International Journal of Innovation and Scientific Research
ISSN 2351-8014 Vol. 28 No. 2 Jan. 2017, pp. 93-100

http://www.ijisr.issr-journals.org/

An Improved Cloud Storage Access Control using JAAS on Stack of Kerberos Protocol

Md. Mahmudul Hasan, Utpaul Sarker, and Md. Kislu Noman

Department of Computer Science and Engineering,
Pabna University of Science and Technology,
Pabna, Bangladesh

ABSTRACT: Cloud storage service is increasingly important and beneficial to the internet users. Users around the world enjoy the high performance cloud servers that are reliable, cost-effective and highly available. Communication vulnerability is also rising to destroy the confidentiality and integrity of user data and information while accessing via network. To resolve these attacks, cloud service providers offer gateways of secured access to the storage services. Though the existing systems are in reasonable standard, investigating a better approach towards security is always worthy. In this paper, we propose a two-step access control system based on Kerberos authentication protocol implemented using Java based authentication service, JAAS. This system provides a well-built and more powerful encryption process by applying Kerberos in two steps. Finally, we build a real time access control system that implements our proposed system.

KEYWORDS: Storage security; Kerberos protocol; Ticket granting; RSA encryption; Access control.

1 INTRODUCTION

Cloud computing is an internet based emerging platform that provides many services in diverse areas of human day-to-day utility. In recent times, internet based services becomes vital and viral that we directly depend on to store sensitive information in various storage services including social networks such as Facebook, Twitter, YouTube and so on. According to reports [1] [2], it has been observed that the involvement of internet users has grown rapidly over years. In a span of 6 years (2010 – 2016), the amount of active users has risen from 400 million to 1.27 billion in Facebook and 30 million to over 317 million in Twitter; YouTube channel views has risen from 120 million to 650 million in this period. Everyday users from these entire social networks store and share numerous amounts of data and content such as photos, word documents, confidential messages etc. Apart from social networks many storage services such as Dropbox, Amazon S3 and EC2, Microsoft Azure, OneCloud and many more services are available in competitive market. With the growing demand of storage services, storage service providers are increasing on the stack. Few of them demand themselves as highly secured services though we cannot guarantee a system is hundred percent secure. Therefore, security measurement and implementation of up-to-date security model is always a matter of concern.

Cloud storage is very useful system to remove the need of maintaining hardware resources such as tape or disk storage device at onsite locations because of its strong ability to increase storage capacity easily when we require it. From the significance of using cloud storage one can centralize data access for better recovery capabilities and can administer day to day storage tasks with ease. Storage security involves configuring a group of security parameters to allow authorized users accessing storage facility via trusted networks. It can also be applied to hardware, programming communication protocols and organization policy. There are two important criteria that can help in determining the effectiveness of a storage security methodology: i) less costly secured system based on the cost involves in users data, ii) discourage hackers by letting them understand that the time and money spent on hacking does not worth much more than the value of the information they are looking for.

Corresponding Author: Md. Mahmudul Hasan

2 BACKGROUND

The basic foundation of cloud enables users to store information in cloud storage instead of using local storage of owner's physical machines. In the process of storing data, several key security components are involved such as security related to application, data transmission, data storage, and using third party resources [3]. Several studies have been conducted on each of these components. In this paper, we concentrate on providing security in application level, access control system in particular. Cloud Service Providers (CSP) offers various application programming interfaces (APIs) and access controls, Cloud Infrastructure Management Interface (CIMI) for example [4], to administer storage tasks via an authentication and authorization process. Each of these processes requires suitable protocols for transferring data securely over internet. One of the popular and cryptographically strong protocols is Kerberos [5] that has been used in many studies for this purpose. In Kerberos, a Ticket Granting Server (TGS) manages service tickets based on the user credentials and validity period which is later used to access the service. While accessing the service, however, there might be another tap in the network as there is no further checking of the validity of the service ticket to CSP server.

In this study, we propose another Kerberos step on stack of existing one where the CSP keeps track of the security measure generated by TGS and validate the service ticket by its own security measures along with TGS measures. Therefore, we implement a two-step Kerberos authentication: first step is for the authentication that Kerberos follows for generating service ticket; the second step confirms the validity of the service ticket used in the service provider application. We consider a cloud file server as a CSP for this purpose. We also focus on security concern on multiple machines in private network. In private network, the machines are independent and so as the users. We attempt to restrict users to use multiple machines for accessing cloud storage. For this purpose, we use hardware address instead of standard client id along with client credentials.

In the remainder of this paper, we present the background theory on cloud storage security and access control in Section 3. Section 4 describes the two-step Kerberos authentication process followed by the implementation of the proposed system in Section 5. The results and discussions are elaborated in Section 6 and Section 8 concludes the paper with system limitations and future recommendations in Section 7.

3 CLOUD DATA STORAGE SECURITY AND ACCESS CONTROL

3.1 CLOUD STORAGE SECURITY

Cloud storage is a collection of plenty of storage devices grouped by network, distributed file systems and other storage middleware to provide scalable, elastic and cost-effective storage service for users [6][7]. The architecture of cloud storage consists of data owner, end user, cloud server (CS), and Third Party Agencies (TPA) [8]. Some of the giant cloud storage providers are Google file system (GFS), Hadoop distributed file system (HDFS) and Amazon Simple Storage Service (S3) [9] that are often utilized by other major storage service providers such as Dropbox [10]. Cloud-storage providers offer users easy, clean and simple file-system interfaces that hide underlying complexities of hardware management [11], software, and personnel maintenance [12]. In addition, it provides flexibility, highly availability on-demand universal data access disregard the geographical locations [8].

A cloud storage service is said to be secured if the overall system maintains the confidentiality and integrity of the storage data [13] [14]. This also includes certification, authority, audit and encryption [6]. Typical security measures can be operated on hardware, software applications, data, information and very often, the network channels [6]. To maintain the security properties several protocols have been used in different studies. Kerberos is one kind of network authentication protocol proposed by MIT that is designed to provide strong authentication for client-server applications by using secret-key cryptography [15].

The client-server Kerberos authentication protocol consists of three main components: the client (user-client machine), the Key Distribution Center (KDC), and the server (rendering the services) [16]. The procedure of this protocol involves a ticket granting system (TGS) that provides service to the client to allow them to establish a secure connection to the server by verifying their identity over a period of time. The KDC is responsible for issuing and checking tickets for clients to be granted to the server. Once the TGS verifies the identity of the client it provides service ticket or credentials to login to the CSP server. We propose another layer of security over the service ticket that is maintained by the CSP server cooperated with TGS.

3.2 ACCESS CONTROL

An access control system allows flexibility in specifying differential access rights of individual users and privileges data consumers to access data relevant to their identity. In cloud, the data owner does not have direct access to the data or file because the files are not stored in owners local machine. For the purpose, CSP offers access control of data stored on untrusted cloud servers by encrypting data through a particular cryptographic formula and exposing decryption keys only to authorized users [17].

The access control system is initiated at the data consumer end where they provide their credentials to be authenticated in remote cloud server. The credentials travel through a secure channel using various protocols to the central authentication server. After verification users are allowed to access the CSP server. JAAS is a Java based an authentication and authorization service that helps to build the login module in access control system [18]. It provides the platform for Kerberos protocol to be implemented in access control system.

4 PROPOSED SYSTEM

We propose a two-step Kerberos authentication process to build more secure access control system for cloud storage service. At first step, we implement the basic Kerberos authentication process using JAAS login module and JGSS (Java Generic Security Service Application) authorization module. This step involves the activities of KDC architecture that includes the user, Authentication Server (AS) and ticket granting center or service, TGS. The second step involves granting access to CSP file server by cooperating with TGS. Fig. 1 presents the high level architecture of the proposed two-step access control system. It is noted that the users are required to create accounts in the AS with their identity such as email id and password. During the registration process, the email and password is hashed using MD5 hashing algorithm [19] and stored in a central database linked to the AS.

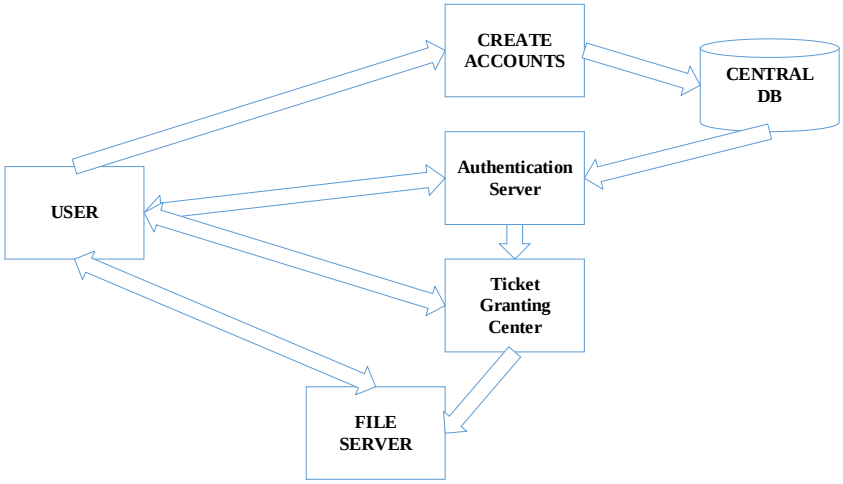


Fig. 1. Proposed Two-step Access Control System

4.1 KDC ARCHITECTURE

KDC has two essential parts: AS and TGS. Users at client site are connected to both of these server parts. To initiate the authentication process, user sends credentials (email id and password) to AS. Prior to travelling to AS the JAAS login module create the hash values of email id and password. The AS verifies the user credentials in the database and issues two messages, A and B, if the verification is successful; one is a TGS session key ($K_{c,tgs}$) which is the public key for TGS, and another is a ticket-granting ticket (TGT), $T_{c,s}$. In addition, the TGS generate a private or secret key (K_{tgs}) and the AS generates a client secret key (K_c) using user's credentials. The TGT includes hardware address, client network address, ticket validity period or timestamp, and $K_{c,tgs}$. It is noted that, instead of client id we use the hardware addresses such as machine's motherboard address, RAM address, hard drive address etc to provide more security in terms of device usability.

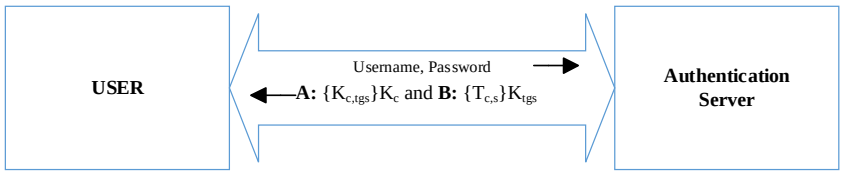


Fig. 2. Passing Ticket Granting Ticket to User

The TGT is now encrypted using TGS secret key and the TGS session key is encrypted using client secret key. The encryption message can be expressed as follows-

$T_{c,s}$ = [hardware address, IP address, timestamp, $K_{c,tgs}$]

A: { $K_{c,tgs}$ } K_c and *B*: {Tc,s} K_{tgs}

These two messages, *A* and *B* is passed through the user (presented in Fig. 2) for retrieving the TGS session key and encrypting new message content of timestamp value and the hardware address. In this process, at first the user decrypts the message *A* using her secret key K_c and retrieve the TGS session key, $K_{c,tgs}$. It is noted that, message *B* is still not be decrypted as the message is encrypted by the TGS secret, that means only TGS can decrypt the message. The next phase is to verify user in TGS server using the TGS session key and encrypted message *B*.

4.2 User Verification

In this phase, user sends two messages, *C* and *D,* to TGS. The message *C* carries hardware address, IP address and timestamp encrypted using $K_{c,tgs}$ that we obtained in last phase. The message *D* is literally the message *B* in last phase. The encrypted messages can be expressed as follows-

A_c = [hardware address, IP address, timestamp]

C: {A_c} $K_{c,tgs}$ and *D*: {$T_{c,s}$} K_{tgs}

Once these two messages travel to TGS, the server decrypts the message *D* using its secret key K_{tgs}. Now, the TGS has previously sent encrypted message $T_{c,s}$ which contains hardware address, IP and timestamp. TGS is also able to decrypt message *C* using its session key $K_{c,tgs}$. That means TGS has the new timestamp and other parameters. TGS now checks the hardware address, IP address and validity period between messages *C* and *D*. The message passing process is presented in Fig. 3. If the validity period does not expire and the hardware and IP addresses are identical in each message then TGS issues a service ticket like the traditional Kerberos system does. We propose a second step encryption in this stage at CSP server to encrypt the service ticket again which is the client server session key. The next subsection describes the second step of our proposed model.

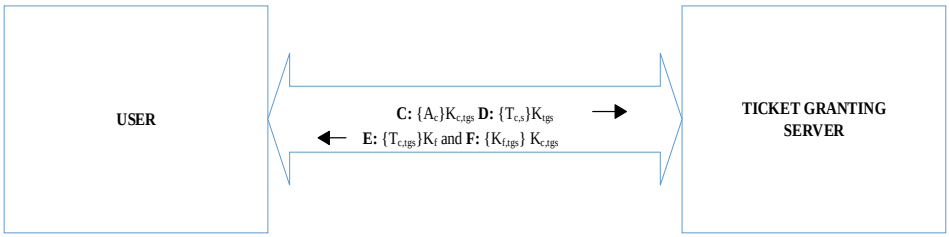


Fig. 3. Communication between User and TGS

4.3 Second Step Encryption

In this step, TGS generate new keys for file server: one is the file server secret key, K_f and another is file server public key or session key $K_{f,tgs}$. Again, TGS sends two messages, *E* and *F* to user for further checking prior to provide access to file server. Message *E* is generated by encrypting client information along with client-server session key, $T_{c,tgs}$, resides in TGS. This encryption is done by the file server secret key. In message *F*, the file server session key is encrypted by TGS session key. The communication is shown in Fig. 3. The encrypted message can be expressed as follows-

$T_{c,tgs}$=[hardware address, IP address, timestamp, client server session key]

E: $\{T_{c,tgs}\}$ K_f and *F*: $\{K_{f,tgs}\}$ $K_{c,tgs}$

Once the messages have been transferred to the user end, the user decrypts the message *F* using the TGS session key but unable to decrypt the message *E* because she does not have the file server secret key. At this point, user follows the same process for providing client information along with the encrypted message to file server. We assume the messages are *G* and *H*. They can be expressed as follows-

$T_{c,f}$ =[hardware address, IP address, timestamp]

G: $\{T_{c,tgs}\}$ K_f and *H*: $\{ T_{c,f} \}$ $K_{f,tgs}$

When user sends these messages back to the file server, server can decrypt the client information using its session key for message *H*. Using the file server secret key the *G* message is decrypted. If the client information from message *G* and *H* are identical then file server issues a true message to the user that is encrypted by file server session key. User has the server session key and easily decrypts the message. The communication process is presented in Fig. 4. Based on the true message, the client issues service request to the file server and operate further operations. In this process, it is guaranteed that there is no chance to lose the authentication data or be tapped in anywhere of the network.

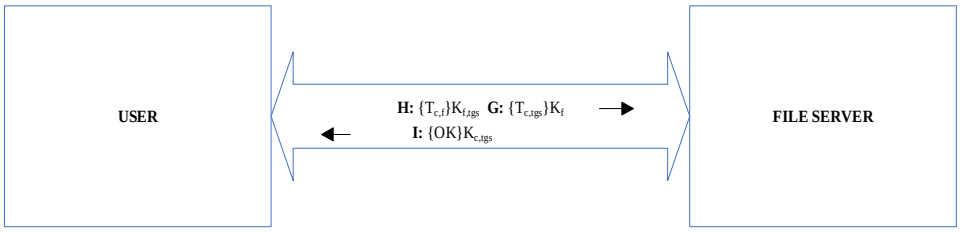


Fig. 4. Communication between User and File Server

5 SYSTEM IMPLEMENTATION

5.1 DEVELOPMENT ENVIRONMENT

We build a Java based access control system where the Kerberos protocol has been implemented using JAAS in login module. We use 64-bit JDK, version-8 and Eclipse IDE (Integrated Development Environment) for writing the source code. The version *Luna* has been used because it supports WindowBuilder for using bi-directional Java GUI (Graphics User Interface) design. We use light weight SQLite database as the central database for storing user information.

5.2 USER REGISTRATION AND LOGIN

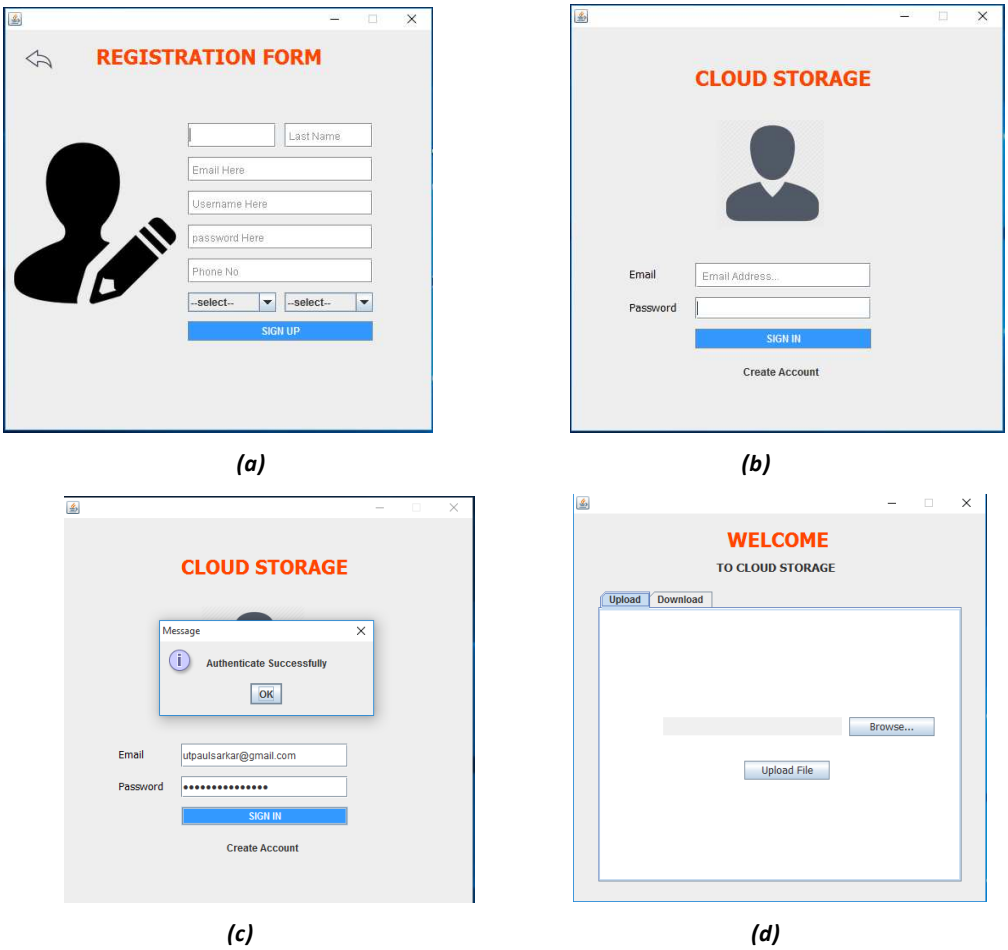
We design and develop a registration module and implemented login module using JAAS. Fig. 5(a) shows the registration and Fig. 5(b) shows the login forms for the access control system we build in this study. User can enter their first and last name, email id, username, password, phone number, gender (left dropdown) and country (right dropdown). Once they enter and submit information the email id and password are hashed and stored into the central database along with other information.

The login screen allows us to provide the credentials for the access control system. Once the sign in button is clicked the Kerberos operation is initiated and once completed it shows the success message on the screen (in Fig. 5(c)). After receiving the grant for accessing the file server user can view, upload and download files from the corresponding GUIs (in Fig. 5(d)). The figure Fig. 5(e) displays the Logcat or console response of ticket granting ticket data and aother security related information through the process.

5.3 UPLOAD AND DOWNLOAD FILES

We design and develop GUIs for uploading to and downloading data from the file server. Fig. 6 consolidates the request and response functionality via the upload and download forms. There are two tabs named ‘Upload’ and ‘Download’ in the

welcome screen. To upload data, we click browse button to select files from the explorer. The interactive screens show the uploading progress and the completion. Similarly, for downloading we choose file from the list of uploaded files; we click on the files to download to our personal computer.



(a) *(b)*

(c) *(d)*

```
javax.crypto.spec.SecretKeySpec@17c3a
Plantext in Client sent to AS:CNU1284P8K-PCBZT00QE0ZABT-110629PCK404GLHVX8XK-00000000-192.168.0.104
1479648514036
Shared key in E in AS: 766712519043816200492392186994751574539382869793680930326213952749948360282021154216667852979825532400874341875516731563195943975484949131 9
Shared key in N is AS: 142810006885692014759029541691854256183538541226592999450903363773765737024361376754400092863588103684090158987107035416604958313250574466 7
Key value is:  PO3HmVN2MUDC5401H2p3gQ==
javax.crypto.spec.SecretKeySpec@17c3a
Encrypted E in bytes: [B@79fe7647
Encrypted N in bytes: [B@727f4df4
Hardware Address in AS: CNU1284P8K-PCBZT00QE0ZABT-110629PCK404GLHVX8XK-00000000-192.168.0.104-1479648514036
e in client: [B@79fe7647
n in client: [B@727f4df4
TGS in client in Bytes: [B@150eeb21
Getting key form clientsite :  PO3HmVN2MUDC5401H2p3gQ==
Getting key form clientsite :  e91cb7d27406afb70ec2f0fea6fa8d5a
orginalKey in clientsite: javax.crypto.spec.SecretKeySpec@17c3a
Decrypt String in N clientsite: 142810006885692014759029541691854256183538541226592999450903363773765737024361376754400092863588103684090158987107035416604958313 2
Decrypt String in E clientsite: 766712519043816200492392186994751574539382869793680930326213952749948360282021154216667852979825532400874341875516731563195943975 4
BigInteger E value in Clientsite: 766712519043816200492392186994751574539382869793680930326213952749948360282021154216667852979825532400874341875516731563195943975 7
BigInteger N value in Clientsite: 142810006885692014759029541691854256183538541226592999450903363773765737024361376754400092863588103684090158987107035416604958315 1
CNU1284P8K-PCBZT00QE0ZABT-110629PCK404GLHVX8XK-00000000
TimeStamp in Clientsite: 1479648514657
Client plantext is: CNU1284P8K-PCBZT00QE0ZABT-110629PCK404GLHVX8XK-00000000-192.168.0.104-1479648514657-e91cb7d27406afb70ec2f0fea6fa8d5a
Bytes value in Plantext: [B@4dfbd9a3
User information in bytes: [B@4dfbd9a3
Ticket from AS: [B@150eeb21
Decrypted String: CNU1284P8K-PCBZT00QE0ZABT-110629PCK404GLHVX8XK-00000000-192.168.0.104-1479648514657-e91cb7d27406afb70ec2f0fea6fa8d5a
Decrypted String Which came from AS: CNU1284P8K-PCBZT00QE0ZABT-110629PCK404GLHVX8XK-00000000-192.168.0.104-1479648514036
CNU1284P8K PCBZT00QE0ZABT 110629PCK404GLHVX8XK 00000000 192.168.0.104 1479648514657
Email: e91cb7d27406afb70ec2f0fea6fa8d5a
Time Difference: 0
1479648514657 1479648514657
TimeStamp in Clientsite: 1479648514735
```

(e)

Fig. 5. ***(a) Registration Form; (b) Login Form; (c) Successful Authentication; (d) Welcome Screen with Upload and Download Features; (e) Logcat Response of the Authentication Process***

Fig. 6. (a) Upload Form; (b) Upload In Progress; (c) Download List; (d) Download Complete;

6 Results and Discussions

The proposed system has two steps to complete the authentication and authorization process. During the authentication, a set of client information is sent to AS. From Fig. 5 (e), we observe the information is sent as plaintext to AS that includes hardware address, timestamp and client IP address. The shared keys are generated at AS and encryption is completed using the client information and shared keys. Few lines down the order in Logcat, we see that the system decrypts the messages at client site and identify the keys sent from the AS. We also check timestamp difference in each step of the authentication process.

The implemented access control system works very well with the authentication process and file service operation. In a private organization, where individual machines are restricted to use by different users, this system works perfectly. For example, a user who is registered from a single machine and login from that machine cannot lose her authentication or nobody in the network can tap the encrypted message from other machine in the same network. In our model, the hardware address is maintained instead of client ID that has been used in previous studies. It restricts the network intruders to provide false hardware address to the server to control anything from their machines.

7 Limitations and Recommendations

Our proposed access control system performs significantly well and provides strong encryption. We use RSA encryption method for this purpose where two keys (public and private keys) contribute in the authentication process. However, the system cannot prevent password guessing attacks. A weak password is always a risk, even though it is not transmitted in clear over the network. The Kerberos protocol itself has recommendation to not use common password for Kerberos and non-Kerberos applications [15]. Also, if KDC is compromised, the entire network can be hacked. So, the KDC must be a dedicated, secured, protected server configured with minimal access.

The proposed system works very well in private organization where machines are restricted to use by individual user. This may be a serious concern as a cloud user which limits anybody to access anything in cloud from anywhere. We recommend using client ID instead of hardware address in this case. But the requirement of the hardware address towards cloud security is not a matter of ignorance as it is important in many organizations.

8 CONCLUSION

Cloud storage service is essentially a distributed file storage system which is used by millions of users over internet now-a-days and the number is increasing. Communication network is always vulnerable and user's identity can be compromised in the network while accessing the cloud services that leads to the loss of confidentiality and data integrity. In this paper, we proposed a two-step Kerberos operation using JAAS and aimed to improve the existing access control system by removing possible attacks during the communication process. We implemented an access control system and observed the encryption and decryption process in real time. The system successfully authenticates and authorizes users to the CSP system. In future, we aim to investigate other cryptographic algorithms such as DES, PBE and BlowFish to use in the access control.

REFERENCES

[1] K. Chard, S. Caton, O. F. Rana and K. Bubendorfer, "Social Cloud: Cloud Computing in Social Networks," *IEEE 3rd International Conference on Cloud Computing*, pp. 99–106, July 5-10, 2010.

[2] Statista, *Monthly active users*, 2016. [Online] Available: https://www.statista.com (October 16, 2016).

[3] S. Subashini and V. Kavitha, "A survey on security issues in service delivery models of cloud computing, " *Journal of network and computer applications*, vol. 34, no. 1, pp. 1-11, July 2011.

[4] D. Davis and G. Pilz, "Cloud Infrastructure Management Interface (CIMI) Model and REST Interface over HTTP," vol. DSP-0263, May 2012, [Online] Available: http://dmtf.org/standards/cloud (October 19, 2016).

[5] B. C. Neuman and T. Ts'o, "Kerberos: An authentication service for computer networks," *IEEE Communications magazine*, vol. 32, no. 9, pp. 33-38, 1994.

[6] W. Zeng, Y. Zhao, K. Ou and W Song, "Research on cloud storage architecture and key technologies," *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human*, pp. 1044-1048, November 24-26, 2009.

[7] A. W. C. S. Zaffos, "Cloud storage: Benefits, risks and cost considerations," *Gartner*, April 2009.

[8] C. Wang, K. Ren, W. Lou and J. Li, "Toward publicly auditable secure cloud data storage services," *IEEE network*, vol 24, no.4, pp. 19-24, 2010.

[9] Y. Hu, H. C. H. Chen, P. P. C. Lee and Y. Tang, "NCCloud: applying network coding for the storage repair in a cloud-of-clouds," *IEEE Transactions on Computers*, vol. 63, no. 1, pp. 31-44, 2014.

[10] I. Drago, M. Mellia, M. M. Munafo and A. Sperotto, "Inside dropbox: understanding personal cloud storage services," *Proceedings of the 2012 ACM conference on Internet measurement conference*, pp. 481-494, November 14 - 16, 2012.

[11] K. D. Bowers, A. Juels and A. Oprea, "HAIL: a high-availability and integrity layer for cloud storage," *Proceedings of the 16th ACM conference on computer and communications security*, pp. 187-198, November 09 - 13, 2009.

[12] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, "Above the Clouds: A Berkeley View of Cloud Computing," *University of California, Berkeley*, Technical Report No. UCB/EECS-2009-28, February 2009.

[13] N. Cao, S. Yu, Z. Yang and W. Lou, "Lt codes-based secure and reliable cloud storage service," *Proceedings IEEE INFOCOM*, pp. 693-701, March 25-30, 2012.

[14] R. A. Popa, J. R. Lorch, D. Molnar and H. J. Wang, "Enabling Security in Cloud Storage SLAs with CloudProof," *USENIX Annual Technical Conference*, vol. 242, 2011.

[15] Kerberos M. I. T. "Kerberos: The Network Authentication Protocol," (2005).

[16] S. Pambalath and AK Pandey, "Connect your apps to DB2 with high-security Kerberos," (2015).

[17] S. Yu, C. Wang, K. Ren and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," *Proceedings IEEE INFOCOM*, pp. 1-9, March 14-19, 2010.

[18] C. Lai, L. Gong, L. Koved and A. Nadalin, "User authentication and authorization in the Java TM platform," *IEEE Proceedings of 15th Annual Computer Security Applications Conference*, pp. 285-290, December 6-10, 1999.

[19] R. Rivest, "The MD5 message-digest algorithm," (1992).