC (EDC-J → Cloud $\{E_{K_c}\left(E_j \parallel E_{K_j}(E_i)\right)\}$). Subsequently the cloud authenticates the EDC-J and uses the secret key to find the EDC-I ID. If cloud found EDC-I is a authenticated node by checking its' own database, subsequently cloud respond the ACK to EDC-J $(E_{K_j}(E_i \parallel K_i))$ by encrypting with recipient's secret key. In case of unauthenticated EDC-I, the cloud responds to EDC-J declining the authentication request. Cloud confirm everything about the authentication process, followed by EDC-J keeps EDC-I details as an authenticated EDC. Then EDC-J concatenates associate key with its own ID and encrypts them using source associated key such as $\left(E_{K_i}(E_j \parallel K_j)\right)$. Upon receiving the encrypted packets by EDC-I, it uses own secret key for decryption and send authentication details to the cloud for EDC-J verification. The encrypted packet is of the format $\left(E_{K_c}\left(E_i \parallel E_{K_i}(E_j)\right)\right)$, where E_j is encrypted with source EDC-I associate key. This combines own ID to generate an encrypted packet using cloud shared key. EDC-J decrypts the encrypted packet using own shared key after receiving confirmation from cloud datacentre and then retrieves the associated key of $E_j(E_j \rightarrow K_j)$ and validate EDC-J. Once the process is validated, cloud concatenates E_j, the associated key (K_j) and encrypts with EDC-I associated key (K_i) to send it back to EDC-I. EDC-I decrypts it to find the key (K_j') after receiving the encrypted packet, which compares with the associated key received from EDC-J. If a match found i.e. $K_j = K_j'$, then EDC-I combines the ID of EDC-I and EDC-J and use destination associate key (K_j) for encryption. This combined packet is received by the EDC-J, which confirms both EDC-I and EDC-J are now authenticated to each other for load balancing. Proposed authentication model follows a straight forward method to get authentication information from cloud datacentres. Algorithm 1 gives stepwise process of authentication between edge datacentres. Followed by, individual EDCs generate their key pairs, i.e. public $\left(PuK_{i/j}\right)$ and private $\left(PrK_{i/j}\right)$ key pairs and broadcast the public key $\left(PuK_{i/j}\right)$ for further use by the EDC.

Algorithm 1. Secure Authentication

Input: Cloud Initialized the shared secret key (K_c) and identification to EDCs ($E_i,\ K_i$).

Output: All the EDCs are authenticated with each other to securely balance the loads.

Procedure:

1. Cloud →EDC-I $\{E_i,\ K_i$ and $K_c\}$; Cloud unicast the individual EDC's ID and common shared key.
2. EDC-I broadcast $\{E_{K_c}(E_i \parallel K_i)\}$
3. EDC-J → Cloud $\{E_{K_c}\left(E_j \parallel E_{K_j}(E_i)\right)\}$; Recipient authenticated node connects to Cloud for EDC-I authentication.
4. Cloud→EDC-J $\{E_{K_j}(E_i \parallel K_i)\}$; Cloud responds to EDC-J with EDC-I authentication details.
5. EDC-J → EDC-I $\{E_{K_i}(E_j \parallel K_j)\ \}$; EDC-J responds to EDC-I by encrypting with destination secret key.
6. EDC-I authenticates EDC-J with Cloud by following Step 3 (Go TO STEP 3)
7. EDC-I → EDC-J $\{E_{K_j}(E_i \parallel E_j)\}$; After successful authentication EDC-I sends ACK to EDC-J.

Now EDC-I and EDC-J are authenticated with each other to generate the private/public key pairs.

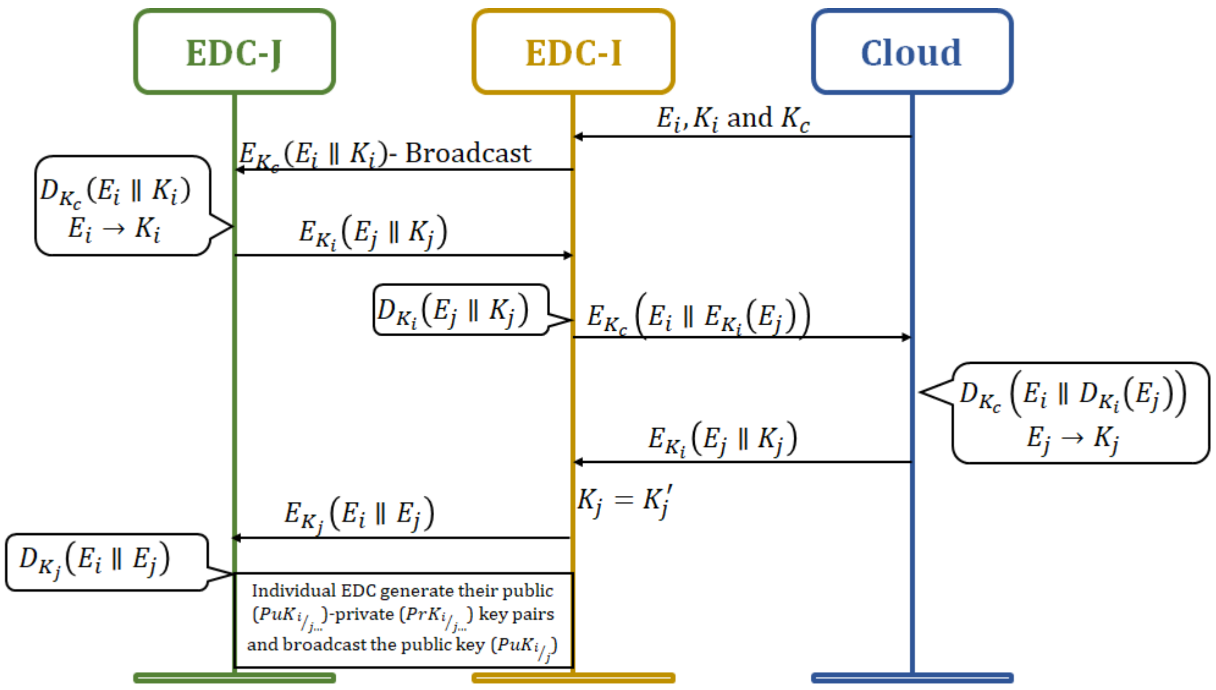

EDC-J
EDC-I
Cloud
E_i, K_i and K_c
$E_{K_c}(E_i \parallel K_i)$- Broadcast
$D_{K_c}(E_i \parallel K_i)$ $E_i \to K_i$
$E_{K_i}(E_j \parallel K_j)$
$D_{K_i}(E_j \parallel K_j)$
$E_{K_c}\left(E_i \parallel E_{K_i}(E_j)\right)$
$D_{K_c}\left(E_i \parallel D_{K_i}(E_j)\right)$ $E_j \to K_j$
$E_{K_i}(E_j \parallel K_j)$
$K_j = K_j'$
$E_{K_j}(E_i \parallel E_j)$
$D_{K_j}(E_i \parallel E_j)$
Individual EDC generate their public ($PuK_{i/j...}$)-private ($PrK_{i/j...}$) key pairs and broadcast the public key ($PuK_{i/j}$)

Figure 3. Stepwise information and credential flow for the secure authentication of EDCs.

3.2 *Secure Load Balancing*

Our proposed solution implies the breadth first search (BFS) technique for the design of the load balancer. The scheme uses two parameters m and n to keep a check on the load on each EDC; where the current load is defined by m and the processing capacity in defined by n. To calculate the current load on an EDC, the parameter p is used where $p = {}^{m}/_{n}$. Each EDC receives a load balancing request from its neighboring EDCs.

For instance, if EDC-I overloads, it will broadcast a control packet, containing its own ID and load information (*Ei*, *Li*), as a request to other EDCs in its vicinity. Here, the *Ei* defines the ID of the EDC sending the request and *Li* defines the received load information. A neighboring EDC (referred to as EDC-J) will perform a check on the received ID by comparing it with its own database. If a match is found, EDC-J will look for load information from the control packets, however if a match is not found the EDC will ignore the packet in order to avoid a possible DDoS attack.

While processing the load sharing information at EDC-I, the recipient EDC (i.e. EDC-J) checks the load information using value of *p*. If the value of *p* is less than or equal to 0.6 (i.e. close to 40% free resources) and the available resource index (i.e. *n-m*) to execute the invited tasks from EDC-I, then EDC-J initiate to prepare the positive ACK to EDC-I. If recipient i.e. EDC-J found the available resource is more than the required resource to process the invited task, then EDC-J processes the positive response packet to the EDC-I. Otherwise, EDC-J remain silent and never respond to the EDC-I. If both the condition satisfies, EDC-J prepares response and includes own identity (E_j), associated key and the status of datacenter available resource (i.e. *p*). Finally, the response packet is encrypted with the public key of the destination EDC-I i.e. kpu_i, $\left(E_{Kpu_i}(E_j||K_j||p)\right)$ and sends it to the EDC-I for further processing of load sharing. Upon receiving the encrypted data packets, EDC-I apply own private key i.e. Kpr_i to decrypt the data packets $\left(D_{Kpr_i}(E_j||K_j||p)\right)$. Followed by, EDC-I verifies the source ID (E_j) of encrypted packet and likens with its own database to find match for authentication. If a match is found, EDC-I excerpts the source device key (k_j') and compares the associated key with the received key ($k_j' = k_j$). I case of match, EDC-I accepts the ACK from EDC-J, else ignore to avoid DDoS attack. In an analogous way, EDC-I receives numerous replies from various EDCs in the region. EDC-I compares the values of *p* from all the authenticated response (ACK) to finds the less loaded EDC with lowest value of *p*. Finally, EDC-I sends tasks to the selected authenticated EDCs to process them. Algorithm 2 defines the stepwise procedure of secure load balancing process.

Algorithm 2. Load Balancing Technique

```
1. If (EDC-I is overloaded)
2. EDC-I broadcast ($E_i$, $L_i$)
3. EDC-J (neighbor EDC) verifies:
4. If ($E_i$ is in database) & ($p \leq 0.6$ & $L_i << (n-m)$)
5.         Response $E_{Kpu_i}(E_j||K_j||p)$
6. EDC-I perform $D_{Kpr_i}(E_j||K_j||p)$
7. $k_j' \leftarrow E_j$
8. If ($k_j' = k_j$)
9.         EDC-I select EDC-J for load balancing.
```

4. Security Evaluation

The proposed authentication model is evaluated using a combination of theoretical analysis and formal verification, both of which are discussed below in detail.

4.1 Security Proof

Definition (Attack on authentication). An intruder *"Ma"* launches an attack on the authenticity and has the capability to monitor, intercept and introducing itself as an authenticated EDC in order to start the load balancing process. Possible attacks for this category include impersonation attack and identity-based attacks [3].

Definition (Attack on confidentiality). A malicious attacker *"Mc"* is an unauthorized party which has the ability to access or view the task information while sharing loads between EDC-I and EDC-J [3].

Definition (Attack on integrity). An attacker "*Mi"* can attack on reliability if it is capable of monitoring the load information and trying to access and/or modify the tasks between EDCs [3].

Theorem 1: It is impossible for an attacker *Ma* to read the secret credentials of EDC to introduce itself as an authenticated EDC to participate in load balancing.

Proof: Following the above definition of attack on authenticity and computational hardness of TPM module (a secure module of EDC), we believe that attacker *Ma* cannot get the secret information such as *Ei*, *Ki* and *Kc* initiated by the cloud. All the secure information for authentication is initiated by the cloud datacenter during the EDC deployment. When EDCs start authenticating each other, they use cloud shared key (*Kc*) to encrypt the initial authentication packet $\left(E_{K_c}(EDC_i \parallel K_i)\right)$ followed by individual associate keys of EDCs (*Ki/j*). Proposed technique follows AES encryption during the initial authentication. Which is already proved that it would take years to break the transaction with current processing capabilities [3]. In our real-world experiment, we found AES 128-bit keys take 1.9e19 days and AES 64-bit keys take 11415 days to find all possible keys.

Thus, it is close to impossible for network intruders to monitor the network thoroughly and get the authentication credentials. During the authentication process, individual EDCs use their secure module (such as TPM) to keep their keys for encryption or decryption. Hence, it is nearly impossible to get either process or keys from secure module, following TPM properties. Consequently, we conclude that an attacker *Ma* cannot attack on authenticity during load balancing.

Theorem 2: An attacker *Mc* and *Mi* cannot read the tasks and/or load information during load balancing to break the data confidentiality and integrity.

Proof: Proposed load balancing technique use standard asymmetric key cryptography for encryption and decryption process after secure authentication. It is already proved that asymmetric is much more secure than symmetric cryptography, whereas asymmetric cryptography needs more computational power compared to symmetric cryptography . While load balancing, EDC-I broadcasts a request packet with its own identity and load information i.e. (Ei, Li) in format $\left(\mathrm{E}_{\mathrm{K_c}}(\mathrm{EDC_i} \parallel \mathrm{K_i})\right)$ from Algorithm 1. Which is proved to be secure by maintaining authenticity (See Theorem 1). Upon receiving load information, recipient EDC responds to the EDC-I by encrypting with the destination EDC's public key i.e. $E_{Kpu_i}(E_j||K_j||p)$. Subsequently, EDC-I use its private key to decrypt the response $\left(D_{Kpr_i}(E_j||K_j||p)\right)$ and find the details about the recipient EDC's identity (*Ej*). In summary, after authentication EDCs use public private key pairs (PuK_i/PrK_i) for encryption and decryption.

In the whole authentication process, intruder *Mc* and *Mi* cannot participate in the load balancing process because of the hardness of the asymmetric key cryptography. So, the process is secure against attacks on confidentiality and integrity. The stepwise key exchange and load balancing is shown in Algorithm 2.

Theorem 3: Proposed load balancing method provide sustainable result by choosing less loaded recipient EDC for task sharing.

Proof: The reposed load balancing technique follows BFS (Breadth First Search) method to share loads. Where, the graph defines as *G (V, E)* for complete network and *N(v)* defines for the number of neighbor nodes, where *V* is the *EDC. σ(EDC1, EDC2, … EDCn)* are set of neighbors from the network. $\forall EDC \in V$, all the n number of *EDCs* are in the range for load sharing. Source *EDC* broadcasts the overloaded information to get the recipient *EDC* to share the load. Followed by, *EDC* received *x* number of responded where $\forall 1 \leq x \leq n$.

As proposed technique following authentications method during initial broadcast (refer Algorithm 1), EDC-I receives the response in a format i.e. ($E_{Kpu_i}(E_j||K_j||p)$) from *x* number of EDCs. This packet shows the information about the recipient EDC's current load (p) after identifying the authenticity of received data packets. More importantly, recipient authenticated EDCs respond to this request only if they have enough resources ($p \leq 0.6$ & $L_i <<$ (n-m)) to handle the multiple tasks.

To conclude, our method not only secures the process but also identifies the less loaded recipient EDCS to share loads.

4.2 Forward Secrecy

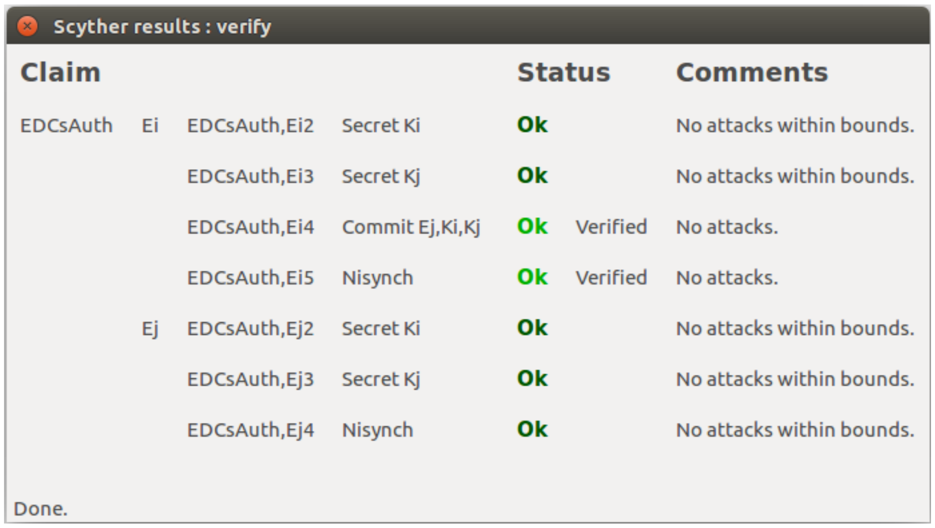
By following a standard symmetric key cryptography procedure for initial authentication of EDCs, shared keys are used for a specific period of time before they are broken by any potential attacker [3]. EDCs use the same key to verify the recipient EDC authentication with cloud. As the authentication process is happens only once at the beginning of the EDCs initialization. However, if an intruder gets the key for authentication, it is useless after initial authentication. Asymmetric key technique (i.e. public/private keys) in initialized to encrypt the information in load balancing.

4.3 Formal Security Verification

We use the Scyther simulation environment [14] for the formal verification of our proposed secure authentication scheme. The security methods and flows are designed using Security Protocol Description Language (.spdl) as supported by Scyther. The roles of *Ei* and *Ej* are defined following the Scyther features; where the authentication initiator (EDC-I) is defined by *Ei* and the destination EDC for authentication (EDC-J) is defined by *Ej*. Our verification scenario considers that both *Ei* and *Ej* possess the security information, which have been initiated by the cloud. The process begins with *Ei* sending the packets to *Ej* and accordingly *Ej* responds to *Ei* with the load information. During this verification scenario we introduce an authentication attack where a malicious adversary acquires the authentication details of *Ei* and using it sends a malicious packet to *Ej* in order to initiate the load balancing process. This scenario is run 100 times with checks at intervals of 10 to verify any possible attack on authenticity. In addition, the default properties of the Scyther tool is also used to run the simulation.

Even though an attack model can comprise of various possible attacks, for this particular scenario we focus more towards authentication attacks. Authentication attacks can involve an attacker being able to observe the communication between EDCs in order to discover and ma authentication patterns. Our assumption is that any malicious EDC is able to observe and replicate the behavior of an authenticated EDC for load balancing. As discussed previously, our proposed solution uses trusted modules such as the TPM on an EDC to store sensitive information such as the rekeying process and the secret keys.

We ran the experiment using the Scyther environment for 100 iterations with checking at intervals on 10, as described above. The simulation did not result in any successful authentication attack during the entire runtime. Figure 4 displays the security verification result obtained from the Scyther environment showing that the proposed security solution is secured against authentication attack.



Scyther results : verify

Claim				Status		Comments
EDCsAuth	Ei	EDCsAuth,Ei2	Secret Ki	Ok		No attacks within bounds.
		EDCsAuth,Ei3	Secret Kj	Ok		No attacks within bounds.
		EDCsAuth,Ei4	Commit Ej,Ki,Kj	Ok	Verified	No attacks.
		EDCsAuth,Ei5	Nisynch	Ok	Verified	No attacks.
	Ej	EDCsAuth,Ej2	Secret Ki	Ok		No attacks within bounds.
		EDCsAuth,Ej3	Secret Kj	Ok		No attacks within bounds.
		EDCsAuth,Ej4	Nisynch	Ok		No attacks within bounds.

Done.

Figure 4. Scyther formal security verification result page.

5. Experiment and Results

We have evaluated the performance of the proposed architecture in two different environments such as in Matlab simulation environment and in real time testbed.

5.1 Simulation Findings

We have used Matlab simulation environment to evaluate the performance of our proposed load balancing solution [15]. The simulation was run on a Dell system with Intel Core i7 processor and 8GB of RAM. Each simulation instance was run 10 times and an average value of the results obtained has been used for the validation of our proposed scheme. For the experiment, 10 EDCs were initialized to evaluate the overall performance of the proposed scheme. We have considered task arrival rate λ_i with Poisson arrival process. The simulation process EDC-I is assumed to be currently overloaded and would perform load balancing upon receiving additional tasks. Once EDC-I receiver the additional tasks, it starts the load balancing process by initiation the authentication to receive the destination EDC's load information in order to locate an idle or less loaded EDC to allocate the tasks. Once EDC-I receives the load information from its neighboring EDCs, it can assign the tasks to the least loaded ones. The simulation also includes and evaluates benchmarks; random allocation, proportional allocation, static and greedy allocation load balancing technique. Along with benchmarking approach, we compared with one recently published approach i.e CTOM [22]. For the random allocation technique, the mobile cloudlet offloads its tasks to a randomly selected neighbor. In comparison, the proportional allocation technique uses a global load information query and selects the most optimal option from its neighbors to offload the tasks. Greedy load balancing technique follows greedy algorithm from graph theory to allocate tasks to recipient EDC [23]. However, the static allocation technique allocated the task to the same specified destination every time. Finally, CTOM follows ball and bin theory to for efficient load balancing.

The initial results from the simulation are generated for our proposed load balancing solution where a suitable EDC is selected based on its current load. This is achieved using the same scenario simulation setup as described above. Figure 5 displays the simulation results for a successful destination EDC discovery. The success of finding a destination EDC is measured in percentage (%) and as shown in the results, the proposed load balancing solution obtained a 100% success rate. Proposed method collects recipient EDC's load information at the time of authentication. After successful authentication, overloaded EDC select less loaded EDC as the recipient for load balancing. As a result, proposed method achieves sustainable load balancing without addition communication. Due to the information collected by the proposed scheme to find a suitable destination EDC, it is able to achieve sustainable load balancing without additional communication. When compared to other schemes, the CTOM, proportional and greedy allocation scheme performed better than the random and static schemes as it also takes into account the load on the destination EDC before allocation it a task. We also discovered that the proposed scheme performed consistently with the increase in the number of tasks. Hence, we can imply that the proposed load balancing solution is bothe secure and efficient is destination EDC selection for load sharing.

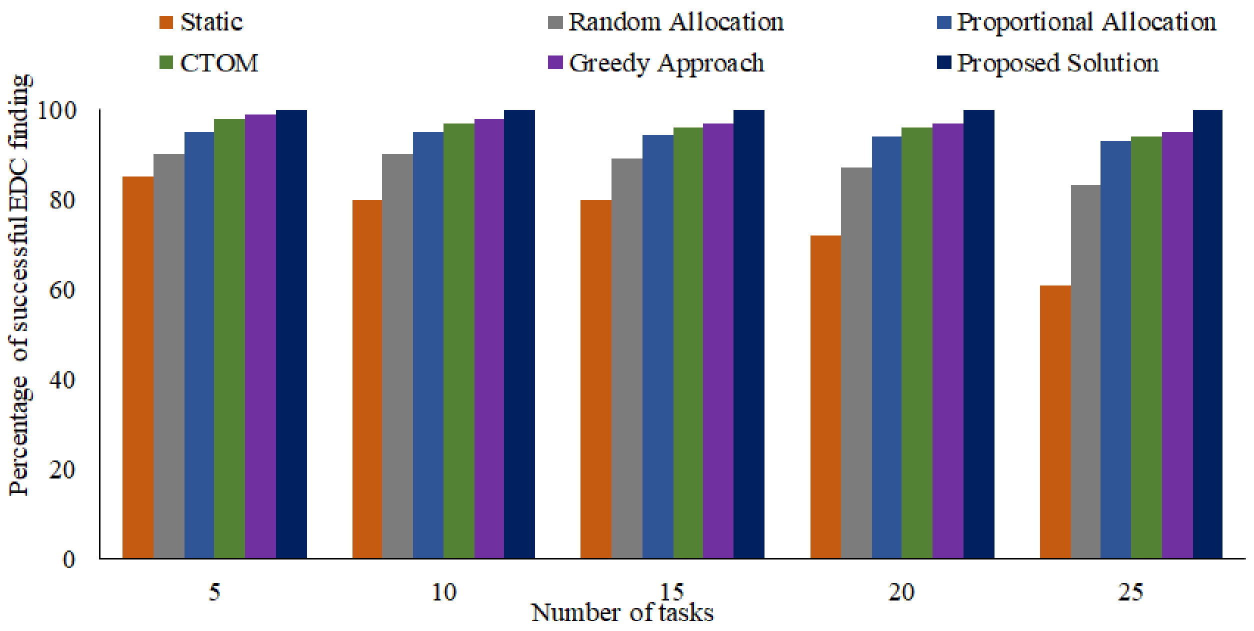


Figure 5. Percentage of successfully selecting the destination EDCs for load balancing.

Response time of the destination EDC also plays a vital role in improving the efficiency of the overall processing time. We also considered the same simulation setup as above to compute the response time. In a similar way, we compared the performance of the proposed load balancing solution with CTOM, static, random, proportional and greedy allocation. The result of the response time performance metric is shown in Figure 6. From the result, we found that proportional and greedy allocation always provides better performance compared to the other two existing techniques. However, the proposed load balancing performs better than the proportional allocation technique. At the same time, the proposed solution also authenticated the destination EDCs before load balancing. This shows that the proposed load balancing solution has better response time compared to other existing techniques even after the secure authentication process.

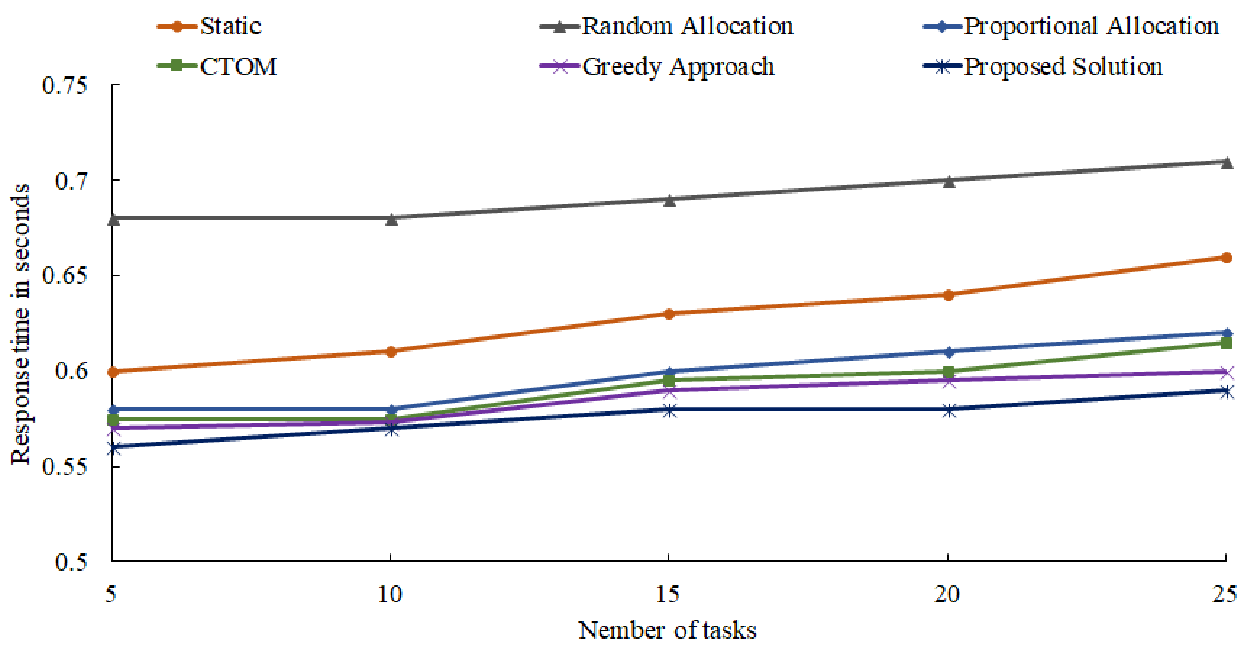


Figure 6. Response time of destination EDC for load balancing.

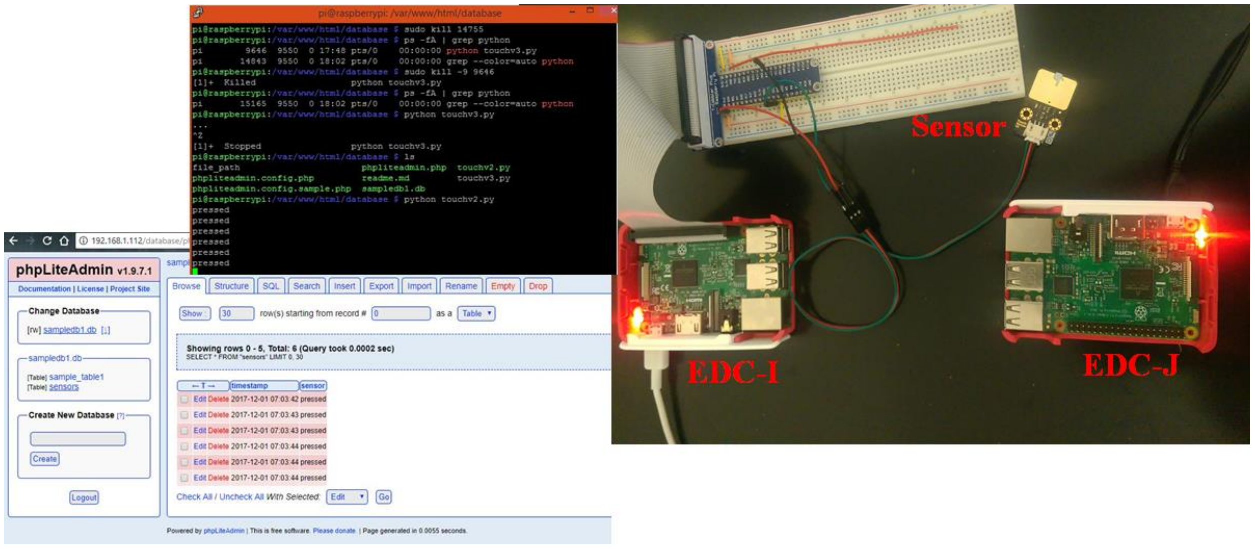
This paper integrated the authentication mechanism with load balancing for first time in distributed edge datacenters. So, prefer to compare performance of our proposed method with the benchmark schemes such as static, random, greedy and proportional allocation. Along with that, we also found our technique perform better than CTOM method. Finally, we conclude that, our method performs efficiently for load balancing even after additional authentication properties.

5.2 Testbed Implementation

The overall real-time evaluation of the proposed method is evaluated in this subsection. This testbed evaluation section is divided into two parts such as testbed configuration and testbed performance as follows.

5.2.1 Testbed Configuration

We have deployed the testbed in a real-time environment to compute the performance of the model proposed in this paper. This paper has considered Raspberry Pi as the edge datacenters and a computer with DigitalOcean cloud setup. The cloud configured computer configuration is a Dell computer with Intel Core i7 processor and 8 GB RAM. The Raspberry Pi configuration is 1.2 GHz 64/32-bit quad-core ARM Cortex-A53 CPU, 1 GB LPDDR2 RAM at 900 MHz and Broadcom BCM2837 system-on-chip. All the devices (i.e. both cloud and Raspberry Pi) are connected to the internet for recent updates and EDCs (Raspberry Pi) are in wifi communication to share their loads. In order to pass loads, there are three sensors (such as temperature, humidity and touch sensors) connected to a Raspberry Pi (EDC-I) to collect data streams for evaluation. Where, EDCs work as intermediate devices to cloud with limited resources. When the first Raspberry Pi (EDC-I) is overloaded it sends a request to other two Raspberry Pis for sharing loads. Cloud has been deployed with MySQL database and Kafka server to process data, whereas Raspberry Pi deployed with lightweight MySQL (SQLite) database and Kafka server to process data. Figure 7 shows the complete setup for the testbed performance.



```
pi@raspberrypi: /var/www/html/database
pi@raspberrypi:/var/www/html/database $ sudo kill 14755
pi@raspberrypi:/var/www/html/database $ ps -fA | grep python
pi        9646  9550  0 17:48 pts/0    00:00:00 python touchv3.py
pi       14843  9550  0 18:02 pts/0    00:00:00 grep --color=auto python
pi@raspberrypi:/var/www/html/database $ sudo kill -9 9646
[1]+  Killed                  python touchv3.py
pi@raspberrypi:/var/www/html/database $ ps -fA | grep python
pi       15165  9550  0 18:02 pts/0    00:00:00 grep --color=auto python
pi@raspberrypi:/var/www/html/database $ python touchv3.py
...
^Z
[1]+  Stopped                 python touchv3.py
pi@raspberrypi:/var/www/html/database $ ls
file_path                     phpliteadmin.php  touchv2.py
phpliteadmin.config.php       readme.md         touchv3.py
phpliteadmin.config.sample.php  sampledb1.db
pi@raspberrypi:/var/www/html/database $ python touchv2.py
pressed
pressed
pressed
pressed
pressed
pressed
```

192.168.1.112/database/p

phpLiteAdmin v1.9.7.1

Documentation | License | Project Site

Change Database: [rw] sampledb1.db [↓]

sampledb1.db: [Table] sample_table1, [Table] sensors

Create New Database [?]

Showing rows 0 - 5, Total: 6 (Query took 0.0002 sec)
SELECT * FROM "sensors" LIMIT 0, 30

←T→	timestamp	sensor
Edit Delete	2017-12-01 07:03:42	pressed
Edit Delete	2017-12-01 07:03:43	pressed
Edit Delete	2017-12-01 07:03:43	pressed
Edit Delete	2017-12-01 07:03:44	pressed
Edit Delete	2017-12-01 07:03:44	pressed
Edit Delete	2017-12-01 07:03:44	pressed

Powered by phpLiteAdmin | This is free software. Please donate. | Page generated in 0.0055 seconds.

Figure 7. Testbed implementation of EDCs for load balancing

5.2.2 Testbed Performance

All the Raspberry Pis are initialized with predefined shared and public/private key pairs by omitting the initial authentication process in our testbed. The Raspberry Pi (EDC-I) connected to sensors receive the continuous data streams. EDC-I store sensing data in the data base and evaluate in real-time. We set the threshold at 60% of the total size of the database, once the amount of data reaches that threshold value, the initial Raspberry Pi (EDC-I) starts sending requests to other authentication Raspberry Pis (EDCs) in the communication range to share loads as shown in Figure 7. The response time for load balancing between the EDCs is shown in Figure 8. This figure concludes that the response time always depends on the network size, so it is showing small response time while we are considering a small network using only three Raspberry Pis. As is also shown in the simulation results from Figure 6, the proposed load balancing achieves comparatively better performance than standard techniques. We have achieved equivalent results from testbed implementation. The last bar (named Proposed Solution) combines the authentication technique during load balancing. Even after addressing secure authentication, the proposed technique performs better in comparison to other traditional techniques.

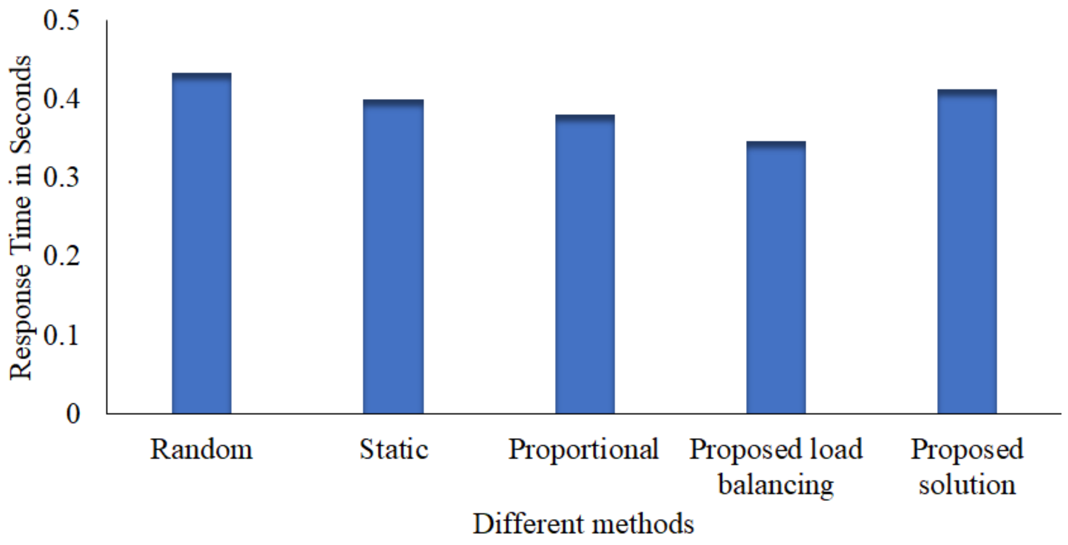


Figure 8. Load balancing performance in a simple three EDC testbed implementation

From the above theoretical and experimental evaluation, we conclude that the proposed load balancing solution is not only sustainable, but also secured. This improves the load balancing performance of EDCs in fog computing environments. This testbed implementation aims to provide the validation of proposed technique in real-time networks. The performance of proposed technique does not achieve high performance compared to standard techniques. However, proposed technique performs efficiently after introducing security mechanism to load balancing i.e. our technique is secure and sustainable in load balancing.

6. Conclusion

This paper presented our proposed load balancing solution for EDCs that is both secure and sustainable and oriented towards a fog computing environment. The proposed load balancing solution is a combination of two major components; the first component focuses on providing a secure authentication of the EDCs in the locality using a cloud initiated credentials while the second component enhances the performance by collecting information for improving load balancing to avoid overloading of some EDCs while other EDCs are idle or on a low load. We conduct a theoretical analysis along with an experimental evaluation to evaluate our proposed load balancing solution. Based on the results obtained from the performance evaluation and comparison, we can conclude that our proposed solution is both secure and sustainable. This is achieved by collecting the current workload on each destination EDC during the authentication process. In addition, the proposed scheme also addresses the security concerns related to the remote deployment of the EDCs by proposing security solutions that prevent outsider attacks and the authentication scheme that avoids malicious EDCs from gaining access to the network.

The future scope of the proposed solution is in expanding the research areas and proposing lightweight security solutions to improve the security as well as the load balancing performance and efficiency for EDCs in fog computing environments. In addition, the authors also aim to build a real-time testbed in order to implement the proposed security and load balancing schemes.

References

[1] O. Osanaiye, S. Chen, Z. Yan, R. Lu, K-K R. Choo, and M. Dlodlo. "From Cloud to Fog Computing: A Review and a Conceptual Live VM Migration Framework." *IEEE Access*, Vol. 5, pp. 8284-8300, 2017.

[2] A. Alakeel. "A guide to dynamic load balancing in distributed computer systems." International Journal of Computer Science and Information Security, Vol. 10, no. 6, pp. 153-160, 2010.

[3] D. Puthal, S. Nepal, R. Ranjan, and J. Chen. "DLSeF: a dynamic key-length-based efficient real-time security verification model for big data stream." *ACM Transactions on Embedded Computing Systems,* Vol. 16, no. 2, pp. 51, 2017.

[4] D. Puthal, X. Wu, S. Nepal, R. Ranjan, and J. Chen. "SEEN: A Selective Encryption Method to Ensure Confidentiality for Big Sensing Data Streams." *IEEE Transactions on Big Data*, (In press),2017.

[5] M. Jia, W. Liang, Z. Xu, and M. Huang, “Cloudlet load balancing in wireless metropolitan area networks,” in *Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on*, pp. 1–9, 2016.

[6] M. Willebeek-LeMair, and A. Reeves. "Strategies for dynamic load balancing on highly parallel computers." *IEEE Transactions on parallel and distributed systems*, Vol. 4, no. 9, pp. 979-993, 1993.

[7] L. Tong, Y. Li, and W. Gao. "A hierarchical edge cloud architecture for mobile computing." In *Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on*, pp. 1-9, 2016.

[8] Y. Zhang, L. Deng, M. Chen, and P. Wang. "Joint Bidding and Geographical Load Balancing for Datacenters: Is Uncertainty a Blessing or a Curse?" In *Computer Communications, IEEE INFOCOM 2017-The 36th Annual IEEE International Conference on*, pp. 1-9, 2017.

[9] D. Puthal, S. Nepal, R. Ranjan, and J. Chen. "Threats to networking cloud and edge datacenters in the Internet of Things." *IEEE Cloud Computing*, Vol. 3, no. 3, pp.64-71, 2016.

[10] I. Butun, M. Erol-Kantarci, B. Kantarci, and H. Song. "Cloud-centric multi-level authentication as a service for secure public safety device networks." *IEEE Communications Magazine*, Vol. 54, no. 4, pp. 47-53, 2016.

[11] D. He, and S. Zeadally. "Authentication protocol for an ambient assisted living system." *IEEE Communications Magazine*, Vol. 53, no. 1, pp. 71-77, 2015.

[12] D. He, S. Zeadally, N. Kumar, and J-H. Lee. "Anonymous authentication for wireless body area networks with provable security." *IEEE Systems Journal* (In Press), 2016.

[13] D. Puthal, S. Mohanty, P. Nanda, and U. Choppali. "Building Security Perimeters to Protect Network Systems against Cyber Threats." *IEEE Consumer Electronics Magazine*, Vol. 6, no. 4, pp. 24-27, 2017.

[14] Scyther, [Online] (accessed on: November 1, 2017). http://www.cs.ox.ac.uk/people/cas.cremers/scyther/

[15] Matlab, [Online] (accessed on: November 1, 2017). http://au.mathworks.com/products/matlab/

[16] D. Puthal, M. S. Obaidat, P. Nanda, M. Prasad, S. Mohanty, and A. Y. Zomaya. " Secure and Sustainable Load Balancing of Edge Datacenters in Fog Computing" *IEEE Communications Magazine*, Vol. 56, no. 5, pp. 60-65, 2018.

[17] R. Tripathi, S. Vignesh, V. Tamarapalli, A. T. Chronopoulos, and H. Siar. "Non-cooperative power and latency aware load balancing in distributed data centers." *Journal of Parallel and Distributed Computing*, Vol.107, pp.76-86, 2017.

[18] C. Qu, R. N. Calheiros, and R. Buyya. "Mitigating impact of short-term overload on multi-cloud web applications through geographical load balancing." *Concurrency and Computation: Practice and Experience*, Vol. 29, no. 12, pp. 1-15, 2017.

[19] A. Paya, and D. C. Marinescu. "Energy-aware load balancing and application scaling for the cloud ecosystem." *IEEE Transactions on Cloud Computing*, Vol. 5, no. 1, pp. 15-27, 2017.

[20] M. Jan, P. Nanda, M. Usman, and X. He. "PAWN: a payload-based mutual authentication scheme for wireless sensor networks." *Concurrency and Computation: Practice and Experience*, Vol. 29, no. 17, pp. 1-10, 2017.

[21] K-K R. Choo, R. Lu, L. Chen, and X. Yi. "A foggy research future: Advances and future opportunities in fog computing research." *Future Generation Computer Systems,* Vol. 78, no. 2, pp. 677-679, 2018.

[22] X. Fan, X. He et al. "CTOM: Collaborative Task Offloading mechanism for mobile cloudlet networks" In *IEEE International Conference on Communications (IEEE ICC),* Kansas City, MO, USA 2018.

[23] L. A. Wolsey. "An analysis of the greedy algorithm for the submodular set covering problem." *Combinatorica*, Vol. 2, no. 4, pp. 385-393, 1982.

[24] A. Bundy, and L. Wallen. "Breadth-first search." In *Catalogue of Artificial Intelligence Tools*, pp. 13-13. Springer, Berlin, Heidelberg, 1984.

[25] D. Georgakopoulos, P.P. Jayaraman, M. Fazia, M. Villari, and R. Ranjan. "Internet of Things and edge cloud computing roadmap for manufacturing." *IEEE Cloud Computing*, Vol. 3, no. 4, pp. 66-73, 2016.