A secure and efficient authentication technique to access cloud services in Openstack

Raghavendra K[1], B Ramesh[2]
[1]*Assistant Professor, Dept of Computer Science and Engineering, NIEIT Mysuru-570018 , Karnataka India*
[2]*Professor, Dept of Computer Science and Engineering, Malnad College of Engineering, Hassan Karnataka India*

Abstract **- Nowadays cloud services have gained more interest. The main advantage of the cloud is, it reduces management costs and efficient usage of resources. The major role of efficient authentication means not only providing authentication for a single service in a cloud environment it should provide AAA in a federated identity management environment also, called as single sign-on(SSO). There are many authentication mechanisms most of them were only for a web application. Here in this paper, we focus on both web and non-web applications for efficient AAA in a cloud environment. For non-web application services of cloud, ABFAB has created an architecture for federated identity management environment.**

ABFAB also defines how to use existing EAP/AAA and GSS-EAP for both web and non-web-based applications. In this paper, we mainly focus on efficient AAA to access cloud services. To demonstrate the proposed system, we use the moonshot from Github, freeradius an opensource, and Openstack for cloud service with our proposed authentication method. We have done the performance analysis of our authentication method compared with the other authentication mechanism to access cloud services. This analysis shows a significant reduction in the computation time required for authentication and authentication traffic.

I.INTRODUCTION

OAuth, OpenID, and SAML are the few web-based authentication technologies to access cloud services and also in federated identity management technologies used by Google, Amazon, and Microsoft. RADIUS and EAP methods support AAA for both web and non-web (generic) applications in accessing cloud services and also in federated identity management environments, in AAA federated environments each organization should deploy AAA server and deployed organizations should interconnect with all AAA servers, in this way federation of identity and AAA can be provided. For authentication, an extensible authentication protocol framework is used with an extensible set of “EAP methods (e.g EAP-TLS, EAP-TTLS, EAP-MD5)”. AAA is widely used for network access in federated environments. Eduroam is an example which is using AAA infrastructure for providing internet access through WiFi for the members of federated organizations students and research scholars. By the federated identity management, if the user of the same organization (home organization) once authenticates, he or she can access services provided by other federated organizations. Today more interests gaining in providing AAA for services, provided by cloud and internet access. An example of providing internet access using AAA infrastructure is eduroam, here RADIUS provides federated infrastructure, and the Extensible authentication protocol provides authentication. The success of eduroam has gained more interest to use RADIUS infrastructure for AAA for any type of application services for example SSH, HHTP, and cloud services, including network access too. For providing AAA using RADIUS, EAP and to access any kind of services in the federated environment is defined in ABFAB. “Generic Security Service Application Program Interface (GSS-API)” is a new mechanism specified by ABFAB. “GSS-API is based on an Extensible authentication protocol”. “GSS-API” is already included and supported by many of the application services. In GSS-EAP authentication, several authentication message exchanges take place with the end-users home organization. In typical EAP authentication, first, there will be the establishment of tunnel TLS/SSL (using the

EAP-TLS method) between the AAA server(home) and the end-user.

1.1. EAP/AAA

An EAP is an authentication framework used mainly for network access. Extensible authentication protocol architecture is very flexible because we can use any authentication techniques known as extensible authentication protocol methods. "These EAP methods are executed between an EAP peer and an EAP server through an EAP authenticator", EAP authenticator just forwards the messages between the EAP peer and the EAP server. Usually, Extensible authentication protocol is deployed over RADIUS or Diameter and also called as (AAA) infrastructure. AAA provides a framework for 3 security services, "authentication, authorization, and accounting". In AAA infrastructure if the user wants to get to access a network service, the user will be registered in an IdP(Identity Provider), provided by an SP.

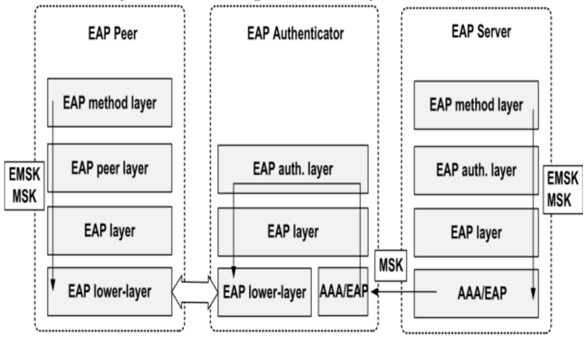

Figure 1.1.1: Architecture of Extensible authentication protocol.

1.2. GSS-API

A set of generic security functions are defined by the GSS-API [11] for server and client applications. The generic set of security functions makes client and server application-independent for any kind of security mechanism. This makes a client application (GSS Initiator) and a server application (GSS Acceptor) to establish a security context between each other. The context exchange between client and server results in GSS tokens and transported using application protocol. These tokens' format and content are derived from the underlying authentication mechanism. In the phase of security context exchange (establishment), both the client and server (initiator and acceptor) are mutually authenticated, services like confidentiality, mutual authentication and application data protection is also provided. the protocols namely XMPP.1, SMTP, SSH, HTTP are all supported by the GSS-API.

1.3. *Application Bridging for Federated Access Beyond Web(ABFAB)"*

ABFAB [12] makes AAA infrastructures support both "web and non-Web application" services (e.g. file storage, grid, remote access, and cloud infrastructures, etc.).

Specifically, the "Application Bridging for Federated Access Beyond Web" joins existing protocols, for example, RADIUS/Diameter protocol, "GSS-API / SAML Security Assertion Markup Language" and EAP [13]. "The Application Bridging for Federated Access Beyond Web (ABFAB)" architecture consists of 3 entities: User application, who requests for the service, the "Relying Party (RP)", accessing services will be controlled by this entity, (IdP) Identity Provider, who is responsible for verifying the credentials of the user and performs authorization process. If the end-user needs to access any service, it executes the Client application, as a component of an access control method, a security context (GSS-API) should be established between the GSS initiator (client) and the GSS acceptor(RP).

GSS context does two things i) mutual authentication between RP and client, ii) identity information will be given to RP of the user, and credentials for "AA authentication and authorization". In RP's viewpoint, GSS-API performs access control and EAP performs the authentication process, in EAP authentication client acts as an EAP peer, Idp acts as an EAP server, RP acts as an EAP authenticator. Using GSS-API EAP packets are transported between RP and Client for this purpose GSS-API mechanism is defined by ABFAB for EAP known as (GSS-EAP) it also defines how EAP packets sent over GSS tokens and how credentials (keying materials

At last, RADIUS protocol, used between the Relying Party and the identity provider, provides a federation substrate, by implementing the trust associations. 2 resolutions of using RADIUS protocol has: i) it conveys EAP packets to RP and the IdP. ii) it transports authorization data about the end-user from the IdP to the RP. This information is represented using SAML.

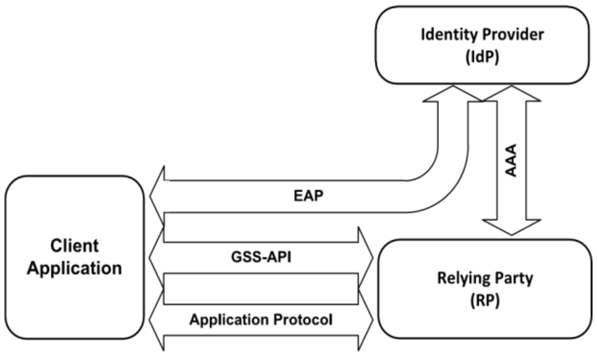

Fig. 1.3.1: ABFAB protocol and entities.

1.4. Moonshot

Moonshot is based on AAA and it is separated into a few modules, the important and applicable is the one that actualizes the "GSS-EAP" system. The moonshot has 2 unique libraries.

The EAP authenticator and EAP peer are implemented in the library named libeap. The libeap [15] has been modified to broaden the EAP, consolidating the advances and usefulness related to REAP. GSS-EAP methods functionality is implemented in the library mech_eap. For the EAP functionality, mech_eap uses the libeap library.

1.5. FreeRadius

FreeRadius is utilized to execute as a home AAA server (IdP), which does the EAP confirmation among clients and connects the cloud administration with the "AAA infrastructure". "FreeRadius" has a lot of modules. To help EAP strategies, In the EAP module of free-range (called rlm_eap) all the usefulness of EAP exists. Little changes are made to this module to help the EAP-REAP and use related to EAP in the IdP and to deal with the various keys in IDP.

1.6. OpenStack

Openstack is made up of different modules, each module provides different cloud services like virtual machines, object storage, file (swift services), and networking. Using HTTP RESTFUL APIs all the modules will communicate with each other for providing services. Authentication and authorization and identity management for the cloud are provided by the module called keystone. Keystone supports a variety of user "identity credentials e.g X.509 certificates" and username/passwords. It also supports different authentication technologies (LDAP, Kerberos, etc). Besides, on account of the endeavors of the "CLASSe project" [12], it additionally underpins "federated authentication" using Apache [16] validation, modules, "mod_shib for SAML based authentication", and "mod_auth_kerb" meant for "ABFAB based authentication", federated user credentials received are plotted to "OpenStack" roles and clusters for authorization purposes. Figure 1 shows how to access cloud services with ABFAB based validation provided by the OpenStack. In this process user first contacts the keystone and gets the unscoped token. This keystone is protected by the apache server and receives the authentication request and passes it to the authentication module(mod_auth_kerb). Then a legacy EAP authentication is performed between the user and the Identity provider, in this process RADIUS and GSS-EAP is utilized to pass the packets between them over the apache server. As soon as the user is authenticated, initially, the HTTP authentication request is allowed by the apache server to reach the keystone. which thus produces and gives the unscoped token to the user. Lastly, the user can use the issued "token to request " supplementary "scoped" tokens from the keystone. After getting the scoped tokens from the keystone it is submitted to the cloud services to get access.

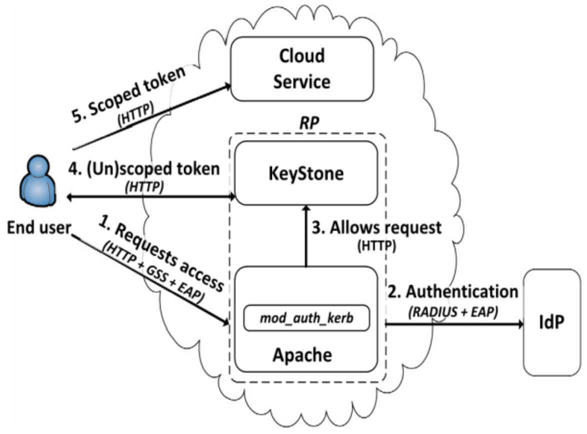

Fig. 1.6.1: Working of Openstack

II. LITERATURE SURVEY

2.1. Related works and comparisons of different AAA mechanisms

In this paper, we are comparing EAP methods with the REAP method. We are comparing the requirements specified in RFC 4017, key properties, and maintenance of certificates. The disadvantage of EAP

MD5 is, it is vulnerable to man-in-the-middle attack and dictionary attack [1]. EAP MD5 will not afford session key generation and mutual authentication.

"EAP-FAST, EAP-PEAP, EAP-TLS, and EAP-TTLS" are the EAP methods based on certificates. These methods mainly rely on certificates. The EAP method known as EAP-FAST provides both session key generation and mutual authentication. EAP-FAST is also not prone to MITM and dictionary attacks. Another EAP method is known as EAP-LEAP is vulnerable to dictionary attacks [2].

User identity protection, During the authentication process user's credentials, should be secured, for this purpose users' credentials are encrypted in the authentication process. To secure users' identity and credentials "EAP methods" like "EAP TLS, EAP TTLS, EAP PEAP, and EAP Fast" establishes a secure tunnel. users' credentials are encrypted and transmitted via the secure tunnel therefore users' identity/credentials are hidden by these EAP methods. User identities/credentials are also protected in EAP TLS and EAP SEM since they use TLS tunnels for hiding the user's identity. EAP-MD5, EAPLEAP, and EAP-SPEKE do not establish secure tunnels so these methods do not hide user identity/credentials.

Urien et al, Badra et al [4], Rescorla [5] et al, and Dierks et al discussed how EAP TLS secure user credentials/identity by tunneling. Moreover, the first EAP method of Juang et al, the EAP method proposed by Park et al, Yoon et al. are prone to the dictionary attack and do not provide hiding of user identity. Hence, their methods do not provide and meet the specification of identity privacy. In REAP user identity is encrypted while exchanging authentication messages so it meets the specification of identity privacy.

Another requirement specified in RFC during authentication is Fast reconnection capability that will reduce the number of round trips which improves the performance. All the EAP methods which are based on Certificates provide fast reconnections capabilities. In these types of EAP methods, the client and server quickly establish a secure connection for communication. Thus reducing the number of round trips or message exchanges when authenticating. EAP-FAST, EAP-SEM, EAP-double- TLS, and EAP-SRP supports "fast reconnections". The REAP method also provides this feature.

The attackers can easily calculate the pre-shared session key at the user side if he gets the long term key. In this way it only gives "half-forward secrecy". "forward secrecy" is not provided by the EAP method MD5. Many methods rely on certificates for providing authentication called Certificate-based EAP methods. However, EAP-TLS entails that all users should apply for the certificates. All users are required to install certificates. This increases the administrative burden and needs to maintaining and manage the certificate and adds additional time for authentication. In symmetric key-based Authentication, the server will authenticate the users by using the shared secrets that reduce authentication time and processing time.

The number of round trips i.e EAP request/response. EAP-MD5 takes two round trips EAP request/response, but it is not capable of providing mutual authentication. EAP-TLS takes Four round trips (EAP request/response)for authentication, but the server and client both are required to install certificates. EAP-TTLS and EAP-PEAP take five and seven round trips (EAP request/response) for authentication, in these methods it is not necessary to install certificates at both server and the client sides, once server authentication is done using TLS handshake, EAP methods like EAP MD5is used to authenticate a client.

In EAP-LEAP it performs (MS-CHAP) "Microsoft Challenge Handshake Authentication Protocol". So, it chooses four round trips. EAP-FAST establishes a secure tunnel by TLS handshake. After the handshake MS-CHAP or One-Time Password (OTP) [5] is used for authentication. So, EAP-FAST takes 5 round trips (EAP request/response) for authentication. REAP method takes only two round trips EAP request/response and satisfies all the requirements.

In "EAP-TLS", "EAP-TTLS", "EAP-PEAP" and "EAP-FAST" both server and client communicate with each other securely using a handshake procedure by performing a key exchange, Furthermore, In "EAP-TLS", "EAP-TTLS", "EAP-PEAP" and "EAP-FAST" user must verify the certificate of the server, In EAP TLS certificate signed by the client, is verified by the server. in EAP-TLS. These processes depend on the asymmetric cryptographic algorithms, such as RSA, DSA, or Diffie-Hellman computations.

In Table 2, we are comparing REAP with other widely used EAP methods for the properties like computation time taken for the handshake, round trips for complete authentication.

In most of the EAP methods the “key exchange algorithm” and “certificate verification” are implemented by Diffie-Hellman and RSA, respectively. AES and SHA-256 algorithms are used to implement the REAP method. Therefore, we are comparing the performance of the REAP method with that of the 2byte DH key agreement and 2byte RSA verification.

The legacy EAP methods use asymmetric cryptographic algorithms such as RSA, DH, for key computation at both the server and the client sides which require more CPU cycles for computing keys and time-consuming also decrease the performance. REAP method uses only symmetric encryption/decryption without asymmetric ones, Therefore, the computation cost of REAP is reduced by depending on different types of EAP methods as associated to the other EAP methods that have the same level of security as REAP.

(REAP) Polynomial based keys are the sequence of keys that are computed using one-time symmetric key cryptography. In this technique, all the messages are encrypted or decrypted by using the sequence of keys generated. Since all the messages are encrypted or decrypted in this technique, if an attacker uses the compromised key, then it can be easily detected. Unlike sharing the keys to entities, in this technique, the keys are generated effortlessly using polynomial expression each time and used for encryption and decryption. In session-based keys generation [6], for each session, separate keys will be generated and exchanged with the entities. In this technique, there is no concept of exchanging the key nor key trade in each session. Both timestamp and polynomial expressions are used to compute/generate the chain of keys used for encryption/decryption. The timestamp is passed as a parameter by the client to server and at both the end, polynomial expressions exist, which is used to compute the sequence of keys to encrypt/decrypt the messages.

That is how our technique utilizes the idea of Polynomial based keys [7] When the succession of Polynomial based keys is spent, another succession of Polynomial based keys is produced by using Polynomial expression. both the server-side and the user side performs the same procedure [8].

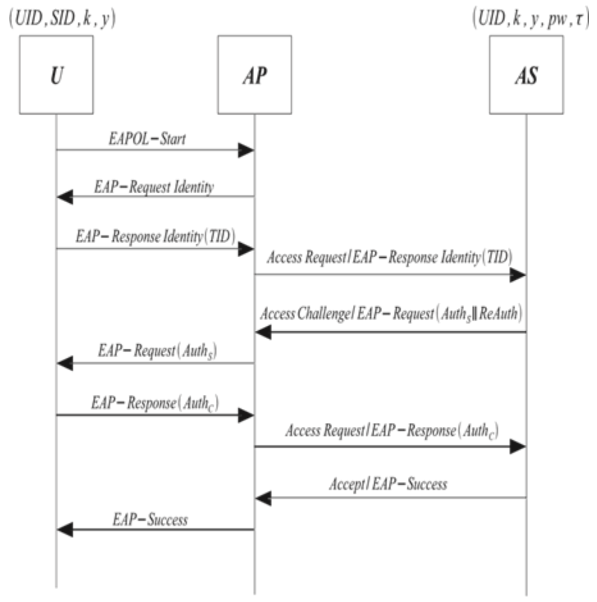
(UID, SID, k, y)
(UID, k, y, pw, τ)
U
AP
AS
EAPOL−Start
EAP−Request Identity
EAP−Response Identity(TID)
Access Request/EAP−Response Identity(TID)
Access Challenge/EAP−Request(Auth_S‖ReAuth)
EAP−Request(Auth_S)
EAP−Response(Auth_C)
Access Request/EAP−Response(Auth_C)
Accept/EAP−Success
EAP−Success

Fig. 2.1: REAP Authentication process

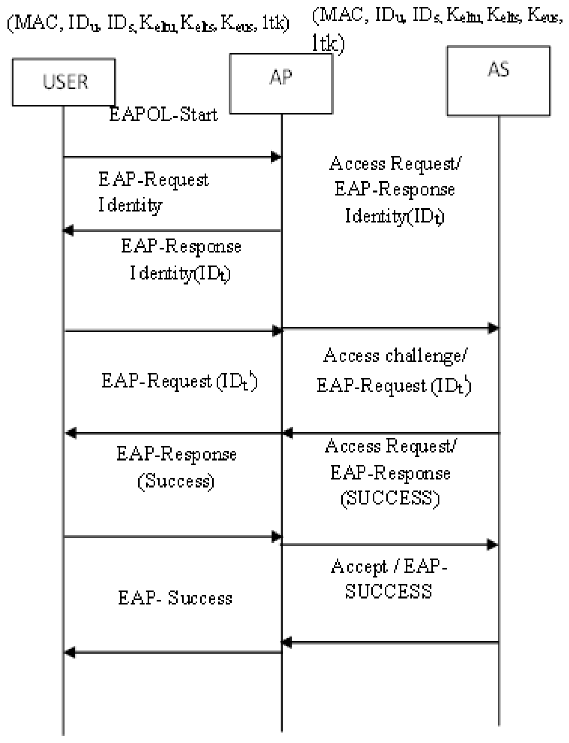
(MAC, IDu, IDs, Kelu, Kels, Keus, 1tk)
(MAC, IDu, IDs, Kelu, Kels, Keus, 1tk)
USER
AP
AS
EAPOL-Start
EAP-Request Identity
Access Request/ EAP-Response Identity(IDt)
EAP-Response Identity(IDt)
EAP-Request (IDt')
Access challenge/ EAP-Request (IDt')
EAP-Response (Success)
Access Request/ EAP-Response (SUCCESS)
Accept / EAP-SUCCESS
EAP- Success

Fig. 2.2: EAP method proposed by [9]

2.1.2 Related works of EAP methods used in cloud scenario for AAA.

In the cloud scenario, the services provided by the cloud will be secured and access control is controlled by the access control mechanisms. Different cloud solutions will use different access control mechanisms and security mechanisms.

For example, OpenStack [20] underpins numerous kinds of credentials for users like "username, password", "certificates x.509" and mechanisms for authentication like "Kerberos" and "LDAP". Through the expanding significance of "identity federations" cloud computing is beginning to show interest for access control as an approach to streamlining client connections to moderate the client the board exertion. OpenStack has incorporated the OS_FEDERATION augmentation [17], which allowed federated clients by performing the assignment of roles and groups all together with cloud access. It also provides a feature of federated access through which an efficient and secure authentication from the industry and R&D for a long time. The advantages of secure and efficient authentication are, it gives regarding expanded convenience for clients (users need not have to present their certificates/credentials every time in the authentication process), too as the eminent decrease on the multifaceted nature of the verification the measure required when accessing to the cloud services have made a significant "Identity Management" (IDM) theme. From the individual perspective, a large portion of the secure, efficient, and SSO-empowered access control mechanisms are just centered around applications of the web. "SAML" [18] and "OAuth" [19] are outstanding models, permitting clients and services of applications to access various resources.

Some other important web services are "OpenID" [20], "OpenID Connect" [21],[22] these have been only essentially framed for applications of web. The current exertion inside the "IETF Kitten WG" [23] to characterize "SAML" Client "SASL" and "GSS-API Mechanisms for non-Web applications" [24] may give more extensive convenience in the coming days.

Kerberos [25] gives a conventional access control convention, in light of the circulation of confirmation tickets, that is broadly upheld by numerous applications & that gives an authentication and "SSO" feature. Even though the standard doesn't especially care about security, there are a few propositions, for example, PrivaKerb [26] or KAMU [27], that give augmentations to supporting improved security. Moreover, Kerberos underpins an activity mode, "Kerberos cross-domain" organizations has not been generally conveyed because of some perceived issues [28], just as to the reality of establishing an autonomous foundation aside those effectively settled for the access to web applications (for example "SAML-based") & access to the network (for example "AAA-based") [29].

Based on the literature survey we have selected 2 authentication methods (EAP-methods) that are secure, efficient, and recent technology. We are comparing these methods with the REAP method integrated to access cloud services. Also, performance analysis and results are discussed below.

III. PROPOSED SYSTEM

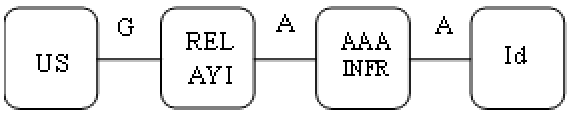

Fig 3.1 Proposed System

The cloud services are provided to the user, for example, SaaS (e.g Openkm) using OpenStack. The "application service supporting ABFAB employing the extension of GSS-EAP". The components in the proposed system are listed below.

- "User" (U). The user is an entity who is intent on using the cloud services provided by the cloud service provider. The user will use his username/password for authentication by the AAA infrastructure and with EAP methods.
- "Identity Provider (IdP)". IdP uses a username/password to verifies the user, and it will act as an "AAA server", known as "IdP in ABFAB".
- "Cloud service/Relying Party (RP)". Also known as RP provides cloud services to the User. GSS-EAP manages access control.

IV. TESTBED CONFIGURATION

In the testbed, the entities are deployed using VM and the entities used in this testbed are listed below.

- "FreeRADIUS" is open-source software that is installed in the VM.
- IdP (RADIUS server). IdP handles the moonshot.test.in the realm and executes a "FreeRADIUS" instance.
- The indicators A and B are measured for the performance analysis, elements A and B are the components: User, RP, and IDP:
- ATC: the time consumed by component A to accomplish the necessary process. Network delays are excluded in this indicator. We have

determined the estimation of this indicator as the total time spent in entity A between the receiving of a message till the transmission of another message.

- ATW: Total time spent by entity A for requests sent and waiting for the responses. It is calculated by the value as the total time spent in element A amongst transmitting the request message till the response message is received for the request.
- ATD(A, B):- The time taken(spent) to deliver the messages between components A and B. ATD(A, B) = ATW(A) − ATC(B) − ATW(B).

  TT:- The total time needed by the user (U) to access the services. TT = ATC(U) + ATD(U, RP) + ATC(RP) + ATD(RP) + ATD(IDP) + ATC(IDP). It is similar to ATC(U) + ATW(U).
- ADB(A, B). Amount of data in bytes that is communicated between elements A and B. the calculation of this indicator is calculated by taking the size of the responses or requests i.e communicated between elements A and B.
- TAT. The total amount of data termed as TAT is communicated in the network for the process of accessing the services. TAT = ADB (U, RP) + ADB(RP) + ADB (IDP).

To measure these indicators, we have used the open-source tool Wireshark. By using this tool, we have captured the packets in the network (network traffic) of all the entities in the process of accessing cloud services. The data in this file are the messages captured during the accessing services. It also provides the size, type of packet (RADIUS), and timestamp for each message that has been sent and received. These files are analyzed by using python scripts that extract and read all the lines and also calculates values for all the indicators with respect to time spent between messages.

V. RESULTS

The results are obtained by executing the following testbed configurations.

- Access to cloud services using the Moonshot implementation.
- Access to cloud services using Moonshot with REAP (EAP method).

In particular, we have executed testbed configuration with legacy EAP method used in moonshot and with configuration using REAP method. Since virtual machines are used to configure our testbed, the challenges and performance considerations that have been mentioned in [24] apply. Before dissecting the outcomes, it should be noted that EAP authentication is only the process of accessing the cloud services (Openstack) other processes are the creation of OpenStack tokens and GSS-API packet encapsulation in keystone messages. So by using "Amdahl's law" [33], the total time to complete the entire process of cloud services accessing won't be relatively influenced by the decreases we bring into the EAP authentication. Alternately, the general decrease will be restricted by the total time spent in the EAP authentication.

Testbed Configuration	Moonshot using legacy EAP method [7]	Moonshot using efficient EAP method proposed in [9]	Moonshot using REAP
ATC(U)	142.72 ms	130.94ms	127.43 ms
ATD(U, RP)	408.38 ms	374.66 ms	364.63 ms
ATC(RP)	206.92 ms	189.83 ms	184.75 ms
ATD(RP)	0.44 ms	0.40 ms	0.39 ms
ATD(IDP)	195.27 ms	179.15 ms	174.35 ms
ATC(IDP)	2.45 ms	2.25 ms	2.19 ms
Total authentication time taken (TaT)	956.18 ms	877.23 ms	853.73 ms

Table 1: Shows the results based on" time-based performance".

Testbed Configuration	Moonshot using legacy EAP method [7]	Moonshot using efficient EAP method proposed in [9].	Moonshot using REAP
ATB(U,RP)	18097.88 ms	16603.56 ms	16158.83 ms
ATB(RP,IDP)	2233.68 ms	2049.25 ms	1994.36 ms
TAT	22594.72 ms	18652.81 ms	18153.18 ms

Table 2: Results obtained based on "data-based performance indicators".

In Table 1, both Moonshot with legacy EAP method and the Moonshot with REAP configurations shows the overall time (TT) authentication time of the 3

methods discussed above. The time required for all three methods is shown in table 1. In the initial authentication process, it is required to execute the complete authentication process "(EAP authentication) with the identity provider". Execution of the authentication method (REAP authentication process) reduces the authentication time to 854ms approximately 11% compared with the legacy authentication method. The reduction of overall time includes processing time and accessing the OpenStack cloud services. If we consider the "total time spent" in the "AAA infrastructure (i.e. ATD(IDP) + ATC(IDP))", we can observe the reduction by using the REAP authentication method.
From Table 1 we can observe that where these reductions1 are obtained. The 2 authentication methods are quite similar if we consider the computational times (ATC). In Table 2 we can observe that by the use of the REAP authentication method the total amount of authentication data transmitted (TAT) is reduced by 32% and 18% respectively. 11

VI. CONCLUSIONS

The proposed work expects to give security and an optimized / efficient authentication process to access (SaaS) cloud service and also in the situations of federated access control environments where ABFAB is used, reducing the authentication data traffic. Also, we have actualized a proof-of-idea model that shows that our proposition can be effectively used to decrease the authentication time needed to access the cloud service (SaaS) implemented using OpenStack. At last, we have done a performance investigation to look at our evidence of-idea executed utilizing the ABFAB arrangement. After obtaining the result by the performance analysis, the work proposed will be able to reduce the authentication time by the use of the REAP method and to access the cloud services around 9 to 11%. Regarding the aggregate sum of network traffic, the decrease is fundamentally the same as, giving a general decrease of 18% if just the traffic on the AAA infrastructure is thought of. In future work, we try to implement the proposed method in Dockers and containers environment." 1.

REFERENCES

[1] Arash Habibi Lashkari, F. Towhidi, R. S. Hoseini, "Wired Equivalent Privacy (WEP)", ICFCC Kuala Lumpur Conference, 2018.

[2] Dictionary Attack on Cisco LEAP, http://www.cisco.com/warp/public/707/cisco-sn-20030802-leap.shtml, 2018.

[3] M. Badra and P. Urien, "Adding Client Identity Protection to EAP-TLS SmartCards," Proc. IEEE Wireless Comm. and Networking Conf., 2017.

[4] T. Dierks and E. Rescorla, "The TLS Protocol Version 1.2," RFC 5246, Aug. 2015.

[5] N. Cam-Winget, D. McGrew, J. Salowey, and H. Zhou, "The Flexible Authentication via Secure Tunneling Extensible Authentication Protocol Method (EAP-FAST)," RFC 4851, May 2014.

[6] C. Alexandra, G. Laura, R. Daniel, "A practical analysis of EAP authentication methods," in Proc. 9th Roedunet International Conference (RoEduNet), 2017, pp.31-35

[7] Alejandro, Pérez, Méndez and Rafael Marín López, Gabriel López Millán , "Providing efficient SSO to cloud service access in AAA-based identity federations" Elsevier, Future Generation Computer Systems, 2016.

[8] Chun-I Fan, Member, IEEE, Yi-Hui Lin, and Ruei-Hau Hsu, "Complete EAP Method: User Efficient and Forward Secure Authentication Protocol for IEEE 802.11 Wireless LANs," IEEE transactions on parallel and distributed systems, VOL. 24, NO. 4, APRIL 2016.

[9] P. Calhoun, J. Loughney, Diameter base protocol, in IETF RFC 6733, October 2012.

[10] J. Linn, Generic security service application program interface version 2, in IETF RFC 2743, January 2000.

[11] Application bridging for federated access beyond web (abfab) in IETF Working Group. http://datatracker.ietf.org/wg/abfab/charter/.

[12] S. Cantor, J. Kemp, R. Philpott, and E. Maler (Eds.), Assertions and Protocols for the OASIS Security Assertion Markup Language, SAML v2.0, March 2005.

[13] S. Hartman, J. Howlett, A GSS-API mechanism for the extensible authentication protocol, in IETF RFC 7055, December 2013.

[14] Patches for the libeap, mech_eap, and FreeRadius to introduce ERP support. http://www.um.es/classe/download.html

[15] Raghavendra K, Ramesh B. "Managing the Digital Identity in the cloud: The current scenario", 2015 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT), 2015.

[16] Configuring keystone for federation. http://docs.openstack.org/developer/keystone/configure_federation.html.

[17] S. Cantor, J. Kemp, R. Philpott, and E. Maler (Eds.), Assertions and Protocols for the OASIS Security Assertion Markup Language, SAML v2.0, March 2005.

[18] E. Hammer-Lahav, D. Recordon, D. Hardt, The OAuth 2.0 authorization protocol, October, in RFC 6749, 2012.

[19] Alejandro Pérez Méndez, Rafael Marín López, Gabriel López Millán. "Providing efficient SSO to cloud service access in AAA-based identity federations", Future Generation Computer Systems, 2016.

[20] N. Sakimura, J. Bradley, M. Jones, B. de Medeiros, C. Mortimore, OpenID connect basic client profile 1.0-draft 24, March 2013. http://openid.net/specs/openidconnect-basic-1_0.html.

[21] M. Goodner, A. Nadalin, WS-Federation 1.2, May 2009. http://docs.oasisopenorg/wsfed/federation/v1.2/ws-federation.pdf.

[22] Common Authentication Technology Next Generation (kitten) IETF Working Group.

[23] S. Cantor, S. Josefsson, SAML Enhanced Client SASL and GSS-API Mechanisms,in IETF Internet-Draft, October 2015. draft-IETF-kitten-sasl-saml-ec-14.

[24] C. Neuman, T. Yu, S. Hartman, K. Raeburn, The Kerberos network authentication service (V5), in IETF RFC 4120, July 2005.

[25] F. Pereniguez, R. Marin-Lopez, G. Kambourakis, S. Gritzalis, A.F. Gomez, PrivaKERB: A user privacy framework for Kerberos, Comput. Secur. 30 (6–7) (2011) 446–463. Elsevier.

[26] F. Pere níguez García, R. Marín-López, G. Kambourakis, A. Ruiz-Martínez, S. Gritzalis, A.F. Skarmeta-Gómez, KAMU: providing advanced user privacy in Kerberos multi-domain scenarios, Int. J. Inf. Secur. 12 (6) (2013) 505–525, 2017.

[27] S. Sakane, K. Kamada, S. Zrelli, M. Ishiyama, Problem Statement on the Cross-Realm Operation of Kerberos. in: IETF RFC 5868, May 2010.

[28] Rafael Marín-López, Fernando Pere níguez, Gabriel López, Alejandro Pérez-Méndez, Providing EAP-based Kerberos pre-authentication and advanced authorization for network federations, Comput. Stand. Interfaces 33 (5) (2011) 494–504, 2017.

[29] Alejandro Pérez-Méndez, Fernando Pere níguez-García, Rafael Marín-López, Gabriel López-Millán, Out-of-band federated authentication for Kerberos based on PANA, Comput. Commun. 36 (14) (2013) 1527–1538.

[30] D. Forsberg, Y. Ohba, B. Patil, H. Tschofenig, A. Yegin, Protocol for Carrying Authentication for Network Access (PANA), in IETF RFC 5191, May 2008.

[31] Marisol García-Valls, Tommaso Cucinotta, Chenyang Lu, Challenges in real-time virtualization and predictable cloud computing, J. Syst. Archit. 60 (9) (2014) 726–740.

[32] Gene M. Amdahl, Validity of the single processor approach to achieving large-scale computing capabilities, in Proceedings of the April 18–20, 1967, Spring Joint Computer Conference, AFIPS '67 (Spring), ACM, New York, NY, USA, 1967, pp. 483–485.

[33] Alejandro Pérez Méndez, Gabriel López Millán, Rafael Marín López, David W. Chadwick, Ioram Schechtman Sette. "Integrating an AAA-based federation mechanism for OpenStack-The CLASSe view", Concurrency and Computation: Practice and Experience, 2017.