*Computers, Materials & Continua*
DOI:10.32604/cmc.2022.020213
***Article***

Tech Science Press

# Cloud Security Service for Identifying Unauthorized User Behaviour

**D. Stalin David[1], Mamoona Anam[2], Chandraprabha Kaliappan[3], S. Arun Mozhi Selvi[4], Dilip Kumar Sharma[5], Pankaj Dadheech[6] and Sudhakar Sengan[7,*]**

[1]Department of Computer Science and Engineering, IFET College of Engineering, Villupuram, 605108, Tamil Nadu, India
[2]Department of Computer Sciences and Software Engineering, Faculty of Basic and Applied Sciences, International Islamic University, Islamabad, Pakistan
[3]Department of Information Technology, Bannari Amman Institute of Technology, Sathyamangalam, 638401, Tamil Nadu, India
[4]Department of Computer Science Engineering, DMI St. John the Baptist University, Mangochi, Lilongwe, Malawi
[5]Department of Mathematics, Jaypee University of Engineering and Technology, Guna, 473226, Madhya Pradesh, India
[6]Department of Computer Science and Engineering, Swami Keshvanand Institute of Technology, Management and Gramothan (SKIT), Jaipur, 302017, Rajasthan, India
[7]Department of Computer Science and Engineering, PSN College of Engineering and Technology, Tirunelveli, 627152, Tamil Nadu, India
[*]Corresponding Author: Sudhakar Sengan. Email: sudhasengan@gmail.com
Received: 15 May 2021; Accepted: 18 June 2021

**Abstract:** Recently, an innovative trend like cloud computing has progressed quickly in Information Technology. For a background of distributed networks, the extensive sprawl of internet resources on the Web and the increasing number of service providers helped cloud computing technologies grow into a substantial scaled Information Technology service model. The cloud computing environment extracts the execution details of services and systems from end-users and developers. Additionally, through the system's virtualization accomplished using resource pooling, cloud computing resources become more accessible. The attempt to design and develop a solution that assures reliable and protected authentication and authorization service in such cloud environments is described in this paper. With the help of multi-agents, we attempt to represent Open-Identity (ID) design to find a solution that would offer trustworthy and secured authentication and authorization services to software services based on the cloud. This research aims to determine how authentication and authorization services were provided in an agreeable and preventive manner. Based on attack-oriented threat model security, the evaluation works. By considering security for both authentication and authorization systems, possible security threats are analyzed by the proposed security systems.

**Keywords:** Cloud computing; user behaviour; access control; security model

## 1 Introduction

Cloud computing has emerged dramatically as a new paradigm of Information Technology. Cloud computing is recognized on the already prevailing Internet technologies platform and functions as a self-service system. Compared with present conventional distributed systems, the prime features of a cloud computing environment are perception and virtualization, enabling technology to understand and apply fully in a distinct manner. From users and developers, the cloud-computing environment extracts the particulars of execution of services and systems. Infrastructure as a Service (IaaS), Software as a Service (SaaS), and Platform as a Service (PaaS) are the three popular cloud services [1].

The following cloud deployment models possess each cloud service: Public, Private and Hybrid. Some advantages like flexibility, cost efficiency, performance, etc., are the benefits of using cloud computing because the user can use the services anytime and anywhere. However, some basic things like pooling, sharing, and resources outsourced to a remote server are the causes of security concerns. In the Cloud computing environment, the security aspects are associated with the security of virtualization technology, disseminated services, services accessibility, traffic control, application security, control of accessibility, and authentication. In addition, a policy execution mechanism, service levels, isolation, etc., are required by multi-tenancy.

On the cloud security solutions, both the cloud deployment and service model possess a more significant influence, and various prominence on multi-tenancy is caused. Cloud computing security risks begin with the cloud service. The IaaS platform provides computing infrastructure, storage space, and network security as a service. Users can use the resources to create their recommended cloud infrastructure through virtualized computing services. To deliver the platform as a service, the PaaS model places one more layer on the top of IaaS along with application development frameworks and tools. In turn, to deliver application (software) as a service, SaaS places one more layer on the top of PaaS that users use through a browser or other client program [2]. Thus, there is a trade-off between the service delivery model and security solutions. The service provider's responsibility for security solutions is more in higher service levels.

The users alone do not control their data and cloud identity in the cloud. There is often complexity in installation procedures, and safe execution of the basic infrastructure and safe utilization of the Service assured by security solution is not complete and adequate. Users and administrators face various authentication and authorization systems, distinct login messages, and all individual login credentials are coordinated with distinct authorization, particularly in the multi-platform clouds and inter-clouds where there is access for multiple cloud systems, apart from implementing authentication methods at the user level, manipulating and creating security features for various platforms and technologies being challenging. These limitations are addressed by the technique called Identity federation and Single Sign-On (SS-O) by permitting interchange of authentication and authorization of information between two parties like Identity Provider (IDP) and a Service Provider (SP). Still, some real-time problems avoid their direct and easy execution in the multi-platform cloud environments, whereas in literature, the identity federation and Single Sign-On notions are well covered. The effort taken for designing and developing a solution, which would have reliable and secure authentication and authorization service in such a situation is elucidated in this paper. OpenStack and Virtual Machine software (VMware) platforms are implemented with Single Sign-On principle-based proposed solutions [3]. Our attempt represented a design for a cloud computing environment that would be more trustworthy and provide secure authentication and authorization with the help of multi-agents.

## 2 Related Works

The roles of users or providers in cloud computing adopt the notion that cloud computing is assumed to be a sum of SaaS and utility computing. From the cloud computing concept's maiden appearance, John McCarthy said that in the 1960s [4], computation might sometimes be structured as a public utility. When providers began to use virtual private networks for data communication in the 1990s, the term "Cloud" emerged from the world of telecommunication services. Information Technology infrastructures are the spread of wired/wireless communication networks and excessive speed, different sorts of toolsets, the spread of free software, and much more.

Whether the proper entity or person has access to the given data from a cloud technology provider is assured by authentication in cloud computing. The cloud service provider is proved of the user's identity when retrieving the stockpiled information in the cloud when there is an assurance of authentication in cloud computing. Public and private clouds apply distinct designs for Rivest, Shamir, Adleman (RSA)-based authentication. The RSA cryptosystem supports a wide range of authentication models, including two-factor, knowledge-based, and adaptive authentication. The transfer of confidential information between the web server and the browser, which includes a virtual private cloud, is focused on by Amazon Web Services (AWS) [5]. Multiple authentication schemes like AWS identity, access management, multifactor authentication, etc., are executed in this context. The users are vulnerable to honeypot and dictionary attacks by using this technique. Leading Information Technology firms like Google, Microsoft, and Facebook use this technique.

Data encryption in the cloud is enabled by another vendor known as Wuala cloud. Here, personal computers transfer data before the enabling of encryption. Even the provider cannot access the data because this provides improved security. For hierarchical attribute-based cloud computing, the authors proposed an encryption method. High performances and more access control are the features of this proposed security model for privacy. Authors are proposing the encryption method to manipulate the data they own in the cloud. For the assurance of confidentiality of data in the cloud, authors in [6] proposed an access control mechanism as far as the user authentication model is concerned. The two protocols viz., (Attribute-Based Encryption (ABE) for data privacy and Attribute-Based Signature (ABS) for user authentication), is based on this mechanism. ABE and ABS are merged to maintain the privacy of users who store their data in the cloud. In a decentralized way, key attributes and distribution were performed. However, on DES, various attacks have been detected. With the latest encryption algorithms, these techniques are no longer effective since they introduce an avalanche effect.

For the combined storage and maintenance of client data, the authors in this paper consider the presence of multiple Cloud Service Provider (CSP) [7]. Besides, the reliability and accessibility of stored data are done using cooperative provable data possession. The following is the description of the verification process: first, the secret key is used by the client for pre-processing the flex that contains the accumulation of *'n'* blocks. Set of public verification information is then generated and stockpiled in Trusted Third Parties (TTP), and later on, some flex and verification tags are transmitted to Cloud Service Providers, and its local copy is deleted. The data storage and management are provided with certain specific functions by this model; however, the storage management's complexity would be hiked by these functions, such as overlapping data blocks and skipping [8]. According to the agent's concept, the user authentication processes' performance is enhanced using an agent-based authentication model in cloud computing environments. The suggested model's increase in the trusted rate in cloud-based environments is apparent in the theoretical analysis. According to agents' capabilities, using agents in the authentication process can be very effective and trustworthy [9].

## 3 Research Motivation

Understanding the technological environment's base and its present situation requires content such as prerequisite analysis introduced in this section.

### *3.1 User Identity Management*

There should inevitably be some idea about the users of an application, whether it runs on-premises or in the cloud. Ultimately, it is demanded by the application that each user offers a digital identity, which is a set of bytes that define the user. The application's decision of things, such as identifying the user and the things users allow them to do, is identified based on these bytes and authenticated. The identity information is furnished using multiple online premises applications that trust on-premises infrastructure services like Active Directory. Generally, when a user accesses a cloud application, or an on-premises application accesses a cloud platform, an on-premises identity will not work. And what happens if an application is made on a cloud foundation? From where does its identity information is obtained? An identity service in the cloud addresses these drawbacks since users, on-premises, and cloud applications use digital identity provided by identity service. In multiple situations, a cloud identity service can be used. The availability of the number of cloud identity services, indeed, indicates the significance of this sort of identity service. To present an Amazon-defined identity, accessing Amazon cloud services like Amazon Simple Storage Service is essential; for instance, a Google account is required for using Google App Engine. Microsoft and other Windows Live Identity applications are provided by Microsoft, whereas BizTalk Services' identity service can be combined [10]. Since a specific identity provider is frequently secured with cloud computing platforms, developers lack absolute freedom. However, there is clarity in terms of identity as a cloud service.

### *3.2 Methods for User Authentication and Authorization*

The cloud resources for listed entities should be provided with identity management and access control by Identity and Access Control Service. Users, software processes, or other systems are such entities. Initially, an entity's identity should be verified to provide an accurate access level in which the authentication process is performed before the authorization process. Regarding the application's authentication and access efforts, to track all successful and failed operations, an audit logging system should be used in addition to authentication and authorization processes. Various Encryption (ENCRYPT) [11] procedures enhance confidentiality, usually encoding data with the help of cryptographic algorithms. The security of sensitive and private data will be guaranteed by rendering such a service, and the proposed entity can only be decoded. Cryptographic algorithms that are rendered as a service and ensure privacy and undeniability in a cloud environment are computationally unable to be integrated with ENCRYPT and decryption processes like hashing, digital signatures, key exchange, and management that form an ENCRYPT system.

### *3.3 Cloud Security Model for User Authentication and Authorization*

Prevention of unnecessary infiltration of unauthorized users at the threshold level is the main objective of security provision. Therefore, all the new ones do not permit the accessibility to data or resources by not proving their distinctiveness. Then, the cloud servers receive the request, which the users first encrypted. The following is the description of the algorithms that were used in the encryption process:

### 3.3.1 Rivest, Shamir, Adleman

The communication is protected through an RSA [12] encryption algorithm. Usually, when the user's requests are sent to the CSP system, they are encrypted. The RSA algorithm uses the system's public key for encryption. The RSA encryption algorithm encrypts the file system through the user's public key whenever they request it. Later, while logging in to the system, with the help of the RSA algorithm, an encrypted file will be decrypted by the user's browser once the file is received from the system. In the same way, when the system receives an encrypted file using its private key, it will decrypt it right away. Eventually, there will be secure communication between the user and the system.

### 3.3.2 Advanced Encryption Standard and Message Digest 5 Hashing Algorithm

The Advanced Encryption Standard (AES) encryption algorithm facilitates encrypting files by the system server when a user uploads a file. The key bits like 128, 192, 256 can be used in AES. The system server randomly generates the key. Only one time a single key is used. A user's file, for that instance, is encrypted and decrypted using a unique key. Later on, in any instance, this key is no longer used. Besides the user account name, the key is kept in the system server's database table. The Message Digest 5 (MD5) [13] hashing is used to hash the user account name before its insertion. This assures that by merely getting access and track the system server's database table, an unauthorized person can never regain the key for decrypting a specific key for a specific user. As a result, the key to the specific file is concealed and secured.

Along with the login page, the encrypted file's path is retained in the storage server's database table when the encrypted file is re-uploaded for storage. The user's identity synchronizes the database tables on the primary and storage servers. The storage server where the encrypted files are stored is not inserted correctly.

### 3.3.3 One Time Password

In this algorithm, the user is authenticated using a One Time Password (OTP) [14]. The user account is kept protected and confidential from unauthorized users using the OTP. The user-defined OTP can, however, be compromised. In the proposed security model, OTP is used to overcome this complexity. A new OTP will be given to the user whenever he logs in to the system to use it in the next login. The system itself usually provides this, and this OTP generation happens randomly. The user's old OTP will get deleted from the system whenever a new OTP is generated. The specific user will be updated with a new OTP. For each login, a single OTP is used. The authorized e-mail account of the user will get the OTP. Simultaneously, the user's authenticity is also verified. Ultimately, the users with a valid e-mail account can be connected with the cloud system.

### 3.3.4 Data Encryption Standard

Data Encryption Standard (DES) algorithm is a symmetric key encipherment algorithm. The encryption and decryption performance is done using a single (secret) key in symmetric-key encryption, a cryptosystem. It is evident that in DES security, a significant role is played by secret keys because of the need for an excellent key generation unit. The key generated using the dynamic key generator features changeability and non-recurrence. Therefore, the dynamic key generator achieves greater speed and reduced logic complexity by adopting this technique [15].

#### *3.3.5 Rijndael Encryption*

The encryption of delicate information is achieved using the standard symmetric key encryption algorithm known as Rijndael, an iterated block cipher. The iteration of a particular transformation accomplishes the cryptography of a data block. One-dimensional 8-bit byte arrays, which generate data blocks, are adopted by Rijndael as input. The input is plaintext, and later it is mapped onto state bytes. Also, the cipher key acts as a one-dimensional 8-bit byte array. The multiple transformations function in an array on interim cipher states and an iterated block cipher [16].

### *3.4 User Authentication and Authorization Process and Protocols*

The cloud resources should be provided with identity management and access control by identity to register the entities. The entities can be software processes, users, or other systems. First, the entity's identity, which is an authentication process that comes before the authorization process, ought to be checked so that a proper level of access to a resource can be assumed. The application's monitoring of all successful and failed operations in connection with authentication and access is performed using an audit logging mechanism in addition to authentication and authorization processes. Confidentiality, which is nothing but data encoding using cryptographic algorithms, is achieved by various encryption mechanisms [17]. The personal and sensitive data and its security will be ensured by offering such a service, and an intended entity can do its decoding. By doing the process of encryption and decryption, key exchange and management, digital signatures, certificates, and hashing create an encryption system that can be rendered as a service, and privacy and non-refutation in a cloud environment are assured.

### *3.5 Authentication Protocols*

*(a) Challenge-Response Authentication System Based on Password*

Regarding computer security, challenge-response authentication is an origin of protocols where a query is modelled by one party ("challenge"), and the other party must give an authorized answer ("response") to be validated. Password authentication is the most accessible illustration of a challenge-response protocol where getting the password is challenging, and the correct password is an authentic response. Similarly, an adversary who obviously can eavesdrop on a password authentication can, later on, authenticate on its own. The issue of multi-passwords with an identifier marked on each of them is the solution. The verifier's passwords can be requested, and the prover must possess that correct password for the identifier. Thinking that the passwords are chosen individually, an adversary with no clues to facilitate a distinct challenge at a different time interrupts one challenge-response message pair [18].

*(b) Lightweight Directory Access Protocol*

In certain kinds of Lightweight Directory Access Protocol Servers, the critical information of many companies is stored. The delegate is provided with the authentication process by SaaS providers to the customer's internal Lightweight Directory Access Protocol/Active Directory Server so that control can be retained on the users' management by the companies.

*(c) Single Sign-On (SSO) Protocol*

The cloud environment's shared security system contains this protocol. A Security Assertion Markup Language (SAML) server has consisted of the system that renders application service providers with SSO services: SAML ticket is issued by SAML [19] server that proclaims the client's identity verification, assuring whether there is proper authentication or not. Access to various authorized resources at the location of various application providers with no necessity for re-authentication for each area can be requested by the user as soon as he/she is authenticated. The access to registered users alone was restricted using a single-page application in many current works. Moreover, to modify each user's practice and view the quantity and kind of data to the personal roles and access levels, efforts were taken. Otherwise, the models' attempt to Authenticate and Authorize individual users was made. Users verify that someone whom they demand to be is meant by authentication. Determining which resources, a user can access and what the user should do with those resources is meant by authorization. An application in most situations will require quite both. A user can log in to one application on social media platforms like Facebook/Google with a set of identifications. Similar sites or applications can later be logged using the same set of identifications.

Similarly, along with the links to intranet sites related to health insurance, timesheets, or organizational news, an internal-facing employee portal should be owned by any business. It is a better way for the employee to log in at a portal that authenticates the user automatically with the other intranet sites rather than the login by an employee at the various website. The concept that allows the user to enter one username and password for accessing various applications is called Single Sign-On. The user is greatly benefited. They manage just one username and password to benefit the associated website-linked identities. Since the user can prevent multiple logins, they have a better experience. Rather than storing multiple authentications around multiple databases, user authentications (single set) will be in one database. It simply means that there is no need for the developers of multiple applications to store passwords. Alternatively, from an authenticated source, a proof of identity or authorization can be accepted by them. The implementation of SSO has multiple solutions. SAML, Open Authorization (OAuth), and Open-ID are the three frequently used web security protocols. Already, there is an existence of implementations and libraries in multiple languages, and rather than a custom solution, going by a systematized protocol gives better interoperability.

*(d) OpenID (O-ID)*

The non-profitable O-ID Foundation promotes OpenID, an open criterion for authentication. O-ID is used by companies like Google, WordPress, Yahoo, and PayPal to authenticate users, and there were more than a billion O-ID-enabled accounts on the Internet through an OpenID (e.g., Google) from which a user must obtain an O-ID account. Any website that adopts OpenID authentication will later be signed into by the user using that account. The layout for communication, which must occur between the identity provider and the trusted third party, is provided by the O-ID standard.

*(e) OAuth2 (OA-2)*

Contrarily, for authorization, Open Authorization-2 is an open standard. Surprisingly, OA-2, which is also fundamental for O-ID Connect, provides O-ID (Authentication) onto OA-2 (Authorization) for a much better security solution. In early 2014, O-ID Connect was generated. Instead, this primer will focus on OA-2, not as a fragment of O-ID Connect. A protective delegated access will be provided by OA-2, which means that an application known as the "*Client*" can take action or get access to resources on a resource server by representing a "*User*" and not allocating their authentications with the application. OA-2 performs this by permitting an identity provider to issue the tokens to these third-party applications by getting the user's approval. Then, on behalf of the user, the client utilizes the token for accessing the resource server.

*(f) Security Assertion Markup Language*

SAML, which was developed in 2001 with a major update in 2005, is the old standard of the three. SAML pronounced as *"Sam-el."* Both authentication and authorization are provided by this open standard method. SAML describes a principle attempted to be decrypted by the end-user related to the definitions used in the other two standards. *The identity provider* is a server that contained the identities and credentials of the principal.

*(g) OpenID Connect*

OA-2 family specifications-based interoperable authentication protocol is known as OpenID Connect. Carrying the goal of "*making simple things simple and complicated things possible,"* the direct Representational State Transfer architectural style (REST)/JavaScript Object Notation (JSON) message is used by OpenID Connect. Compared with other preceding Identity protocols, the integration is strangely easy for developers. By not owning and managing password files, the developers are allowed by OpenID Connect to authenticate their users through websites and apps. It gives a safe and verifiable answer to the app developer's question: "*What is the identity of the user who is at present using the browser or native app linked to me?"* OpenID Connect permits the launching of sign-in flows and receiving verifiable statements regarding the signed-in users' identity for clients, including browser-based JavaScript and native mobile applications.

$$(User\ Identity, User\ Authentication) + OAuth\ 2 = OpenID\ Connect$$

Disowning the responsibility to store and manage passwords in the face of an Internet, which is denser with users attempting to reconcile with the user's account for their benefit, allows app and site developers to authenticate users. Being very developer-friendly is the objective of OpenID Connect and simultaneously intensifying the set of a use case where it can be used, and it has also succeeded in this. On a considerable scale, the deployments of production are being operated. Using the standard crypto signature-verification libraries, a programmer with adequate experience in sending and receiving JSON messages on Hypertext Transfer Protocol (HTTP) should be capable of implementing OpenID from scratch. Fortunately, since there are excellent commercial and open-source libraries, which have more concern over authentication mechanics, most of them need not go to that extend. OpenID 2.0 and OpenID Connect have many architectural similarities, and indeed the same set of problems are solved by protocols.

Consequently, at times, OpenID 2.0 applications would inexplicably deny interoperation because sometimes the practical use of Extensible Markup Language (XML)-based OpenID 2.0 and a custom message signature scheme are proved to be difficult for developers to rectify. The encryption inevitable to the built-in Transport Layer Security (TLS) infrastructure of Web globally executed on both server and client is outsourced by OAuth 2.0, the substrate for OpenID Connect. JSON Web Token (JWT) is utilized by OpenID Connect when there is a requirement for a signature. As a result, the implementation of OpenID Connect is made dramatically more accessible for developers, and much better interoperability has resulted in practice.

## 4 Problem Statement

The attributes of distributed systems like dynamic scalability, unlimited virtual resources, and cost-effective business organizations besides on-demand services to clients are provided by cloud computing. From business and technological perceptions, security challenges, which emerge inside the computing environment, led to many attacks. For a cloud environment, there is a continuous enhancement of security solutions in addition to many challenges. The cloud environment is provided with security solutions professionally and centralized by a new security technique as a Service. Since the Security as a Service (SaaS) model is wide and not an actual application, but at present, it has been progressing. A few cloud providers possess a centralized security infrastructure system to fulfill clients' security concerns. There is a huge requirement of the apparent and most accessible cloud security infrastructure to offer cloud-based software services with identity management services.

## 5 Proposed Model

According to user information, the validation of user's authorized identities and acquirement of their control of access benefits for the sources can be performed by the proposed model known as User Access Control and Authentication Model (UACAM), which consists of proper tools. At the time of user authentication and access control processes, the notion of multi-clouds-agents and Right-to-Access and Authentication-Software-as-a-Service (RTAA-SaaS) has been considered to develop a secured algorithm. Fig. 1 represents a concise introduction to the proposed model. To design a cloud-based SaaS and control the accesses and user authentication processes, this model uses the trustworthiness of the public and private cloud computing environment and the concepts of multi-clouds. The proposed model uses one client-based multi-agent to develop a secure algorithm during services, communications, and processes. Increasing trustworthiness is the prime objective of these agents. Therefore, with RTAA-SaaS and OID-SaaS applications, the other agents have been linked, and every individual agent possesses distinct performance. In the following activities, each agent and UACAM's responsibilities have been explained.

The dependency of this method on cloud server systems is one of the most challenging issues in the existing user identification model. Thus, before cloud environment accessing, the users' identity is managed by an application known as a client-based user authentication agent. For this purpose, the installation of UACAM is done to register devices authorized by the user. The device registration algorithm in UACAM is shown in the following Fig. 2.

After signing up the device with RTAA-SaaS, the device will accept login credentials. The access code will be entered and verified with the RTAA-SaaS database at the installation time. The installation of the application will be done once confirmation is received and Media Access Control (MAC)-ID registers the device. For other processes, an extension ought to be installed by the client-based application on the registered device's web browser. The cloud server can be accessed very securely by the users once a device is registered by fixing UACAM and approving it. The performance of the UACAM multi-agent is shown in detail by the following algorithm.

In Fig. 3, the following is the performance of UACAM:

*Step 1*. A REQ (request) is sent to a cloud-based agent by a client-based multi-Agent for check authentication.

*Step 2*. If the client-based access code is accepted, the REQ is received and confirmed by a client-based multi-Agent.

*Step 3*. For authentication of the cloud-based user, account details like username and password are sent by Client-based multi-Agent.

*Step 4*. The user can access the cloud server after confirmation.

*Step 5*. Communication between users and the Cloud server occurs, and there is the transmission of relevant data.

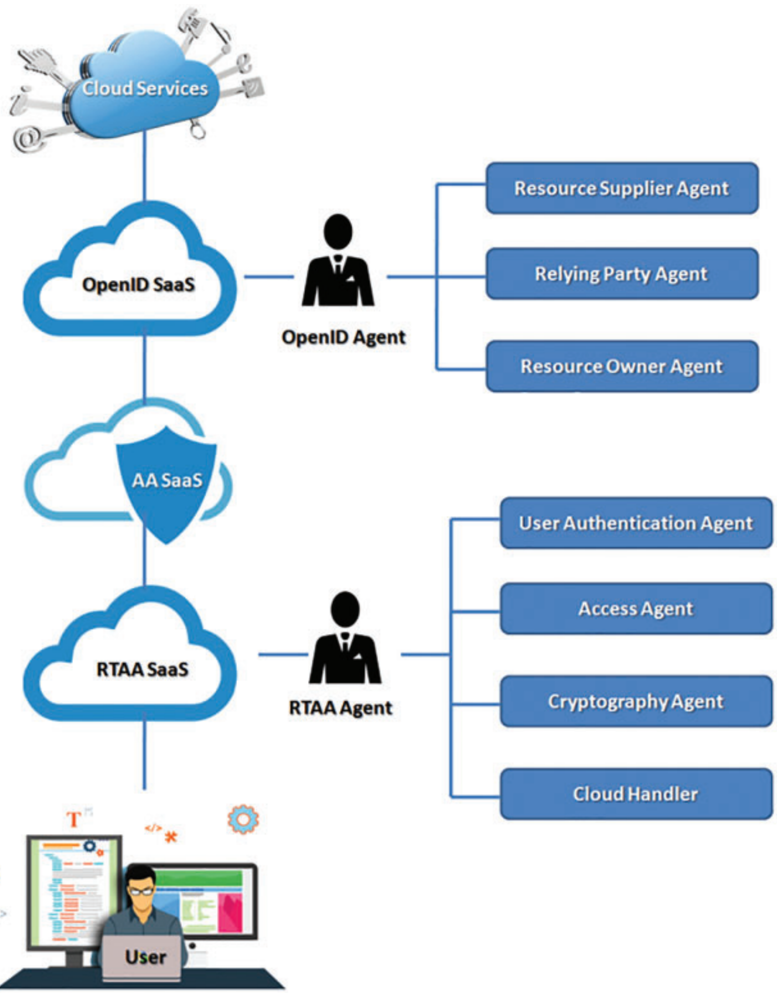

**Figure 1:** User access control

The fundamental purpose of this algorithm is to improve the cloud user's security through multi-agent.

### *5.1 Cloud User Authentication Multi-Agent*

Using modern techniques, a layout sign-in to cloud servers is offered by an agent called Cloud User Authentication Multi-Agent (CUAMA). The reliability of user authentication processes is increased by doing this method. However, the most crucial problem in cloud-based environments is that the user authentication process is less efficient than the existing model's assessment process.

### *5.2 Access Control for Multi-Agent*

The trustworthiness of presumed registering environments is reduced by monitoring access control, one of the most common issues stated by investigating the completed query. Besides these lines, the assembling of monitoring services in cloud-based environments is performed by setting an opening control authority in this model. The stand-out data header will elongate data access management in the access control authority. This assumes that the trust pace in cloud servers is increased, and a unique data header will represent for management to get to. In this access control model, a couple of areas will be assigned with the data, and several cloud servers will take each part into account. Likewise, at the Authentication and Authorization SaaS cloud server, which encourages getting confirmation from O-ID, the disseminated data will be linked for users' use.

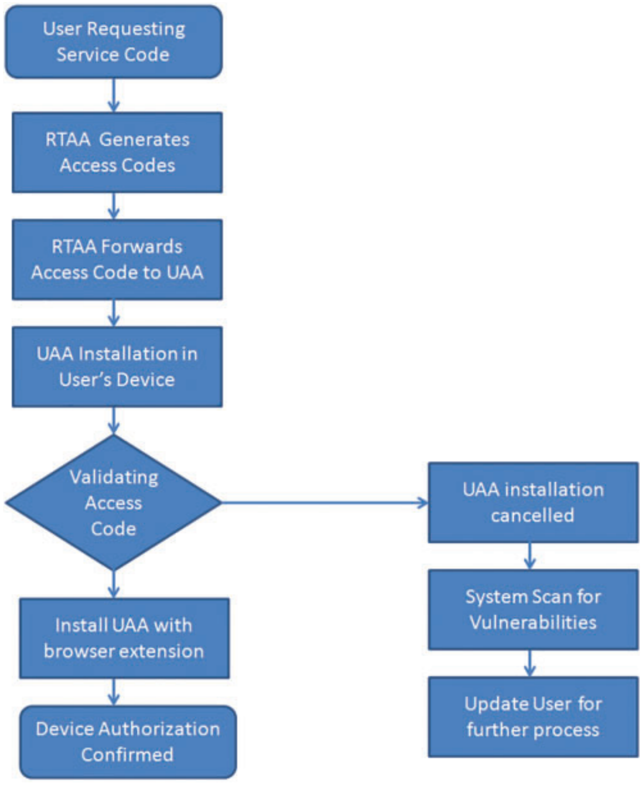
User Requesting Service Code
RTAA Generates Access Codes
RTAA Forwards Access Code to UAA
UAA Installation in User's Device
Validating Access Code
UAA installation cancelled
System Scan for Vulnerabilities
Update User for further process
Install UAA with browser extension
Device Authorization Confirmed

**Figure 2:** Algorithm of access device registration authentication model

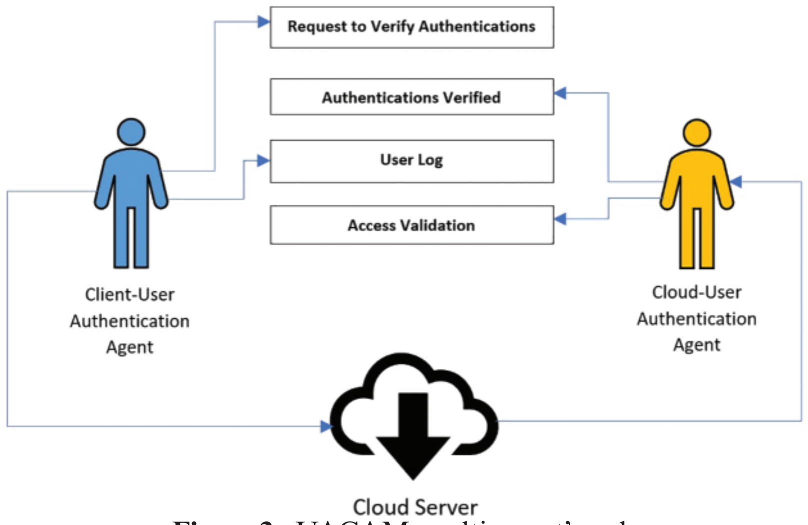
Request to Verify Authentications
Authentications Verified
User Log
Access Validation
Client-User Authentication Agent
Cloud-User Authentication Agent
Cloud Server

**Figure 3:** UACAM multi-agent's role

### 5.3 *Agent for Cryptography*

The utmost open response to work up security is accomplished through cryptography algorithms in a cloud computing environment. In various situations, both asymmetric and symmetric keys models are used by cryptography control.

#### *5.3.1 Private Files by AES-128 Bits*

AES-128 that is unable to be distributed is incessantly reasonable in private data. Furthermore, because the Security of AES-128 requires extra time and an unconfirmed cracking opening, it will increase the reliability of the cloud server.

#### *5.3.2 Sharing Files by RSA-1024 Bits*

Communication between open keys and authorized users is carried out to stretch the security of sharing data using private and public keys by RSA-1024. In this computation, the extension of trust value occurs when the private key cannot transmit among users.

### 5.4 *Cloud Computing Manager*

The bare reality is that the quick progression of using cloud managers has taken place in the current years. The users venture to buy from various cloud managers for multi-purposes since they are in need. The open and private keys of cloud servers are employed for multiple purposes. In cloud servers, handling information is one of the most challenging problems, which reduces the success rate in the distributed cloud computing environment. A cloud manager developed in RTAA manages the relationships and responses between private and open cloud computing.

### 5.5 *OpenID Multi-Agent*

Using portable devices, a cloud system of sign-in to cloud servers is provided by an operator, and the cloud user authorization agent determines the technique for constructing a standard quality of user verification procedures. According to the present model's study method, the missing value in the user validation process will be the most pivotal problem in environments based on the cloud. Similarly, along with specific virtual devices, a cloud-based user validation agent has been managed in RTAA.

### 5.6 *OpenID-Connect-as-a-Service (O-ID CaaS)*

- End-User (EU): A user needs access to the Service.
- Resource Holder (RH): Giving access to a secured resource is enabled by it.
- Relying Party (RP): Authentication and permission from the user for access are the application's requirements.
- Resource Server (RS): Resources and user metadata are managed.
- O-ID Provider: After authentication of the EU and acquiring O-ID successfully from the user, O-ID, an authorization server is responsible for issuing tokens to RP.

Imagine that test data exams have been taken by a user and store the user logs at RS (For example, picture archiving and communication systems database). The personalized consent policies, which stated that the patient desires to share his information, are also defined by the patient. If the EU and RO are different users, then RO may be offline, and it cannot grant or refuse to accept access requests. Hence, as shown in Fig. 4, it is a must that access control policies based on patient consent be determined and incorporated with the O-ID connect process flow.

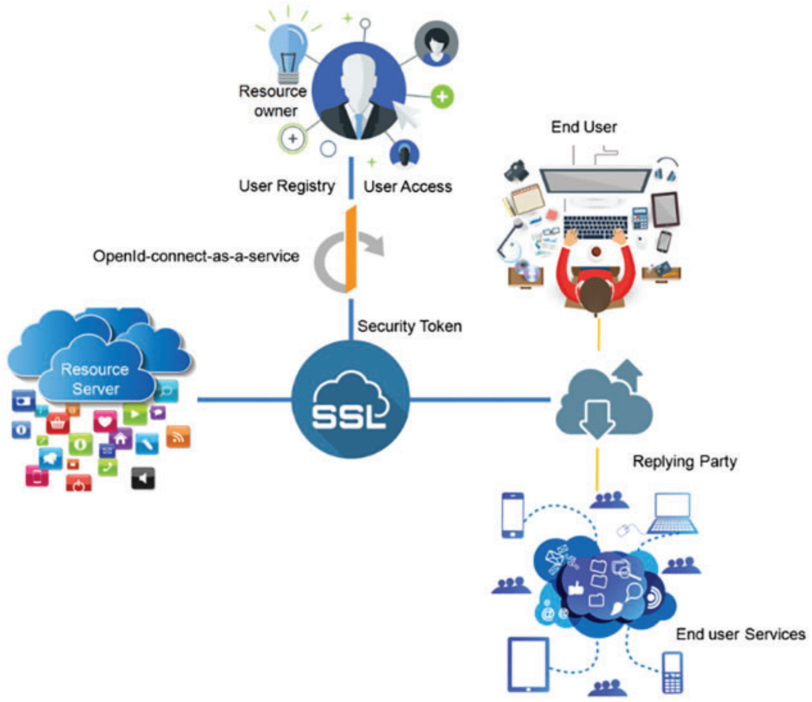

**Figure 4:** O-ID CaaS architecture

### *5.6.1 Initiate Access Request (REQ)*

Assume that the EU has already registered an account. Then, the user O-ID identifier to the medical imaging service is provided after the user initiates an access request. By non-disclosure of any service, username and password are continued by the user. However, rather than a reliable system, malevolent Service could try to phish EU passwords by offering its interface. Hence, all information in the authentication exchange can be captured, including username and password. Regarding phishing attacks, the EU ought to be educated. For example, access to reliable RP alone.

### *5.6.2 Delegate Authentication*

According to the URL *"http://www.cloudcomputing.com"* the location of the identical user is discovered dynamically by Service. Service assigns the user to authenticate the EU and request an access token if access is permitted. Thus, access to a secured resource is obtained, and a malevolent Service can impersonate another RP. Whenever the user is assigned authentication, he must authenticate Service.

### *5.6.3 EU Authentication*

The user performs authentication by redirecting the EU's browser to a user's login page. A strong authentication method can be used in addition to a username and password. However, to assure that the repeated request is from the original Service and not from an impersonator, the user should not authenticate the Service to avoid repeated requests.

### *5.6.4 Grant Access*

The EU is furnished with information regarding RP and requested access by the user if the EU and RO are the same users. EU assesses the information to offer or refuse access requests and restricts the scope and lifetime of access. Sometimes, great privileges might be acquired by Service. This is because the EU may not understand the scope of the granted access and to whom the necessary scope must be explained explicitly and perfectly. Then, only the minimum scope essential to Service should be granted by the EU. Identifying an objective service by the user is also assisted by EU engagement in the authorization.

#### *5.6.5 Apply Consent Policies*

Contrary to service predefined consent policies, the user analyzes the access request after authentication. The consent policies are compulsory if Service and EU are different users as RO is not run-time present and there is an omission of Step (4).

#### *5.6.6 Authorization Code*

Assuming that Service's access request is granted, the EU's browser is redirected back to RP by the user and reverts RP, a less spanned and one-time authorization code. After assuring the authorization code issued to a similar RP, the user authenticates RP.

#### *5.6.7 Issue Tokens*

RP is issued an ID token if the user authenticates EU. An access token is reverted to RP if RO authorizes the resource by RP. A secret key is used for digital signs on an encoded JSON string, an ID/access token. Usually, the string is opaque to the Service. In both transit and storage, ID token and access tokens must be maintained confidentially. From the database of the user, ID token and access token might be obtained by attackers, in case such sensitive information in a database is persisted by the user that results in ruin because there could be a disclosure of all tokens. Hence, there should be storage of token hashes alone, and the database security has to be enforced by the user.

#### *5.6.8 Retrieve Resource*

The user's inability to generate, modify, or guess ID token and access token must be ensured by unauthorized parties. RS must possess the access token and validate the access token obtained from the user before returning the demanded resource to RP that is finally reverted to the EU.

### ***5.7 Authentication and Authorization Flow***

O-ID CaaS can be incorporated with the current authorization workflow within medical imaging systems besides user-centric access control based on consent to implement access control policies specific to the enterprise. In the following, we have defined two services in the architecture of association user and medical imaging services:

#### *5.7.1 Authentication (AuthN) Service*

By performing two operations, the "OpenID Connect Authentication" service is implemented: (a) database services' authentication request; and (b) authorization service's user information query.

#### *5.7.2 Authorization (AuthZ) Service*

The authorization service in the current database systems is represented in this Service. The authenticated users' attributes that provide AuthZ Service for taking access control decisions are provided by AuthN service in this setup. The AuthN Service does the user interactions and controls ID and access tokens. The separation of authentication from authorization is done by O-ID CaaS design. For incorporation, general and simple Application Programming Interface (API)s are provided. To perform the authentication process, AuthN service API alone is needed to be called by the current medical imaging services for regaining information associated with the user (Fig. 5).

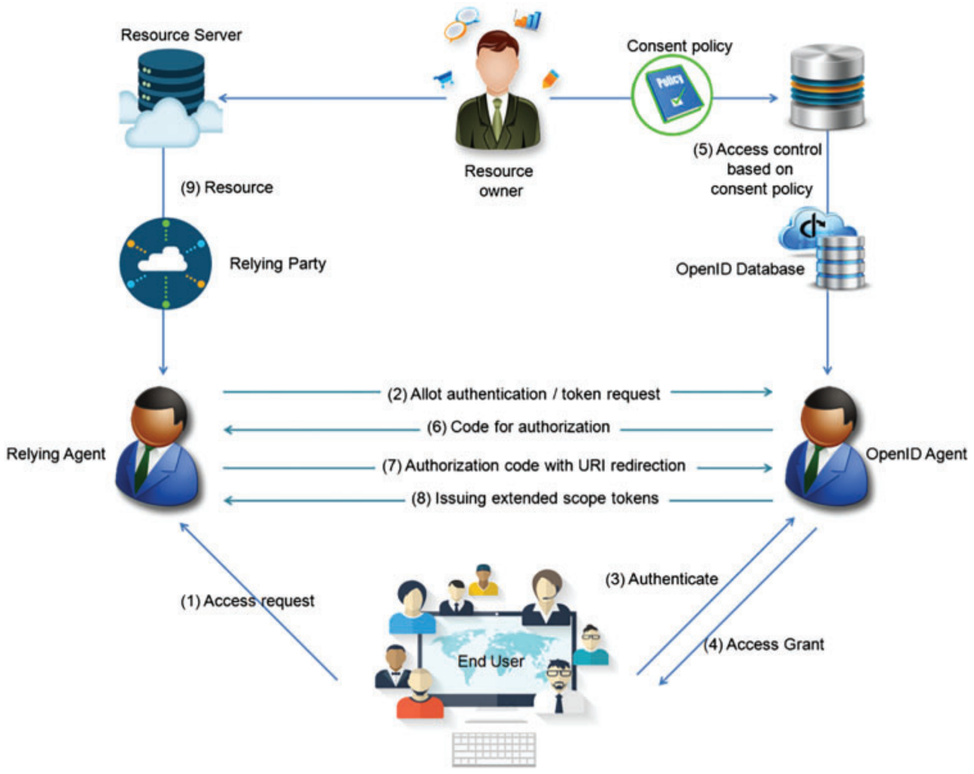
Resource Server
Resource owner
Consent policy
(5) Access control based on consent policy
(9) Resource
Relying Party
OpenID Database
(2) Allot authentication / token request
(6) Code for authorization
(7) Authorization code with URI redirection
(8) Issuing extended scope tokens
Relying Agent
OpenID Agent
(1) Access request
(3) Authenticate
(4) Access Grant
End User

**Figure 5:** O-ID CaaS process

## 6 Proposed Model Implementation

OpenID Connect uses REST/JSON message flows because the integration is very easy for developers while comparing with previous federated identity protocols. JSON, a simple message format based on text, is frequently used along with REST web services. O-D Connect specifications written in Python.30 are implemented in the Open-source Python Reddit API Wrapper (PYOIDC) tool. Members of the OpenID community develop and maintain it. CherryWado, an Open-source tool, simulates the WADO server written in Python.31. The implementation of Python web applications is done using Web Server Gateway Interface (WSGI) by Cherrywado. Python imaging library handles Digital Imaging and Communications in Medicine (DICOM) images and offers image processing and graphics potentials. The information regarding how the Web Access DICOM Objects (WADO) service does not assume/where the DICOM files are stored. The DICOM access layer specifies how to retrieve identical DICOM files from DICOM image repositories. Our prototype implementation is indicated in the class diagram of Fig. 6.

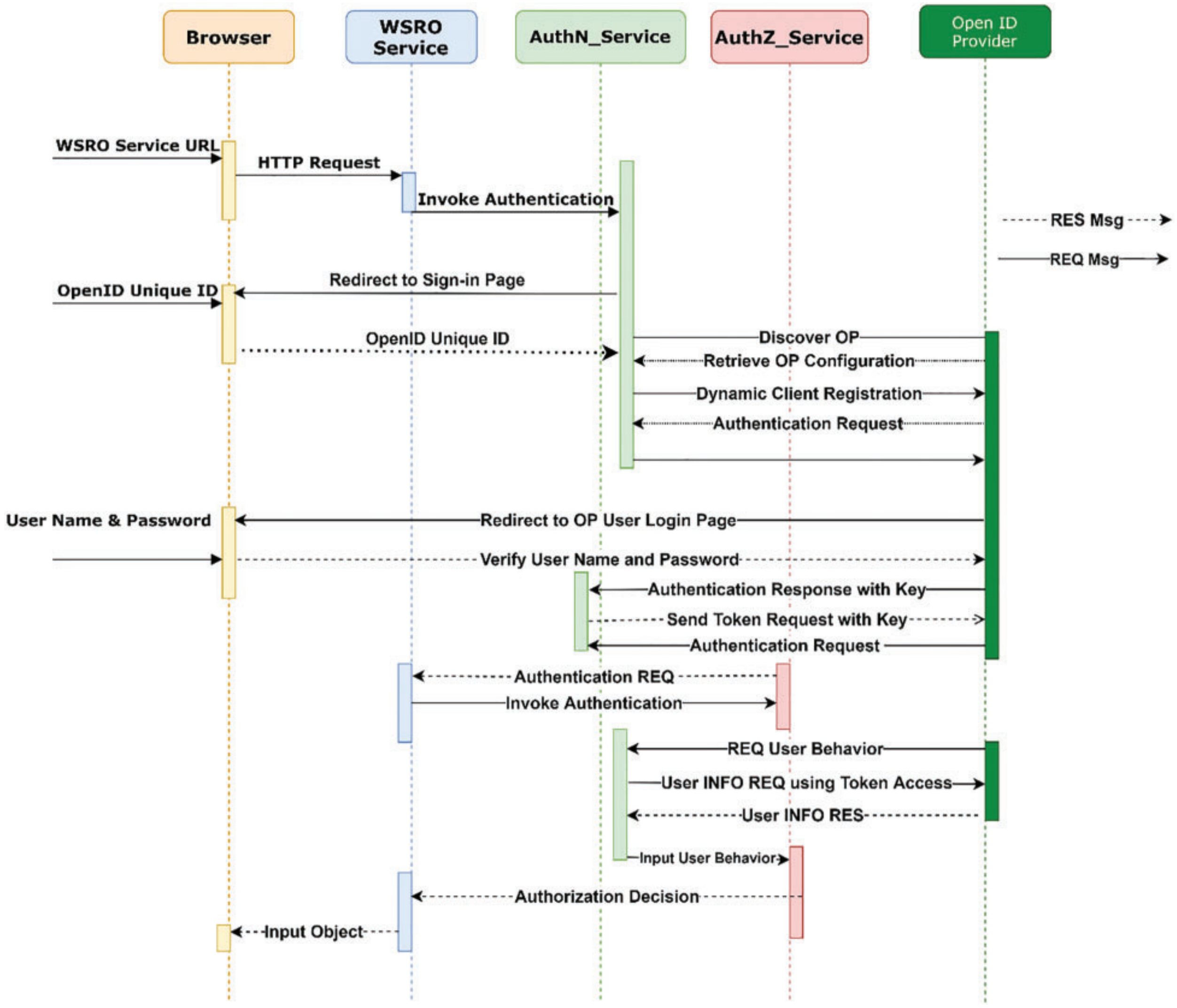

**Figure 6:** The flow chart of our proposed model performance

*(a) OpenID Identifier*

In the program, the user first accesses the Uniform Resource Locator (URL) of WSRO administration. Then, from AuthN Service, HTTP demand is acquired, and a validation activity is evoked by WSRO control. Finally, the user's program is diverted to an SSO page, and an O-ID is requested by AuthN control.

*(b) Delegate Validation and Demand Token*

The utilization of the end-point URL http://testmodel.com/ by OP area is discovered strongly by *AuthN* control. Operating Procedure (OP)'s design metadata is controlled by AuthN and registers with OP as a user.

*(c) Authentication*

After deployment, a validation request is sent to AuthN control by maintaining the ideal request parameters to OP.

(*d) Issue Tokens with Expanded Extension*

After getting an effective validation response, OP retrieves a verification code to *AuthN*. Then, for the protected asset, ID tokens and access token are obtained.

*(e) Access Control Optimal*

To obtain an authentic method, WSRO controls the *AuthZ* command. To obtain the necessary user behaviour, *AuthN* control is invoked by the *AuthZ* command. The resource demands of the authorized user, like the user's full name, photo, e-mail ID, etc., are accomplished by AuthN control that sends a request to the OP to utilize access tokens developed through verification.

### *6.1 Web Service Security and Integration Benefits*

The Web Service technology designs both SSO and authorization services. Furthermore, high-level Web API supports the access of a cloud computing platform, which is service-oriented. This is why, within a cloud environment, incorporating these security services does not create any technological incompatibility problems. Furthermore, by deploying all the merits of service-oriented architecture, it can be employed and manipulated effectively. These are the major merits of Web Service, which the proposed cloud security system acquires from the service-oriented architectural design.

*(a) Loosely Coupling*

The self-encapsulated software modules are known as security services in which, through standard Internet communication protocols, service functionalities are delivered. From service requesters, the particulars of their execution are hidden, and therefore, the requester side will not be affected by any internal system alteration. In addition to that, the consumption of those services by other systems is not impeded by the fundamental system intricacy as only through the high-level interface there is user accessibility.

*(b) Standardized Protocols*

The following layers have consisted of the Web Service protocol stack: service description layer, XML messaging layer, service transport layer, and service discovery layer. The implantation of a specific system allows selecting from the comprehensive collection of distinct standard protocols.

*(c) Interoperability*

Interoperability is the crucial merit of Web Service from where the security system benefits. Through public networks like the Internet, the system is delivered interoperable, separate from its implementation platform. Thus, communication between the platform-independent service provider and requester happens in the absence of all hindrances.

*(d) Usability*

Client applications use Web Service. The client application links with the service end-point and does service calls on that end-point irrespective of the tools and programming languages employed at the implementation time. With the help of serialized request-response messages by considering standard data formats such as XML, communication happens.

*(e) Deployability*

Through standard Internet technologies, security services are used on application servers and SSL over HTTP channel security mechanisms, and outgoing messages can be transmitted through firewalls. Through multiple built-in security mechanisms offered by Web Services Security protocol, the management of security solutions for our system is enabled by Web Service. However, the implementation and deployment of the system become less by these merits. The deported communication between the service requester and provider is the sole demerit caused by Web Service technology to the security system. This is why more mechanisms are required to monitor service requests.

## 7 System Security Assessment

Considering the security for both authentication and authorization methods, the analysis of the proposed security system is done for probable security threats. For both the services, replay attacks, the confidentiality of message information, message alteration, refutation and imitation are the five distinct probable attacks. Whether both services are secured against those security threats is shown in Tab. 1. In both services, the session IDs are generated randomly, and we can avoid replay attacks. For both services, the privacy of the message is protected: messages are passed through a secure channel like Secure Sockets Layer (SSL)/Transport Layer Security (TLS); in the case of authorization service, XML encryption standard encrypts the messages. Using XML digital signature standard, impersonation attack is prohibited for both services since the intended entity's information is the message's origin (Fig. 6).

**Table 1:** Addressing of security hazards

| Type of attacks | Authentication | Authorization |
|---|---|---|
| Replay | Discussed | Discussed |
| Message information disclosure | Discussed | Discussed |
| Message tampering | Discussed | Discussed |
| Masquerade | Discussed | Discussed |
| Denial of service | Discussed | Discussed |

## 8 Conclusion

In this paper, an attempt has been made to design and develop a solution to secure authentication and authorization service in cloud computing. The IaaS model delivers Infrastructure as a Service, computation of infrastructure, and physical storage. On OpenStack and VMware platforms, the SS-O principle-based proposed solution was applied. We attempted to enhance a solution that would give a cloud computing environment a trustworthy and protected authentication and authorization with the help of multi-agents. Monitoring access control reduces the trustworthiness of presumed registering environments, and the analyzes of the performed query are about it. Consecutively, in the environments based on Cloud, an opening control authority is prefixed in this model for assembling the observing skills.

A stand-out data header will extend managing data to represent a unique data header and elevate the trust pace in cloud servers in access control authority. In this access control model, a couple of areas will assign the data, and in multiple cloud servers, every fragment will be taken care of. And the same is done for distributed data, which will be joined at the Authentication and Authorization SaaS cloud server. Besides being user-centric access control based on consent, O-ID IaaS with current authorization workflow is used for imposing access control policies specific for enterprise. Using the benefits of service-oriented architecture, the deployment and exploitation of Web Service Security can be done effectively. The attack-oriented threat model is the basis of security evaluation. Understanding system security assessment is simplified by a threat model that provides a formal approach for ordering latent security problems.

**Funding Statement:** The authors received no specific funding for this study.

**Conflicts of Interest:** The authors declare that they have no conflicts of interest to report regarding the present study.

## References

[1] M. Tebaa and S. E. Hajji, “Secure cloud computing through homomorphic encryption,” *International Journal of Advancements in Computing Technology*, vol. 29, no. 5, pp. 29–38, 2014.
[2] B. Sharma and R. Delhi, “Security architecture of cloud computing based on elliptic curve cryptography (ECC),” *International Journal of Advances in Engineering Sciences*, vol. 3, no. 3, pp. 58–61, 2013.
[3] G. Ateniese, R. Burns, R. Curtmola, J. Herring, O. Khan *et al.,* “Remote data checking using provable data possession,” *ACM Transactions on Information and System Security*, vol. 14, no. 1, pp. 34–34, 2011.
[4] N. Ruangchaijatupon and P. Krishnamurthy, “Encryption and power consumption in wireless LANs-N,” in *The Third IEEE Workshop on Wireless LANs*, Newton, Massachusetts, pp. 148–152, 2001.
[5] N. Ferguson, D. Whiting, B. Schneier, J. Kelsey, S. Lucks *et al.,* “Helix fast encryption and authentication in a single cryptographic primitive,” *Proc. Fast Software Encryption*, vol. 2887, pp. 330–346, 2003.
[6] X. Jing, Z. Yan and W. Pedrycz, “Security data collection and data analytics in the internet: A survey,” *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 586–618, 2019.
[7] D. Eckhoff and I. Wagner, “Privacy in the smart city—applications technologies challenges and solutions,” *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 489–516, 2018.
[8] M. Xu, K. M. Schweitzer, R. M. Bateman and S. Xu, “Modeling and predicting cyber hacking breaches,” *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 11, pp. 2856–2871, 2018.
[9] Z. Tari, “Security and privacy in cloud computing,” *IEEE Cloud Computing*, vol. 1, no. 1, pp. 54–57, 2014.
[10] C. Hyseni, “The proposed model to increase security of sensitive data in cloud computing,” *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 2, pp. 203–210, 2018.
[11] J. Li, Y. Zhang, X. Chen and Y. Xiang, “Secure attribute-based data sharing for resource-limited users in cloud computing,” *Computers & Security*, vol. 72, no. 4, pp. 1–12, 2018.
[12] S. Kumar, S. K. Singh, A. K. Singh, S. Tiwari and R. S. Singh, “Privacy-preserving security using biometrics in cloud computing,” *Multimedia Tools and Applications*, vol. 77, no. 9, pp. 11017–11039, 2018.
[13] Y. Yan, L. Wu, G. Gao, H. Wang and W. Xu, “A dynamic integrity verification scheme of cloud storage data based on lattice and bloom filter,” *Journal of Information Security and Applications*, vol. 39, no. 11, pp. 10–18, 2018.
[14] Q. Zhang, S. Li, Z. Li, Y. Xing, Z. Yang *et al.,* “CHARM: A cost-efficient multi-cloud data hosting scheme with high availability,” *IEEE Transactions on Cloud Computing*, vol. 3, no. 3, pp. 372–386, 2015.
[15] T. Dbouk, A. Mourad, H. Otrok, H. Tout and C. Talhi, “A novel ad-hoc mobile edge cloud offering security services through intelligent resource-aware offloading,” *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1665–1680, 2019.

[16] R. Maeser, “Analyzing CSP trustworthiness and predicting cloud service performance,” *IEEE Open Journal of the Computer Society*, vol. 1, pp. 73–85, 2020.
[17] W. Qiao, Y. Liu, L. Xi, X. Li, Z. Li *et al.,* “A novel method for resource-efficient security service chain embedding oriented to cloud datacenter networks,” *IEEE Access*, vol. 9, pp. 77307–77324, 2021.
[18] W. Ke, Y. Wang, M. Ye and J. Chen, “A priority-based multicast flow scheduling method for a collaborative edge storage datacenter network,” *IEEE Access*, vol. 9, pp. 79793–79805, 2021.
[19] S. Rizvi, J. Mitchell, A. Razaque, M. R. Rizvi and I. Williams, “A fuzzy inference system (FIS) to evaluate the security readiness of cloud service providers,” *Journal of Cloud Computing*, vol. 9, no. 42, pp. 964, 2020.