Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Review

Encrypted token based authentication with adapted SAML technology for cloud web services

I.Indu [a], P.M.Rubesh Anand [a], Vidhyacharan Bhaskar[b,*]

[a] *Department of Electronics and Communication Engineering, Hindustan University, Chennai 603103, India*
[b] *Department of Electrical and Computer Engineering, San Francisco State University, 1600 Holloway Avenue, San Francisco, CA 94132, USA*

ARTICLE INFO

Keywords:
Cloud computing
Identity management
Authentication
Web services
Security

ABSTRACT

Web applications and cloud services are rapidly emerging as the inevitable technology for communication between organizations. Cloud-based solutions are currently deployed to provide improvement in the existing business processes and services. The major challenge involved in cloud is data security that is stored and transferred. Cloud infrastructure requires an extensive authentication mechanism to protect data as well as to ensure that the right person is accessing the right information. In this paper, token based fine grained authentication for cloud web services with the help of adapted Security Assertion Markup Language (SAML) technology is proposed. The entire set of communications between Identity Provider, Service Provider and Cloud Server is encrypted to enhance the security. The combination of SAML and single use access token based verification provides improved security to cloud web services. The proposed adapted SAML authentication mechanism ensures flexibility and scalability of the environment by the provision of adding multiple numbers of trusted sources and web services.

1. Introduction

Cloud computing is an emerging technology that has enormous potential in fulfilling the customer demands with efficient resource utilization and high accessibility. Cloud computing has its applications in many commercial fields that need elastic and dynamic computing resources. Cloud system is based on service-oriented computing architecture which is capable of providing Database-as-a-service (DbaaS), Identity-as-a-service (IDaaS), and Anything-as-a-Service (XaaS) (Jansen and Grance, 2011). While implementing the cloud service model, there are quite a number of factors like flexibility, scalability, interoperability, and control of service to be considered for determining the best option for the cloud service model (Khan, 2016). Another facet of the cloud system is complexity and its associated security challenges. Cloud system is vibrant in nature by considering numerous users, devices, networks, organizations, and resources that are frequently connected and disconnected to/from the system (Sood, 2012; Atwood et al., 2016). The various security challenges, vulnerabilities, attacks, and threats in the area of cloud computing during the sharing and virtualization of resources, resource pooling and public nature of the cloud have been attempted by many researchers (Singh et al., 2016; Subashini and Kavitha, 2011; Sen, 2013; Xiao and Xiao, 2013).

Cloud computing is gaining its importance and usage in accomplishing the organizational requirements. The need for collaboration between different departments in an organization or between various organizations is increasing to ease the tasks (Tari et al., 2015). The organizations are forced to keep data or information in an environment which is easily accessible by the stake holders or authorized persons. In such situations, the suitable solution is to store data in a cloud environment. This helps the organizations to share data easily and cost effectively with others. Organizations are required to develop a secure solution to control data access in a cloud environment to avoid data security issues (Rasheed, 2014). In order to minimize the privacy and security risks in cloud web services, organizations require a strong, flexible, scalable, and accountable Identity Management system (IdM) (Singh and Chatterjee, 2015; Sharma et al., 2016). This can be achieved by ensuring that the right personnel are accessing the information. Cloud service providers (CSPs) provide identity and other kinds of access management for the customers in their cloud infrastructure. However, a large number of data leakage incidents are caused by the vulnerabilities in the existing identity management systems (Bhonsle et al., 2013; Butun et al., 2016; Liu et al., 2015). Presently, the mechanism of identity management is mainly CSP-centred, which hardly meets the requirement of users' fine-grained and flexible access control policy. Cloud computing requires extensive authentication and authorization mechanisms to secure its data and resources due to the complexity of

* Corresponding author.
E-mail addresses: indu.i2043@gmail.com (I. I.), rubesh.anand@gmail.com (R.A. P.M.), vcharan@gmail.com (V. Bhaskar).

http://dx.doi.org/10.1016/j.jnca.2017.10.001
Received 1 March 2017; Received in revised form 21 August 2017; Accepted 2 October 2017
Available online 05 October 2017
1084-8045/ © 2017 Elsevier Ltd. All rights reserved.

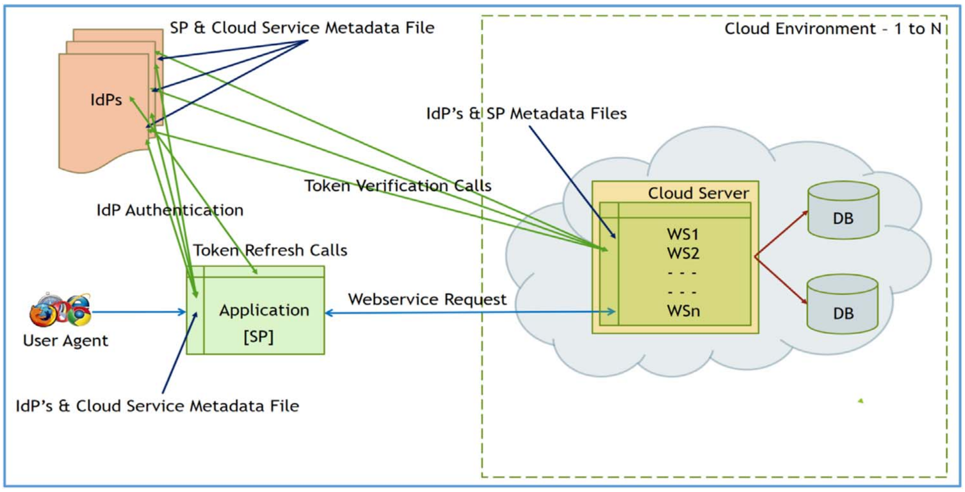


Fig. 1. Overall architecture of the proposed authentication model.

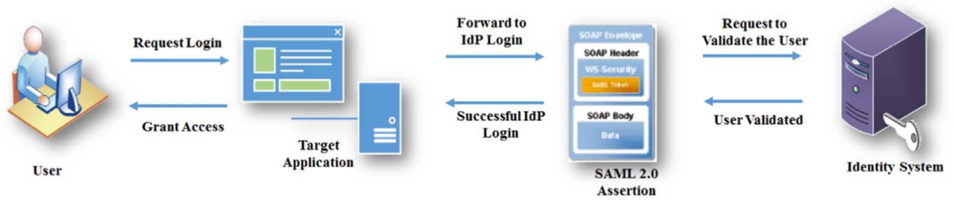


Fig. 2. SAML2.0 SSO (Single Sign On) workflow between user and Identity system.

usage. Lack of this efficient mechanism creates multiple challenges like data security, privacy, trust management, identity management, risk management, compliance, transparency, and data leakage (Lee, 2016). The loss of control and transparency issues are created while storing and processing identity information by the third parties (CSP) or outside the organizational boundaries. Due to these distinct security challenges, the rate of migration to cloud is slow regardless of the assured and attractive features of the cloud (Chang and Ramachandran, 2016). In spite of these issues, the organization has a tendency of reluctance in contributing their critical identity information to cloud (Fernandes et al., 2014). The various limitations that are incorporated with cloud technology like Quality of Services (QoS), interoperability, elasticity, and interruption of services affect the potential of this technology. In order to reduce these limitations, the scientific community and specialized industrial institutions avail the benefits of multi-clouds. The efficient way to access multiple clouds simultaneously needs verification of the identity of the users by each cloud. Cloud web services verify the identity of the users to provide security management and personalized user experience in multi-cloud environment (Liu et al., 2009; Nicanfar et al., 2014).

Single Sign On (SSO) protocols are integrated in web services in order to access multiple services at any time by the end users from the identity providers. Identity Providers (IdPs) utilizes integration tools that allow developers to implement SSO in a few minutes (David et al., 2011; Lewis and Lewis, 2009). Traditionally, web users need to authenticate for all their services, each with their own credentials one after another. The single sign on method permits users to log in once for accessing multiple services through authentication services of services providers. The authentication services and service providers allow cross domain accessibility by enabling the identity information exchange from one domain to another (Nordbotten, 2009). In SSO, web services grant access to users depending on their identity information through different protocols that are used in SSO assertions such as, OpenID, OpenID Connect, OAuth2.0, and

Table 1
Notations and their Interpretations.

Terms or Notations	Interpretations
TR	Token Request
SSO	Single Sign On
U_I	User Identifier
SP_{ID}	Service Provider Entity Identification
T_t	Token expiry time
N	Count
IdP_{ID}	Identity Provider Entity Identification
SYS_TIME	Current System Time
RNG	Random Number Generator
TS_n	Generated Array of Tokens
MF_{IDP}	Identity Provider Metadata File
WS	Web Services

BrowserID (Cirani et al., 2015; Tysowski, 2016). Single Sign On and federation are mainly used for security domains in sensitive data communication. Security Assertion Markup Language (SAML) is one of such methodology that provides SSO and has advantages for use in cross-domain networks (Al-dubai and Khamitkar, 2014). (Waqar et al., 2013) emphasized on security deficiencies and privacy issues in the users' data and exploration of the possibility of deploying anonymity based authentication scheme for preservation of cloud users. (Armando et al., 2013) proposed a single sign-on protocols for the authentication flaw in browser and its remediation. The existing protocols such as SSO protocols namely, the SAML SSO and Open ID suffered from authentication flaw. (Wang, 2015) formalized an identity based system model for certificate management by improving the flexibility and efficiency to realize private verification, public verification and delegated verification based on the client's authorization. An integrated identity and attribute based access management system to alleviate the security issues in authentication was proposed which provides flexibility, accountability and scalability for cloud web services (Indu and Rubesh Anand, 2015). A lightweight authentication and authorization mechanisms for embedding authentication and authorization functionality on constrained smart objects was proposed (Hernandez-ramos et al., 2015). A survey on different cloud federation architectures and their evaluations based on their functional and non-functional properties for the purpose of handling traffic exhibits the recent demand on federation of cloud (Assis and Bittencourt, 2016; Kritikos et al., 2017). Though there are various solutions for authentication related problems, the authentication mechanisms for cloud based networks still suffer in their security aspects. The motivation of this research is to address some of the major challenges which are currently affecting the cloud based authentication such as, man-in-the-middle attacks, insider attacks, password/key compromise and replay attacks. In this paper, a token based authentication for cloud web services with the help of SAML technology is proposed. The access tokens utilized for identity management are encrypted to provide improved security and to also prevent the tokens from multiple usage. The SAML methodology is modified for identity provider and service provider to ensure that the authentication mechanism provides flexibility and scalability for multiple users and web services.

Further, the rest of the paper is organized in five sections. The conceptual framework details of the proposed authentication technique are provided in Section 2. The various authentication mechanisms utilizing SAML are explained in Section 3. The experimental results with comparative analysis and discussions of proposed mechanisms are included in Section 4. Finally, Section 5 summarizes and provides the conclusions.

2. Conceptual framework

2.1. Architectural overview

We propose an integrated identity management system for cloud web services by combining SAML and token based verification to provide improved security. In the proposed system, identity provider(s), service provider(s) and web services provider(s) are integrated in a single architecture. The possibilities of securing data in a cloud environment are explored through a hybrid model of encrypted SAML assertions for authentication and access tokens for web services. SAML based communication is established for the purpose of authentication with the help of metadata files. A metadata file defines the targets, binding method and attributes which are needed to be passed in a SAML request. The predefined metadata files are saved in the Identity Provider (IdP), Service Provider (SP) and Web Services (WS) provider for authentication. In the proposed scenario, in order to obtain access to the cloud web services, the user has to authenticate through an identity system from a client application (service provider). The successful authentication results in the generation of access tokens by the identity system. These access tokens are stored in the identity provider and service provider. Access tokens are shared for verification while accessing the web services from cloud servers. In the cloud web server, the validation of the access token is performed directly with the identity system through a mechanism based on a verification protocol. In this proposed model, it is possible to integrate multiple numbers of identity providers (organizations), service providers (applications) and cloud web servers/services through enhanced scalability feature as shown in Fig. 1. The proposed solution has the capability to add or remove any web servers/services, client applications (service providers) and organizations (identity providers) to or from this architecture with minimal modification. This also ensures to avoid direct user interactions with web servers/services which are hosted in cloud infrastructure.

Functional Feature 1*: The authentication model supports high scalability through addition or deletion of multiple numbers of web-servers/services, identity providers and service providers.*

2.2. SAML 2.0 based Federated SSO

SAML 2.0 is widely used in authentication for web based applications by aiding the service provider applications to verify and validate the access requests against the corresponding Identity Provider (IdP) system. The communications between the target application and IdP system are happening in a secure way with the help of encryption and some predefined services which are already defined in the metadata files. The predefined metadata files are stored in the application (service provider) as well as in the identity provider. The work flow between the user and the IdP system initiates when a user in an organization tries to access a client application (service provider) as shown in Fig. 2. The client application (service provider) requests authentication to the user through organization's identity provider. The user navigates to the authentication screens of the identity provider and provides the authentication information. Once the identity provider authenticates the user against the organization identity store, it constructs a SAML2.0 assertion with user information and sends the assertion to the client application (service provider). The client application (service provider) validates the SAML2.0 assertion and authenticates the user for federated SSO.

2.3. Token generation and token refresh

Once the SSO authentication through SAML is successful, the identity provider generates access tokens and stores a copy of the same at the identity provider. The identity provider system shares the generated access tokens with the associated service provider. The service provider stores these access tokens and uses any one of the access token for the web service request. Once a token is utilized for a request, the same access token is prohibited from reuse. The used access token is removed from the service provider's token store. While verification happens in identity provider system, the corresponding

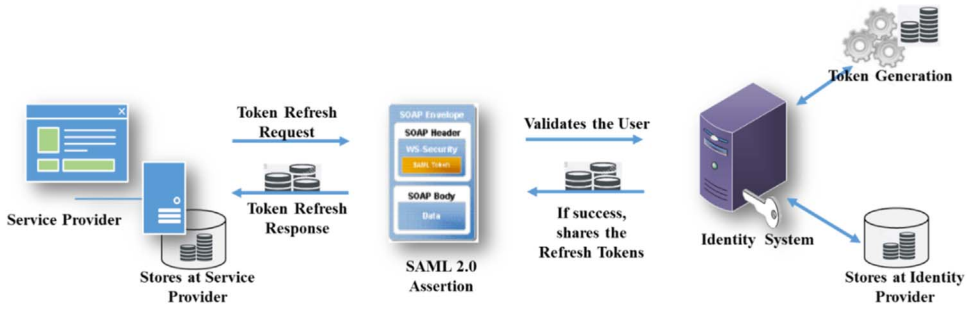


Fig. 3. Token generation and refresh workflow between service provider and identity system.

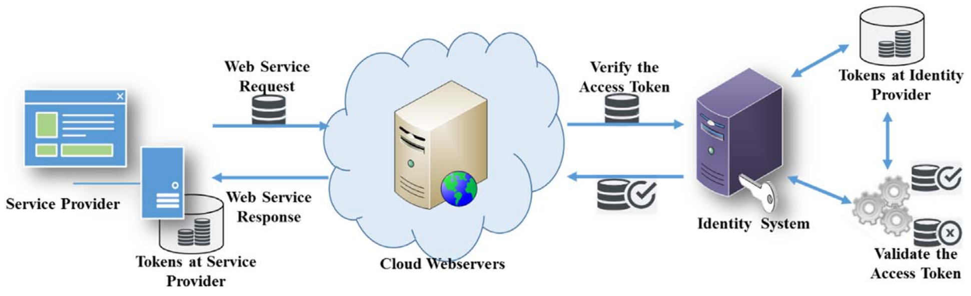


Fig. 4. Workflow of the token verification by service provider with the identity system.

token needs to be removed from the identity token store as well. This helps to ensure that the tokens are used only once. The tokens are refreshed based on the request from service provider. The service providers request for new set of tokens either after consumption of the available tokens or after expiration of the token validity period. The access token is a hexadecimal random string which is generated by the identity provider system in predefined numbers. The identity provider considers the parameters like identity of the user, service provider entity id, current time and token expiry time for the generation of access tokens. The access token is generated on the basis of random numbers, user information, and IdP-SP identity as in Algorithm 1. The hashing algorithm for generating the access tokens is based on SHA-2. The user identity information is not stored in the access tokens. Table 1 represents the notations and their interpretations that are used in the proposed authentication model.

Algorithm 1:. Generation of Access Tokens

```
Input: TR, SSO, $U_I$, $SP_{ID}$, $T_t$, N, $IdP_{ID}$, SYS_TIME, RNG;
Output: $TS_n$;
                    // Process at Identity Provider //
01: if (TR & SSO = TRUE) then
02:     $IdP_{ID}$ = GET IdPEntityId ();
03:     SYS_TIME = GET SysTime ();
04:     RNG = GET RandomNumber ();
05:     $T_t$ = SYS_TIME + 60 min;                      // Set Token Expire time //
06:     if ($U_I$ & $SP_{ID}$ = TRUE) then              // Check for User and Service provider validity //
07:         for i = 1 to N do
08:             Let $TS_n$(1....N) be new array
09:     Repeat:N = XOR ($SP_{ID}$, $U_I$, $IdP_{ID}$, SYS_TIME, RNG);
10:             $TS_n$ [i] = HASH (N);
11:             if ( $T_t$ < SYS_TIME is TRUE for $U_I$) then
12:                 GENERATE ($TS_n$ [i]);
13:                 STORE (TSn [i]);
14:                 SET Expiry ($T_t$);
15:             else
16:                 Return $TS_n$[i];
17:             end if
18:         end for
19:     else
20:         CALL (SP Metadata File);
21:         INVOKE (TokenRefreshResponse);
22: else
23:     REJECT (TR);
24: end if
```

The token generation and refresh workflow is initiated from the service provider through the client application as shown in Fig. 3. Service Provider initiates a token refreshing SAML request to the user's identity system. The identity system validates the user for refreshing the access token. When the validation is successful, the identity system generates a set of access tokens and also stores the same in the identity provider.

The generated access tokens are shared to the service provider along with the SAML response. The client application (Service Provider) validates the response and stores the newly generated tokens in Service Provider. The access token generation request is invoked in the below listed cases:

Use Case 1: **Successful SSO authentication by a user identity -** Identity Provider system automatically invokes the token generation request once a user successfully authenticates through a SAML SSO request. The generated access token is stored in the identity provider and the same is passed to the connected service provider.

Use Case 2: **Expiry period of the current tokens -** Identity Provider system invokes the token generation request to generate a new set of tokens after the expiry period of the current set of access tokens.

The process happens only for users who are not inactive for a specific period of time. The identity provider shares the token details to the corresponding service provider in the same way as in the above case.

***Use Case 3:* Token Refresh Request from Service Provider -** When the available access tokens are utilized by the user, the service provider request for a new set of access tokens from identity provider. The identity provider invokes the token generation process based on the request from service provider system and shares the newly generated access token with SP.

Functional Feature 2*: The valid access tokens are used only once and invalid tokens are removed from the identity provider store.*

Additional considerations in the Token refresh request and response are possible with the inclusion of some additional conditions in the SAML assertion like, validity, target audience and specific time interval. A sample schema of SAML assertion with URL (Uniform Resource Locator) is given as

```
< saml:Conditions
  NotBefore="Date & Time"
  NotOnOrAfter=" Date & Time " >
   < saml:AudienceRestriction >
   < saml:Audience > SP URL < /saml:Audience >
   < saml:Audience > IdP URL < /saml:Audience >
   < /saml:AudienceRestriction >
   < /saml:Conditions >
```

2.4. Storage mechanism

The access tokens are stored as a nested object in the identity and service providers for quick access and faster response. In the proposed authentication model, the access tokens are stored in the Identity Provider system as well as in the Service Provider system with suitable storage mechanisms like database, flat files, or objects depending upon the coding language used in the system development.

Functional Feature 3*: The storage of access tokens as nested object provides quick and faster response.*

The schema of the nested object in Identity Provider System:

Map < **User Identity** as key, **Map** < **SP Entity ID** as key, **Array** < Index, Token > > >

The schema of the nested object in Service Provider System:

Map < **User Identity** as key, **Map** < **IdP Entity ID** as key, **Array** < Index, Token > > >

2.5. Access token verification

The token verification happens at the Identity Provider for every access token before its usage as in Algorithm 2. The client application (service provider) initiates a SAML based webservice request along with the access token. The cloud web server receives the web service request along with the access token from the client application (service provider). The cloud server passes the access token to the Identity Server with the verification request. The IdP system validates the access token with its token store and determines the authority. In the event of successful verification of the token, the identity provider system removes the access token from its store and simultaneously responds to the cloud server with the verification result. The removal of the access token from IdP system and SP ensures that the tokens are utilized for one-time consumption. This process also helps to increase the security as duplication of tokens are prohibited. The cloud server verifies the response from the IdP and responds back to the client application (service provider) as shown in Fig. 4. At the other end, the access tokens are removed from the service provider store once they are used.

Functional Feature 4*: All access tokens are verified at the identity provider before their usage and the verified tokens are removed from identity provider and service provider systems instantaneously.*

Algorithm 2:. Verification of Access Tokens

```
Input: $TS_n$, $MF_{IDP}$, $U_I$, $SP_{ID}$, $T_t$;
Output: WS;
                              // Process at Identity Provider//
01: GENERATE (TokenverificationRequest ← ServiceProvider);
02:   SEND_REQ (IdentityProvider ← VALIDATE ($SP_{ID}$));
03:     CALL ($MF_{IDP}$);
04:     if ($SP_{ID}$ = TRUE) then
05:           VALIDATE ($TS_n$);
06:       if ($TS_n$ = TRUE) then
07:             SEND_ACK (ServiceProvider ← IdentityProvider);
08:             WS ← ENABLE;
09:           else
10:             if ($T_t$ < SYS_TIME) then
11:                   INVOKE (TokenRefreshRequest);
12:     else
13:       SEND_NEG_ACK (ServiceProvider ← IdentityProvider);
14:       INVALID ($TS_n$);
15:       INVALID ($U_I$);
16:       DELETE (TokenverificationRequest);
17:     else
18:       DELETE (ServiceProvider_REQ);
19:     end if
```

3. Token based authentication mechanism

3.1. Existing services and protocols

The implementation of the proposed authentication architecture needs a set of services and protocols to be defined for use at Identity Provider, Service Provider and Cloud Webservers. The existing SAML2.0 protocols are considered with some modifications (Pérez Méndez et al., 2016). The features like token refresh, token verification and web service request require modifications for new services and protocols in the proposed architecture. There is a set of existing protocols which are defined by OASIS for the implementation of SSO based authentication (Cantor et al., 2005a, 2005b). The major protocols available for authentication are Authentication Request Protocol, Artifact Resolution Protocol, Name Identifier Management Protocol and Single logout. *Authentication Request Protocol* is used when a user or web agent tries to access a service provider for web services. *Artifact Resolution Protocol* is utilized by a message sender for sending requests and responses through SAML protocol as a small piece of data called an *artifact* using SAML binding by reference. *Name Identifier Management Protocol* is used by identity provider for creating a name identifier of a user entity or changing the value and/or format of the identifier or to indicate the validity of the name identifier. This protocol informs the service providers of any changes by sending them a *< ManageNameIDRequest >* message. *Single Logout*

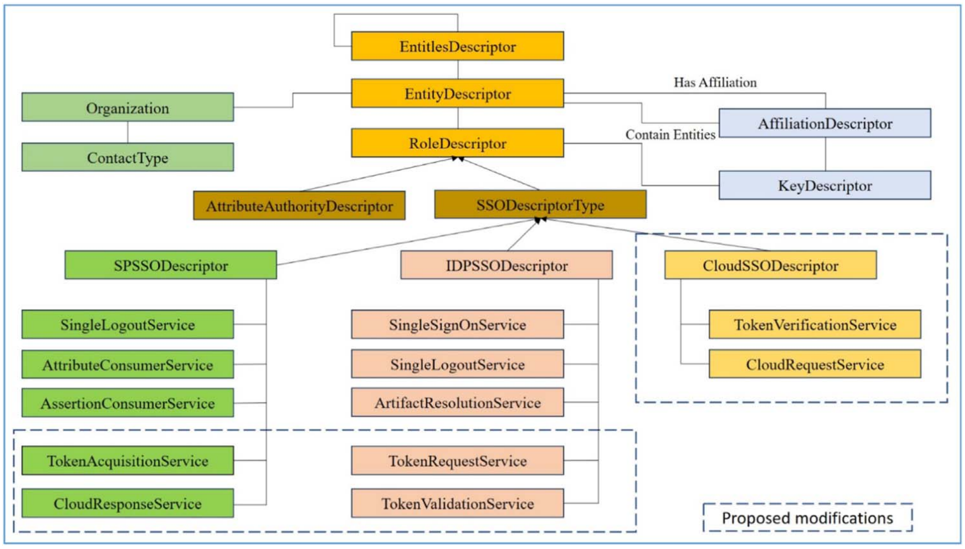


Fig. 5. Simplified Metadata class schema of the proposed authentication model.

Protocol is responsible for the immediate termination of all sessions provided by a particular session authority through a message exchange protocol. This works when a logout or time out of an active session of the principal or web agent is initiated.

3.2. Token refresh protocol

The Token refresh protocol provides the mechanism for refreshing access tokens, which is generated and stored by the identity provider. These tokens are valid for a certain time period. On or before expiration of the time period of the tokens, identity provider generates the next set of tokens and stores the same in the identity system. Unique set of access tokens are generated for each identity, which has an active login session with the identity provider. This service may have been invoked in two different ways. It could be initiated by a Service Provider for a particular identity which has an active session at the identity provider. This happens when the service provider is requesting for a new set of access tokens for the connected identity. The 'TokenRefreshService' end point URL to access this service is shared through the IdP metadata settings. The service provider should provide mandatory information like user identifier, service provider entity ID while raising an access token refresh request.

Functional Feature 5*: Token is refreshed by identity provider only if 'TokenRefreshRequest' is received from service provider.*

In the other way, the Identity Provider itself automatically invokes the access token refresh request in certain cases. This happens when the validity period of the existing set of access token expires for a particular identity and service provider combination. The token refresh protocol is a part of the IdP server configuration. In single sign in process, the identity provider generates hexadecimal reference tokens (by invoking the token refresh service) for the logged in identity, and stores the same in the identity store. Based on the request from service provider, IdP shares a copy of these access tokens to the service provider. Once the identity provider receives a 'TokenRefreshRequest' from the connected service provider with required parameters like user identity and SP entity ID, the identity provider should share the refreshed tokens to the service provider's 'TokenAcquisitionService" end point URL which is specified in the predefined Service Provider metadata file.

3.3. Token request and acquisition

The Token Request and Acquisition protocol describes the mechanism for placing the 'TokenRefreshRequest' from the service provider to the connected Identity Provider as well as acquiring the refreshed tokens from the identity provider for an active identity. The service provider invokes this service at any time based on the need. After the successful single sign on, the service provider requests for the initial set of access tokens. These tokens are valid for a certain time period. The service providers must place the subsequent request for next set of access tokens on or before the expiring time period of the existing access tokens or after consuming all the access tokens. In order to place the request successfully, it is required to share mandatory parameters like identity and SP entity ID to the Identity provider. This helps the identity provider to identify the user details as well as the connected service provider. After successful verification, identity provider shares the refreshed tokens to the service provider. The 'TokenRefreshRequest' end point URL is specified in the predefined Identity Provider metadata file which is available in the service provider server. The service provider need to store these access tokens for future communications with the cloud web service providers.

3.4. Token verification and validation

The token verification and validation provides the mechanism for verifying and validating an access token in between the web server and the identity provider. The service provider must place the web service request along with an active access token, identity provider entity ID and the user identifier to the identity system. The cloud web server identifies the details of the identity provider from the predefined IdP metadata file with the help of IdP entity ID. The web server places the 'TokenValidationRequest' through the 'TokenValidationService' end point URL. The web server shares the required information like access token, user identifier and the web server entity ID to the identity provider for successful processing of the access token validation. Once the access token is consumed for obtaining a service, that token will be removed from the identity and service provider token store. The 'TokenVerificationService' is designed at web servers for handling the token verification response from the identity provider. The details of the end point URL for this service is available in the webserver

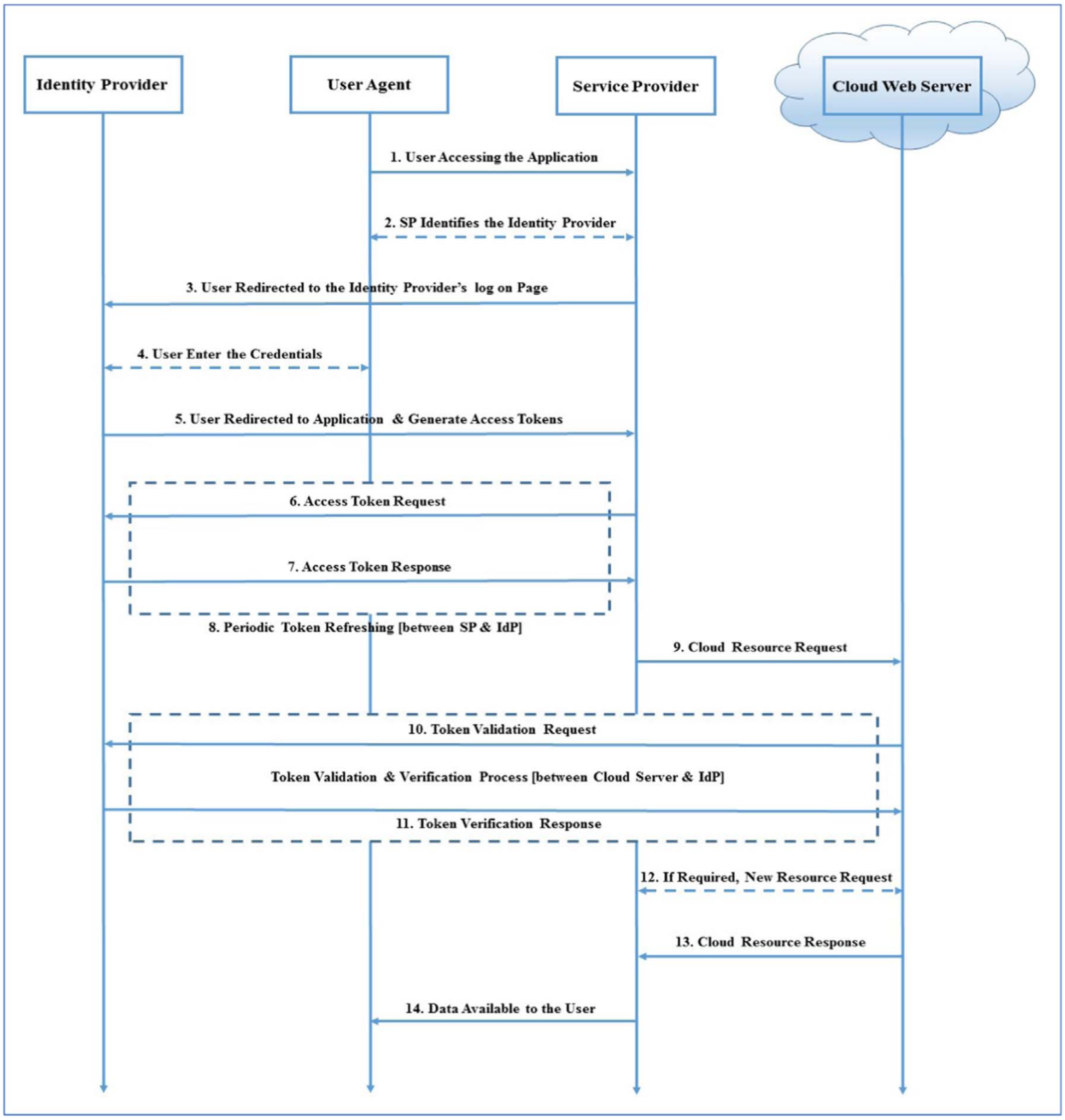

Identity Provider
User Agent
Service Provider
Cloud Web Server
1. User Accessing the Application
2. SP Identifies the Identity Provider
3. User Redirected to the Identity Provider's log on Page
4. User Enter the Credentials
5. User Redirected to Application & Generate Access Tokens
6. Access Token Request
7. Access Token Response
8. Periodic Token Refreshing [between SP & IdP]
9. Cloud Resource Request
10. Token Validation Request
Token Validation & Verification Process [between Cloud Server & IdP]
11. Token Verification Response
12. If Required, New Resource Request
13. Cloud Resource Response
14. Data Available to the User

Fig. 6. Use case diagram of the proposed authentication model.

metadata file which is stored in the IdP system. The cloud web server grants or deny the service to the user based on the positive or negative response from the IdP server.

3.5. Resource request and response protocol

The Resource Request and Response Protocol provides the mechanism for placing the web service request to the cloud servers from service provider and gathering the response from cloud servers. The service provider must share mandatory parameters like active access token, user identifier and IdP entity ID to the cloud webserver for successful placing of the request. Once the access token is used to place a web service request, the service provider must remove/delete that access token from the access token store. Thus, the access token is ensured for single-use semantic such that once it has been utilized, it can no longer be used by any other party. The request placing end point URL of the cloud service is identified from the predefined cloud web server metadata file which is stored in the service provider. Further, the cloud webserver identifies the response gathering end point URL of the service provider from the predefined service provider metadata file which is stored in the Cloud webserver. The servers send the response only to the specified end point URLs depending up on the end point service URLs in metadata files. The URL redirections attacks are eliminated by this mechanism to a certain level.

3.6. Metadata files

In the existing Federated SSO scenario, there are two systems namely, Identity Provider and the Service Provider. Currently, two metadata files are required to establish the SAML2.0 based federated SSO (Cantor et al., 2005a, 2005b). In this proposed model, an additional system of cloud web server is considered. All systems interact with one another and the information required for each system is different as well. We propose modifications in the existing metadata files and few new metadata files. Each system requires to store the metadata information about the other two systems which leads to six metadata files. These metadata files contain the necessary certificates which are used for encrypting all communications between IdP, SP and cloud web service provider.

3.6.1. Modifications in the existing metadata

3.6.1.1. IdP metadata file for Service Providers. IdP metadata file contains information about the Identity Provider and needs to be stored in the Service Provider. Apart from the existing services and settings, information about the newly proposed service '*TokenRequestService*' needs to be included. The metadata schema for the proposed service used by the service providers to place the access token refresh request is given as

```
< md:TokenRequestService
  index="0"
  isDefault="true"
  Location="__End point url__"
  Binding="url:oasis:names:tc:SAML:2.0:bindings:SOAP" / >
```

3.6.1.2. SP metadata file for identity providers. SP metadata file contains the information about the Service Provider and needs to be stored in the identity provider server. In the proposed model, this metadata is modified to add one more service information which is used by the identity provider to share the access token refresh response as given below:

```
< md:TokenAcquisitionService
  index="0"
  isDefault="true"
  Location="__End point url__"
  Binding="url:oasis:names:tc:SAML:2.0:bindings:SOAP" / >
```

3.6.2. Proposed metadata files

3.6.2.1. IdP metadata file for Cloud Service Providers. IdP metadata file contains information about the Identity Provider and needs to be stored in the Cloud Service Provider. This metadata file is similar to the IdP metadata file for SP. The change is in the 'IDPSSODescriptor'. Here, the required information is stored to establish an IdP – Web Server communication. In this model, necessary information to establish an IDP-Web server communication and SP-Web server communication is stored by changing the parameters in 'IDPSSODescriptor' and 'SPSSODescriptor' respectively. This service is used by the cloud web servers to verify the access token. The web server gets information from the 'IDPSSODescriptor' tag of the metadata file as given below:

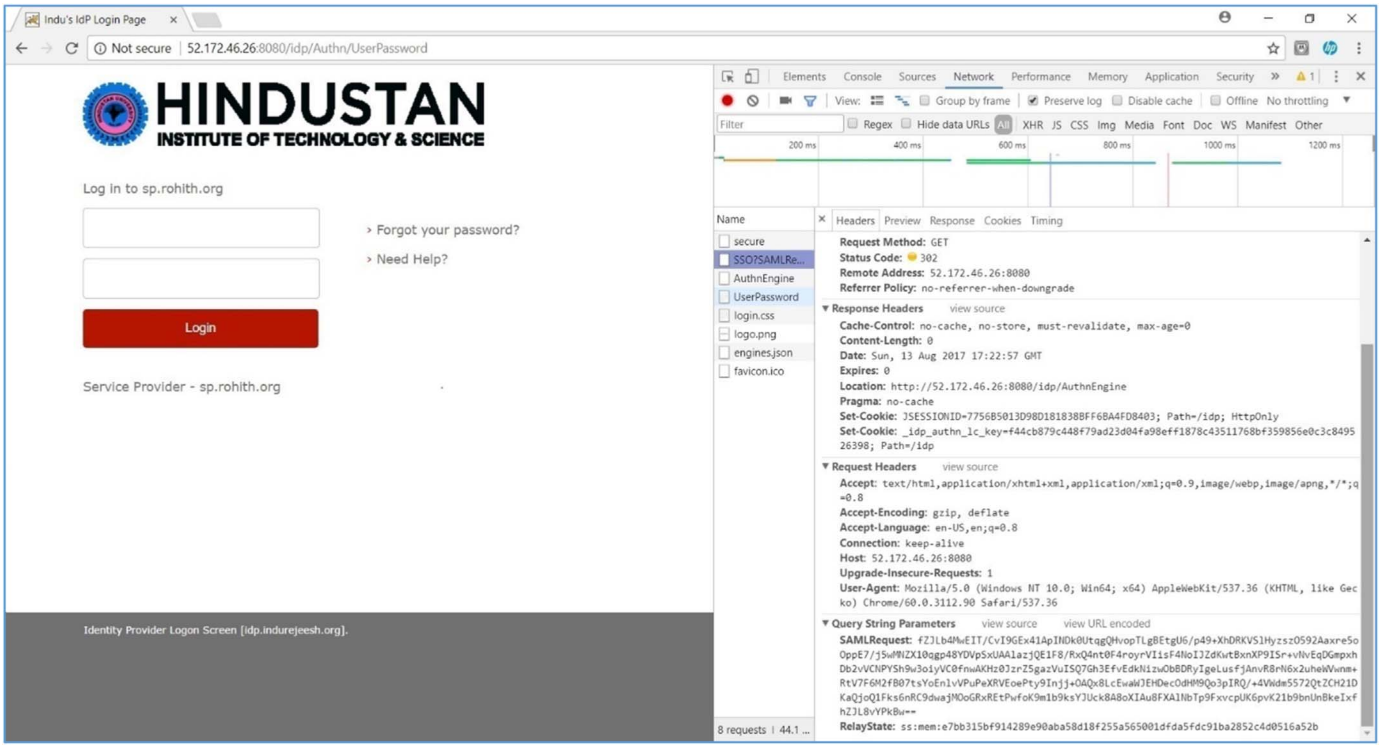
```
< md:TokenValidationService
  index="0"
  isDefault="true"
  Location="__End point url__"
  Binding="url:oasis:names:tc:SAML:2.0:bindings:SOAP" / >
```

3.6.2.2. SP metadata file for Cloud Service Providers. This metadata file contains the information about the Service Provider that needs to be stored in the Cloud Service Provider. This metadata file is similar to the SP metadata file for IdP. The change is in the 'SPSSODescriptor'. Here, the required information is stored to establish an SP – Web Server communication. This service is used by the cloud web servers for responding to the service providers. The web server receives this information from the 'SPSSODescriptor' tag of the metadata file as given below:

```
< md:CloudResponseService
  index="0"
  isDefault="true"
  Location="__End point url__"
  Binding="url:oasis:names:tc:SAML:2.0:bindings:SOAP" / >
```

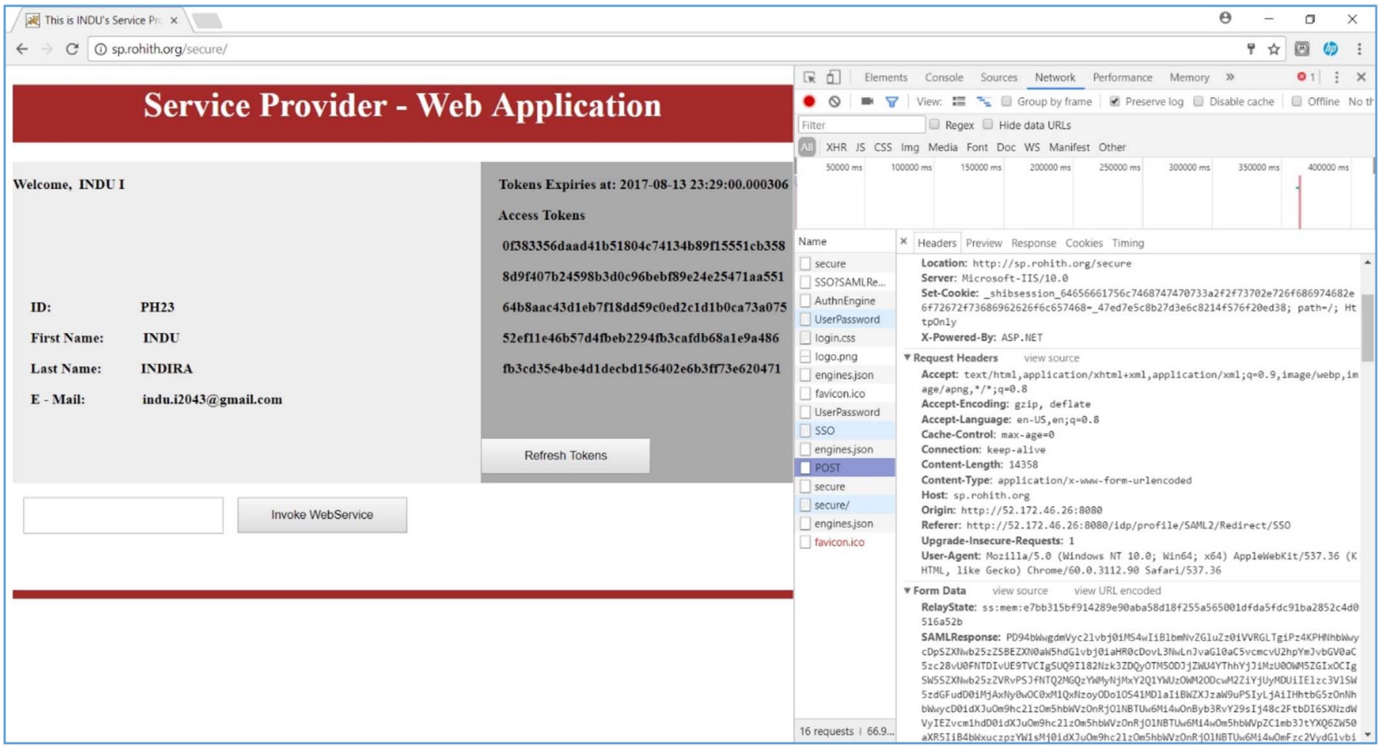
Table 2
Use Case sequences and their explanation.

Different steps in sequence diagram	Explanation
User accessing the application	The user requests access to a target web application (Service Provider). This could be from a user agent (web browsers)
SP identifies the identity provider	When the target application (Service Provider) is configured against more than one Identity Providers, the application web page displays all the available IdPs. The service provider recognises its Identity Provider with the help of the user.
User redirected to IdP login page	The Service Provider redirects the user agent to the Identity Provider login page. Now, the user is ready for IdP login web interface.
User enters the login credentials	The user now is able to provide the IdP credentials for authentication. The IdP validates the credentials. If successful, IdP generates the SAML2.0 'AuthenticationRequest'
User redirected to application & Generate access tokens	After the successful IdP authentication, the user is redirected to the service provider application. The IdP creates a set of access tokens and stores the same in identity system.
Access Token Request	If the access token is expired, then SP raises the request for new set of access tokens.
Access Token Response	The Identity Provider response to the Service Provider's access token request. The identity Provider shares a new set of access tokens and stores the same in identity and service provider.
Periodic Token Refreshing	The access tokens are periodically generated and synchronised in between the Identity Provider and the Service Provider based on the token expiry time or demand.
Cloud Resource Request	Service Provider places the cloud resource request along with an access token. Once an access token is utilized, it needs to be removed from the SP access token storage
Token Validation Request	The web server places the access token validation request to the Identity Provider. The Identity Provider validates the access tokens against the IdP access token store.
Token Verification Response	If IdP finds a matching token, then it responds back to Web Server, and destroys that access token from the IdP token store.
If Required, New Cloud Resource Request	When the token validation fails due to any reason, the web server shares that information to the Service Provider. If required, the service provider can place a new request along with a new access token and the steps 9, 10 and 11 are repeated.
Cloud Resource Response	After the successful access token validation and verification, the Cloud Web Server shares the requested data to the service provider.
Data Available to the User	The service provider displays the data through the user agent. The authenticated user gets the requested information.


Indu's IdP Login Page
Not secure | 52.172.46.26:8080/idp/Authn/UserPassword
HINDUSTAN
INSTITUTE OF TECHNOLOGY & SCIENCE
Log in to sp.rohith.org
› Forgot your password?
› Need Help?
Login
Service Provider - sp.rohith.org
Identity Provider Logon Screen [idp.indurejeesh.org].
Elements Console Sources Network Performance Memory Application Security
secure
SSO?SAMLRe...
AuthnEngine
UserPassword
login.css
logo.png
engines.json
favicon.ico
8 requests | 44.1 ...
Headers Preview Response Cookies Timing

```
Request Method: GET
Status Code: 302
Remote Address: 52.172.46.26:8080
Referrer Policy: no-referrer-when-downgrade
▼ Response Headers     view source
Cache-Control: no-cache, no-store, must-revalidate, max-age=0
Content-Length: 0
Date: Sun, 13 Aug 2017 17:22:57 GMT
Expires: 0
Location: http://52.172.46.26:8080/idp/AuthnEngine
Pragma: no-cache
Set-Cookie: JSESSIONID=7756B5013D98D181838BFF6BA4FD8403; Path=/idp; HttpOnly
Set-Cookie: _idp_authn_lc_key=f44cb879c448f79ad23d04fa98eff1878c43511768bf359856e0c3c8495
26398; Path=/idp
▼ Request Headers     view source
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q
=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.8
Connection: keep-alive
Host: 52.172.46.26:8080
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gec
ko) Chrome/60.0.3112.90 Safari/537.36
▼ Query String Parameters     view source     view URL encoded
SAMLRequest: fZJLb4MwEIT/CvI9GEx41ApINDk0UtqgQHvopTLgBEtgU6/p49+XhDRKVSlHyzsz0592Aaxre5o
OppE7/j5wMNZX10qgp48YDVpSxUAAlazjQE1F8/RxQ4nt0F4royrVIisF4NoIJZdKwtBxnXP9ISr+vNvEqDGmpxh
Db2vVCNPYSh9w3oiyVC0fnwAKHz0JzrZ5gazVuISQ7Gh3EfvEdkNizwObBDRyIgeLusfjAnvR8rN6x2uheWVwnm+
RtV7F6M2fB07tsYoEnlvVPuPeXRVEoePty9Injj+0AQx8LcEwaWJEHDecOdHM9Qo3pIRQ/+4VWdm5572QtZCH21D
KaQjoQ1Fks6nRC9dwajMOoGRxREtPwfoK9m1b9ksYJUck8A8oXIAu8FXAlNbTp9FxvcpUK6pvK21b9bnUnBkeIxf
hZJL8vYPkBw==
RelayState: ss:mem:e7bb315bf914289e90aba58d18f255a565001dfda5fdc91ba2852c4d0516a52b
```

Fig. 7. A sample of the encrypted SAML request for IdP authentication.



Service Provider - Web Application

Welcome, INDU I

ID:	PH23
First Name:	INDU
Last Name:	INDIRA
E - Mail:	indu.i2043@gmail.com

Tokens Expiries at: 2017-08-13 23:29:00.000306

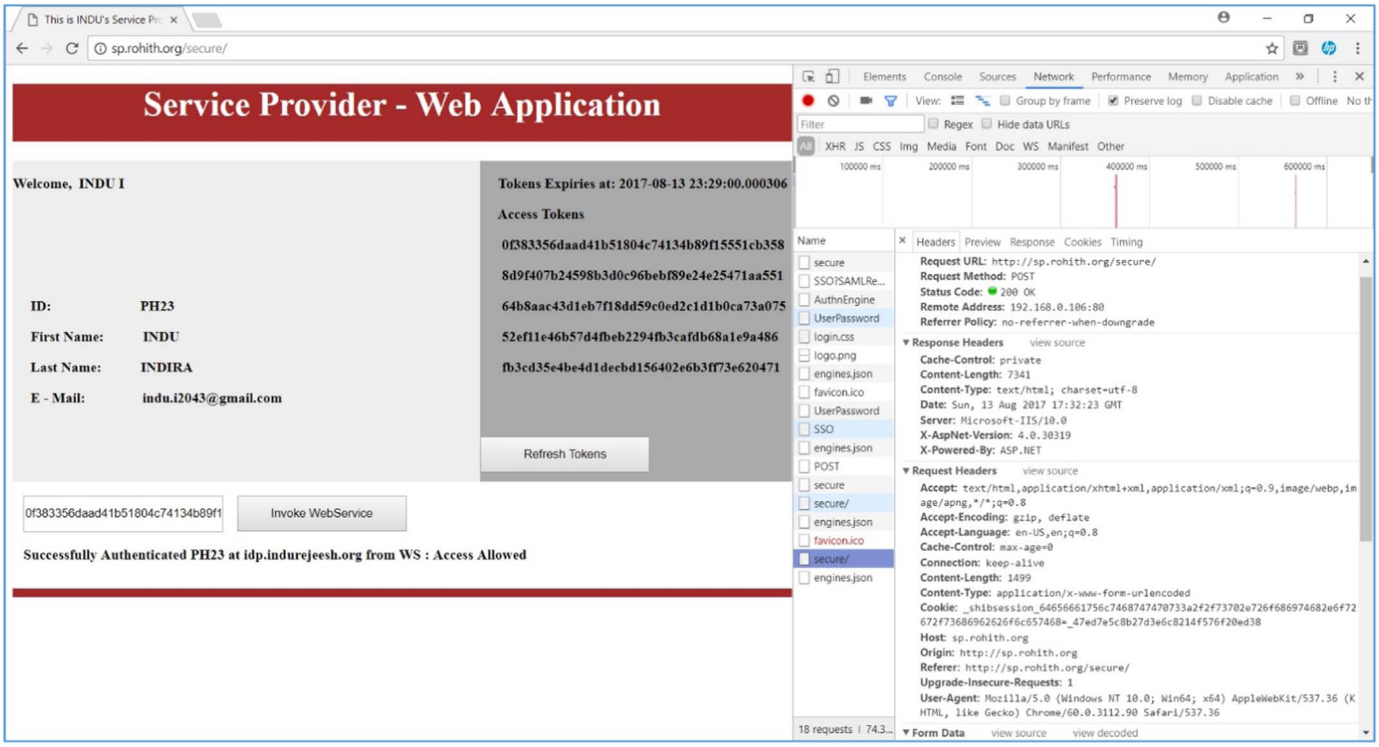
Access Tokens

0f383356daad41b51804c74134b89f15551cb358
8d9f407b24598b3d0c96bebf89e24e25471aa551
64b8aac43d1eb7f18dd59c0ed2c1d1b0ca73a075
52ef11e46b57d4fbeb2294fb3cafdb68a1e9a486
fb3cd35e4be4d1decbd156402e6b3ff73e620471

```
Location: http://sp.rohith.org/secure
Server: Microsoft-IIS/10.0
Set-Cookie: _shibsession_64656661756c7468747470733a2f2f73702e726f686974682e
6f72672f73686962626f6c657468=_47ed7e5c8b27d3e6c8214f576f20ed38; path=/; Ht
tpOnly
X-Powered-By: ASP.NET
▼ Request Headers     view source
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,im
age/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.8
Cache-Control: max-age=0
Connection: keep-alive
Content-Length: 14358
Content-Type: application/x-www-form-urlencoded
Host: sp.rohith.org
Origin: http://52.172.46.26:8080
Referer: http://52.172.46.26:8080/idp/profile/SAML2/Redirect/SSO
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (K
HTML, like Gecko) Chrome/60.0.3112.90 Safari/537.36
▼ Form Data     view source     view URL encoded
RelayState: ss:mem:e7bb315bf914289e90aba58d18f255a565001dfda5fdc91ba2852c4d0
516a52b
SAMLResponse: PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz4KPHNhbWwy
cDpSZXNwb25zZSBEZXN0aW5hdGlvbj0iaHR0cDovL3NwLnJvaGl0aC5vcmcvU2hpYmJvbGV0aC
5zc28vU0FNTDIvUE9TVCIgSUQ9I182Nzk3ZDQyOTM5ODJjZWU4YThhYjJiMzU0OWM5ZGIxOCIg
SW5SZXNwb25zZVRvPSJfNTQ2MGQzYWMyNjMxY2Q1YWUzOWM2ODcwM2ZiYjUyMDUiIElzc3VlSW
5zdGFudD0iMjAxNy0wOC0xM1QxNzoyODo1OS41MDlaIiBWZXJzaW9uPSIyLjAiIHhtbG5zOnNh
bWwycD0idXJuOm9hc2lzOm5hbWVzOnRjOlNBTUw6Mi4wOnByb3RvY29sIj48c2FtbDI6SXNzdW
VyIEZvcm1hdD0idXJuOm9hc2lzOm5hbWVzOnRjOlNBTUw6Mi4wOm5hbWVpZC1mb3JtYXQ6ZW50
aXR5IiB4bWxuczpzYW1sMj0idXJuOm9hc2lzOm5hbWVzOnRjOlNBTUw6Mi4wOmFzc2VydGlvbi
```

Fig. 8. A sample of encrypted SAML response from IdP after successful authentication.

3.6.2.3. Cloud metadata file for Identity Providers. Cloud metadata file contains the information about the Web Service Provider and needs to be stored in the Identity Provider. This newly created metadata file must follow the specifications similar to the SP and IdP metadata files. We propose a new descriptor type 'CloudSSODescriptor' for this purpose. The required service information for establishing the IdP –



sp.rohith.org/secure/

Service Provider - Web Application

Welcome, INDU I

Tokens Expiries at: 2017-08-13 23:29:00.000306

Access Tokens

0f383356daad41b51804c74134b89f15551cb358
8d9f407b24598b3d0c96bebf89e24e25471aa551
64b8aac43d1eb7f18dd59c0ed2c1d1b0ca73a075
52ef11e46b57d4fbeb2294fb3cafdb68a1e9a486
fb3cd35e4be4d1decbd156402e6b3ff73e620471

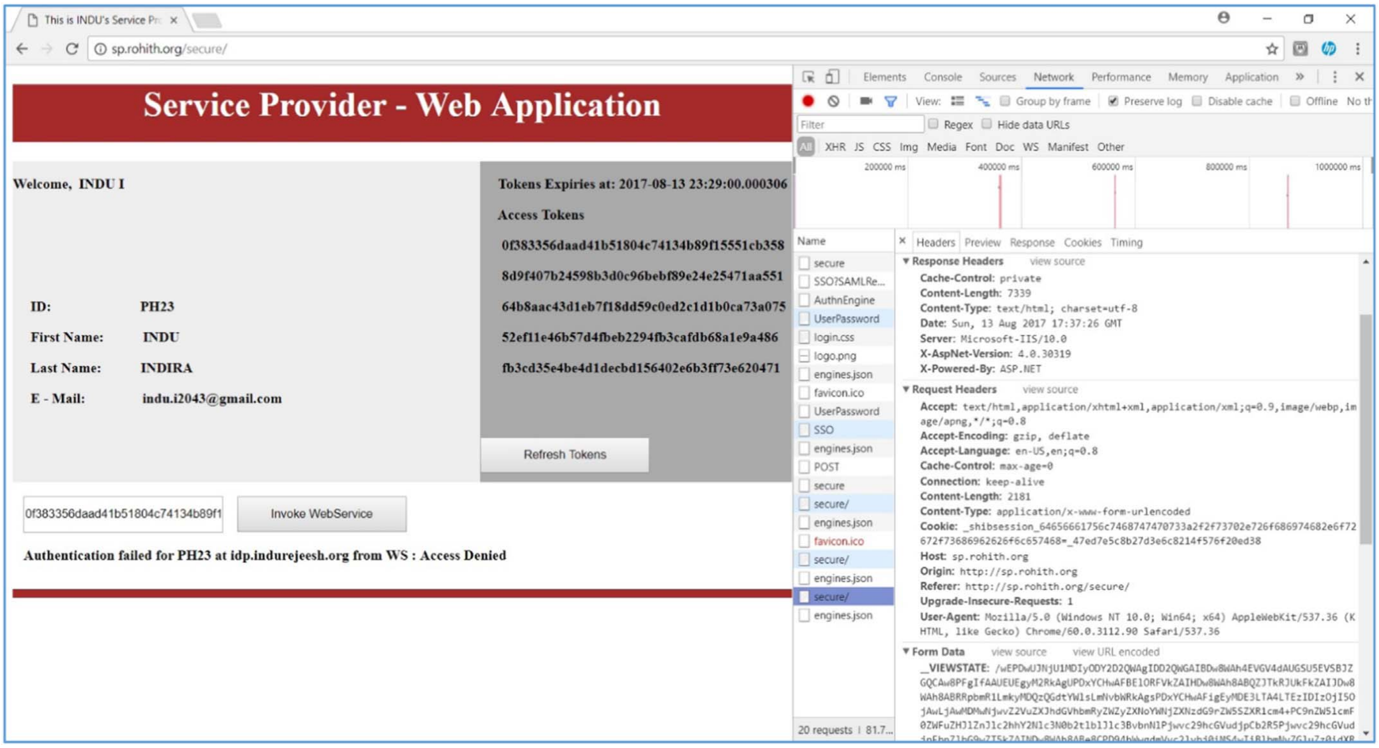
ID:	PH23
First Name:	INDU
Last Name:	INDIRA
E - Mail:	indu.i2043@gmail.com

0f383356daad41b51804c74134b89f1

Successfully Authenticated PH23 at idp.indurejeesh.org from WS : Access Allowed

```
Request URL: http://sp.rohith.org/secure/
Request Method: POST
Status Code: 200 OK
Remote Address: 192.168.0.106:80
Referrer Policy: no-referrer-when-downgrade
▼ Response Headers    view source
  Cache-Control: private
  Content-Length: 7341
  Content-Type: text/html; charset=utf-8
  Date: Sun, 13 Aug 2017 17:32:23 GMT
  Server: Microsoft-IIS/10.0
  X-AspNet-Version: 4.0.30319
  X-Powered-By: ASP.NET
▼ Request Headers    view source
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,im
  age/apng,*/*;q=0.8
  Accept-Encoding: gzip, deflate
  Accept-Language: en-US,en;q=0.8
  Cache-Control: max-age=0
  Connection: keep-alive
  Content-Length: 1499
  Content-Type: application/x-www-form-urlencoded
  Cookie: _shibsession_64656661756c7468747470733a2f2f73702e726f686974682e6f72
  672f73686962626f6c657468=_47ed7e5c8b27d3e6c8214f576f20ed38
  Host: sp.rohith.org
  Origin: http://sp.rohith.org
  Referer: http://sp.rohith.org/secure/
  Upgrade-Insecure-Requests: 1
  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (K
  HTML, like Gecko) Chrome/60.0.3112.90 Safari/537.36
▼ Form Data    view source    view decoded
```

Fig. 9. A sample of token based cloud webservice request after successful authentication.



sp.rohith.org/secure/

Service Provider - Web Application

Welcome, INDU I

Tokens Expiries at: 2017-08-13 23:29:00.000306

Access Tokens

0f383356daad41b51804c74134b89f15551cb358
8d9f407b24598b3d0c96bebf89e24e25471aa551
64b8aac43d1eb7f18dd59c0ed2c1d1b0ca73a075
52ef11e46b57d4fbeb2294fb3cafdb68a1e9a486
fb3cd35e4be4d1decbd156402e6b3ff73e620471

ID:	PH23
First Name:	INDU
Last Name:	INDIRA
E - Mail:	indu.i2043@gmail.com

0f383356daad41b51804c74134b89f1

Authentication failed for PH23 at idp.indurejeesh.org from WS : Access Denied

```
▼ Response Headers    view source
  Cache-Control: private
  Content-Length: 7339
  Content-Type: text/html; charset=utf-8
  Date: Sun, 13 Aug 2017 17:37:26 GMT
  Server: Microsoft-IIS/10.0
  X-AspNet-Version: 4.0.30319
  X-Powered-By: ASP.NET
▼ Request Headers    view source
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,im
  age/apng,*/*;q=0.8
  Accept-Encoding: gzip, deflate
  Accept-Language: en-US,en;q=0.8
  Cache-Control: max-age=0
  Connection: keep-alive
  Content-Length: 2181
  Content-Type: application/x-www-form-urlencoded
  Cookie: _shibsession_64656661756c7468747470733a2f2f73702e726f686974682e6f72
  672f73686962626f6c657468=_47ed7e5c8b27d3e6c8214f576f20ed38
  Host: sp.rohith.org
  Origin: http://sp.rohith.org
  Referer: http://sp.rohith.org/secure/
  Upgrade-Insecure-Requests: 1
  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (K
  HTML, like Gecko) Chrome/60.0.3112.90 Safari/537.36
▼ Form Data    view source    view URL encoded
  __VIEWSTATE: /wEPDwUJNjU1MDIyODY2D2QWAgIDD2QWGAIBDw8WAh4EVGV4dAUGSU5EVSBJZ
  GQCAw8PFgIfAAUEUEgyM2RkAgUPDxYCHwAFBE1ORFVkZAIHDw8WAh8ABQZJTkRJUkFkZAIJDw8
  WAh8ABRRpbmR1LmkyMDQzQGdtYWlsLmNvbWRkAgsPDxYCHwAFigEyMDE3LTA4LTEzIDIzOjI5
  jAwLjAwMDMwNjwvZ2VuZXJhdGVhbmRyZWZyZXNoYWNjZXNzdG9rZW5SZXR1cm4+PC9nZW5lcmF
  0ZWFuZHJlZnJlc2hhY2Nlc3N0b2tlblJlc3BvbnN1Pjwvc29hcGVudjpCb2R5Pjwvc29hcGVud
```

Fig. 10. A sample of token based cloud webservice request for failed scenario (token already used).

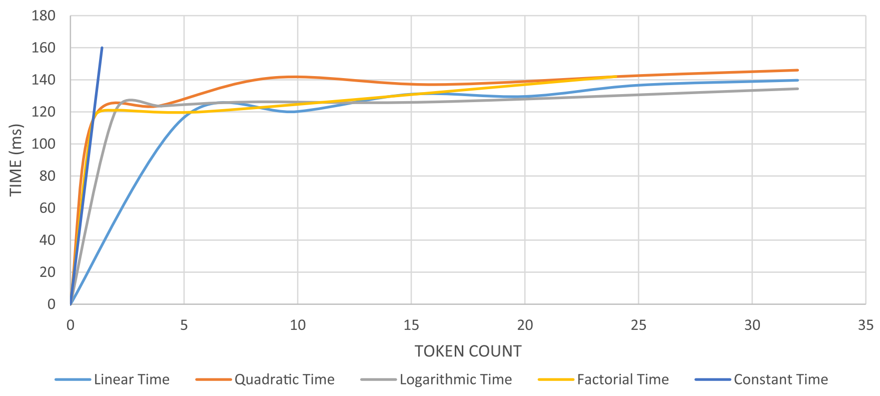


Fig. 11. Time complexity analysis of token generation algorithm.

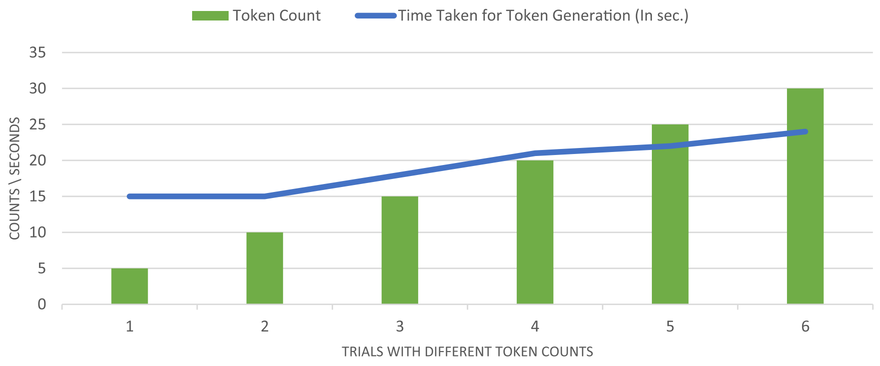


Fig. 12. Comparison of time taken for different trials with variable token counts.

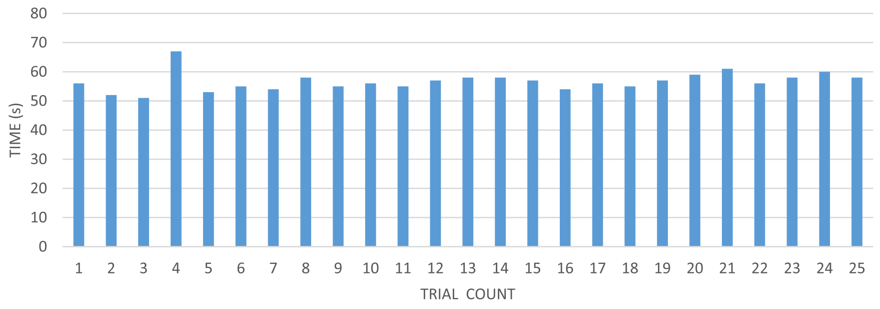


Fig. 13. Average time taken for web service authentication process.

Cloud server communication is kept inside the 'CloudSSODescriptor' tag. This service is used by the Identity Provider to respond back to the Cloud Web Server's token validation requests. The schema for storing this information as shown below:

```
< md:TokenVerificationService
  index="0"
  isDefault="true"
  Location="__End point url__"
  Binding="url:oasis:names:tc:SAML:2.0:bindings:SOAP" / >
```

3.6.2.4. Cloud metadata file for Service Providers. This metadata file contains the information about the Web Service Provider, and needs to store this file in the Service Provider. This newly created metadata file must follow the specifications similar to the SP and IdP metadata files. A new descriptor type 'CloudSSODescriptor' is proposed. The required service information for establishing the SP – Cloud server communication is kept inside the 'CloudSSODescriptor' tag. The schema for storing this information is shown below:

```
< md:CloudRequestService
  index="0"
  isDefault="true"
  Location="__End point url__"
  Binding="url:oasis:names:tc:SAML:2.0:bindings:SOAP" / >
```

The service provider should use this service for placing all the requests to the cloud web service. The number of parameters passing through this service varies based on the requirements and request types. But all the requests must pass mandatory parameters like User Identity, Access Token, and Identity Provider Entity ID.

3.6.2.5. Schema diagram of the proposed metadata files. A metadata file contains all information required for establishing a secure communication between the source and the destination. In order to

Table 3
Security comparison of the authentication models.

Characteristics / Authentication Model	Log-On Credentials	OpenID	OAuth 2.0	SAML 2.0	Proposed Model
SP & IdP Initiated Login	**NA**	**NA**	**Yes** Supports SP & IdP initiated login	**Yes** Supports SP & IdP initiated login	**Yes** Supports SP & IdP initiated login
Attributes	**NA**	Open ID connect scopes	**Yes**	**Yes**	**Yes**
Discovery Service	**No**	**Yes** Single discovery service	**Yes**	**No** Predefined metadata	**No** Predefined metadata
Privacy Preserving	**Yes**	**No**	**No**	**Yes**	**Yes**
Authentication Mechanism	**Biometrics or Password** sharing through network	**Credential Based** Password sharing through network	**OAuth Claim based** No password sharing	**SAMLToken based** No password sharing	**Token based** A combination of encrypted SAML & access token, No password sharing
Digital Signature	**NA**	**NA**	**No**	**Yes** Encrypted & digitally signed SAML communications	**Yes** Encrypted & digitally signed SAML communication
Metadata	**NA**	**NA**	**No** Client registration process is in place	**Yes** Prepopulated metadata file for establishing communication between IdP & SP	**Yes** Prepopulated metadata file for establishing communication between IdP, SP and Webserver
Single Sign On Support	**No** Each application needs separate Sign On	**Yes** Uses only consumer key and consumer secret	**Yes** Supports Web and Native application Single Sign On	**Yes** Supports Web Single Sign On	**Yes** Supports Single Sign On for web applications as well as the web services.
Verification with Identity system	**NA**	**Yes**	**Yes**	**Yes** Verification through SP initiated log in	**Yes** Verification through SP initiated log in and Web Service requests

ensure the security and confidence of the data, different encoding and encryption methods are used. The digital certificates and related information are kept in a structured way in the metadata file. The source or destination system uses these information for encrypting, decrypting or signing the data. Further, the binding methods for each specified service is declared in the metadata file. Any one of the binding methods like, SOAP (Simple Object Access Protocol), HTTP-POST, Artifact, or HTTP-Redirect are used based on the user's needs and requirements. A combined and simplified class schema diagram for the proposed metadata files are shown in Fig. 5. In the schema, the necessary tags are included which are required to establish the proposed architecture in a secured manner.

3.7. *Proposed methods for authentication*

3.7.1. *Authentication request*

This request is raised by the service provider to the identity provider when a user is trying to access service provider resources. This is a part of the single sign on protocol which is defined for SAML communication. The service provider contracts a SAML assertion request with all the required parameters for authentication and submit the same to the Identity Provider. IdP validates the request and redirect the user to the IdP login page. Once the login is successful, IdP responds back to SP through the Authentication Response.

3.7.2. *Authentication response*

Authentication Response is working in association with the Authentication Request. The Authentication Response is generated at the Identity Provider and processed at the Service Provider. In the proposed model, IdP needs to invoke the 'GenerateToken' functionality along with the Authentication Response.

3.7.3. *Token refresh request*

This request is raised by the service provider to the Identity Provider with the help of 'TokenRequestService'. The mandatory parameters required for this request are the 'User Identifier' and 'Service Provider Entity ID'. The Service Provider constructs a SAML2.0 assertion for token refresh request and sent it to the corresponding IdP service. The 'TokenRequestService' processes this request at IdP server and invoke the 'GenerateToken' functionality for access token generation.

3.7.4. *Token refresh response*

The token refresh response is initiated from the Identity Provider. Whenever the 'GenerateToken' functionality is invoked, the tokens are created and stored in the IdP. Once the tokens are stored with the help of 'SP entity Id', IdP navigates the 'TokenAcquistionService' endpoint URL from the corresponding metadata file. A SAML2.0 assertion is then constructed along with the generated tokens which is sent to the identified 'TokenAcquistionService' endpoint URL.

3.7.5. *Cloud resource request*

The 'CloudResourceRequest' is initiated from the Service Provider. This request is also in the SAML 2.0 assertion format and uses the 'CloudRequestService' for communicating the details to the cloud web server. Service provider identifies the end point URL from the predefined cloud metadata file which is already stored in the service provider. The mandatory parameters that needs to be shared to Cloud web server are 'SP Entity ID', 'User Identifier', 'IdP Entity ID', 'Access Token' and 'Request Type'. The service provider shares the request specific parameters (if any) along with mandatory parameters.

3.7.6. *Cloud resource response*

The 'CloudResourceResponce' SAML assertion is generated at the Cloud web server based on the 'CloudResourceRequest' which is initiated from the Service Provider. The 'CloudResourceResponce' is posted to service providers 'Cloud Response Service' end point URL.

3.7.7. *Token verification request*

The 'TokenVerificationRequest' assertion is generated from Cloud web server to the Identity Provider server based on the 'CloudResourceRequest' from service provider. The cloud web server shares the details of 'User identifier', 'SP Entity Id' and the 'Access Token' as the parameters of 'TokenVerificationRequest'.

3.7.8. Token verification response

The 'TokenVerificationResponse' assertion is generated from Identity Provider, and contains the authentication decision. The Cloud server processes the cloud resource request and provides the response to the Service Provider based on the response from the identity provider.

3.8. Use case analysis

The steps involved in the proposed token based authentication model with encrypted requests and responses are shown as a use case diagram in Fig. 6. All communications happening in the sequences are secured by using encryption mechanism. Table 2 explains about the sequence of steps used in the proposed authentication model. The commonly used encryption mechanisms for SAML assertions are SHA-256, RSA, Blowfish and AES.

4. Results and discussions

4.1. Implementation details

The proposed model is implemented with the help of Shibboleths' Java open source code for IdP system. The cloud environment is implemented using Microsoft Azure cloud platform. The identity provider and the webservice servers are configured in separate Azure Windows 2008 Server virtual machines which is in different networks with static public IPs. Tomcat 8.5 is used as HTTP webserver for hosting identity provider and webservices. The verification and validation mechanism in the cloud web service system is developed with the help of Java technology in the cloud environment. The service provider is developed using .NET technology in the Windows 10 machine at lab environment which is exposed to internet. Internet Information Services (IIS) webserver used as the hosting server for service provider. All the communications between these systems are based on encrypted SAML to ensure the security of the systems and protection of data as shown in Figs. 7–10.

4.2. Performance evaluation

4.2.1. Time complexity analysis of the token generation algorithm

The first stage of the proposed model is to generate the access tokens at the Identity provider after successful authentication. The performance or complexity of an algorithm is described by Big O notation. The time complexity analysis through total execution time of the token generation algorithm is shown in Fig. 11. Constant time analysis is calculated through the execution of the algorithm with fixed time, independently from the size of the input. The function describing the run-time behaviour is O(1). Linear time describes the number of operations that vary in proportion to the first degree of the run time of the proposed algorithm. The linear time analysis is expressed as O(n). The expression for logarithmic time describes the run-time behaviour with respect to logarithmic function of the number of operations in the token generation algorithm. O-notation for logarithmic function is expressed as O(log(n)). Quadratic time explains the relationship between the running time and square of the number of size of input for the token generation and is expressed as $O(n^2)$. Factorial time explains the relationship between computation time and the factorial value of the number of tokens and is expressed as O(n!). The time complexity analysis of the proposed token generation algorithm shows that the generation of a set of tokens at an instance is beneficial and less time consuming than generating single token during every request. The analysis also helps to identify the optimal number of tokens that provides the maximum performance of the authentication system. The time complexity analysis helps in determining the expiry time for the access tokens based on the number of tokens and the frequency of web service access requests.

4.2.2. Time taken for token generation and storage

The second stage of the proposed model is to share the generated token to the service provider along with the SAML authentication. The generated tokens are stored in the identity provider and a copy of the access tokens are communicated to service provider. The approximate time taken to generate and store access tokens in the identity provider as well as the service provider is calculated through multiple trials performed with different counts of access tokens as shown in Fig. 12. The system is capable of generating the required number of access tokens depending upon the needs of the environment.

4.2.3. Time Taken for web service authentication

The final stage of the proposed model is to access the cloud web services with the help of the access tokens. The service provider requests for the web service along with the required parameters like identity provider entity id, web service related parameters and access token. The cloud web servers pass the access token to the identity provider and based on the verification results, the system authenticates the user to access the web service. The average time taken for the successful web service authentication is calculated with different trails as shown in Fig. 13.

4.3. Comparison of the authentication models

The authentication models presently used for the purpose of web applications are log-on credentials, OpenID, OAuth 2.0 and SAML 2.0. The existing models use various methods for providing privacy and security with the defined parameters. Table 3 compares the characteristics and parameters of the existing authentication models with the proposed adapted SAML based authentication model. The main scope of the proposed model is to fill the security gaps that exist in each of the aforementioned authentication models. The proposed token based model with adapted SAML considers the advantages of token based models and SAML assertions in the encrypted form for communications between the identity provider, service provider and cloud web services. The implementation of the proposed model is accomplished with the existing SAML authentication model by the addition of single time usage access tokens.

4.4. Security analysis of the adapted SAML protocol

The proposed adaptations in the encrypted SAML communication for IdP authentication and webservice access do not require any kind of credential information exchange between the participating systems. In the existing SAML federated systems, identity provider acts as the trusted source for service provider authentication. After the successful authentication of the service provider, all the web service requests initiated from service provider is considered as trusted requests. The proposed model considers identity provider as the only trusted party for service provider authentication and verification of the web service requests. The security of the authentication system is further improved through digitally signed SAML requests. In addition, the proposed model introduces single use access token with expiry time for verifying the authenticity of the requested user for cloud webservice access. These adaptations in the existing SAML protocol helps to address the cloud based authentication challenges.

(i) **Man-in-the-Middle (MITM) attacks:** In cloud systems, any attacker can intercept the communication between the systems and manipulates the data without their knowledge. In the proposed model, the SAML request for web service is encrypted and additionally a one-time access token is used for verifying the identity of the user against the identity store. This prevents the attackers from involving in the malicious activities of MITM based attacks.

(ii) **Insider attacks:** These attacks are launched by someone who is inside the security perimeter and compromising the security. In the adapted SAML protocol, both the SAML request and the access token have an expiry period. After the expiry time, the access tokens and the SAML request become invalid. Moreover, the identity provider has the capability to clean up the access tokens from its store at any point of time whenever an attack or system breach is sensed. These mechanisms help to reduce the impact of insider attacks.

(iii) **Password/Key compromise:** In the traditional authentication model, an attacker gets access to the web services credentials and can misuse the cloud services with the acquired access rights. As the adapted SAML protocol is not storing or using any kind of credentials like password or secure shell (SSH) keys for accessing the cloud web services, the possibility of password or key compromise is resolved.

(iv) **Replay attacks:** In cloud services, the attacker creates an authentication request from the authentication message which was previously exchanged between an authorized user and the target system. In the proposed model, all the authentication messages are unique due to the single use access tokens. Moreover, the authentication is achieved through the encrypted SAML request and single use access tokens with expiry time which helps to prevent the replay attacks.

4.5. Integration with the existing SAML federated systems

The proposed mechanism is integrated to the federated SAML system and the cloud webservice by modifying the existing IdP & SP SAML components and integrating the cloud webservice SAML component. In addition, modifications are required in the IdP & SP metadata files to accommodate the proposed changes and inclusion of the webserver metadata file. The required modifications in IdP SAML component are the accommodation of token generation, refresh and verification services. The SAML component at service provider is required to modify in such a way to accommodate the token refresh and webservice SAML requests. The webserver SAML component handles the token verification against the identity system and the webservice request from the service provider. The proposed system can be implemented in any cloud environment such as public cloud, private cloud and hybrid cloud based on the authentication requirements and the level of the security/criticality of the data to be protected. Since the model involves identity verification against the identity provider, it requires a session persistence load balancer in the case of multiple hosting servers used for cloud services.

5. Conclusions

In this paper, cloud based data and web services are protected by the fine-grained adapted SAML authentication model with the help of access tokens. Once the access token is used to place a web service request, the service provider and the Identity Provider removes/deletes that particular access token from the access token store. This ensures that the access token is used as a single-use semantic such that once it has been utilized, it is no longer be used by any other party. The extensive use of encrypted SAML for all the communications between IdP, SP and Cloud Webservice ensures secure data transfer between the systems. Security analysis in comparison with the existing authentication models demonstrates that the proposed model is secure in terms of insider and outsider cyber-attacks. A prototype of the proposed model is implemented as a proof of concept with test-bed experiments conducted to analyse the time complexity and stability of the system.

References

Armando, A., et al., 2013. **An authentication flaw in browser-based Single Sign-On protocols: Impact and remediations. Comput** Secur. 33, 41–58.

Al-dubai, Y.F., Khamitkar, S.D., 2014. Kerberos: secure single sign-on authentication. Glob. J. Comput. Sci. Technol. B Cloud Distrib. 14, 10–16.

Assis, M.R.M., Bittencourt, L.F., 2016. A survey on cloud federation architectures: identifying functional and non-functional properties. J. Netw. Comput. Appl. 72, 51–71. http://dx.doi.org/10.1016/j.jnca.2016.06.014.

Atwood, C.A., Goebbert, R.C., Calahan, J.A., Hromadka, T.V., Proue, T.M., Monceaux, W., Hirata, J., 2016. Secure web-based access for productive supercomputing. Comput. Sci. Eng. 18, 63–72. http://dx.doi.org/10.1109/MCSE.2015.134.

Bhonsle, M.V., Poolsappasit, N., Madria, S.K., 2013. ETIS - Efficient trust and identity management system for federated service providers. In: IEEE Proceedings of the 27th International Conference on Advanced Information Networking and Applications, AINA, Barcelona, Spain, pp. 219–226. ⟨http://dx.doi.org/10.1109/AINA.2013.13⟩.

Butun, I., Erol-kantarci, M., Kantarci, B., Song, H., 2016. Cloud-centric multi-level authentication as a service for secure public safety device networks. IEEE Commun. Mag., 47–53. http://dx.doi.org/10.1109/MCOM.2016.7452265.

Cantor, S., et al., 2005a. Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0. [online] Available: ⟨http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf⟩.

Cantor, S., et al., 2005b. Metadata for the OASIS Security Assertion Markup Language (SAML) V2. 0. [online] Available: ⟨https://docs.oasis-open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf⟩.

Chang, V., Ramachandran, M., 2016. Towards achieving data security with the cloud computing adoption framework. IEEE Trans. Serv. Comput. 9, 138–151. http://dx.doi.org/10.1109/TSC.2015.2491281.

Cirani, S., Picone, M., Gonizzi, P., Veltri, L., Ferrari, G., 2015. IoT-OAS: an OAuth-based authorization service architecture for secure services in IoT scenarios. IEEE Sens. J. 15, 1224–1234. http://dx.doi.org/10.1109/JSEN.2014.2361406.

David, B.M., Tonicelli, R., Nascimento, A., Amaral, D., Peotta, L., 2011. Secure single sign-on and web authentication. IACR Cryptol. EPrint Arch. Report. 246, 1–14.

Diogo A. B. Fernandes, Soares, Liliana F.B., Gomes, Joao V., Freire, Mario M., P.R.M.I, 2014. Security issues in cloud environments: a survey. Int. J. Inf. Secur. 13, 113–170 . http://dx.doi.org/10.1007/s10207-013-0208-7.

Hernandez-ramos, J.L., Pawlowski, M.P., Jara, A.J., Skarmeta, A.F., Ladid, L., 2015. Toward a lightweight authentication and authorization framework for smart objects. IEEE J. Sel. Areas Commun. 33, 690–702. http://dx.doi.org/10.1109/JSAC.2015.2393436.

Indu, I., Rubesh Anand, P.M., 2015. Identity and access management for cloud web services. IEEE Recent Adv. Intell. Comput. Syst. RAICS, 2015. http://dx.doi.org/10.1109/RAICS.2015.7488450.

Khan, M.A., 2016. A survey of security issues for cloud computing. J. Netw. Comput. Appl. 71, 11–29. http://dx.doi.org/10.1016/j.jnca.2016.05.010.

Kritikos, K., Kirkham, T., Kryza, B., Massonet, P., 2017. Towards a security-enhanced PaaS platform for multi-cloud applications. Futur. Gener. Comput. Syst. 67, 206–226. http://dx.doi.org/10.1016/j.future.2016.10.008.

Lee, C.A., 2016. Cloud federation management and beyond: requirements, relevant standards, and gaps. IEEE Cloud Comput. 3, 42–49. http://dx.doi.org/10.1109/MCC.2016.15.

Lewis, K.D., Lewis, J.E., 2009. Web single sign-on authentication using SAML. J. Comput. Sci. 2, 41–48.

Liu, H., Ning, H., Xiong, Q., Yang, L.T., 2015. Shared authority based privacy-preserving authentication protocol in cloud computing. IEEE Trans. Parallel Distrib. Syst. 26, 241–251. http://dx.doi.org/10.1109/TPDS.2014.2308218.

Liu, W., Tan, Y., Zhang, E., 2009. Service token for identity access management. In: Proceedings of the 2009 IEEE Asia-Pacific Services Computing Conference, APSCC 2009. IEEE, pp. 34–39. ⟨http://dx.doi.org/10.1109/APSCC.2009.5394143⟩.

Nicanfar, H., Jokar, P., Beznosov, K., Leung, V.C.M., 2014. Efficient authentication and key management mechanisms for smart grid communications. IEEE Syst. J.. http://dx.doi.org/10.1109/JSYST.2013.2260942.

Nordbotten, N.A., 2009. XML and web services security standards. IEEE Commun. Surv. Tutor. 11, 4–21. http://dx.doi.org/10.1109/SURV.2009.090302.

Pérez Méndez, A., Marín López, R., López Millán, G., 2016. Providing efficient SSO to cloud service access in AAA-based identity federations. Futur. Gener. Comput. Syst. 58, 13–28. http://dx.doi.org/10.1016/j.future.2015.12.002.

Rasheed, H., 2014. Data and infrastructure security auditing in cloud computing environments. Int. J. Inf. Manag. 34, 364–368. http://dx.doi.org/10.1016/j.ijinfomgt.2013.11.002.

Sen, J., 2013. Security and privacy issues in cloud computing. Archit. Protoc. Secur. Inf. Technol., 42. http://dx.doi.org/10.1109/HICSS.2011.103.

Sharma, A., Sharma, S., Dave, M., 2016. Identity and access management- a comprehensive study. In: Proceedings of the 2015 International Conference on Green Computing and Internet of Things, ICGCIoT 2015. Haryana, India, pp. 1481–1485. ⟨http://dx.doi.org/10.1109/ICGCIoT.2015.7380701⟩.

Singh, A., Chatterjee, K., 2015. Identity management in cloud computing through claim-based solution. In: 2015 Proceedings of the Fifth International Conference on Advanced Computing & Communication Technologies. pp. 524–529. ⟨http://dx.doi.org/10.1109/ACCT.2015.89⟩.

Singh, S., Jeong, Y.-S., Park, J.H., 2016. A survey on cloud computing security: issues, threats, and solutions. J. Netw. Comput. Appl. 75, 200–222. http://dx.doi.org/10.1016/j.jnca.2016.09.002.

Sood, S.K., 2012. A combined approach to ensure data security in cloud computing. J. Netw. Comput. Appl. 35, 1831–1838. http://dx.doi.org/10.1016/j.jnca.2012.07.007.

Subashini, S., Kavitha, V., 2011. A survey on security issues in service delivery models of cloud computing. J. Netw. Comput. Appl. 34, 1–11. http://dx.doi.org/10.1016/j.jnca.2010.07.006.

Tari, Z., Yi, X., Premarathne, U.S., Bertok, P., Khalil, I., 2015. Security and privacy in cloud computing: vision, trends, and challenges. IEEE Cloud Comput. 2, 30–38. http://dx.doi.org/10.1109/MCC.2015.45.

Tysowski, P., 2016. OAuth standard for user authorization of cloud services. Encycl. Cloud Comput., 406–416. http://dx.doi.org/10.1002/9781118821930.ch34.

Wang, H., 2015. Identity-based distributed provable data possession in multicloud storage. IEEE Trans. Serv. Comput. 8, 328–340. http://dx.doi.org/10.1109/TSC.2014.1.

Waqar, A., Raza, A., Abbas, H., Khurram Khan, M., 2013. A framework for preservation of cloud users data privacy using dynamic reconstruction of metadata. J. Netw. Comput. Appl. 36, 235–248. http://dx.doi.org/10.1016/j.jnca.2012.09.001.

Wayne Jansen and Timothy Grance, 2011. Guidelines on Security and Privacy in Public Cloud Computing, NIST Special Publication 800144, [Online] ⟨http://csrc.nist.gov/publications/nistpubs/800144/SP800-144.pdf⟩.

Xiao, Z., Xiao, Y., 2013. Security and privacy in cloud computing. IEEE Commun. Surv. Tutor. 15, 843–859. http://dx.doi.org/10.1109/SURV.2012.060912.00182.

I. Indu is a Research Scholar in Electronics and Communication Engineering at Hindustan University, India. She received her B.Tech. degree in 2009 and M.Tech. degree in 2012 from Cochin University of Science and Technology, India. Her research interests include communication networks, cloud computing, cryptography and network security.

Dr. Rubesh Anand is a Professor in the Department of Electronics and Communication Engineering at Hindustan University, India. He received his B.E. (Electronics and Communication Engineering) degree from Periyar University, India in 2002, M.Tech. (Advanced Communication Systems) degree from SASTRA University, India in 2004, and Ph.D. degree from SRM University, India in 2012. His research interests include communication networks, image processing, biometric security, cloud computing, Big Data, cryptography and network security. He has published more than 50 research articles in International Journals and Conferences. He has filed a patent in the area of biometric authentication. He has published two books in the field of network security and cloud computing. He is an active reviewer in the reputed International Journals and IEEE conferences. He is a member of IEEE, IETE, IE(I), BES and ISTE.

Dr. Vidhyacharan Bhaskar received the B.Sc. degree in Mathematics from the University of Madras, Chennai, India in 1992, M.E. degree in Electrical & Communication Engineering from the Indian Institute of Science, Bangalore in 1997, and the M.S.E. and Ph.D. degrees in Electrical Engineering from the University of Alabama in Huntsville in 2001 and 2002, respectively. During 2002–2003, he was a Post-Doctoral fellow with the Communications research group at the University of Toronto, Canada, where he worked on the applications of space-time coding for wireless communication systems. During 2003–2006, he was an Associate Professor in the Department of Information Systems and Telecommunications at the University of Technology of Troyes, France. From Jan. 2007 to May 2014, he was a Full Professor in the Department of Electronics and Communication Engineering at SRM University, Kattankulathur, India. Since 2015, he is a Professor in the Department of Electrical and Computer Engineering at San Francisco State University, San Francisco, California, USA. His research interests include MIMO wireless communications, signal processing, error control coding, computer networks and queuing theory. He has published 112 Refereed Journal papers, presented around 70 Conference papers in various International Conferences and has published 3 books. He is on the Board of Reviewers of Refereed Journal articles published by IEEE, Springer, EURASIP, and Elsevier. He has currently been a reviewer of more than 50 Refereed journal articles. His name was nominated for inclusion in the 2011 and 2018 Editions of Who's Who in the World. He received the Prof. SVC Aiya Memorial Award for outstanding contributions in teaching and research. He also received a National Award for Outstanding Academic from the India Society of Technical Education. He is an IEEE Senior member, member of IET, Fellow of IETE and IE, India.