AN INTEROPERABILITY FRAMEWORK FOR IDENTITY FEDERATION IN MULTI-CLOUDS

Ahmad Mohammad Saied, Ayman Abdel-Hamid,Yasser El-Sonbaty

College of Computing and Information Technology
Arab Academy for science and Technology, Alexandria Egypt

ABSTRACT

Cloud computing is a major trend and a fast growing phenomena in the IT world. Organizations working in different sectors such as education or business are becoming more interested in moving their applications to the cloud to boost their infrastructure resources; increase their applications' scalability and reduce costs. This boost in demand for cloud services led to the need for clouds to federate, in order to flawlessly deliver the required computation power and other services. In addition, there is major trend in delegating identity management tasks to identity providers in order to reduce cost. Managing identity and access control across different domains is a challenge. This paper proposes a framework for managing identity in federated clouds based on the use of a Security Assertion Markup Language (SAML) Agent. The agent acts as an interface for a cloud identity manager where it sends and receives identity assertion requests from and to other clouds in the federation. In addition, the agent assigns roles to users using identity attributes received in assertions and cloud's internal role mapping rules. For testing the agent's capability to scale and sustain load, a prototype implementation was developed. Performance evaluation results demonstrate the viability of the proposed framework.

KEYWORDS

Cloud Computing, Federated Clouds, Federation, Identity Management, Security, Access Control.

1.INTRODUCTION

Nowadays, Cloud computing is a major trend in the IT world. It adopts a service-oriented architecture and leverages the flexibility of virtualization. It is not a new concept; since it has evolved from distributed computing and utility computing [1]. Cloud services can be categorized according to their level of abstraction as follows: Infrastructure as a Service (IaaS) [2]; Platform as a Service (PaaS) [2]; and Software as a Service (SaaS) [2]. IaaS provides basic infrastructure components such as CPUs, memory, and storage, (EC2) Amazon's Elastic Compute Cloud is an example of IaaS. PaaS provides libraries and tools that help in building applications. An example of PaaS is the Google Apps Engine which enables to deploy and dynamically scale Python and Java-based web applications. SaaS provides ready to use software such as Google Apps. Recently, there has been a number of efforts to leverage Cloud Computing potential, e.g., MedCloud [4] and a secure cloud-based service for electronic medical record exchange [5].

However, the resources of any cloud are finite, which leads to the need for cloud federation to leverage other clouds' computing and storage capabilities. Multi-Cloud, Intercloud or Cloud-of-Clouds are similar terms, these terms refer to the concept of avoiding to depend on a single cloud

DOI : 10.5121/ijcnc.2015.7506

and use several clouds to increase service availability [3]. Furthermore, cloud federation allows users to access multiple services even if not all services are deployed through a single cloud provider. For example, Microsoft Azure users can federate their applications with over 50 applications including AnswerHub, Salesforce and Google Apps.

Identity management systems (IMS) are a cornerstone of cloud computing. They are responsible for authenticating users and control access to resources based on user attributes, Identity management systems aim can be defined as "To maintain the integrity of identities through their life cycles in order to make the identities and their related data (e.g., authentication results) available to services in a secure and privacy-protected manner." [6]. There are several parties that interact in the IMS, The main two parties are Identity Providers (Idp) and Service Providers (SP) or sometimes called Relying Parties. Idp are responsible for generating identities and asserting identity attributes. SP are responsible for providing services after receiving identity information from Idp [6].

In federated clouds, a user is allowed to authenticate once in his home cloud and access services in other clouds without needing to re-authenticate which is termed "Single Sign On" (SSO). A single user might have different distinct identities in these systems, establishing a logical link between user's different identities is called Identity federation which is the base to accomplish SSO. However, cloud systems might use heterogeneous identity management systems. This causes problems in interoperability and in access control. Most of current Identity Management Systems are not built to operate in a federated environment.

In this paper, we propose an Interoperability Framework for identity federation in multi-clouds. The solution was designed to connect clouds' identity management systems and transfer identity data between them, and overcome the interoperability problem raised by the different systems and protocols used by these identity managers. Security Assertion Markup Language 2.0 (SAML 2.0) [7] is adopted as a standard for exchanging authentication and authorization data, between agents. The solution is agent-based in order to avoid single point of failure in multi-clouds, and allows direct communication between each agent in the federation which increases scalability. A prototype implementation was developed to compare performance differences between SAML and OAuth [9], and to test the effect of encrypting part of the SAML assertion to preserve the confidentiality of identity data. A number of solutions (section 3) have used SAML for communicating identity info. However, such solutions suffer a single point of failure disadvantage which limits their scalability as all federation requests go through a single point.

The contributions of this work can be summarized as follows:

- The design of an interoperability framework for identity federation in multi-clouds;
- The design of an Identity Agent in a federated cloud setup;
- Scalability enhancing techniques were proposed such as partial encryption of an assertion's sensitive data instead of full encryption; and
- A Prototype implementation has demonstrated that the Identity Agent succeeded in transmitting Identity data in a heterogeneous environment while efficiently scaling.

The rest of this paper is organized as follows: Section 2 discusses the required background concepts. Section 3 outlines related work. Section 4 presents the proposed interoperability framework for identity federation in multi-clouds. Section 5 presents preliminary performance evaluation results through an implemented prototype. Section 6 concludes the paper discussing future work.

2. BACKGROUND

This section concisely summarizes background material including cloud computing, identity management and its standards.

2.1. CLOUD COMPUTING

Cloud computing is based on the idea of delivering computation service as an utility like gas and electricity, Clouds are large pools of virtualized resources that can scale rapidly on demand, self-serviced, with broad network access, and its usage can be measured for monitoring and control, Clouds can be categorized depending on service or deployment model [2].

2.2. IDENTITY MANAGEMENT

Identity management is a combination of processes and Technologies used to manage and secure access to data and resources, and to protect user information. It is composed of two main processes: establishing an identity and user authentication.

Identity has several definitions. [10] defines Identity as "Identities of which each represents the person in a specific context or role. A partial identity is a subset of attribute values of a complete identity, where a complete identity is the union of all attribute values of all identities of this person."

Others [11] defined identity that covers a wider range of subjects—not just people, such as software agents, hardware devices, artificial objects (e.g., daily goods, machine parts, and buildings) and natural objects (e.g., livestock and crops) monitored and managed by sensors.

2.3. IDENTITY MANAGEMENT STANDARDS

There are several standards used in communicating Identity information, such as SAML, It was developed in 2001 and aimed to solve SSO. SAML is an XML based standard. Security Tokens containing XML identity Assertions are sent from SAML Identity Provider to SAML Service Provider. Assertions are sent upon request from SAML Service Provider [7]. The following is an example of an assertion holding name attribute. Figure 1 illustrates a SAML 2.0 operation sequence diagram.

```
<saml:AttributeStatement>
    <saml:Attribute FriendlyName="fooAttrib" Name="SFDC_USERNAME"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
        <saml:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="xs:string">
            User_Name
        </saml:AttributeValue>
    </saml:Attribute>
 </saml:AttributeStatement>
```

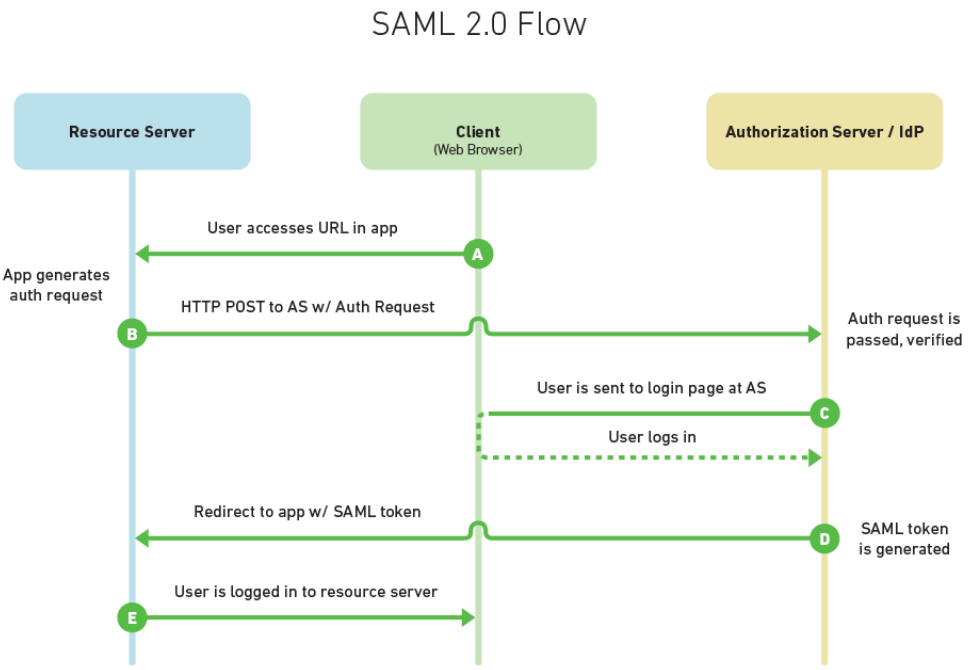

Fig. 1 SAML 2.0 Sequence Diagram [12].

OAuth is another standard which aims to solve authorization problem across different domains. It allows resources owner to grants access to his/her resources to third parties without sharing their credentials. OAuth is widely used in sharing online images, articles, webpages and activities on Social Media applications [13]. Figure 2 depicts OAuth's Standard main components and their interaction.

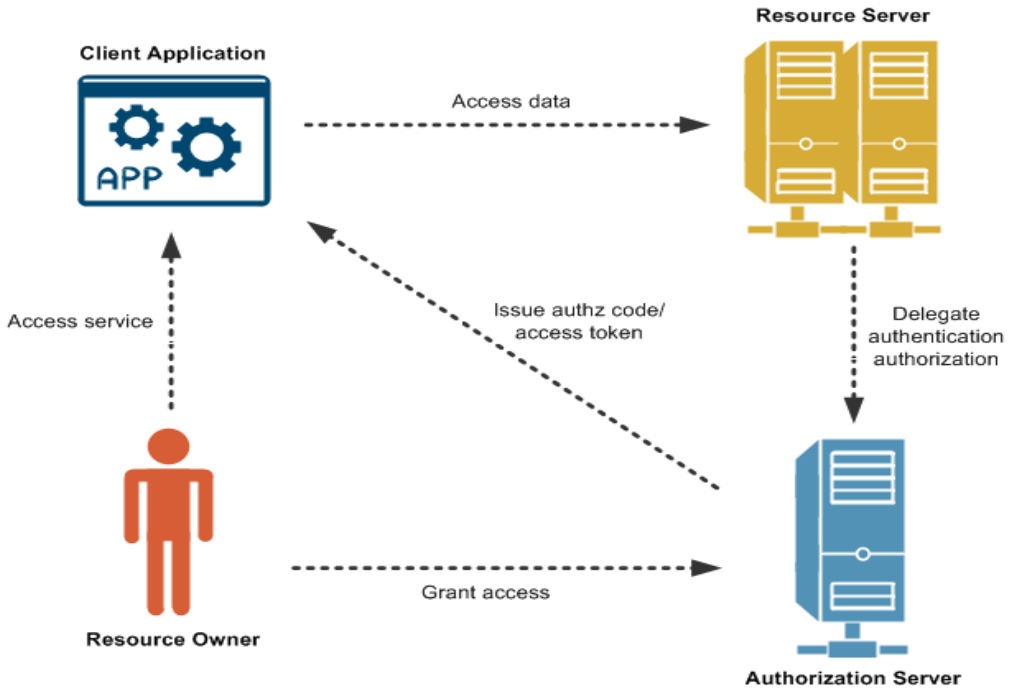

Fig. 2 OAuth Component Interaction [14].

3. RELATED WORK

Some efforts exist to interoperate Identity systems such as developing plugins that allow users authenticated in a domain using certain Identity Provider systems to communicate with other domains using a different Identity Provider through these plugins [15] [16]. Still there is an issue in controlling user's activity in foreign domain after granting him/her access.

Related work can be categorized as Agent-based, Layer-based, Broker-based, and identity as a service solutions as detailed in the following sections.

3.1. AGENT-BASED

In [17], Middleware architecture is centred on three design goals. These design goals are "no changes to the Cloud software", "controlled access to the underlying Cloud" and "ability to add new functionalities, the main goal of authors was to achieve Federated Identity Management in Intercloud. The focus of the architecture is towards a Middleware based Intercloud approach, where additional functionalities could be added into the Intercloud on a need for basis. This solution concerns with service acquiring mechanism between clouds. The identity management part uses OAuth for authorization.

3.2. LAYER-BASED

In [18], the proposed solution consists of two layers: Idm layer and trust layer.

The Idm Layer: manages sessions and user profiles, as well as issuing and processing authentication and authorization requests and responses. The system supports multiple authentication mechanisms (e.g. username/passwords, digital certificates, or delegated credentials). Concerning privacy, the Privacy Engine module is responsible for managing user identifiers and monitoring how user data is being accessed without compromising user's identity. The authors suggest that metadata lists can be thought of as equivalent to trust, the Dynamic Trust List (DTL).

Trust Layer: responsible for distributing, collecting and managing requests and responses of reputation to take federation decisions. It calculates the trust value associated to each entity depending on context, analysis of the risk factors associated with each Transaction, history keeping of past interactions. As trust value will change over time.

The solution proposes a new architecture for computing clouds that can be adopted in future solutions but doesn't solve the identity federation problem while using existing technology.

3.3. BROKER-BASED

In [19], a solution offers a federation that is both horizontally (i.e. between multiple IaaS providers) and vertically (i.e. through all abstraction levels – SaaS, PaaS, and IaaS). The authors assume that other solutions generally deal with relatively homogeneous scenarios (e.g. every resource is a web site) or makes some assumptions (e.g. authentication is interactive and password-based). It is hard to measure with precision how much of the contracted resources a given SaaS user has consumed, because the user is only known inside the SaaS application itself. Without the unification of SaaS user identities and IaaS user identities, it is only possible to obtain estimated (average) individual resource consumption. The authors suggest a solution distributed between three domains, the user application domain (SaaS), the security tokens server is implemented as (PaaS) and the computing resources used by the applications as (IaaS). The problem with this approach is that it has a single point of failure and a scaling problem because all interaction between SaaS and IaaS interaction must first go through the STS.

In [20], a solution to decrease accounts management load due to the use of cloud services and solve scalability problem in pair wise Idp level trust model. The authors suggests an identity federation broker model that introduces a trusted third party as a broker for establishing identity federation between services in cloud and cross clouds in a user centric approach. It consists of a broker server connected to a set of extensible gateways, the gateways acts as an interface between the broker and the federated clouds.

In [21], the authors assume that current federation frameworks do not support dynamic federation. Trust is an obstacle to scale in current systems because it must be preconfigured. SAML, Shibboleth, WS-federation LA and OpenID do not offer proper trust model for dynamic federation because it trusts all comers. SAML is flexible and abstract enough for extending trust model. The authors propose a SAML trust extension that allows entities to include external trust. SAML trust list is turned into Dynamic Trust Lists (DTL) by gathering external trust information and taking into account previous experiences and community knowledge. DTL is updated after certain events such as successful interaction or receiving recommendations. The authors propose to extend SAML entities with trust engine which handles trust data, updating DTL and making trust decisions. This solution works only between clouds with SAML Identity providers and doesn't address the access control problem.

In [22], the authors address the problem of cloud services Quality of Service (QOS). The authors suggest that clouds should be able to acquire resources dynamically to meet sudden demand spikes. The authors identify several challenges such as application service behaviour prediction, flexible mapping of services to resources and integration and interoperability between software on premises and the services in the cloud. A Cloud Coordinator for exporting Cloud services and their management driven by market-based trading and negotiation protocols for optimal QoS delivery at minimal cost and energy. A Cloud Broker responsible for mediating between service consumers and Cloud coordinators. A Cloud Exchange acts as a market maker enabling capability sharing across multiple Cloud domains through its match making services and a software platform implementing Cloud Coordinator, Broker, and Exchange for federation. The architecture proposed cohesively couples the administratively and topologically distributed storage and computes capabilities of Clouds as parts of single resource leasing abstraction. Part of the propped architecture a Cloud Exchange (Chex) brings together service producers and consumers and acts as an information registry that stores the Cloud’s current usage costs and demand patterns., The Cloud Coordinator service is responsible for the management of domain specific enterprise Clouds and their membership to the overall federation. The Cloud Broker, acting on behalf of users, identifies suitable Cloud service providers through the Cloud Exchange and negotiates with Cloud Coordinators for an allocation of resources that meets Quality of Service needs of users.

Shibboleth [6] is an open source SSO system, it allows user to login to several federated systems using single username and password. Shibboleth consists of three individual core components, the service provider (SP), identity provider (Idp) and the discovery service (DS). Trust between these components is achieved using public key cryptography (often simply SSL server certificates) and metadata that describes providers. Shibboleth implements SAML standard. Shibboleth version 1.x implements SAML 1 and the Shibboleth new version 2.x implements SAML 2. Regarding privacy, Shibboleth gives the user the control on which attributes are released to service providers upon signing in. Shibboleth performs SSO by matching attributes supplied by Idp against rules set by SP. Attributes can be any information about the user, users' identity is an attributes he/she can choose to hide it in order to protect his privacy.

Shibboleth Idp decides how it will authenticate the user based on roles established by the service provider saved in XML file. The Idp gathers the needed attributes by the SP, the attributes is filtered so only the necessary ones are passed to protect user's privacy, the attributes are then encoded with the Idp key or in Shibboleth 2 with SP key.

The SP's Assertion Consumer Service decrypts the Idp messages, checks for the required credentials, passes the attributes to an attribute extractor and puts it in a cacheable form after passing through the attribute filter to prevent the SP from violating any policies the filter implements. Shibboleth 2 doesn't yet support Single Logout. In multi-clouds environment, with clouds with different identity systems and environment, Shibboleth can't operate. All federated clouds have to implement Shibboleth as identity provider for Shibboleth to operate.

3.4. IDENTITY AS A SERVICE "IDAAS" SOLUTIONS

Several *IDaaS [6]* solutions such as "Ping Identity"[23] and "Safe Layer" [24] exist nowadays to fullfil the market demand for Identity federation services. They support the most used Identity protocols such as SAML, OAuth and OpenID. They are able to play this role because now most Cloud service providers allow their clients to configure their own identity provider for authenticating their users. They carry out a very important role by allowing users to impelement SSO property for the independent services they use. This approach is valid for independent services. Adminstration effort is needed to configure each service. For federated services, this approach will not give a uniform view of services and require redundant adminstration effort.

4. INTEROPERABILITY FRAMEWORK FOR IDENTITY FEDERATION

Figure 3 illustrates the abstract view of the proposed agent-based interoperability framework for identity federation in multi-clouds. The Identity Agent acts as a connector between different Identity systems. Each cloud in the federation can use different identity providers, whether external (cloud A and C) or internal (Cloud B) as illustrated in Figure 3. In this heterogeneous domain, interoperability problems arise. The proposed identity agent aims to solve such problems by providing a means to communicate in-between agents using a standard protocol "SAML". Each agent in the federation can communicate directly with other agents avoiding any central communication point therefore increasing scalability. Configurations files can be used in order for a new agent to join the cloud federation. This pre-configuration process is essential for defining the role that users of each cloud in the federation will be granted once they gain access to the cloud.

Section 4.1 outlines the identity agent logical blocks. Section 4.2 presents a possible federation scenario. Section 4.3 presents the proposed framework analysis including scalability, security risks and countermeasures, and comparison versus related work.

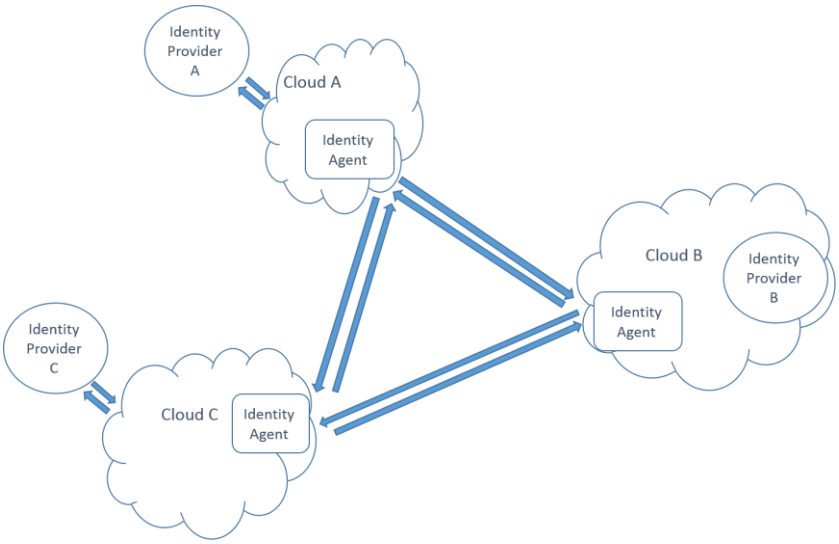

Fig. 3 The abstract view of proposed Interoperability Framework.

4.1. IDENTITY AGENT LOGICAL BLOCKS

The **Identity Agent** handles identity data communication between federated clouds. Each cloud can be using a different identity provider whether internal or external (third party). Figure 4 illustrates the Identity agent logical blocks: request processor, assertion processor, and access controller. Handling trust establishment which is essential for successful operations is currently out of the scope of the proposed framework.

4.1.1. REQUEST PROCESSOR

The Request processor generates SAML Assertion Requests in case an unauthenticated user tries to access the cloud's resources, sends the request to user's Home cloud. In addition, it receives these requests from other clouds in the federation asking to send assertions holding information about its users. The created Assertion includes the URL needed to send the requested Assertion to. In addition, it receives Assertion requests and calls the Assertion Processor component.

4.1.2. ASSERTION PROCESSOR

The Assertion Processor generates Assertion about the Cloud's users, upon the request from other clouds in the federation, Assertions hold user's attributes such as Role, and these attributes are needed to determine user's privileges.

4.1.3. ACCESS CONTROLLER

The Access Controller is responsible of mapping visiting users to cloud's internal role; the mapping depends on the received user's information from his Home cloud. The privileges of the role are determined by the visited cloud. In the proposed framework, Role Based model for Access Control (RBAC) [25] is adopted. In this model, access control is based on the role granted to the user in the system. This way it is easy to grant or revoke privileges to roles and these settings will be applied on all users in the role. Using RBAC allows each cloud to group users accessing its resources from other clouds in the federation by Cloud or by their role in their home cloud or by both. This resiliency, in controlling access, increases scalability by saving computing resources consumed by controlling access.

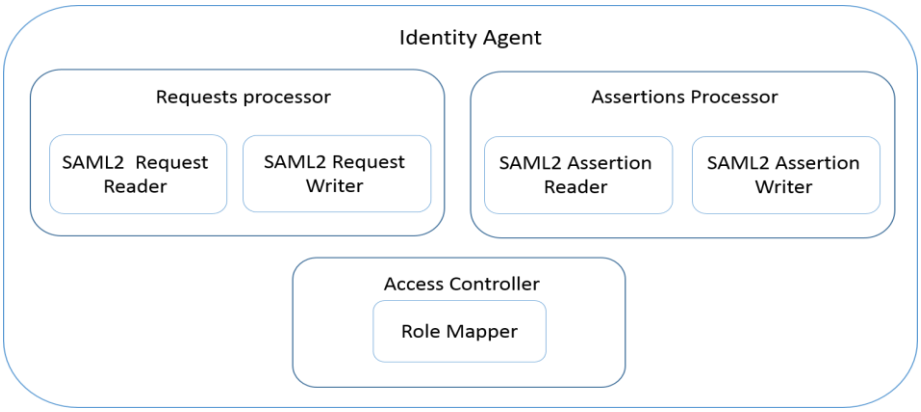

Fig. 4 Identity Agent Logical blocks.

4.2. IDENTITY FEDERATION SCENARIO

The **Identity Agent** aims to solve access control and interoperability problems in InterCloud environment. In a typical scenario, there will be a cloud user requesting resources from a visited cloud. Possible emerging problems: What access rights the cloud users have in the foreign cloud?; How to track such users' activity?; and a potential interoperability problem between identity management systems used by each cloud.

Hence, the Cloud's internal Identity Manager intercepts the request for resources. If the user is not authenticated locally or through trusted Identity provider, the Cloud's internal Identity Manager delegates the authentication request to the Identity Agent.

The **Request Processor** redirects the user to his home Cloud along with an Assertion request. The cloud may ask the user about his home cloud, Where Are You From? "WAYF", or get this information from the HTTP Referrer. At the user's Home Cloud, the Request Processor receives the Assertion requests, then the Request Processor communicates with internal Identity Manager to authenticate the user, After authentication success and checking user's privilege, the Assertion processor creates an Assertion holding the user's Role among other attributes and sends it back to the visited Cloud "the requester". It sends it to the URL it got from the Assertion Request it received earlier.

The visited cloud receives the response and the Assertion Processor extracts user's attributes. The user's data is then delivered to the local Identity Manager including the role attribute to create a session for the user and decide on the user's privileges.

The authentication process takes place in the cloud through an internal service or through and identity provider. SAML Protocol doesn't specify how the authentication process should be carried out. The federation scenario is demonstrated in Figures 5 and 6. Figure 5 illustrates identity federation workflow. Figure 6 depicts identity agents' components interaction.

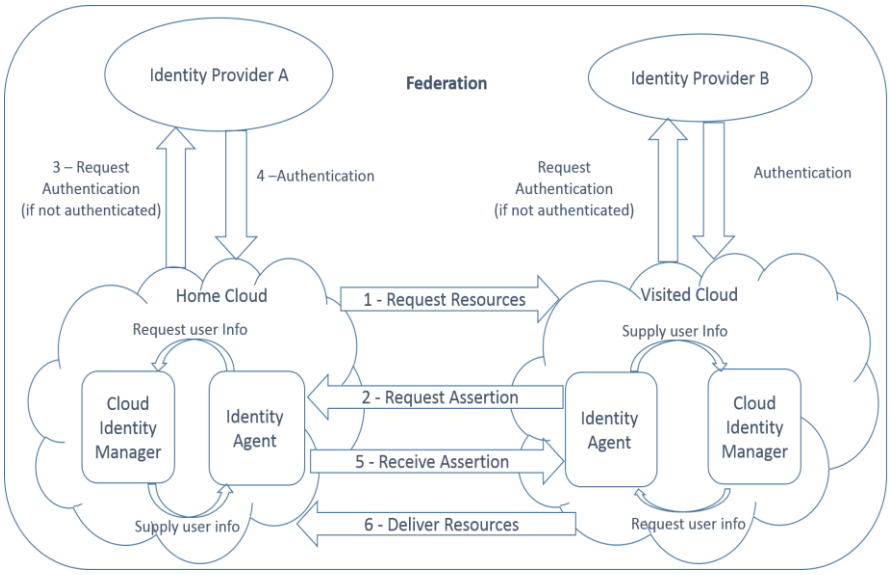

Fig. 5 Identity Federation Workflow

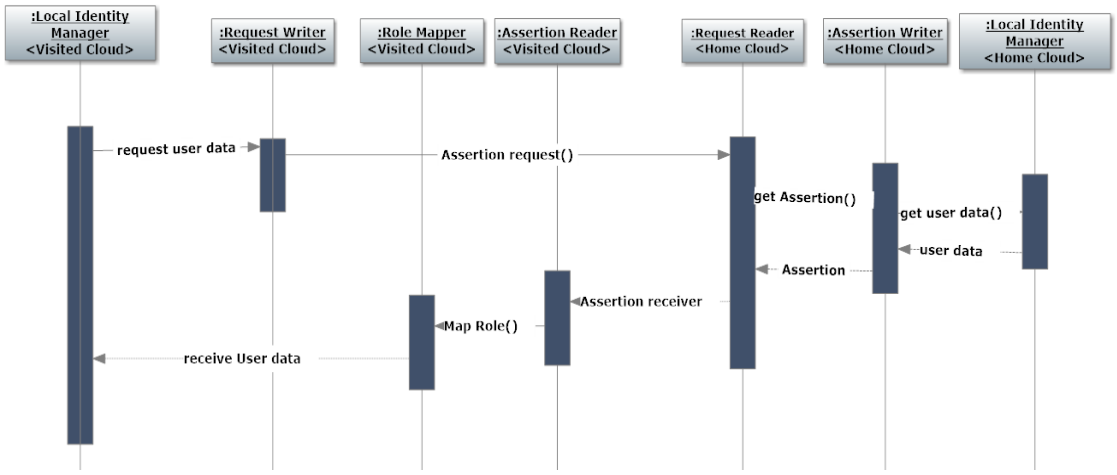

Fig. 6 Identity Agent Components' Interaction.

4.3. INTEROPERABILITY FRAMEWORK ANALYSIS

In this section, the proposed framework is analyzed in terms of its validity, scalability, security risks and countermeasures, and comparison versus related work.

4.3.1. SCALABILITY

Accepting multiple Assertions or sending multiple requests from and to single or multiple federated clouds will be limited by hardware capability. To increase scalability, it is not necessary to encrypt whole requests. Only assertions or the part holding user's information in the assertion needs to be encrypted as each request is handled by a separate thread.

The total identity federation processing time needed during a successful federation scenario is as shown in Equation 1. Processing time components are further explained in Table 1. The inter-agent interaction sequence is illustrated in Fig. 7 depicting different processing time components.

$$t_{idf} = t_{GenerateRequest} + t_{SendRequest} + t_{ProcessRequest} + t_{GenerateAssertion} + t_{SendAssertion} + t_{ProcessAsertion} \quad (1)$$

Scalability can be increased by decreasing messages size and number of messages used in communication. By decreasing message size $t_{GenerateRequest}$, $t_{ProcessRequest}$, $t_{GenerateAssertion}$ and $t_{ProcessAsertion}$ can be decreased. $t_{SendRequest}$ and $t_{SendAssertion}$ are out of the framework's control but can be decreased in general by decreasing number of messages used in communication. Decreasing message size and its processing time is achieved by encrypting only the part containing sensitive data.

Table 1. Identity federation processing time notation and description

t_{idf}	Total identity federation processing time
$t_{GenerateRequest}$	Time needed to generate request
$t_{SendRequest}$	Time needed to send request and reach destination
$t_{ProcessRequest}$	Time needed to process request
$t_{GenerateAssertion}$	Time needed to generate assertion
$t_{SendAssertion}$	Time needed to send assertion and reach destination
$t_{ProcessAsertion}$	Time needed to process assertion

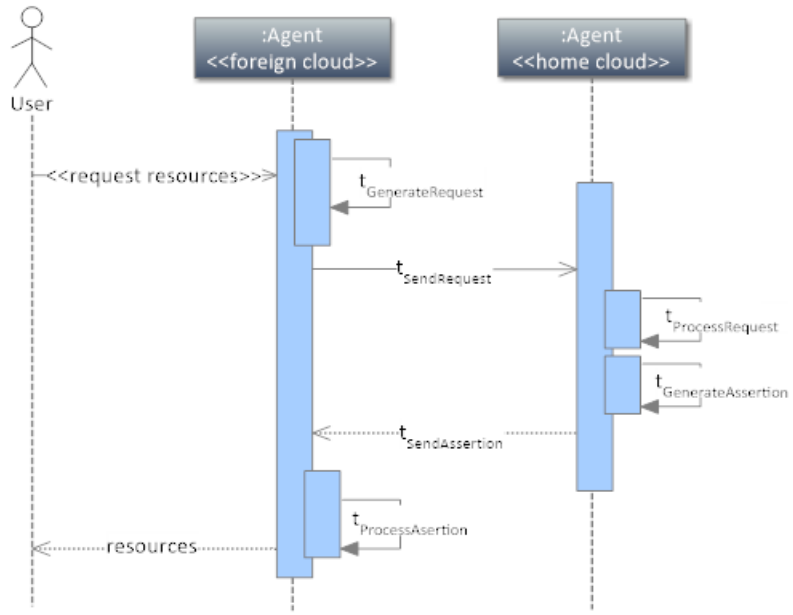

Fig. 7 Inter-Agent Interaction Sequence Diagram.

4.3.2. SECURITY RISKS AND COUNTERMEASURES

The framework may be susceptible to security attacks such as Denial of service attack, Man in the middle attack, and Replay attack. Countermeasures include using Time stamp or nonce for preventing Replay attack, public key infrastructure (PKI) such as TLS in case of web communication, security certifications such as X.509. Authentication and confidentiality can be achieved in inter-agent communication through the use of public/private key pairs and double

encryption as given by equation 2 where E= Encrypt, D = decrypt, KU = Public Key, KR = Private Key, A and B are the communicating Parties, M is a plain message, and C is the generated ciphertext.

$$A \rightarrow B: C = E_{KU(B)} (E_{KR(A)} (M)) \quad (2)$$

B can process the received communication using equation 3.

$$B: M = E_{KU(A)}(E_{KR(B)}(C)) \quad (3)$$

4.3.3. COMPARISON VERSUS RELATED WORK

Table 2 summarizes the comparison between related work and the proposed framework. Comparing SAML to OAuth we find that SAML assertions hold user identity Information while OAuth tokens are just used for authorizing actions. In addition, the OAuth Resource Server needs to make an additional round trip to validate the token with the Authorization Server as previously illustrated in figures 1 and 2.

By comparing SAML with OpenID we find that SAML is based on an explicit trust between Service provider and Identity Provider while OpenID Service Provider accepts identities coming from arbitrary Identity Providers. There is no trust in OpenID. A Service provider can never really know who the user is. Instead, the Service provider can know that a user is the very same person that registered an account.

As shown in table 2, in case of agent failure the other clouds in the system will keep functioning, the failure affects only the Agent's cloud. As soon as the Agent resumes functioning, it can accept requests and send assertions. A log file can be used to recall the Agent's last status.

Table 2. Identity Federation Solutions Comparison

Solutions	Structure	SSO	Protocol	Fault-Tolerance
[17]	Agent	Yes	OAuth	Cloud federation tolerates individual Cloud failure
[19]	Broker	Yes	SAML	Single point of failure
[20]	Broker	Yes	SAML	Single point of failure
[18]	Layer	Yes	SAML	Cloud federation tolerates individual Cloud failure
[21]	Broker	Yes	SAML	Single point of failure
[22]	Broker	Yes	-	Single point of failure
IDaaS [6]	Broker	Yes	Multiple	Single point of failure
Proposed Framework	Agent	Yes	SAML	Cloud federation tolerates individual Cloud failure

5. PERFORMANCE EVALUATION

A preliminary prototype was developed to evaluate and compare the performance of the Agent using SAML with and without partial encryption using RSA [26]. Another evaluation was performed using OAuth, each with different requests' load.

5.1. PROTOTYPE TECHNICAL SPECS

The prototype is implemented using C#. The prototype consists of two web applications simulating Clouds, and a class contains Assertion Manager and Request Manager's functionalities. For authentication and authorization OWIN [27] Forms authentication was used with Authorization Attribute, for creating SAML messages .Net framework 4.5 libraries are used and also XML writing and reading libraries. For the sake of simplicity messages were not decoded and signed, Communication is based on using the HTTP protocol. Single encryption is used for simplicity and as a proof of concept. Figure 8 shows the prototype deployment diagram.

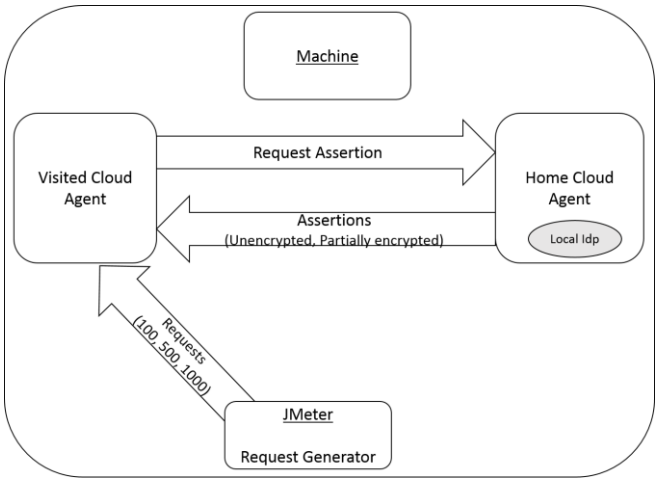

Fig. 8 Agent Prototype Deployment Diagram.

5.2. PERFORMANCE EVALUATION METHODOLOGY AND RESULTS

Load testing is performed by simulating requests from different users' numbers. The test is repeated with part of the assertion encrypted at the Home Cloud and decrypted at the Visited Cloud. Apache JMeter [28] was used for load testing. In addition, retest was performed using RSA encryption. Only a part of the SAML assertion was encrypted using RSA encryption through the use of X.509 certificate [29]. There is no need to encrypt the whole certificate, only the part containing sensitive data needs to be encrypted. The Ramp-Up period (the time JMeter should take to start the total number of threads) is one second. Performance evaluation results were as depicted in Table 3.

Table 3. Performance Evaluation Results

# of Users	Average Response (ms)	response time Increase %	STD. Dev/ms	Throughput / sec	AVG. Bytes
Unencrypted SAML					
100	3441		477	22	8934
500	21624	≈ 528 %	629	20	8934
1000	40557	≈ 88 %	1134	19.9	8934
Partially Encrypted SAML with RSA					
100	4530		570	17.5	9610
500	23026	≈ 408 %	621	19.2	9610
1000	47680	≈ 107 %	1676	17.3	9610
OAuth					
100	1429		330	36	2637
500	12113	≈ 748%	3710	28	2637
1000	22860	≈ 89 %	5154	26	2637

Where

- Average Response (AR) is the average elapsed time for the set of results in milliseconds
- STD. Deviation (SD) quantifies how much response time varies from the average response time
- Response time Increase Percent shows the percentage of increase in response time due to the increase in users' number, the increase percent =((New AR – Old AR) / New AR) %
- The throughput shows how many requests per second can the server handle
- Avg. Bytes (AB) is the average response size.

Analyzing the results we realize that encrypting only part of the assertion makes an increase in AR by 32% for 100 users, 7% for 500 users and 17% for 1000 users. Decrease in SD by ≈ 1% at 500 users and increase by 19% and 47 % for 100 and 1000 users respectively.

Using OAuth compared to unencrypted SAML decreased AR by 58%, 43% and 44% for 100, 500 and 1000 users, respectively. Figure 9 depicts the results in Table 3.

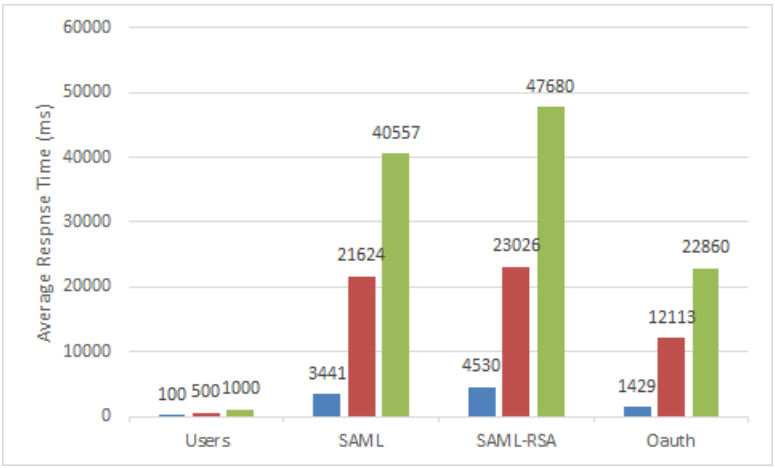

Fig. 9 Visual representation of Table 3 Results

Although in SAML it is a two-step communication: Request for assertion and sending assertion, the size of messages in SAML is relatively large. It can hold detailed information about the user. In OAuth, there is one extra step which is token request prior to request for authorization. Furthermore, OAuth messages are smaller in size.

6. CONCLUSION AND FUTURE WORK

Cloud federation is becoming a necessity in order for clouds to load balance traffic and accommodate spikes in demand. One major obstacle in this path is identity federation. The proposed framework: The interoperability Framework for Identity Federation aims to solve communicating Identity data between Multi-Clouds, where each cloud uses different identity management system. We offered a solution that uses existing technologies to solve this problem. A proof of concept prototype was implemented for validation and for performance evaluation purposes. For real life usage, we believe that custom development is needed for different cloud systems, this is necessary as each cloud solution has different identity management system. SAML was chosen for communication between Identity Agents because of its SSO capability, platform neutrality and security. Identity Agents can communicate directly with other agents in the federation which increases scalability.

As future work, we plan to investigate dynamic trust establishment between Identity Agents to increase scalability. In addition, fault-tolerance issues as related to the identity agent will be researched.

REFERENCES

[1] A. O. Joseph, J. W. Kathrine, and R. Vijayan, "Cloud Security Mechanisms for Data Protection: A Survey" International Journal of Multimedia and Ubiquitous Engineering, vol. 9, pp. 81-90, 2014.

[2] NIST Definition of Cloud Computing, "The NIST Definition of Cloud" Available http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf, September 2011.

[3] M. A. AlZain, E. Pardede, B. Soh, and J. A. Thom, "Cloud Computing Security: From Single to Multi-clouds" in System Science (HICSS), 2012 45th Hawaii International Conference on, 2012, pp. 5490-5499.

[4] D. Sobhy, Y. El-Sonbaty, and M. Abou Elnasr, "MedCloud: Healthcare cloud computing system" in Internet Technology And Secured Transactions, 2012 International Conference for, 2012, pp. 161-166.

[5] A. S. Radwan, A. A. Abdel-Hamid, and Y. Hanafy, "Cloud-based service for secure Electronic Medical Record exchange" in Computer Theory and Applications (ICCTA), 2012 22nd International Conference on, 2012, pp. 94-103.

[6] E. Bertino and K. Takahashi, Identity Management Concepts, Technologies, and Systems: Artech House, 2011.

[7] (2014, 12 November). SAML Specifications. Available: http://saml.xml.org/saml-specifications

[8] (2014, 12 November). OpenID. Available: http://openid.net

[9] (2014, 12 November). OAuth Documentation. Available: http://oauth.net/documentation

[10] A. Pfitzmann and M. Hansen, "A Terminology for Talking About Privacy by Data Minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management" pp. 98, Aug. 2010.

[11] M. Bishop, Computer Security: Art and Science, 1 Ed.: Addison-Wesley Professional, 2002.

[12] (2015, 5 May). Choosing an SSO Strategy: SAML vs OAuth2. Available: http://www.mutuallyhuman.com/blog/2013/05/09/choosing-an-sso-strategy-saml-vs-oauth2/

[13] (2015, 7 May). Chapter 20. OAuth - Authentication with Social Network accounts. Available: https://access.redhat.com/documentation/en-US/Red_Hat_JBoss_Portal/6.1/html/Administration_and_Configuration_Guide/chap-OAuth_-_Authentication_with_Social_Network_accounts.html

[14] 2015, 5 May). Oracle® Fusion Middleware Introduction to API Gateway OAuth 2.0. Available: http://docs.oracle.com/cd/E50612_01/doc.11122/oauth_guide/content/oauth_intro.html

[15] (2015, 7 May). Gluu releases Shibboleth Plugin for OX to enable simultaneous OpenID Connect and SAML sessions. Available: http://www.gluu.org/press-releases/2013/gluu-releases-shibboleth-plugin-for-ox-to-enable-simultaneous-openid-connect-and-saml-sessions/

[16] (2015, 7 May). Shibboleth 2 Contributions and Extensions. Available: https://wiki.shibboleth.net/confluence/display/SHIB2/Contributions

[17] D. N. Sriram, "Federated Identity Management in Intercloud" Master Thesis, Technische Universität München, 2013.

[18] R. Sanchez Guerrero, P. Arias Cabarcos, F. Almenares Mendoza, and D. Diaz-Sanchez, "Trust-aware Federated IdM in Consumer Cloud Computing" in Consumer Electronics (ICCE), 2012 IEEE International Conference on, 2012, pp. 53-54.

[19] M. Stihler, A. O. Santin, A. L. Marcon, and J. S. Fraga, "Integral Federated Identity Management for Cloud Computing" in New Technologies, Mobility and Security (NTMS), 2012 5th International Conference on, 2012, pp. 1-5.

[20] H. Y. Huang, B. Wang, X. X. Liu, and J. M. Xu, "Identity Federation Broker for Service Cloud" in proceedings of the 10th International Conference on Services Sciences, pp. 115-120, Hangzhou, China, 2010.

[21] P. Arias Cabarcos, F. Almenárez Mendoza, A. Marín-López, and D. Díaz-Sánchez, "Enabling SAML for Dynamic Identity Federation Management" in Wireless and Mobile Networking. vol. 308, J. Wozniak, J. Konorski, R. Katulski, and A. Pach, Eds., ed: Springer Berlin Heidelberg, 2009, pp. 173-184.

[22] R. Buyya, R. Ranjan, and R. N. Calheiros, "InterCloud Utility-Oriented Federation of Cloud Computing Environments for Scaling of Application Services" in proceedings of the 10th International Conference on Algorithms and Architectures for Parallel Processing, pp. 13-31, 21-23 May, 2010.
[23] Ping. (2014, 31 December). Single Sign-On for Your Workforce. Available: https://www.pingidentity.com/en/solutions/workforce/single-sign-on.html
[24] Safelayer. (2014, 31 December). Identity Federation. Available: http://www.safelayer.com/en/solutions/identity-federation
[25] NIST RBAC. (2004, 21 April). Role-based Access Control (RBAC) and Role-based Security. Available: http://csrc.nist.gov/groups/SNS/rbac/
[26] EMC2 RSA. (2015, 5 May). RSA. Available: http://www.emc.com/domains/rsa/index.htm
[27] (2015, 5 May). OWIN - Open Web Interface for .NET. Available: http://owin.org/
[28] Apache. (2014, 14 November). JMeter. Available: http://jmeter.apache.org
[29] (2015, 5 May). X.509 : Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks. Available: http://www.itu.int/rec/t-rec-x.509/e

Authors

Ahmad Saied is a software engineer. He has ten years of experience in IT. He received his Computer Science Graduate Diploma from AASTMT in 2009. He is currently working towards his Master of Science in Computer Science in The College of Computing and Information Technology, AASTMT. His research interests include Cloud Computing, Distributed Systems, and Identity Federation.

Ayman Abdel-Hamid is a Professor of Computer Science in the College of Computing and Information Technology, Arab Academy for Science, Technology, and Maritime Transport (AASTMT), Alexandria, Egypt. In addition, he currently holds the position of Head of Computer Science and Software Engineering Departments. He received his Ph.D. degree in Computer Science from Old Dominion University, VA, USA in May 2003. His research interests include mobile computing, network-layer mobility support, computer and network security, distributed systems, and computer networks. He is a member of IEEE, IEEE Computer Society, ACM, and ACM SIGCOMM.

Yasser El-Sonbaty is a Professor of Computer Science in the College of Computing and Information Technology, Arab Academy for Science, Technology, and Maritime Transport (AASTMT), Alexandria, Egypt. In addition, he currently holds the position of Dean of College of Computing and Information Technology. He received his Ph.D. degree in Computer Science from Alexandria University, Egypt in 1998. His research interests include Pattern Recognition, Image Processing, Data Mining and Natural Language Processing.