Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

A Framework for Orchestrating Secure and Dynamic Access of IoT Services in Multi-Cloud Environments

Muhammad Kazim, Lu Liu, Member, IEEE, Shao Ying Zhu, Member IEEE, and Yongjun Zheng

School of Electronics, Computing and Mathematics, University of Derby, Derby UK

Corresponding authors: M. Kazim (e-mail: m.kazim@derby.ac.uk), L. Liu (l.liu@derby.ac.uk).

ABSTRACT IoT devices have complex requirements but their limitations in terms of storage, network, computing, data analytics, scalability and big data management require it to be used it with a technology like cloud computing. IoT backend with cloud computing can present new ways to offer services that are massively scalable, can be dynamically configured, and delivered on demand with largescale infrastructure resources. However, a single cloud infrastructure might be unable to deal with the increasing demand of cloud services in which hundreds of users might be accessing cloud resources, leading to a big data problem and the need for efficient frameworks to handle a large number of user requests for IoT services. These challenges require new functional elements and provisioning schemes. To this end, we propose the usage of multi-clouds with IoT which can optimize the user requirements by allowing them to choose best IoT services from many services hosted in various cloud platforms and provide them with more infrastructure and platform resources to meet their requirements. This paper presents a novel framework for dynamic and secure IoT services access across multi-clouds using cloud on-demand model. To facilitate multi-cloud collaboration, novel protocols are designed and implemented on cloud platforms. The various stages involved in the framework for allowing users access to IoT services in multi-clouds are service matchmaking (i.e. to choose the best service matching user requirements), authentication (i.e. a lightweight mechanism to authenticate users at runtime before granting them service access), and SLA management (including SLA negotiation, enforcement and monitoring). SLA management offers benefits like negotiating required service parameters, enforcing mechanisms to ensure that service execution in the external cloud is according to the agreed SLAs and monitoring to verify that the cloud provider complies with those SLAs. The detailed system design to establish secure multi-cloud collaboration has been presented. Moreover, the designed protocols are empirically implemented on two different clouds including OpenStack and Amazon AWS. Experiments indicate that proposed system is scalable, authentication protocols result only in a limited overhead compared to standard authentication protocols, and any SLA violation by a cloud provider could be recorded and reported back to the user.

INDEX TERMS authentication, IoT, IoT services, multi-clouds, security, secure collaboration, service level agreement, service matchmaking

I. INTRODUCTION

The Internet of Things (IoT) paradigm has revolutionized the IT industry by bringing together technologies such as Radio Frequency Identification (RFID), Wireless Sensor and Actor Networks (WSANs) and ubiquitous computing domains. Internet of Things (IoT) connects billions of devices over the Internet. The heterogeneous IoT objects are provided with sensing and actuation capabilities, that enable them to capture information from physical objects and send it as data streams [1]. Moreover, IoT objects directly co-operate with physical and virtual resources over the internet to deliver data and functionalities to end users and applications. IoT has played a critical role in advancing human lives by bringing applications with usage in the real

world. From users perspective, IoT plays a critical role in application scenarios such as smart homes, healthcare, vehicular networks, and enhanced learning. While from the business viewpoint, the major applications of IoT are in the areas of logistics, transportation, agriculture, retail and smart cities. It is predicted that the growth of the global IoT services market will be at a compound annual growth rate (CAGR) of 24 percent until 2021 [2]. As the number of IoT devices increases and they generate large volumes of big data, it brings forwards the challenges related to data collection, analysis, management, and storage.

Cloud computing has been proposed as a solution that can potentially solve the problem of managing big data in IoT [3]. Some key advantages cloud computing offers are that: it is massively scalable, can be dynamically configured, delivers on-demand services and provides users with immediate access to hardware resources without capital investments [4]. Different companies using cloud have infrastructures that scale over several data centres and cloud also has a simple pricing model that lets you pay as you go and only for the services they are being used. Due to these advantages, cloud vendors including Google (Google Cloud IoT), Amazon (AWS IoT) and Microsoft (Azure IoT Solution Accelerators) are offering services to support IoT devices and services in terms of computing, storage, resource elasticity and data analytics. Despite the benefits offered by the cloud to IoT sector, the variety and proliferation of services offered by the cloud provider raise some challenges relevant to cloud environment. These challenges include portability issues of IoT services on various IaaS and PaaS platforms, interoperability of distributed IoT applications on various cloud platforms, PaaS dealing with the heterogeneity of cloud protocols to support IoT service interactions, and the requirement of geo-diverse platforms [5].

The multi-cloud architecture can provide a solution to these challenges. Multi-cloud environment is dependent on multiple clouds, and a user can be reliant on multiple cloud service providers such as Amazon, Microsoft, or OpenStack which are communicating. IoT applications can benefit from the adoption of multi-clouds from their abilities to run workloads on best-suited platforms, avoiding the need to migrate legacy IoT applications and creating redundancy to avoid vendor lock-in [6]. Multi-cloud providers are increasingly in demand. In a survey by 451-Microsoft, around 50% companies' representatives were looking for providers that could provide one-stop-shopping from various cloud providers and establish contracts with different providers for additional services on their behalf [7].

Multi-clouds provide an increased level of efficiency to cloud providers by enabling them to share their services for improving revenues. In terms of IoT, the services to be shared between multi-clouds can include SaaS, PaaS or IaaS service while the clients using these services can be other clouds, organizations or a single user. Other factors driving the adoption of multi-clouds for cloud provider can vary from dealing with a peak in service requests, having backup servers to diminish downtime scenarios and enhancing its own offers to get a market competitive edge.

In a multi-cloud environment, users access services across multiple cloud providers which changes the traditional cloud landscape. However, a very limited research has been done to support IoT services deployment and access across multi-clouds. Therefore, advanced development frameworks are required that can offer IoT services orchestration across multi-clouds and reduce companies time-to-market to keep cloud services running smoothly. Along with the service orchestration issues in multi-clouds, many security concerns are also related to their adoption and application. The basic authentication solutions that exist for traditional networks fail to meet the need of a dynamic collaboration of clouds and services (such as IoT services) in multi-clouds. Consider a scenario in which a cloud (local cloud) user is accessing IoT service located in another cloud (foreign cloud). That cloud user would have no mechanism to verify that the service being used is trustworthy and neither do they have insights on what is happening with their data being handled by services. In order to trust the cloud services, users depend on their assurances given by the cloud provider. Cloud providers give very limited evidence or accountability to users which offers them the ability to hide some behavior of the service.

In order to address these challenges, we propose a novel framework named called Multi-cloud Collaboration for IoT (MC-IoT) in this paper that can facilitate multi-cloud collaboration and provide guarantees to the user that the software or service (such as IoT service) running on a foreign cloud node is secure and the agreed service level agreements (SLAs) are not being violated. The key challenge in designing this framework is to develop solutions for multi-cloud that can support efficient authentication, authorization of large number of cloud users, enable the users to select most suitable service in foreign cloud according to their requirements as well as to ensure that services in foreign cloud are compliant with the service level agreement (SLA) between user and cloud provider. The proposed framework is based on NIST cloud computing security architecture standard [8]. It satisfies the following conditions: i) rapid provisioning by automated service deployment; ii) mapping authenticated and authorised data and tasks onto VMs; iii) monitoring the cloud resources, operations and performance; iv) metering active user accounts to guarantee that security policies are always enforced; v) maintaining the service level agreement (SLA) established between customers and service providers.

In an IoT based multi-cloud architecture, hundreds of cloud users might be using thousands of IoT services across multi-clouds. The basic authentication solutions that exist

for traditional networks fail to meet the need for a dynamic collaboration of clouds and services in multi-cloud. Therefore, this paper provides a lightweight and novel technique for the dynamic authentication which provides single sign-on to users trying to connect to the foreign cloud. The proposed authentication solution achieves better performance than traditional authentication protocols like SAML and Kerberos while maintaining security. Next, we provide a service selection algorithm to select the best IoT service from multiple cloud providers that best match user quality of service requirements (QoS). In the next stage, service level agreements (SLAs) are used to ensure security and handle service execution in the foreign cloud. The usage of SLA mechanisms ensures that QoS parameters including the functional (CPU, RAM, memory etc.) and non-functional requirements (bandwidth, latency, availability, reliability etc.) of users for a particular IoT service are negotiated and secure collaboration between multi-clouds is setup. The multi-cloud handling user requests will be responsible to enforce mechanisms that fulfill the QoS requirements agreed in the SLA. While the monitoring phase in SLA involves monitoring the IoT service execution in the foreign cloud to check its compliance with the SLA and report it back to the user.

MC-IoT has been designed with the goal to enhance secure multi-cloud collaboration in which cloud providers can easily apply their business model to achieve extended functionalities. The proposed model is based on an architectural solution that can be used to setup multi-cloud collaboration between any clouds irrespective of their underlying implementation. Experiments indicate that the proposed approach supports collaboration among a large number of IoT services across multi-clouds and incurs a minor overhead.

The major contributions of this paper are the following:

- A novel framework is proposed for providing users secure dynamic collaboration and access to IoT services in multi-clouds. The protocols to support the framework and the functionalities of its components responsible for multi-cloud collaboration are presented.
- Dynamic and lightweight authentication protocol to setup single sign-on (SSO) between multi-clouds has been presented.
- A service selection algorithm is proposed that achieves high accuracy by providing distance correlation weighting mechanism among large number of IoT services QoS parameters.
- Mechanisms to setup service level agreements (SLAs) for multi-cloud collaboration have been presented. The various stages in setting up SLA include negotiation, enforcement and monitoring. They help in negotiating QoS parameters for IoT services between the user and foreign cloud provider, enforce a mechanism to comply with the agreed SLAs, monitor client usage of IoT service in the foreign cloud and report back any violation of SLA.
- Business and use cases have been presented to discuss how the proposed framework can be used in various applications

This paper has been organized as follows: Section 2 presents the background of this work and section 3 presents framework design with the detailed description of various protocols for authentication, service selection and SLA management. In section 4 the workflow of system components has been presented. Experimental results of our system have been given in section 5. Section 6 provides the use cases of this work and section 7 details the literature review related to the area of multi-clouds, and IoT based cloud systems. In the end, the conclusion of the paper is presented.

II. BACKGROUND

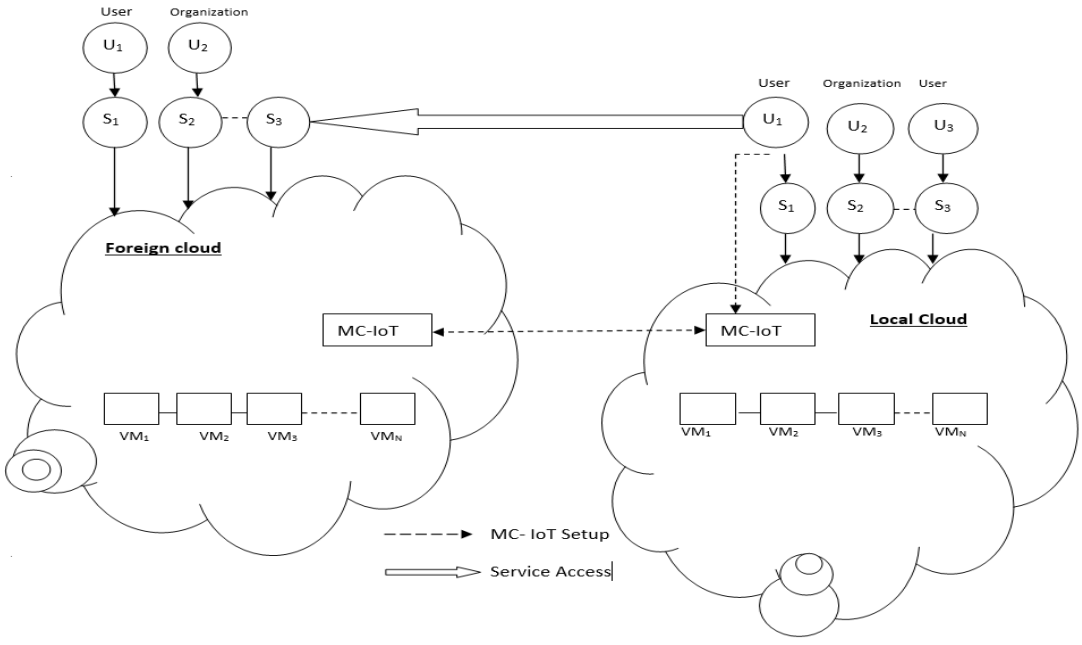

Figure 1: Multi-cloud collaboration scenario where user U_1 made a request to MC-IoT in local cloud to access service S_3 in foreign cloud. The multi-cloud collaboration is setup using MC-IoT after which U_1 can directly access service S_3.

Multiple clouds have two delivery models which are federated cloud and multi-cloud. In federated clouds, there is an agreement between different providers that want to collaborate, and also the user is not aware of the fact if the resources are being used from another cloud. However, multi-clouds provide a way for dynamic collaboration between various clouds as there is no former agreement between participating clouds and collaboration is established at runtime according to requirements. Moreover, in multi-clouds user has the knowledge of all connected clouds and is directly responsible to the provisioning of services from multiple clouds which can be more beneficial from customers and organizations perspective. Therefore, this work focuses on multi-clouds so that users and organizations can dynamically access IoT services across various cloud providers. The multi-cloud communication scenario that provides access of IoT services to users across multi-clouds is shown in figure 1.

mechanisms across different clouds. Along with authentication problems, secure service orchestration is also a challenge. Once a service from a foreign cloud is being used, the cloud users have no mechanism to verify that the service they are using is trustworthy and depend on that provider to ensure service execution. As a user is accessing services in the foreign cloud, the interaction with malicious or faulty service can lead to the manipulation of data processing results, failure to provide advertised services, violation of the security properties such as confidentiality, integrity and availability, and other malicious activities without user consent.

The general mechanisms described in the literature on service security in cloud are based on having a guarantee that the software or service running on a cloud node is similar to its original implementation and it cannot be modified at runtime at foreign cloud. However, cloud customers cannot know if the service functionality has been

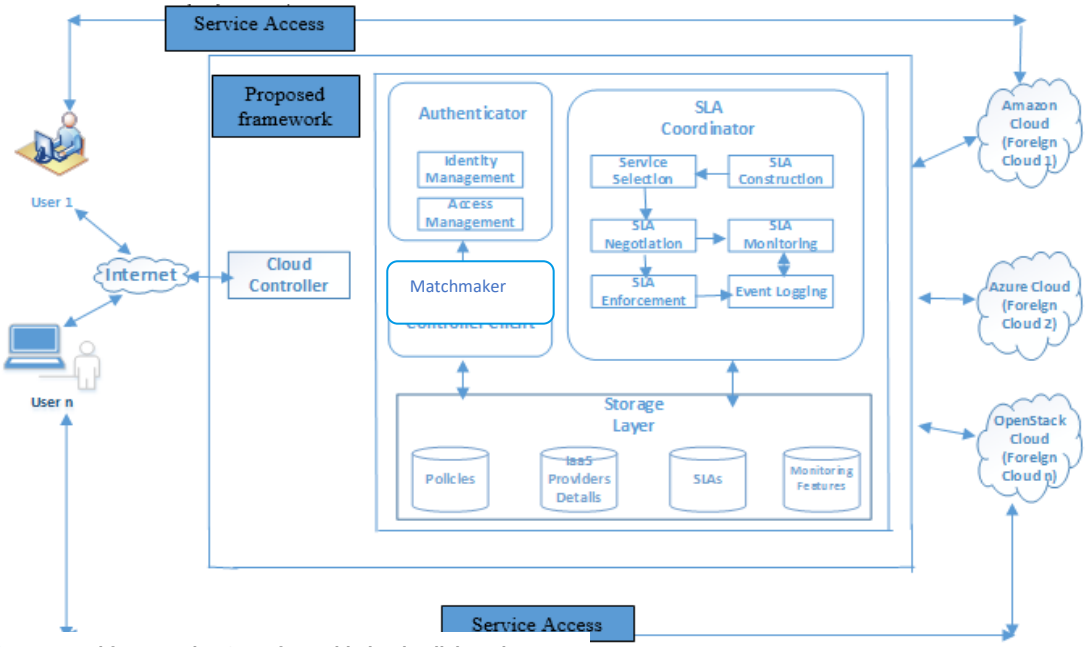

Figure 2: Proposed framework MC-IoT for multi-cloud collaboration

In a multi-cloud environment, the user's access IoT services across multiple cloud providers which changes the traditional cloud landscape**.** Multi-clouds offer greater agility, innovation and more intense collaboration and they can be predicted to become an industry norm to handle IoT big data and their associated applications, however, managing service orchestration is still an open issue. Advanced development frameworks are required that can reduce companies time-to-market.

Along with the service orchestration issues in multi-clouds, multi-clouds bring many security concerns as well. The traditional authentication solutions that exist for networks fail to meet the need of a dynamic collaboration of users and services in multi-cloud due to performance overhead and/or difference of underlying authentication

altered when using services to the foreign cloud. Therefore, advanced mechanisms are required that can support efficient IoT service selection according to client functional and non-functional QoS requirements, provide efficient and secure authentication, and enable service level agreement (SLA) management to ensure that proper mechanisms are implemented to comply with agreed SLA parameters, and monitor the service execution to guarantee that foreign cloud always complies with those parameters.

III. PROPOSED FRAMEWORK (MC-IoT)

Based on heterogeneous requirements of multi-clouds and IoT services, we propose a novel framework named MC-IoT that can enable dynamic collaboration between users and services in multi-clouds. The architecture of MC-IoT

involves various components that have been implemented in each participating cloud to achieve the secure multi-cloud authentication. These components involved in system design are authenticator for managing identity and authorization, controller to manage user requests and communication with external clouds (functionality mentioned in section 4), matchmaker to select suitable IoT service meeting user specifications and SLA coordinator for managing SLA negotiation, enforcement and authorization.

Figure 1 displays various components in a local cloud to communicate and collaborate with foreign clouds. All the system components serve different functions which are described below. In this paper, the communicating clouds are referred to as local cloud (in which user is located) and foreign cloud (to which user needs access and collaboration has to be established).

A. COLLABORATION OBJECTIVE

The objective of multi-cloud collaboration is the maximum number of user requests from local cloud to be handled and successfully granted access to services in foreign cloud. The overall objective for multi-cloud collaboration for M cloud users (j = 1, 2 … M) in local cloud, with requests of S IoT services (i = 1, 2 … N) in a foreign cloud can be formally defined as:

$$\mathbf{O\ (i,\ j) = Max}\ \sum_{i=1}^{N}\ \sum_{j=1}^{M}(R_{ij} - C_{ij})$$

In the above equation, N and M are the numbers of services requested and a number of users making requests respectively. R_{ij} is the required number of user's requests for IoT services to be granted while C_{ij} is the number of IoT services that were actually granted.

B. INITIALIZATION PROTOCOL

The initialization protocol is the first step that is used to set up the system services, parameters and attributes required for multi-cloud collaboration. When the required services are booted in MC-IoT, and user request for multi-cloud access is received, authentication service in the Authenticator component establishes if it has the certificate for that user that can be used for authentication with foreign clouds. If the certificate does not exist in the cloud, Authenticator which is a RESTful web service submits a request on behalf of its cloud to Trusted Party (TP) for certificate generation.

A feature of TP is to generate a certificate for cloud Trusted Party (TP) after receiving a request and cloud parameters and to use a function to map a certificate to client ID which is returned to the requesting cloud. The Authentication Service of cloud receives the certificate and stores it to be used for communication with foreign clouds. Similarly, it is the responsibility of Authenticator component to ensure that the certificate obtained from TP is valid and to get a new certificate if the existing one is revoked or rejected by foreign cloud.

Algorithm 1: System Initialization ()

```
1. BEGIN: Boot the required services to enable multi-cloud collaboration
2. while (the system is running)
3.     For i = 1 to n:
4.         LC (Auth_service) -> Check (Cert) // Checking client's cert
5.         if Valid (Cert):
6.                 goto 17
7.         else: // Request a new certificate to TP
8.                 Auth_service -> Send_request (Cert, ID) -> TP
9.                 LC -> Mapping_data(LC) -> TPi
10.                TPi -> Publish(Cert)
11.                TPi -> Send_certificate (Cert) -> LC
12.                LC (Auth_service) -> Receive (Cert)
13.        end if
14.        LC (Auth_service) -> Check (Cert)
15.        if Valid (Cert):
16.                wait (Request)
17.        else:
18.                goto 7
19.        end if
20.    end for
21. END
```

C. AUTHENTICATION PROTOCOL

This protocol describes how multi-cloud authentication is setup between participating clouds. In a distributed environment usually a large number of clouds are present with each cloud having tens of users, which makes credential management a big challenge. Moreover, in a dynamic communication setup between multi-clouds, each cloud might have different authentication mechanisms. This raises a need to develop a single sign-on (SSO) authentication mechanism by which any cloud user in the local cloud can authenticate itself with the foreign cloud, and access required resources. In our case, we use a Trusted Party (TP) which acts as an identity provider on which a requesting user must hold a digital identity, based on which TP grants a digital certificate to that user that it can use to authenticate with the foreign cloud. Since the foreign cloud also trusts TP, the user is able to authenticate itself and access resources based on that certificate.

We assume that the local cloud's request is composed of two parts namely the certificate and the required cloud service. Initially, a certificate is sent by the local cloud (LC) to the foreign cloud (FC) for proving its identity. This certificate contains a set of attributes including the cloud identifier, digital signature, and validity period of the certificate. This message is encrypted by the public key of FC.

FC checks the validity of the certificate sent by LC. If the certificate is valid, FC then sends a response message to LC that it is authenticated. However, if the certificate is invalid, FC sends the message of failed authentication to LC and waits for a new certificate. This message is encrypted with the public key of LC. In case the message received from FC is that authentication certificate was invalid, LC sends a message with its credentials to the

Trusted Party (TP) to generate a new certificate. TP generates a new certificate and sends it to the LC which is sent from LC to FC.

FC checks the new certificate received from LC. If the certificate is invalid again, the authentication request is terminated. If authentication of LC is successful, both FC and LC exchange nonce messages to agree on a session key using Diffie-Hellman (DH) algorithm [9]. Since DH key exchange is performed after certificate exchange, it is called authenticated DH which is more secure compared to usual DH.

Algorithm 2: Authentication and Authorization ()

```
1. BEGIN
2. Data: request: Communication request received by cloud controller
3. LC -> Send_request (authentication) -> FC //Secured using SSL
4. for j = 1 to n do:
5.      FC -> Verify (Cert, ID)
6.      if Verify (Cert, ID):
7.              goto 17
8.      else:
9.              FC -> Send_request (New_Cert) -> LC
10.     end if
11.     LC -> Send_request ((Cert),Profile) -> TP
12.     TP -> Send (Cert) ->CC //Generates updated certificate for
        LC and sends to LC]
13.     LC -> Send_msg (Cert) -> FC
14.     if Not_valid (Cert):
15.             End
16.     else:
17.             FC -> Send_msg(n) -> LC
18.     end if
19.     FC -> Wait (response) -> LC
20.     if no_resp():
21.             End
22.     else:
23.             LC -> Send_msg(n+1) -> FC encrypted using
        LC- FC session key generated by DH
24.             FC -> Send_msg(request_authorization) -> LC
        encrypted using LC-FC session key
25.             LC-> Send_msg(Send_LCAuthorization_level) -
        > FC encrypted using LC-FC session key
26.             FC -> compute_local_level (LC)
27.             if compute_local_level = True:
28.                     FC->Authentication_local (LC,FC,+)
29.             else:
                        FC ->Authentication_local (LC,FC,-)
30.             end if
31.     end if
27. end for
28. END
```

After cloud authentication, LC sends a message to FC containing client authorization details as well as resources required from FC. As FC receives details of IoT services which are to be accessed and required resources message, it locally computes if the tasks from LC have the authorization to access the required services. The corresponding FC computes the status of the IoT services associated to the request. The status is computed due to the fact that users on a local cloud can have the different status of privileges that can affect their level of access to resources. For example, only doctors might have access to some expensive IoT services and other hospital staff might not have access to them. The return result is one of the following possible statuses:

- Privileged
- Non-privileged

If the result returned is privileged users are granted access to services, otherwise they are not granted access.

D. SERVICE MATCHMAKING

The cloud services states can change dynamically during runtime. Moreover, the dynamic collaboration between users in multi-clouds can make service automatic detection complicated. Cloud customers can have varying requirements and in multi-cloud scenarios best services that can meet their required quality of service (QoS) need to be selected from various providers.

To select the most suitable services, the first goal is the efficient discovery according to the characteristics of services. Service discovery for dynamic multi-cloud collaboration could be hard due to requirements such as satisfying service QoS, functionalities and other metrics. Moreover, lack of central repositories for cloud services makes service selection a challenging task.

There might be cases when a single service would be able to satisfy all user requirements and the service that matches most requirements might need to be selected. This leads to partial matchmaking where the service that matches most required QoS criteria will be selected. In this section, we propose an efficient and dynamic algorithm for selection of cloud services in multi-cloud scenarios based on partial or closest matching of service QoS attributes. This protocol for service selection is an extension of our previous work on partial web service selection for disaster services [10].

The proposed protocol has three essential characteristics.

- Firstly, the proposed protocol provides service selection among all services in a dynamic decentralized environment of multi-clouds with high accuracy.
- Secondly, different QoS requirements of services can be supported. In case, there is no exact match of user QoS requirements with available services, services matching the most requirements are selected using partial matching.
- And thirdly, the protocol is able to support a large number of services and by using distance co-relation weighting mechanism it can support various IoT services QoS requirements such as response time, availability, reliability, cost, energy, throughput, latency and best practices.

Once a cloud controller receives a response from various foreign clouds that can deliver required services, it communicates with service matchmaking module to select the required service.

1) USER REQUEST AND SERVICE QOS ANALYSIS (PHASE 1)

The process of service selection starts with the cloud controller passing the requirements to service matchmaking module which includes the required service and desired QoS. Such as a user might require high throughput compared to cost saving while it might be opposite for another user. Moreover, the module collects the results of available services in the foreign cloud from the controller component.

Here we represent various denotations for request types:

- R_Q represents a set of user functional QoS requirements, $R_Q = \{q_1, q_2, q_3, \ldots., q_n\}$, where n ε N
- S is a set of available services with similar functionality, $S = \{s_1, s_2, s_3 \ldots s_m\}$, where m ε N
- Each service S has Q_S property matrices, $Q_S = \{Q_{S1}, Q_{S2}, Q_{S3} \ldots Q_{Si}\}$, where $Q_{Si} = \{q_{i1}, q_{i2}, q_{i3} \ldots. q_{ij}\}$, i, j ε N. Q_{Si} represents quality matrices for service i.

2) REQUIREMENT MATRIX CONSTRUCTION (PHASE 2)

Once the QoS requirements have been gathered, the module collects all possible service offers and their associated QoS parameters. These are used to construct an accuracy matrix and for the calculation of offers ranks.

In an ideal scenario, user QoS requirements Q_R must be similar to the service QoS parameters mentioned in $Q_{Si.}$ In other words, an ideal service for user request can be represented as,

$$\mathbf{R_Q = Q_{Si}}$$

However, in real case scenario that user requirements R_Q and the number of quality matrices Q_{Si} will be different. Therefore, R_Q is taken as a baseline and quality matrices could be arranged in the following way:

- If the quality service matrix Q_{Si} lacks in user $Q_{R,}$ it is removed and Q_R is assigned 0

To construct accuracy matrix, n consumer requests R_Q are identified along with m available services that can satisfy user requirements, an m*n matrix is constructed which is called R. The columns in the matrix represent QoS parameters R_Q while each available service is represented in a row for the selection process.

Requirement matrix, R can be defined as:

$$\begin{array}{c} \\ S_1 \\ S_2 \\ .. \\ S_m \end{array} \begin{array}{c} \begin{array}{cccc} R_{Q1} & R_{Q2} & \ldots & R_{Qn} \end{array} \\ \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ r_{21} & r_{22} & \cdots & r_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ r_{m1} & r_{m2} & \cdots & r_{mn} \end{pmatrix} \end{array}$$

A service not satisfying the mandatory QoS requirements R_Q is removed from the selection process.

3) ACCURACY MATRIX CONSTRUCTION (PHASE 3)

The calculation of accuracy matrix, A, is dependent on the tendency of QoS parameters. The tendency which describes how the numeric value of a service QoS parameters changes for a service to be observed as better. It indicates whether high or low values of a QoS parameter are preferred in an ideal case. For example, an ideal service will require availability and throughput parameters to be high while its response time and latency should be low.

Using the user described QoS range and service QoS offered, elements of the accuracy matrix is calculated using case dependent formulae as mentioned in equations below:

For values with high tendency:

$$\frac{Q_{ij}}{Q_1} \quad \text{when} \quad Q_{ij} < Q_1$$

$$\frac{Q_{ij} - Q_1}{Q_h - Q_1} + \alpha$$

$$\frac{Q_{ij}}{Q_{max}} + \beta \quad \text{when } Q_{ij} > Q_h$$

For values with low tendency:

$$\frac{Q_h}{Q_{ij}} \quad \text{when } Q_{ij} > Q_h$$

$$\frac{Q_h - Q_{ij}}{Q_h - Q_1} + \alpha \quad \text{when} \quad Q_1 \leq Q_{ij} \leq Q_h$$

$$\frac{Q_{min}}{Q_{ij}} + \beta \text{ when } Q_{ij} < Q_1$$

In the above equations, Q_{ij} is the value of ith QoS property of jth service, Q_1 is the lower limit of user requirements for an attribute, Q_h is the highest limit of user requirements for an attribute. Q_{max} and Q_{min} are respectively the maximum and minimum values of a QoS property being offered by a service. α and β belong to {1, 2, 3, …} where α < β. The results from the above equations are normalized in the range [0, 1].

α and β are used to differentiate between loose range, preferred range and tight range. The preferred range for any service is between Q_l and Q_h. If a value falls in this range, α is added to normalize the value so that results are in range (α, α + 1). The values in the loose range (between Q_{min} and Q_l for high tendency parameters, and between Q_h and Q_{max} for low tendency parameters) are normalized between 0 and 1. While the values in the tight range (between Q_h and Q_{max} for high tendency parameters, and between Q_{min} and Q_l for low tendency parameters) are normalized by adding β so that results are in the range (β, β +1). Therefore, for all the values in accuracy matrix lie between (0, β +1) which helps in consistency. Moreover, β > α which always guarantees that higher range always has a higher value in accuracy matrix than other two ranges.

The results of these equations are used to calculate the accuracy matrix, A. It shows how precisely each service matches the user requirements. After constructing the accuracy matrix, the rank of each service can be calculated in the following way:

$$\mathbf{R_i} = \sum_{j=1}^{n} A_{ij} * W_j$$

In the above equation, R_i represents the rank of service i, A_{ij} represents the accuracy value of the j^{th} QoS property of service i, and W_j represents the weight of the j^{th} QoS property.

Algorithm 3: Service Matchmaking ()

```
1. BEGIN
2. Data: Input: <Client functional and non-functional QoS
requirements (CR)>,
                        <List of services (LS)>
3. Service LS= {1, 2,…., n}; // Total list of available services
4. <Service,CR> ServiceContenderList (SCL) = NULL //List of
services satisfying requirements
5. Service S=NULL // Single service instance
6. CR Q=NULL //Single QoS requirement
7. <Service, CR> O=NULL;
8. For each S in LS do:
9.          if (Satisfy(S ,CR)) //Add to SCL all appropriate services
matching user requirements
10.                 SCL.add (S,CR)
11.         end if
12. end for
13. For each O in SCL do:
14.         for each Q in O.CR do:
15.                 Normalize (AccuracyMatrix (Max(Q),Min(Q) ))
            //Generate accuracy matrix
16.         end for
17. end for
18. For each O in SCL:
19.         Score = Calculate_Score( O.Service ) // Calculate score of
            each service
20. end for
21. SCL.sort(Score) // Rank all services in SCL
22. Return SCL
23. END
```

E. SLA NEGOTIATION

The SLA coordinator receives user requirements and SLA's from the foreign cloud and negotiates a dynamic SLA between them. These SLAs exist within the customer domain that wants to access foreign cloud resources. From the client viewpoint, SLAs define the mechanisms to securely access services while the SLAs are utilized by cloud administrators to manage the mechanisms to offer cloud services. SLA-coordinator negotiates the SLAs on behalf of the user if there is full match of QoS requirements in the stated SLAs. However, as described earlier there might be a partial match after which user can have the ability to negotiate SLAs itself. Therefore, SLA coordinator component in MC-IoT offers added features to customers such as negotiating an SLA or switching to a new provider in a multi-cloud scenario if selected provider and user cannot agree on an SLA.

As discussed earlier, the matchmaking component checks the service specs like base service, features, cost and recommends them to the user. SLA Negotiation involves agreeing to the service terms for SLA and QoS parameters, measuring metrics (service level objectives) and defining how the metrics will be measured. While service providers also check if they can provide requested service and perform basic risk evaluation in case. As provider reputation is on a stake if it fails to provide the service agreed in SLA.

Integrating the security parameters within SLAs is a novel problem and a very limited research has been done in this area. For the case of secure multi-cloud collaboration, we propose a service level objective (SLO) called service identity which can help customers to negotiate the SLAs for secure service execution on the foreign cloud.

1) SERVICE IDENTITY

Service identity as an important property to maintain strong IoT service security and compliance in the foreign cloud. A set j services F_j deployed on a single cloud platform with functional properties $Func_i$ and non-functional properties $NFunc_i$ can be defined as:

$$F_i = \{Func_i, NFunc_i\} \qquad 1 \le i \le j$$

During service execution in the foreign cloud, both functional and non-functional properties of service instances being used by users must be maintained. Functional properties of instances that could be violated include a change in the code or implementation of service to make it do certain other activities affecting the original behavior of service. While a few non-functional issues can include service taking more processing time, charging more cost than agreed or remaining unavailable during required times.

If F is the original service deployed by the service provider in cloud after agreeing SLAs and F' is the instance of that service running in cloud that is being used by client, the service identity can be satisfied only if *F=F'* holds true for that particular instance of F running in the cloud during the entire lifecycle of F from deployment to

decommissioning. The service identity can be described by the following equation:

$$F \equiv F' \quad \dots (a)$$

In order for functional properties of a service instance F' to hold, its functional properties must be the same as original instance. While the case for non-functional properties is more complex as the service states can change dynamically during runtime. Moreover, each user will have different QoS requirements from a service. As an example, users X and Y using different instances of F' of same service F can have varying availability, and cost requirements. Therefore, we define a threshold value for non-functional parameters of a service instance that it must maintain to ensure service identity.

The non-functional parameters of a service agreed in the SLA can be defined as a tuple:

$$NFunc = \{Min_i, Max_i, W_i\}\ 0 \leq i \leq l$$

i is the QoS parameter, Min and Max show the accepted boundaries or threshold values for that parameter, and W denotes the weight assigned to a particular parameter by a user which shows the importance given to that parameter by a user. The range of W is [0,1] with the higher value showing that parameter is important for the user and it will have a larger impact on service quality, and vice versa for the lower value. In case a user does not define i, medium importance is given to that parameter and for that purpose a medium value is chosen in the range of W which is 0.5. For non-functional properties to hold true in an instance, the following condition must be satisfied at all times:

$$Min_i \leq NFunc_i \leq Max_i \dots (b)$$

To comply with functional requirements such as security different techniques can be agreed in the SLA which can ensure that functional behavior of service instances F' will not change. For example, to maintain service identity trusted platform module (TPM) mechanism could be used. The functional property of a service could be defined as:

$$F\text{-}F' = \emptyset \dots (c)$$

If both equations (b) and (c) hold than equation (a) will hold. However, in case if service security is compromised than the equation will become $F' \supset F$ meaning that service identity does not hold.

Meanwhile, various authors have proposed definitions of other functional and non-functional metrics (SLOs) for IoT services that can be agreed between customer and provider during SLA negotiation. These parameters include request latency, availability, accessibility, service throughput, completion time, and mean times to repair and failure, energy cost and financial cost.

The proposed system uses WSDL to express the functional security requirements and non-functional requirements. The XML data structures are generated on the basis of WSDL document, the service interface definition and its implementation. Therefore, QoS tags are associated with a new category to recognize security and other properties. The protocols for SLA management are implemented in the form of a REST based service and API.

F. SLA ENFORCEMENT

Once a user is authorized to access cloud resources, next stage is the enforcement of security mechanisms by the provider. In this stage, mechanisms are implemented that can guarantee SLA assurances. The enforcement of agreed SLA is done in two stages. The first stage involves implementing the software modules that can be activated for the acquisition of resources for enforcing security policies and second stages involve dynamic reconfiguration of the resources after a security alert is generated.

This paper focuses on the implementation of mechanisms for non-functional properties of IoT services to ensure that service complies with the defined SLA policy. The enforcement of policies for SLA enforcement is done by foreign cloud in its infrastructure by acquiring enough resources for service execution and employing required mechanisms. QoS parameters mentioned in SLAs are measured by maintaining current system configuration information and runtime information of parameters that are part of SLOs (measurable metrics). Depending on the client requirements some or all SLA parameters could be measured, and SLOs such as request latency or service throughput could be measured by retrieving resource metrics.

Development of mechanisms for maintaining functional property is not in the scope of this paper. We discuss various mechanisms that exist in the literature that could be deployed for secure service execution such as trusted computing. Trusted computing is a paradigm used to enforce trustworthy behavior of computing platforms. It is based on using a hardware crypto-processor module named Trusted Platform Module (TPM) [11]. This feature can be used to run services on only those cloud nodes whose fingerprints are trusted [12]. Various mechanisms for cloud computing based on TPM have been proposed that are used for security of services, data and other resources. Excalibur [12] is a system that can be used to design trusted computing services for cloud. It uses policy sealed data (data encrypted according to customer policy) that can only be unsealed (decrypted) by nodes whose configuration match the node policy. Excalibur uses Attribute Based Encryption to bind policies and attributes to node configurations. A mechanism that uses a hardened hypervisor to attest that the image of the VM running on a cloud node is the same as the one uploaded originally by the service provider and initiated by cloud was proposed by Santos N. et al. [13]. It confines the execution of VM to secure nodes inside the cloud and guarantees that even the system admin with root privileges cannot tamper with the VM memory. Some other recommendations provided by NIST for hardening the hypervisor include maintaining proper isolation, separating the duties of administrative functions and restricting administrator access to security checks [14].

G. SLA MONITORING

Currently, no solutions exist to check for SLA compliance for user support. However, researchers have recommended using the monitoring mechanisms to check for SLA compliance on the cloud provider which involves, i). verifying that SLAs are followed through infrastructure access, and ii). generating an alert notification if the SLAs are violated to take corrective steps.

Monitoring could either be performed by the client from data received from cloud provider or by the cloud provider at the infrastructure level which is the focus of this paper. The input to monitoring component provided by tbe cloud provider is the formal requirements to be monitored in a formal language such as XML. The monitoring component than derives the pattern of events that could occur during service execution and imply SLA violation. In the proposed system uses event-driven modules to collect all generated events and performs required filtration operations before analyzing them. The description of event captors and monitor is used to monitor SLA parameter.

The analysis is performed based on captured events to check if any generated events show an SLA violation. If a security violation is reported by the monitoring component, it logs the event and estimates the current status of service. Monitor also reports to the user if the foreign cloud is compliant with the signed SLA or not. In case of SLA violations, user can enforce penalties on the provider.

IV. WORKFLOW

This section explains the workflow for the overall system and the details of various components. A user request to connect to the foreign cloud for accessing an IoT service is received by cloud controller that advertises request to connect to multiple foreign clouds and receives their responses. Service Matchmaker selects the best provider based on authenticator is responsible for authentication while the SLA coordinator is responsible for SLA management. Workflow of the proposed system is shown in figure 3.

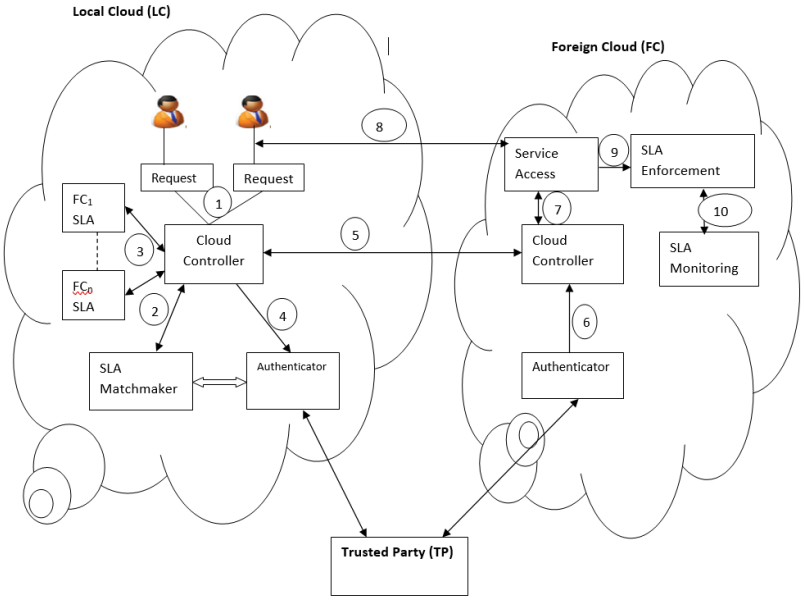

Figure 3: **Workflow of MC-IoT**

A. CLOUD CONTROLLER

This is the major component responsible to handle the multi-cloud communication and authentication. The controller is implemented as RESTful web service in cloud and its responsibilities are two folds. First in the local cloud when it wants to access a foreign cloud and second in a foreign cloud when a connection request is received.

When a user in the local cloud needs to access service in a foreign cloud, it is the responsibility of a controller to establish a connection with the other foreign cloud. Before sending a message to the foreign cloud, it communicates with the local authenticator component to get the certificate. After sending an authentication request, on behalf of local cloud it establishes the communication channel by sharing session keys.

In a foreign cloud, requests for communication from the local cloud are received by cloud controller. Cloud controller is then responsible to check whether, (a) the requested service is available in the foreign cloud, (b) the connecting local cloud is trustworthy, and (c) respond to the foreign clouds request.

B. AUTHENTICATOR

Authenticator component is responsible to manage the authentication of multi-clouds. Once the communication request from the local cloud reaches the foreign cloud, cloud Controller of the foreign cloud connects with the authenticator to verify if the connecting user (of local cloud) is trusted or not. When the authenticator component receives the message containing the identity of local cloud and its digital certificate, it checks whether the certificate is valid and responds to controller component. Based on the response

from the authenticator, cloud controller of the foreign cloud responds to the cloud controller of the local cloud.

In a local cloud, when a collaboration request is to be sent to foreign cloud authenticator is responsible to contact trusted party (TP) which generates the certificate for the local cloud, signs it and returns it to the local cloud. Before sending a communication request to the foreign cloud, local cloud controller gets its certificate from the authenticator.

C. TRUSTED PARTY

Trusted party (TP) is the identity provider responsible to handle the authentication among multi-clouds. It has list of trusted cloud providers, and before establishing session the connecting clouds communicate with it to acquire their certificate. After receiving a certificate request, it generates a certificate, signs it with its private key and returns it to the requesting cloud. Any cloud registered with a TP receiving a certificate signed with a private key of that particular TP considers it true.

D. SLA COORDINATOR

This component is responsible to manage SLA's in the proposed framework. It has features including adaptability and rapid response. It initially selects the suitable service for a client in local cloud based on his requirements using negotiation. Once a foreign cloud provider has been selected, security and QoS parameters are negotiated. The enforcement component is responsible to ensure that service execution in the foreign cloud is according to the QoS parameters agreed in the SLA. Moreover, the monitor is responsible to ensure that the service used by the cloud provider complies with the SLA and in case there is a violation of SLA it reports that violation to the service provider.

V. EXPERIMENTS AND RESULTS

To examine the feasibility of the proposed design empirically, it was implemented on two different clouds. The experiments were conducted to assess the, (i) scalability of the proposed system and (ii) runtime overheads of the system during a collaboration between multi-clouds.

The prototype was tested on two different cloud infrastructures. One of the cloud infrastructures was an OpenStack cloud based in University of Derby. This setup consists of six server machines. Each machine has 12 cores with two 6-core Intel Xeon processors running at 2.4 GHz with 32 GB RAM and 2 TB storage capacity. The cloud nodes on which the experiments were performed had 4 VCPUs running at 2.4 GHz each, 8 GB RAM, and data storage of 100 GB per node. The second cloud was also based on Amazon AWS. The cloud nodes on this machine had 4 VCPUs, 8 GB RAM and 100 GB storage.

Both the Cloud Controller and Cloud Authenticator are employed as web services which help in avoiding tightly bound security. While WS-Agreement was used to implement the SLA components. To enable the interaction among components in the prototype according to the proposed system, cloud controller of local cloud submits requests for resources to other foreign clouds. When the foreign cloud controller initializes and receives a request for available services from a local cloud, it shares exchange information about available services and their characteristics.

In the experiments, to check the scalability of the system initially a large number of service requests were created in the local cloud so that they can be connected to multiple instances in the foreign cloud. To start the communication, cloud controller from a local cloud invokes the cloud controller in the foreign cloud. This is then followed by various operations in the foreign cloud including checking the availability of the required services, verifying if the local cloud user that wants to connect is authorized and SLA negotiation to agree the functional and non-functional requirements of services that need to be satisfied. After performing authentication and communication among multiple instances, a large number of users from local cloud were able to request for multi-cloud collaboration and access service instances in the foreign cloud, and those instances were generated according to negotiated SLA parameters.

To evaluate the overhead caused by protocol, the time taken by different operations was calculated. The time taken by different instances during authentication of instances in the foreign cloud using the proposed system is shown in figure 4.

Figure 4: Authentication time for various instances

To assess the effectiveness of our proposed prototype, we compared the results with other commonly used authentication protocols like SAML [15] and Kerberos [16]. Figure 5 shows that the proposed authentication protocol is very efficient compared to other protocols. 2. The proposed authentication protocol has better performance than traditional protocols like SAML and Kerberos as it is designed specifically for heterogeneous multi-cloud scenarios. Kerberos is a centralized protocol and distributes tickets to all communicating parties which increases its processing time. Although SAML is a distributed authentication protocol, it does not support heterogeneous client attributes, and when used in a secure way (in conjunction with SSL) it takes longer than proposed protocol to perform authentication of multiple clients.

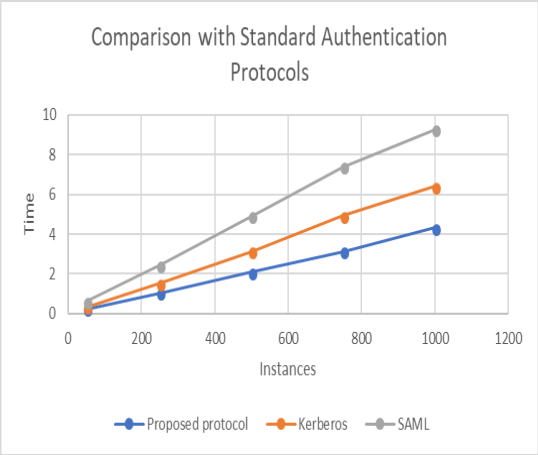

Figure 5: **Performance comaprison of proposed authentication scheme in multi-cloud scenario**

To check the accuracy of the service selection algorithm, service selection requests were made from a large number of services instances, and the algorithm was successfully able to select the service with the highest match of QoS properties using accuracy matrix compared to SPSE and simple additive weighting (SAW) technique [17]. Precision measured as the ratio of a total number of correctly returned services to a total number of returned services of using accuracy matrix compared to SPSE and SAW is shown in figure 6.

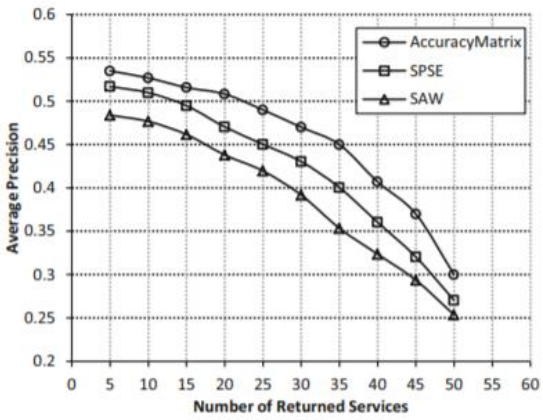

Figure 6: **Precision of Accuracy matrix for service selection**

To measure the performance SLA co-ordinator and effectiveness of monitoring we did experiments to measure the accuracy monitoring component during service execution in the foreign cloud. A basic user interface (UI) was created on the client side to report any SLA violations of the SLA metrics. Figure 7 shows the client UI after accessing a few services in the foreign cloud. The boxes in red are SLA violations that were captured while green boxes indicate the SLA parameters that were successfully implemented and followed.

To measure the delay caused by monitoring, the average time taken to make a decision about the events captured and violations recorded. It is used to measure the difference between the time at which the event leading to the violation of SLA occurred and the time taken by the monitor to decide that a violation has been recorded. The average delay in measuring 1000 events was found out to be 123.34 ms and it remained stable as the number of events increased. Therefore, it can be said that monitoring of SLA parameters take a small amount of time to detect and record violations which can be reported to the foreign cloud so that these violations could be decreased.

VI. USE CASES

IoT has brought revolutionary changes by having applications varying from manufacturing, transport to healthcare and smart homes. The proposed framework MC-IoT offers various advantages and use cases for IoT. Among these is the usage of MC-IoT in e-Healthcare, smart cities, vehicular networks and smart retail. In this section, we present how the proposed framework can be used in e-Healthcare and improving supply chains.

A. E-HEALTHCARE

Healthcare IoT devices such as sensors including implantable, bio-sensors, micro-electromechanical silicon and nano-sensors can potentially bring huge benefits to e-Healthcare industry in the coming years. Some of the benefits offered to patients include remote monitoring of patients with chronic illness, helping in the treatment of diseases, and monitoring of health statistics by patients themselves can help them to steps to improve their health. With the significant advantages offered by using sensor data in health care, the challenge arises with storing huge amount of data generated by sensors. Moreover, e-Healthcare requires data processing, storage and analytics that can be potentially be used by collaborative healthcare entities and

Not secure | sla-client.us-east-2.elasticbeanstalk.com/manageviolation?action=result

Event Id	CPU Core	Memory available(IN MB)	Storage Size(in GB)	Availability(in Percentage(%))	Throughput(in KB)	accessibility(in Percentage(%))
05d3958f-75b8-4cb8-b514-6d7ed31c4003	2	513	292.81	92	1563	65
0d3fd1b0-3d07-4f9a-8599-f059a508a732	4	950	206.82	95	714	96
12af9727-5555-4973-81af-1c31e290f487	8	465	520.94	89	1366	80
15586257-3f60-4ef1-a03d-715bca6f8cb8	16	634	228.08	99	1230	89
1a101c85-240b-4563-88a2-4f2fa8436ef3	8	686	310.18	88	583	89
1a9ab04e-0079-4ebf-b015-7b0ede2d5f85	5	500	2	95	400	65
1af2c190-be8b-4004-bd49-f4758e358fcd	16	407	865.48	96	1603	96

Figure 7: **UI of client side showing SLA parameters compliance in foreign cloud (Red color shows SLA violations while green shows SLA compliance)**

applications.

e-Healthcare solutions enable the delivery of health care services at any required time however, its deployment also raises several challenges. The world population is increasing with the passage of time and more healthcare challenges can be expected in the future. Due to the rise in healthcare cost, more sophisticated procedures such as e-Healthcare are required. Sensor based e-Healthcare systems can monitor patient's health remotely and the doctor can view patients health using e-Health applications without the need of patients visiting a doctor. This ubiquitous monitoring has been predicted as the future of modern healthcare.

Multi-cloud system can provide a service based and application-oriented infrastructure that can be suitable for sensor based e-Healthcare system due to many reasons including the following: sensors generate a large amount of data, number of patient's records being managed is very large, healthcare workers need inter-organizational and collaborative data sharing, some e-Health services need a specific platform to run, healthcare workers might need to use an e-Health service being run on remote platform only for a limited period that will be economically inefficient to be purchased for a long time, and performing data analytics on large datasets of healthcare needs more resources than traditional infrastructure. Based on heterogeneous requirements of multi-cloud and e-Healthcare services, this work proposed framework can enable dynamic collaboration between e-Health services in multi-clouds.

Using MC-IoT based healthcare system, users including patients and healthcare workers will only need to get authenticated by their local cloud and the proposed system will enable them to use services in foreign clouds according to requirement. The proposed system design can revolutionize the healthcare by providing key benefits such as ability to use multiple e-Health services on various platforms, scale computing resources such as storage according to requirements and share collaborative data with health care workers from other clouds.

B. BUSINESS CASE

As described earlier, MC-IoT can be used to enable users of a cloud platform to access services in another cloud. There are many other business cases of this framework that can help to improve the business supply chain.

Consider a case in which an organization named E-Packagers is using cloud resources and services on a cloud service provider. The company needs cloud resources during peak times between 9 am to 5 pm on working days and usage of these resources and their services on weekends is close to none. In this scenario, E-Packagers will have to pay for the time when the usage of their allocated resources is really low. However, using MC-IoT the company can further lease its services to be used by users from other clouds who can directly contact E-Packagers and use their services for a certain time without cloud provider interaction. This can help the company to generate additional revenues and users to access services with lesser conditions in less time.

VII. RELATED WORK

Delivery models for multiple clouds can be classified into two types which are federated cloud and multi-cloud. These models contrast in the level of co-operation between the included Clouds and the way that the client communicates with them [18].

Celesti et al. abridge the prerequisites of identity management across clouds in two classifications [19]:

1). Single Sign-On (SSO) authentication, where a Cloud must have the capacity to verify itself to access the assets gave by federated foreign Clouds having a place to a similar trust setting without further identity checks.

2). Digital identities and third parties, where a cloud must be considered as a subject particularly distinguished by credentials and each cloud must have the capacity to confirm itself with outside clouds utilizing its digital identity.

To address the authentication issues in multi-clouds different architectures were proposed. J. Xu [20] proposed an architecture by which different organizations can collaborate to use business services. The proposed methodology coordinates security pre-requisites in SOA-based business forms and presents techniques for authentication of services from various domains for SOA-based business forms at runtime. Their architecture requires neither credential exchange nor foundation of any validation for creating a business session. The accuracy of the convention is formally broke down and demonstrated, and an observational review is performed utilizing two creation quality Grid frameworks, Globus 4 and CROWN.

Celesti et al. [21] propose a design to empower cloud federation in view of a three-stage model. These stages are named as discovery, matchmaking and authentication. The design includes a matchmaking agent which facilitates brokering, given by a match-production operator, whose errand is picking the more helpful Cloud(s) wherewith to set up an organization in view of data gathered both at the IaaS layer (e.g., CPU or RAM memory) and higher layers (e.g., QoS level). The proposed inter-cloud identity management infrastructure extends from XMPP, and XACML to SAML [22].

Bohli et al. [23] give a study of security and protection arrangements that expand on the idea of the synchronous use of multiple clouds. Pearson et al. [24] talk about how the ideas of privacy, security, and trust develop with the emergence of cloud, and propose conceivable ways to deal with their insurance and administration.

Hussain et. al. [25] developed an authentication scheme that can be used to build up certain trust connections among these business intelligence service instances and clients by sharing a typical session key to all members of a session. The distribution and generation of secret keys were managed by a central authority called session authority. The

correctness of the protocol was verified and performance overhead was evaluated using a trusted third party.

The concept of IoT backed by cloud was introduced as the advantages of cloud including unlimited storage and processing can significantly improve IoT performance. IoT based clouds have introduced concepts such as smart things, things as a service and sensor as a service (SenaaS) [26]. Due to benefits offered by cloud in IoT, several new concepts were proposed.

The idea of cloud federation using IoT has been presented by authors in three stages [27]. The first stage includes embedded devices to be connected to IoT cloud systems, the second stage includes cloud providers leveraging IoT as a service while the third stage includes federation of IoT providers to extend their services and achieve more flexibility.

Authors in [28] have proposed a dynamic data-driven architecture that is able enough to ensure service provisioning in cloud federation with minimum violations of service level agreement (SLAs). The author provided the simulation studies to validate the proposed approach. In [29], the author has introduced a novel approach named SPECS. The SPECS approach helps to offer various mechanisms to access security features that have been offered by CSPs, specify security requirements and to integrate the security services with cloud services to form security as a service approach.

Despite the considerable amount of research in multi-clouds, establishing dynamic communication to access services (particularly IoT services) in heterogeneous clouds is still an open research problem. Current work lacks the protocols and frameworks that can be used for dynamic multi-cloud service collaboration and this research aims to solve this problem.

VIII. CONCLUSION

Multi-clouds offer a promising solution to efficiently deliver IoT services, but their adoption also raises challenges due to lack of supporting frameworks. This paper provides a novel framework to establish secure collaboration across multi-clouds to access services running in the foreign cloud. An authentication scheme is presented by which communicating clouds can authenticate each other dynamically. Service matchmaking technique is proposed to select the best IoT service matching user requirements among multiple foreign clouds, and SLA approach is used to ensure service execution in the foreign cloud is according to the agreed SLA parameters between the user and the provider. Moreover, we also present the detailed system design to implement these protocols and framework. The experiments are performed on two cloud systems based on OpenStack and Amazon AWS and the results show that our protocols only result in a limited overhead. Furthermore, the use case scenarios are presented to show applications of the proposed framework.

ACKNOWLEDGMENT

This work was partially supported by the National Natural Science Foundation of China under Grants No. 61502209 and 61502207, Natural Science Foundation of Jiangsu Province under Grant BK20170069, and UK-Jiangsu 20-20 World Class University Initiative programme.

REFERENCES

[1] Gubbi, J., Buyya, R., Marusic, S. and Palaniswami, M., 2013. Internet of Things (IoT): A vision, architectural elements, and future directions. Future generation computer systems, 29(7), pp.1645-1660.

[2] 2017 Roundup Of Internet Of Things Forecasts, https://www.forbes.com/sites/louiscolumbus/2017/12/10/2017-roundup-of-internet-of-things-forecasts/#3a4c69321480, last accessed: 14/08/2018

[3] Botta, A., De Donato, W., Persico, V. and Pescapé, A., 2014, August. On the integration of cloud computing and internet of things. In Future internet of things and cloud (FiCloud), 2014 international conference on (pp. 23-30). IEEE.

[4] Buyya, R., Yeo, C.S. and Venugopal, S., 2008, September. Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. In High Performance Computing and Communications, 2008. HPCC'08. 10th IEEE International Conference on (pp. 5-13). Ieee. Vancouver

[5] Paraiso, F., Haderer, N., Merle, P., Rouvoy, R. and Seinturier, L., 2012, June. A federated multi-cloud PaaS infrastructure. In Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on (pp. 392-399). IEEE.

[6] Ferry, N., Rossini, A., Chauvel, F., Morin, B. and Solberg, A., 2013, June. Towards model-driven provisioning, deployment, monitoring, and adaptation of multi-cloud systems. In Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on (pp. 887-894). IEEE.

[7] 451 Research, https://451research.com/images/Marketing/press_releases/Pre_Re-Invent_2018_press_release_final_11_22.pdf, last accessed: 15/08/2018

[8] Liu, F., Tong, J., Mao, J., Bohn, R., Messina, J., Badger, L. and Leaf, D., 2011. NIST cloud computing reference architecture. NIST special publication, 500(2011), pp.1-28.

[9] Bresson, E., Chevassut, O., Pointcheval, D. and Quisquater, J.J., 2001, November. Provably authenticated group Diffie-Hellman key exchange. In Proceedings of the 8th ACM conference on Computer and Communications Security (pp. 255-264). ACM.

[10] Ahmed, M., Liu, L., Yuan, B., Trovati, M. and Hardy, J., 2015, October. Context-Aware Service Discovery and Selection in Decentralized Environments. In Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on (pp. 2224-2231). IEEE.

[11] Awad, A., Kadry, S., Lee, B. Zhang, S., "Property Based Attestation for a Secure Cloud Monitoring System", Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing, 2014.

[12] Santos, N., Rodrigues, R., Gummadi, K.P., Saroiu, S., "Policy-Sealed Data: A New Abstraction for Building Trusted Cloud Services", USENIX Security Symposium, 2012.

[13] Bouchenak S, Chockler G, Chockler H, Gheorghe G, Santos N, Shraer A. "Verifying cloud services: present and future", ACM SIGOPS Operating Systems Review, 2013.

[14] Kazim, M., Zhu, S. Y., "Virtualization Security in Cloud Computing", Guide to Security Assurance for Cloud Computing, Springer, 2015.

[15] SAML, https://developers.onelogin.com/saml. Last accessed: 26-08-2018

[16] Kerberos, http://web.mit.edu/kerberos/. Last accessed: 16-07-2018

[17] Afshari, A., Mojahed, M. and Yusuff, R.M., 2010. Simple additive weighting approach to personnel selection problem. International Journal of Innovation, Management and Technology, 1(5), p.511.

[18] Petcu, D., 2013, April. Multi-Cloud: expectations and current approaches. In Proceedings of the 2013 international workshop on Multi-cloud applications and federated clouds (pp. 1-6). ACM.

[19] Celesti, A., Tusa, F., Villari, M. and Puliafito, A., 2010, July. How to enhance cloud architectures to enable cross-federation. In Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on (pp. 337-345). IEEE.

[20]Xu, Jie, et al. "Dynamic Authentication for Cross-Realm SOA-Based Business Processes." IEEE Trans. Services Computing 5.1 (2012): 20-32

[21] Celesti, A., Tusa, F., Villari, M. and Puliafito, A., 2010, July. Three-phase cross-cloud federation model: The cloud sso authentication. In Advances in Future Internet (AFIN), 2010 second international conference on (pp. 94-101). IEEE.

[22] Celesti, A., Tusa, F., Villari, M. and Puliafito, A., 2010, June. Security and cloud computing: Intercloud identity management infrastructure. In Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE), 2010 19th IEEE International Workshop on (pp. 263-265). IEEE.

[23] Bohli, J.M., Gruschka, N., Jensen, M., Iacono, L.L. and Marnau, N., 2013. Security and privacy-enhancing multicloud architectures. IEEE Transactions on Dependable and Secure Computing, 10(4), pp.212-224.

[24] Pearson, S., 2013. Privacy, security and trust in cloud computing. In Privacy and Security for Cloud Computing (pp. 3-42). Springer London.

[25] Al-Aqrabi, H., 2016. Cloud BI: A Multi-party Authentication Framework for Securing Business Intelligence on the Cloud (Doctoral dissertation)

[26] Cavalcante, E., Pereira, J., Alves, M.P., Maia, P., Moura, R., Batista, T., Delicato, F.C. and Pires, P.F., 2016. On the interplay of Internet of Things and Cloud Computing: A systematic mapping study. Computer Communications, 89, pp.17-33.

[27] Celesti, A., Fazio, M., Giacobbe, M., Puliafito, A. and Villari, M., 2016, March. Characterizing cloud federation in IoT. In 2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA) (pp. 93-98). IEEE.

[28] Leitner, P., Ferner, J., Hummer, W. and Dustdar, S., 2013. Data-driven and automated prediction of service level agreement violations in service compositions. Distributed and Parallel Databases, 31(3), pp.447-470.

[29] Rak, M., Suri, N., Luna, J., Petcu, D., Casola, V. and Villano, U., 2013, December. Security as a service using an SLA-based approach via SPECS. In Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on (Vol. 2, pp. 1-6). IEEE.