Current Trends in Computer Sciences & Applications

DOI: 10.32474/CTCSA.2019.01.000121

ISSN: 2643-6744

Research Article

Multidimensional: User with File Content and Server's Status Based Authentication for Secure File Operations in Cloud

Jims Marchang[1]*, Jing Wang[1], Abayomi Otebolaku[1], Timi Enamamu[1], Daniel Porter[2] and Benjamin Sanders[3]

[1]*Department of Computing, Sheffield Hallam University, United Kingdom*

[2]*Content Guru Limited, Berkshire, United Kingdom*

[3]*Winchester University, United Kingdom*

***Corresponding author:** Jims Marchang, Department of Computing, Sheffield Hallam University, United Kingdom

Received: November 13, 2019 **Published:** November 22, 2019

Abstract

The popularity of data storage in cloud servers is getting more and more favored in recent times. Its ease of storage, availability and synchronization of personalized cloud file storage using client applications made cloud storage more popular than ever. In cloud storage system, using a basic authentication method like username and password are still one of the most popular forms of authentication. However, the security ensure by such traditional authentication method is weak and vulnerable because the username and password can be compromised by intruders or the user account can be left open by forgetting to logoff in public computers, leading to exposure of information to unauthorized users and hackers. In recent years, using a two-factor authentication has become a trend throughout network-based cloud services, online banking system and any form of services that requires user authentication. Here, in this paper a second layer authentication in the form of session key is used to ensure the authenticity of the activities of the user after user's web-based account is logged-in successfully. The interesting and the critical contribution in this paper is the way the session key is generated and delivers to the authentic user. The key is generated by using the hash value of the file content, file size, file last modified, pseudo-random generated by the server using CPU temperature, clock speed, system time, and network packet timings, and user based 8 digit random position selection from a 32 digit Hex to mitigate against the attacker while performing vital file activities which may lead to data lost or data destruction or when user's credentials are compromised.

Keywords: Cloud file server; session key; un-authorized users; intruders, second factor authentication

Introduction

To use any form of network services (storing, sharing, viewing, downloading, editing, creating, deleting) from a cloud server or a standalone system, authentication of the user is mandatory to validate the authenticity of the user and in-order to protect the information from unauthorized users. Without authentication it will be hard to protect user's data and becomes easier to steal information. Different methods of authentication can be used, such as token-based (card information), biometric-based (fingerprint, retina scan, facial recognition) and knowledge-based (username and password) [1]. In a cloud-based file server, knowledge-based authentication seems to be the most common authentication method adopted. Once the authentication is successful, then the file server is open to anyone who login successfully and the server cannot differentiate between an authentic user and an un-authentic. So, any activities and operations can be performed by the one who has the credentials irrespective of the knowledge of the authenticity of the authentic user. Adopting a mixture and layers of different types of authentication methods can protect the system better from unauthorized users. Some of the most common second factor authentication methods adopted by financial institutions like banks are use of PIN sentry, memorable word, and OTP (One Time Password) [2–5]. Financial organizations like Barclays even uses second or third factor authentication depending on the online activity and the session keys are generated based on factors like knowing user memorable PIN, transaction amount, smart card information, physical card availability etc. using Chip Authentication Program like PIN sentry, but detail internal methods and techniques are not available in public domain due to security reasons and one such security framework for card reader is presented by [2]. In a PIN sentry approach, it reads the card details and using the secret pin, one-time 8-digit pin password is generated to authenticate.

The advantage of this technique is that the card, pin and card reader should all be available at same place and at the same time to generate one-time authentic session key and the key is generated offline. The drawback is that without the card reader it will not work, and the unused session key generated earlier may be usable for future authentication.

It will be inconvenient to use such approach for activities concerning file server system. Memorable words are static in nature for a long period of time unless it's replaced, so it's vulnerable as time passes by. So, a session key based on OTP is more appropriate to use as a second factor authentication concerning the authenticity test of the users when various activities and operations are conducted on files in a cloud server. However, the critical aspect is the method used or adopted for generation of an OTP session key. Some of the critical questions regarding OTP generation are as follows:

a. How unique are the session keys and how they are generated?

b. What is the degree of randomness or degree of correlation between two consecutive session keys?

c. What is the seed or factor or parameter or input source or the method used to generate such session key?

Moreover, there are applications which can be installed in personal laptop or desktop as client for cloud services e.g. Apps for Dropbox, OneDrive, and Google Drive etc. In such situation, the likelihood of an intruder trying to be accessing the cloud client directly through physical laptop or the desktop is minimal because of the lack of physical availability of the devices with the intruder unless the device is stolen, and the login credentials of the personal device is compromised. So, in this study, the second factor authentication is activated only when the cloud server is accessed using web-browser and the current user tries to conduct operation on files e.g. opening a file, deleting a file or downloading or save as etc. The reason why this work doesn't initiate a second factor authentication during the initial login stage is because the real risk is not when someone login, rather the thread is when someone tries to harm or access the content of the files and documents of the user. In this study, it is not only about generating a second factor authentication session key, but it is all about when to generate, what and how to generate is the main aspect of the contribution of this work.

Thus, in this paper, whenever an important file operation like opening a file, deleting a file or downloading a file is to be conducted, the server triggers the user to verify the authenticity of the user by generating and sending a session key to the registered email address or an SMS so that activities of an authorized users are protected from any intruders. If an intruder wanted to change the file name, it does not affect the integrity of the file, so in this study, operation of changing a file name is not considered a critical activity. The file cloud server uses a unique and a novel technique to generate a session key depending on the content of the file (hashing technique) upon which the operation is to be performed, the server's status and user based random input. In this work, email is used as a means of sending the session key. Thus, even if an unauthorized user managed to get through the initial authentication phase of login, the intruder would not be able to steal, temper or alter the file or file content with the proposed approach and the session keys are dependent on computer status based random pseudo random number, the file content, file modification time and file size, and finally the user random input. Thus, the uniqueness and the correlation of the session keys within and across users are more random and even if a same operation is repeated on the same file.

Background study

Over the years, to safeguard intrusion or prevent unauthorized access to user account, multi-factor authentication technique is adopted through different means including use of One Time Password (OTP) as session key or memorable word or use security questions technique which are expected to be known only by the authorize user. The authors of [6] discusses the potential issues with security in relation to cloud computing. It discusses the need for using security measures such as encryption, access control and third-party auditing. Other authors like that of paper [7], identifies the security measures for user authentication. The authors discuss about different attacks such as password discovery, session hijacking, customer fraud attacks and more. In fact, because of using the same password over many authentication platform and online accounts, the password becomes easy to get revealed. So, the authors of [8] proposed a social- network based email system, which allowed users to authentication themselves. The system can let the user change their password on the primary server if it was ever compromised. Different user protection and validation techniques are adopted by various authors. The authors of [9] proposed a secured system using AES based encryption with an asynchronous key system. The work of [10] proposed an authentication method which uses a session key for a second factor authentication. Much like other previous solutions, the session key is randomly generated using the encrypted value of the user ID. However, this approach is potentially unsafe, because if an attacker figures out the seed, which is where the value is being generated from, the intruder can generate their own two-factor session keys and access the user's files. A new method of authentication is designed in [11], the method adds an additional layer of security to the Active Directory Server. However, the method does not consist of any two-factor authentication methods, so if an attacker manages to get victim's password, the whole system would be compromised. There are authors like the one listed in [12] who believes that single authentication technique like using password is safe if the password is selected well and believed that such method is safe from malware and man-in- middle attacks. However, majority of the cloud users don't change their password for a very long time and people tend to use easier password because it's hard to remember complex ones, which makes such approach vulnerable.

An authentication method derived from Diffie-Hellman's RSA model to share an encrypted key is designed in [13]. Nevertheless, it is clear to see how this algorithm would prove useful to implement into authentication for a cloud system. There are other authors like the ones listed in [14] who also derive their work from Diffie-Hellman's RSA model. However, use of a second factor smart card-based authentication has a limitation of availability and access because it will work only when the card is available otherwise access will be blocked. A paper using single sign on protocols using the Kerberos V5 protocol, which works on ticket distribution, is designed in [15]. The work deters attacks to the cloud-based server, however it would not be secure for the end user because of using only a single sign on. However, using the Kerberos V5 protocol would be a user-friendly way to access the cloud-based system. The authors of [16] discusses a multi-factor authentication protocol which initially uses a single logon to authenticate, and the server challenge for a sequence or pattern to authenticate. The server matches the pattern to authenticate. If the pattern considered is the pattern of mouse movement then there is a chance of matching with the intruder's movement pattern, because mouse movement is generally conducted over a limited space. An attribute-based authentication system was proposed in [17] using Quantum Key Distribution. This method uses quantum mechanics to create an irregular secret key. This secret key is known only to the user and the server, so it is expected to be highly secure. The authors of [18,19] designed a secure method for online financial transactions that uses a two-factor authentication method that tracks mouse activity and text typed activity of the user. So, it's easier to identify and differentiate between the user and the intruder. A method that uses a key stroke measurement for user authentication is designed by the authors of [19]. The method works by measuring the duration of holding a key, and each key hold would be a feature to differentiate the users and segregate between the authentic user and the intruder. However, the situation, keyboard type, physical health condition of the user may result in different key holding style and duration and can even lead to authentication failure.

A different authentication technique that uses a hybrid text-image based authentication method is designed in [20]. The method works by showing the user an image and the user types in the word they have associated with the image to authenticate themselves. This is a very innovative way of achieving authentication to a cloud service, however if an attacker gained access to the associated words of the image, then the authentication could be compromised. A two-factor authentication system using biometric authentication technique is designed by [21]. In this technique, a fingerprint is used to grant the user access. This could potentially be a viable solution for a two-factor authentication scenario; however, it is not cost effective and biometrics can be exploited if an attacker gained access to the user's biometric information. A method using a one-time password which generates the password based on the user's local clock is proposed by authors of [22]. However, in such approach clock synchronization could be a potential issue. The authors of [23] designed a mechanism using two-factor authentication method where the user's email is used to pass the session key to authenticate. The user input user's identification number and the key are generated. The server side uses a dynamic token from the hash table in generating the key. The authors of [24] proposed an authentication system which takes advantage of biometric systems. The mechanism proposes a system that uses fingerprint systems to let the user gain access to the system. Biometrics is a very secure and an individual way of authentication, if an attacker gains the biometric information, an attacker could potentially conduct self-authentication. In order to secure a healthcare information over cloud, a linear network coding technique and re-encryption method is used by [25] in the form of hybrid approach. However, no one has taken an approach of using a unique session key which is controlled by the Operating System of the server, hash value of a file and user based random positional selection makes this paper unique from the rest.

The rest of the paper is organized as follows, section 3 covers the detail proposed model, and an analysis and discussion about the proposed model is elaborated in section 4. Qualitative and quantitative analysis are conducted in section 5 and finally the conclusion and future direction are highlighted in section 6.

Proposed Multidimensional Alerting and Authentication System

This paper proposes a multidimensional authentication for a cloud-based file server when performing important file activities like downloading or saving or opening or deleting files from a remote cloud server to protect user's information from intruders. The multi factor authentication is initiated only when user login to the cloud sever via a web-browser rather than client applications installed in a mobile phone or computer. Second factor authentication may not be necessary if file operations are conducted via installed client application in user's device because accessing the device itself needs an authentication and moreover using multi factor authentication for conducting important file operation like deleting or opening etc. in client application installed in user device can be inconvenient, but surely a convenient authentication technique can also be adopted for client application based file activities, but it's not covered in this paper. The main attacks of user data in cloud server will happen online and from anywhere, but it does not rule out possibility of attacks through a client application which is already installed in a user device if the device is stolen or not logout or auto- login is saved or attacks can also be carried through VPN or through remote login etc. The designed trustable secure file system is called Multidimensional: User with File Content and Server's status-based Authentication for Secure File Operations in Cloud (M-UFS authentication). The uniqueness of this approach is the way the second factor authentication is generated and adopted in securing the information stored in the cloud server when file operations are conducted only via a web browser. In this method, when a session key is generated, it uses the file content upon which the file operation is to be conducted, file size, file last modified time, system generated pseudo-random and user supplied positional values and makes the system multidimensional for initiating multi-factor authentication.

The intruder could get into the cloud server of an authentic client by having or knowing the login credentials of the user or it could be because the user forgot to logout from public or hot desk computers. If a session is inactive for a certain period, then the session can be terminated and logout from the server, but when credentials are compromised then nothing can be done or when user saved the login credentials. To protect the information stored in the cloud server after an intruder managed to get into the file directory of a client is by invoking a second authentication step, if the intruder tries to perform activities that can harm the user's data e.g. downloading or deleting file. Triggering an alarm or invoking a second layer authentication upon file modification can also be addressed, but in this study, it is assumed that the server does not have installed applications to open or read or view file contents, so such actions will not arise. To view or read or open a file, in this study the file must be downloaded first. Hence, to safeguard files stored in the cloud file server, a secret session key is used to authenticate and allow activities concerning downloading or deleting a file. The main contribution in this paper is the way the server and user combined to generate a novel secret session key from the following factors:

Operating System (OS) based pseudo random number

A pseudo random number is generated based on CPU temperature, clock speed, system time, and network packet timings. One such method is used by executing OS based random generation and the output of the pseudo random code is generated in a hexadecimal value. Interestingly, this value is hard to reproduce, and it is also unpredictable because it all depends on the OS running in the server. The entropy (the degree of randomness in the system) generated for random is different for each device because the entropy is derived from device drivers and noises generated in the computer. An OS based random generating a 32-character hexadecimal value $R_{0\text{-}31}$ is used. The random can take up-to 256 bits of entropy to generate a pseudo-random string.

SHA512 hash value of a file

The second component used as part of the source in generating the novel secret key is the hash value of the file upon which the operation of downloading or deleting is taking place. The hash algorithm used in this model is known as SHA512. The interesting feature about SHA512 is that it generates a fixed 128-digit hexadecimal irrespective of the file type or size for either text or binary file type. In this work, all the 128 digit values are not used, rather to infuse randomness, the 128 digit is divided into four 32 digit segments and then an XOR operation of those four 32 digit Hex segments are conducted and the resultant 32 digit Hex is assigned to $F_{0\text{-}31}$ to align with the 32 digit random values generated i.e. $R_{0\text{-}31}$. In this work, the SHA512 hash checksum is used as a digital signature to ensure the integrity of the file when upload and download activities is carried out. Example: if the content of the file is the title of this paper i.e. "Server State and File Content Based Two-Factor Authentication for Secure Cloud File Servers" then its corresponding hash value of the SHA512 algorithm is a 32-digit hexadecimal i.e. 4 0 5 F 9 7 2 3 2 9 8 8 D 9 B B E F 7 8 E 2 D 2 D D E 1 C 7 2 4 9 C A D E 4 2 D E 4 9 7 A 1 C 4 3 1 E 3 4 9 D A C 1 C A A C A 5 2 2 E 3 3 2 7 7 D C 0 D 2 7 7 D F B 5 0 B F 3 2 E 6 9 5 E F C 8 0 3 C B 4 A 3 7 2 F 1 4 7 0 9 6 7 6 8 3 A E 0 9 0 7 4 D B 9 B 2 (used: https://passwordsgenerator.net/sha512-hash-generator/). The interesting novel aspect of the designed is that the pseudo random generated by OS and the hash value generated from the file is not used directly, rather further an XOR operation is conducted over the two sets of 32-digit Hex values i.e. $R_{0\text{-}31}$ and $F_{0\text{-}31}$ by mixing. The resultant $C_{0\text{-}31}$ is derived by performing XOR with $R_{0-7} \oplus F_{16-23}$, $R_{8-15} \oplus F_{24-31}$, $R_{16-23} \oplus F_{0-7}$ and $R_{24-31} \oplus F_{8-15}$ as shown in Figure 1. An eight-character long session secret key is selected from this 32-character Hex value $C_{0\text{-}31}$ and more detail is elaborated in the next paragraph.

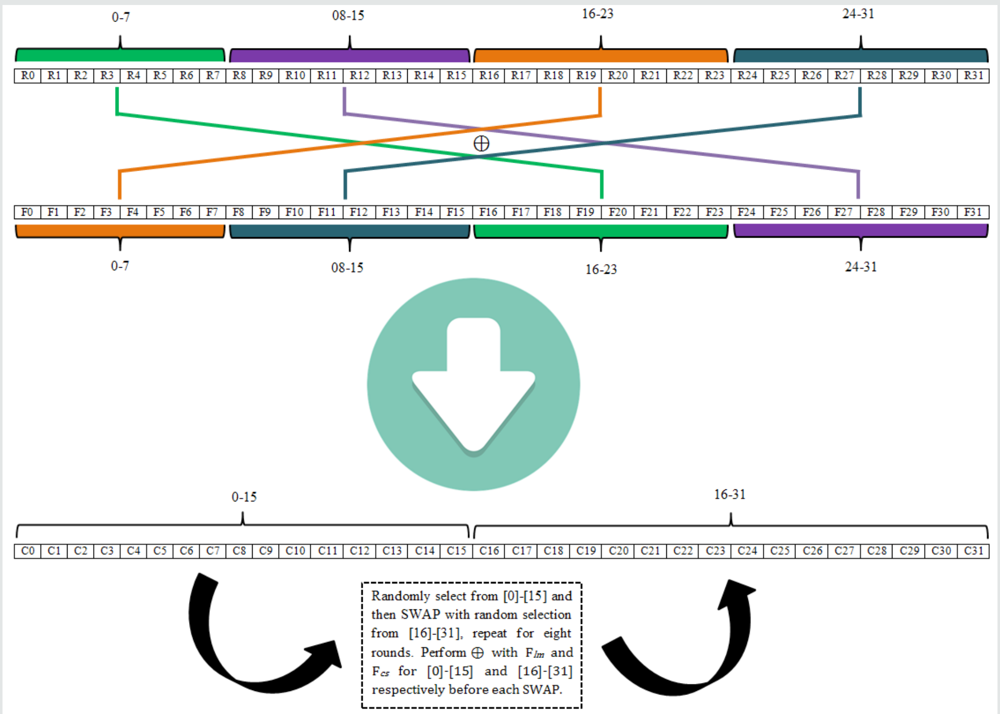
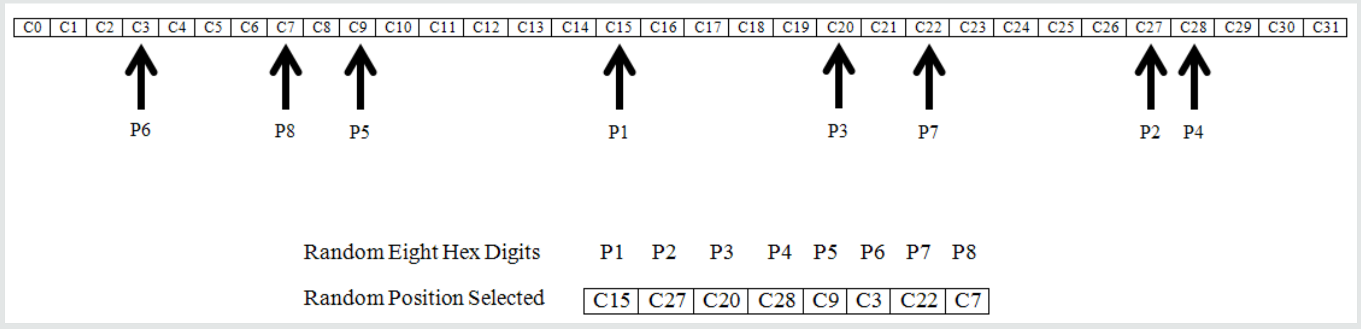

0-7
08-15
16-23
24-31
R0 R1 R2 R3 R4 R5 R6 R7 R8 R9 R10 R11 R12 R13 R14 R15 R16 R17 R18 R19 R20 R21 R22 R23 R24 R25 R26 R27 R28 R29 R30 R31
⊕
F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 F10 F11 F12 F13 F14 F15 F16 F17 F18 F19 F20 F21 F22 F23 F24 F25 F26 F27 F28 F29 F30 F31
0-7
08-15
16-23
24-31
0-15
16-31
C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 C10 C11 C12 C13 C14 C15 C16 C17 C18 C19 C20 C21 C22 C23 C24 C25 C26 C27 C28 C29 C30 C31
Randomly select from [0]-[15] and then SWAP with random selection from [16]-[31], repeat for eight rounds. Perform ⊕ with F_lm and F_cs for [0]-[15] and [16]-[31] respectively before each SWAP.

Figure 1: Digesting using XOR operation between Pseudo-random Hex and File Hash Hex.

Digesting with File Modification Time and Last File Size before swapping positions

The third component is digesting the 32 Hex derived from the system generated random number and the 32 Hex hash values generated from the file content. In this operation using a pseudo random number, an index C_i is selected from $C_{0–15}$ and swapped with C_j which is selected from $C_{16–31}$. This iteration is repeated eight times and in each iteration before swap (C_i, C_j) function is executed; an XOR operation of $C_i \oplus F_{lm}$ and $C_j \oplus F_{cs}$ is calculated. Where, F_{lm} and F_{cs} stands for file last modified time and file size respectively. If a file is editable or viewable then there exists a likelihood of capturing the file modification time, changing file size or file access time and these factors can add to increase the dynamic degree of randomness. In this model, file size and file last modified time are disabled to view but read directly from the system. Moreover, the values of F_{lm} and F_{cs} are converted into single digit Hex value by performing XOR with the Hex digits and its resultant, when the Hex value has more than one digit. Only the micro-second portion of the time of the system is extracted for the last modified time. Then the final generation of the secret session key is detailed in the following section.

User based 8-digit random Hex selections: All the resultant 32 digits of Hex are not used as the session key; rather random 8-digits out of the 32-digits are selected based on the one-time 8 random digit positions supplied by the user during the account creation. These 8 random digit positions need not be remembered by the user, rather these digit positions are remembered by the cloud server for each user. Moreover, each selected position is not directly stored in plaintext, rather it is salted by conducting XOR operation with a master key (M_k), so that the information is not directly visible even at the database level. Thus, the randomness of the selection of the 8- digits out of the 32-digits are not controlled by the server, but its user dependent, which made is harder for other users or an intruder to guess or predict, because digit position selection of one user does not have a correlation with the selection pattern used by other users and the content of the positional values are dynamic. An example of the random selection of the 8-digit during user's registration is given in Figure 2. The approach of selecting the random digit position by a user is adopted, so that the randomness of the selection of the 8 digits Hex is distributed over the N users rather than letting the single server decide for N users after the server has already taken part in generating the 32 digit random Hex based on the OS parameters. In fact, to ensure a higher degree of unpredictability, the random selection of the 8-digit position supplied by the users can be revised and up- dated on the server from time to time. Thus, the Hex position selection from P1 to P8 is conducted in random and the resultant combinations of the corresponding Hex values are used as the secret session key. Since, the 32 digit Hex generated based on the OS running in the system is random, even if an operation is repeated on a same file (same file leads to same hash in SHA512), the resultant 32 digit Hex will not be the same, unless the OS generates a same random Hex for a same file, however such case is very highly unlikely when a 32 digit Hex is considered.


C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 C10 C11 C12 C13 C14 C15 C16 C17 C18 C19 C20 C21 C22 C23 C24 C25 C26 C27 C28 C29 C30 C31
P6
P8
P5
P1
P3
P7
P2
P4
Random Eight Hex Digits P1 P2 P3 P4 P5 P6 P7 P8
Random Position Selected C15 C27 C20 C28 C9 C3 C22 C7
Figure 2: A random selections of the 8-digit positions by the users.

In the model designed in this paper, if a session key is entered over thrice then it can be assumed that the authentic user is either not getting the secret session sent or is received but the user is not aware of the session key being delivered. In such case, since the user is not aware of the situation, the server initiates and invokes a series of security questions which were recorded during the account generation. If the security questions could not be matched, then the user account is logged out and the user is not allowed to login using the current account password until the security questions are answered correctly. To avoid, any form of relay attack, any session key generated by the server is recorded and never allowed to use again irrespective of the user using it or not. In case a session key is decrypted or hijack by an intruder or intercepted by a middleman and wants to perform a relay attack, the key should be un-operational. Moreover, to keep the system communication secure, SSL is used with a mail server "smtp.gmail.com" on port 465. Since, SSL is used, a secured certificate with a key is provided by the server to secure the communication channel, so the session key and any form of communication between the client and the server is secured. The SSL server certificate used for a secure communication was issued by "Let's Encrypt X3" with a serial number: 03 : 2F : DD : 6D : FE : EC : DD : 90 : F1 : A0 : 7B : 6C : 40 : 0E : C5 : 16 : 76 : D0 (It is expired as of now).

This second layer authentication system protects the data stored in the cloud server, even if, the user credentials are compromised, because the intruder would not be able to directly perform any important activity that can harm user's data stored in the cloud server, because to proceed with any vital activities that can lead to harm or lost, the activity has to be authenticated first as shown in Figure 3. In the process, the main aim of this paper is the way how the secret session key is generated. Since the session key is sent to the user email or SMS, if the session key is entered over thrice, then the server suspects an intrusion and challenge the user with security questioned which were set during account creation, so it is expected that the authorized user knows the answers. This challenged is raised by the server to block the intruder and logged the intruder out if the challenged is not reciprocated with the correct answers. However, if the security questions are correct then a new set of session key is generated again and send again to the mail or SMS of registered email or phone number, because the server wants to make sure that the new set of session key generated by the server is entered correctly in the second attempt, even if it has failed to enter the session key correctly within the first three attempts in the earlier round. If the intruder could not enter the correctly session key, even if the intruder knows the security questions, it will not be able to delete or download a file. If the intruder failed to answer the security questions and failed to enter the correct session key, then the server logout the intruder and can ask for another round of security question by reporting the situation to the authentic user asking to reset its credentials. So, even if the authentic user does not notice the session key through his email or SMS, the server makes sure that unauthentic users are not allowed to conduct any damaging act to the user's file even if the login credentials are stolen or compromised. Security questions should be something which only the user knows, but questions like: Mother's maiden name, first school, first job etc. are personal, but these are common, and these kinds of questions are known by many people who knows the user, so it's not safe if the intruder is someone close to the user. So, security questions can be set by self and answer by self rather than supplying by the server. Some of the self-set questions could be something random events but personal which no one else knows, so that it will be hard to predict e.g. What was your first investment company? What was the first foreign food you had? Etc.

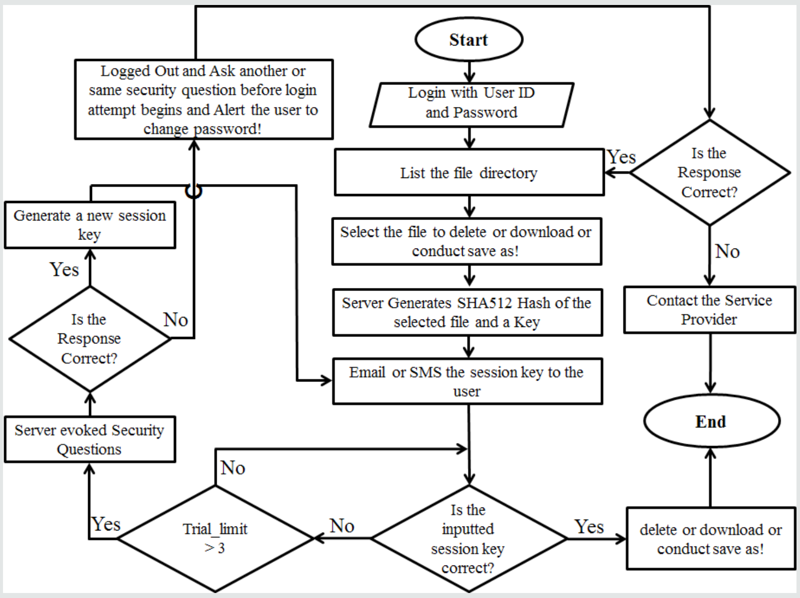

Start
Login with User ID and Password
List the file directory
Select the file to delete or download or conduct save as!
Server Generates SHA512 Hash of the selected file and a Key
Email or SMS the session key to the user
Is the inputted session key correct?
Yes
delete or download or conduct save as!
End
No
Trial_limit > 3
No
Yes
Server evoked Security Questions
Is the Response Correct?
Yes
Generate a new session key
No
Logged Out and Ask another or same security question before login attempt begins and Alert the user to change password!
Is the Response Correct?
Yes
No
Contact the Service Provider

Figure 3: A flowchart to protect the Cloud File Server from an unauthorized user.

Analysis and Discussion

In this secure cloud server system, when important operation like download and delete are to be conducted then the server invokes a secure 8-digit Hex and mail or SMS the registered authentic user. The interesting novelty of the approach is the way those 8-digit Hex values are derived and how the user is informed while the intruder is kept at bay even if the user's credentials are compromised. In the analysis, let's say the following two sets of 32-digit Hex are produced from the OS based random function i.e. R_{0-31} = C 5 C 7 9 D 6 B 7 72 1 6 8 9 E 5 F C 9 5 C 4 1 C D A B 2 0 3 E and the hash value of the file to be downloaded or deleted using SHA512 i.e. F_{0-31} = 6 E B 0 1 4 5 0 E 0 B A 4 2 5 1 7 D 0 3 D 6 B E E 3 4 9 B 4 3 6.

The next step is to perform an XOR operation over the 32 pseudo random Hex generated by the computer based on the OS status and the SHA512 value of the file as highlighted in Figure 1, by converting the Hex values to its corresponding binary values, and the resultant C_{0-31} : B 8 C 4 4 B D 5 9 4 6 8 D C A 8 3 1 7 9 4 8 1 1 2 D 11 6 2 6 F is shown in Figure 4. The XOR operation is conducted over the two sets of 32-digit Hex values i.e. R_{0-31} and F_{0-31} by mixing. The resultant C_{0-31} is derived by performing XOR with $R_{0-7} \oplus F_{16-23}$, $R_{8-15} \oplus F_{24-31}$, $R_{16-23} \oplus F_{0-7}$ and $R_{24-31} \oplus F_{8-15}$. This action of conducting an XOR operation leads to another round of digestion of the random 32-digit Hex generated by the operating system with the 32-digit hash of the file. So, the degree of unpredictable of the session key increases. The intruder neither knows the hash of the file nor knows the random Hex generated by the OS. So, it becomes impossible to guess or predict the resultant of the XOR operation of two unknown Hex values when the lifespan of the session key is short. Since, the file content is not known; it will become extremely hard to guess the corresponding hash value of the file, even if the random Hex generated by the server is known, in reality that is also a challenge because the 32 random Hex is generated based on the status of the CPU temperature, clock speed, system time, and network packet timings. Over a period of time, the CPU temperature, system time and network packet timings keep changing even if the clock speed of the CPU is constant, so the degree of randomness is high. With the help of the XOR binary operation, same resultant value can be derived from two different sets of input and makes less predictable. The session key is generated using this resultant 32-digit Hex. An example of the detail operation of the XOR of the input Hex of R_{0-31} and F_{0-31} is given in Figure 4. So, by looking into the resultant, the input sources cannot be predicted directly or reconstruct easily when XOR operation is used because same Hex result can be constructed from many possible inputs. Example: Binary XOR operation of (1010 ⊕ 0101) = 1111, (0011⊕ 1100) = 1111 and (1001 ⊕ 0110) = 1111.

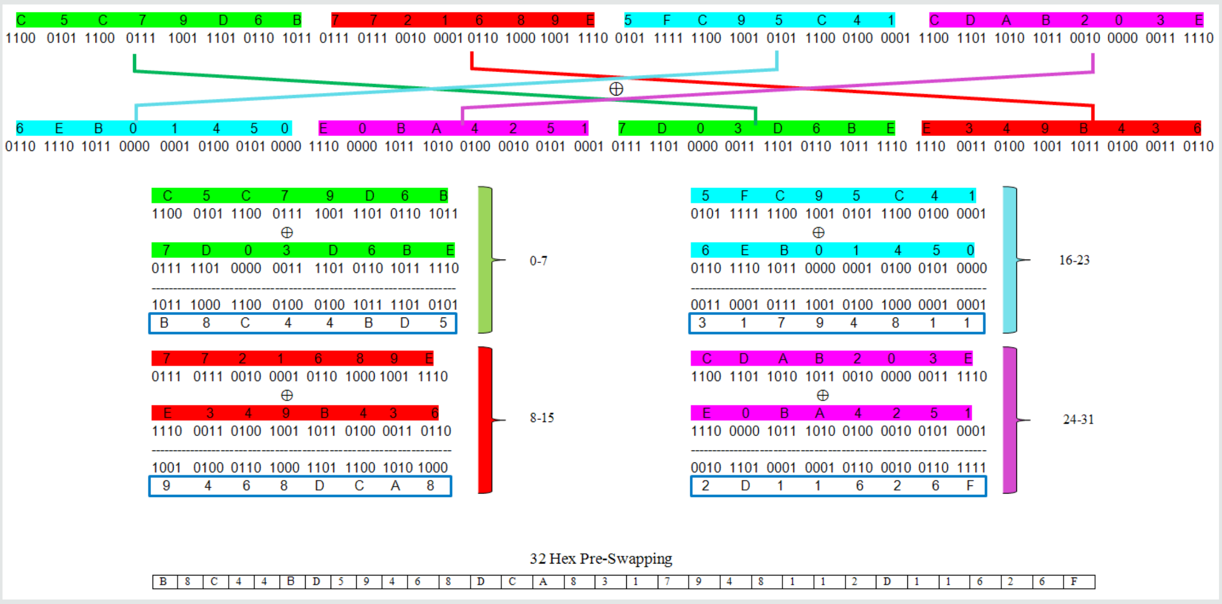


Figure 4: The XOR operation of the 32-digit random Hex and 32-digit F_{0-31} resultant from 128-digit SHA512 file Hex.

Table 1: Digesting of the 32 Hex values based on File Modification Time and File Size.

Iteration. Count	C_i	$F_{lm} \oplus C_i$	C_j	$F_{cs} \oplus C_j$	AFTER Swap(C_i, C_j)
I_1	C[3]=4	A	C[21]=8	B	B8CB4BD59468DCA8 :: 31794A112D11626F
I_2	C[6]=D	3	C[26]=1	2	B8CB4B259468DCA8 :: 31794A112D31626F
I_3	C[0]=B	5	C[17]=1	2	28CB4B259468DCA8 :: 35794A112D31626F
I_4	C[15]=8	6	C[25]=D	E	28CB4B259468DCAE :: 35794A112631626F
I_5	C[2]=C	2	C[18]=7	4	284B4B259468DCAE :: 35294A112631626F
I_6	C[6]=2	C	C[31]=F	C	284B4BC59468DCAE :: 35294A112631626C
I_7	C[5]=B	5	C[17]=5	6	284B46C59468DCAE :: 35294A112631626C
I_8	C[12]=D	3	C[29]=2	1	284B46C594681CAE :: 35294A112631636C

Table 2: The response rates about usability, trust and complexity study.

Computer Knowledge Level	Q1	Q2	Q3	Q4	Q5
$Computer_k$	100%	100%	60%	100%	100%
$Computer_{nk}$	78%	80%	94%	94%	90%

However, this resultant 32 Hex is not the final Hex from where the final eight-digit Hex session key is generated. The 32-digit Hex is further digested by performing XOR using last modified time of a file with randomly selected Hex values from C_{0-15} and XOR with the file size and randomly selected values between C_{16-31} and later the values are swapped. Thus, the 32 Hex Pre-Swapping is divided into two sections i.e. B 8 C 4 4 B D 5 9 4 6 8 D C A 8 and 3 1 7 9 4 8 1 1 2 D 1 1 6 2 6 F for C_{0-15} and C_{16-31} respectively. Thereafter, a random position of C0−15 is selected and XOR with Flm i.e. last file modified time and swapped with one of the digit positions of C_{16-31} after an XOR operation is performed with F_{cs} i.e. last modified file size. Since, random selection is done from C_{0-15} and C_{16-31}, same positions could be selected multiple times, but the values keep changing in each iteration unless the value to be swapped is same with the earlier value as shown in (Table 1). As shown in the (Table 2), the random eight positions elected from C_{0-15} could be C_3, C_6, C_0, C_{15}, C_2, C_6, C_5, and C_{12} and the other eight random selections could be C_{21}, C_{26}, C_{17}, C_{25}, C_{18}, C_{31}, C_{17} and C_{29} from C_{0-31} and performs an XOR with the file last modified time and file size respectively before the Hex values are swapped. Example: F_{lm} = 54324 micro-second and F_{cs} = 3132 bytes, so Hex values of the corresponding decimal for F_{lm} and F_{cs} is D434 and C3C respectively and converting them to a single Hex digit by performing XOR i.e. D⊕4⊕3⊕4 = E and C⊕3⊕C = 3. Thus, before swapping C_i is randomly selected from C_{0-15} where i = 0 to 15 and perform Flm ⊕ Ci operation and then swapped with $F_{cs} \oplus C_j$ where j = 16-31 and C_j is randomly selected from C_{16-31}. The operation is repeated eight rounds and in each round a random index of i and j are selected, so same position can also be selected multiple times. The probability of knowing the right order of picking those eight positions Pe_{0-15} from the 0 to 15 positions of C_{0-15} is given in (1), and the probability of picking the correct eight positions from the C_{16-31} is the same Pe_{16-31} as shown in (2), where a and b =16. So, the probability of correctly predicting both the pair of eight positions is Pe_{0-31} is shown in (3). In this example, the final resultant R is 2 8 4 B 4 6 C 5 9 4 6 8 1 C A E 3 5 2 9 4 A 1 1 2 6 3 1 6 3 6 C from where the 8 digit Hex session key is selected as per the positions provided by each user and the probability of selecting all the eight positions correctly is given by Psk_{0-31} in (4). The probability of selecting all the right eight Hex values after the eight positions of the 32 Hex positions are predicted correctly is given by (5). So, probability of selecting all the eight positions correctly and predicting all the Hex values of each eight positions correctly is given in (6).

$$Pe_{0-15} = \prod_{i=0}^{i=7} \frac{1}{a_i} \qquad (1)$$

$$Pe_{16-31} = \prod_{j=0}^{j=7} \frac{1}{b_j} \qquad (2)$$

$$Pe_{0-31} = \prod_{i=0}^{i=7} \frac{1}{a_i} X \prod_{j=0}^{j=7} \frac{1}{b_j} \qquad (3)$$

$$Psk_{0-31} = \prod_{k=0,s=32}^{k=7} \frac{1}{(s-k)_k} \qquad (4)$$

$$P\upsilon = \prod_{m=0}^{m=7} \frac{1}{h_m} \qquad (5)$$

$$P_r = \prod_{k=0,s=32}^{k=7} \frac{1}{(s-k)_k} X \prod_{m=0}^{m=7} \frac{1}{h_m} \qquad (6)$$

The final session key is not the entire 32-digit Hex key as highlighted before, rather a random eight-digit Hex selected from the resultant 32-digit Hex (R) and the selection of the index positions are provided by user. Thus, the digit positions of the 8-digit Hex are randomly supplied by the user as shown in example of Figure 2 and the random digit positions are communicated to the server once, but it can be updated from time to time. A user can select the 8-digit positions [P1 to P8] from the resultant (R) 32 digit Hex in random (Example: A user can select the index positions R_{15}, R_{27}, R_{20}, R_{28}, R_9, R_3, R_{22}, and R_7 for the eight digit session key Hex i.e. P1, P2, P3, P4, P5, P6, P7, and P8 respectively). The main reason of letting the user allow selecting the session key digit positions is to introduce an element of randomness from different perspective and to deduce the correlation between the session keys by letting each user of the N users to uniquely and independently decide the digit positions and introduce the randomness among the N users rather than the single server deciding the randomness for all the N users. The server has already involved in generating a 32-digit random Hex, so a new dimension of randomness is infused from human perspective. By doing so, the randomness of the session key is derived from the random Hex generated by the server and content of the file in one hand and on the other hand the degree of final randomness is enhanced by letting each user to select their own random digit positions from the available 32 digit R which is the resultant of the 32 digit random Hex generated by the server and the 32 digit Hex of the SHA512 hash of the file content. In this study a master key (M_k) is used to be salted with the selected eight positions provided by the user e.g. (M_k) used is A, 1, 2, F, B, C, E, and 7 for Hex positions P1, P2, P3, P4, P5, P6, P7 and P8 respectively. If the selected eight Hex values are E, 1, 4, 6, 4, B, 1 and 5 then XOR operations of E ⊕ A, 1 ⊕ 1, 4 ⊕ 2, 6 ⊕ F, 4 ⊕ B, B⊕C, 1⊕E, and 5⊕7 are performed to salt the digit positions. In order to get back to the original digit positions of each user, XOR operations are conducted with M_k's corresponding Hex digit values. In an event, an intruder access someone else's account using stolen credentials, the user will be triggered and informed by sending the 8-digit secret session key if the intruder tries to harm or steal information by trying to delete or download a file. So, while the intruder is busy guessing the secret session key, since the channel is secured by SSL communication, the intruder cannot decrypt the secret session key, rather it has to guess the right session key in three attempts to complete the action of downloading or deleting a file. In a meantime, upon receiving the secret session key, the authentic user could reset the user credential as shown in Figure 5 and logout before the intruder could inflict any form of damage to user information stored in the cloud. If the user did not notice the triggered session key delivered through the email or SMS by the server, still the server invokes a series of security questions which were set during account creation after the intruder fails to successfully guess the secured session key within three attempts. It is expected that the intruder would not be able to answer the personalized security questions, so the account would eventually log out. However, if the security questions are guessed correctly by the intruder and the server is requested to initiate a new session key, but still then the session key will be sent to the authentic user and the intruder would still have to guess the secret session key.

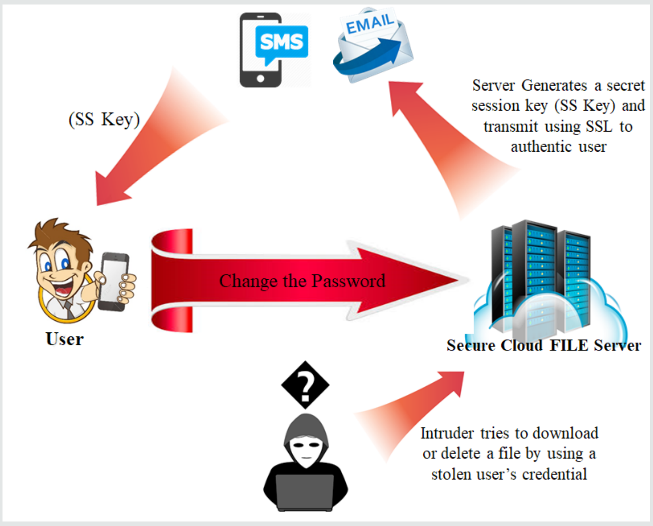

SMS
EMAIL
(SS Key)
Server Generates a secret session key (SS Key) and transmit using SSL to authentic user
User
Change the Password
Secure Cloud FILE Server
Intruder tries to download or delete a file by using a stolen user's credential

Figure 5: User and the server interaction during the generation of the secret session key.

Usability and Trust Analysis of the System

After the designed system is tested, a survey is conducted with five basic questions pertaining to the designed system to understand its usability, trust and complexity. The survey is conducted with fifty (50) individuals who studied computer science or related course ($Computer_k$) and on the other hand it is also surveyed with fifty (50) individuals who never studied computer related course but uses computers and cloud services ($Computer_{nk}$). Moreover, all the individuals who took part in the survey have used or have cloud-based file storage account like Dropbox, OneDrive, and Google Drive etc. In order to capture the context of the usability, trust and complexity analysis of the proposed authentication mechanism the following five questions were asked to the participants i.e.

Q1. Do you know, if your cloud server login credential is compromised, your files are at risk?

Q2. Do you worry if your files stored in the cloud are compromised?

Q3. Is the proposed approach inconvenient to use?

Q4. If the designed authentication system is more secure, then will you compromise about the number of steps involved to secure your data?

Q5. Do you feel safer using the proposed model compared to a single authentication system?

It is interesting to find that that 22% of those who don't study or know about computer security did not realize about the risk exposed to the file's stores in cloud when login credentials are compromised. However, majority of those individuals understand that their files are at risks, if credentials are compromised.

It was 100% clear to those who know about computers that login credentials are the first point of defense to files safety and security. Unlike those who have the knowledge about computer, 20% of those who doesn't know about computers do not worry about the files stored in the cloud servers, it could mainly be due to un-realizing the level of damage or threads which can be caused if files are stolen damage or manipulated etc. It could also be due to the fact that they don't store any valuable personal files, or it could be due to complete lack of awareness about data breach (it could not be captured in this study).

It is obvious that using a simpler security steps like user credentials for authentication are convenient, using a second factor authentication could be inconvenient to an extent. Thus, 60% - 94% of both the parties, who knows about computers and those who do not, thinks that additional steps used in the proposed security model is inconvenient, however 100% of those who knows about computers agrees not to compromise on security despite the requirement of additional steps to secure the cloud file server activity. It is also found that 100% of those who have the knowledge about computers feel safer in using the model of activating a secure activity key during file download or upload or save as operation. Among those who don't have much knowledge about security or computer, 90% still thinks that the proposed model will make the server activity safer even if majority of them feels that the steps added will make it inconvenient to use.

Conclusion and Future Direction

This paper deigned a more secure multi-factor cloud server authenticating technique when file access attempt is made through web-browser when important file operations like deleting, downloading or saving is attempted. This will ensure safety of the files stored over a cloud server when an unauthorized user and intruders tries to damage or access or steal the unauthorized files when user's credentials are compromised or when user forgot to logoff from the system or when user saved the credentials on a system. A secure session key is delivered to the registered email address or phone, if important activities are conducted over the files stored in the cloud.

In this paper, the session key generated by the server to complete any important activities related to files stored in the cloud and it is unique and different from any known session key generation techniques because it is generated based on the hash value of the file upon which the activity is conducted, file size, file last modified time, random value generated the server which is based on the OS (CPU temperature, clock speed, system time, and network packet timings etc.) running in the system and user positional input for key generation. The session key is active for a limited duration, thereafter it expires. If the current user could not supply the correct session key, then the user can be suspected as an intruder and the server initiate other personalized security question-based challenges as a next line of defense. Thus, the server invokes different layers of security to safeguard the files stored in a cloud and the authentic user is informed about the activity with a corresponding secure session key, so that while the intruder is busy trying to guess the session key or the security questions, the authentic user could easily block the intruder by setting new login credentials. Thus, the technique secures the files stored in a cloud server from any unauthorized access.

This work will be further extended with a design for client-based application. This work will also be extended in solving the issue if a same session key is generated twice for the same user at any given point of time.

Compliance with Ethical Standards

Ethical approval: This article does not contain any studies with human participants performed by any of the authors.

(Or) Ethical approval: This article does not contain any studies with animals performed by any of the authors.

(Or) Ethical approval: This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Nilesh A Lal, Salendra Prasad, Mohammed Farik, (2016) A Review of Authentication Methods. International Journal of Scien- Tific & Technology Research Volume 5(11): 246-249.
2. Drimer S, Murdoch SJ, Anderson R (2009) Optimized to fail: Card readers for online banking. In International Conference on Financial Cryptography and Data Security.
3. Kwon T (2001) Authentication and Key Agreement via Memorable Password. In NDSS, India.
4. Kuo C, Romanosky S, Cranor LF, (2006) Human selection of mnemonic phrase-based passwords. In Proceedings of the second symposium on Usable privacy and security.
5. Lee YS, Kim NH, Lim H, Jo H, Lee HJ, (2010) Online banking authentication system using mobile-OTP with QR-code. In Computer Sciences and Convergence Information Technology (ICCIT), International Conference on.
6. Liu Y, Sun YL, Ryoo J, Rizvi S, Vasilakos AV (2015) A Survey of Security and Privacy Challenges in Cloud Computing: Solutions for Future Directions. Journal of Computing Science and Engineering.
7. Dhange M, Sajjan R, Ghorpade V (2017) A Survey on User Authentication Techniques and Attack Taxonomy in Cloud Computing. International Journal of Advanced Research in Computer Science and Software Engineering 7(2): 12.
8. Nicolosi AR (2007) Authentication Mechanisms for Open Distributed Systems. Department of Computer Science New York University, USA.
9. Nafi KW, Kar TS, Hoque SA, Dr Hashem MMA (2012) A Newer User Authentication, File encryption and Distributed Server Based Cloud Computing security architecture. IJACSA 3(10): 181-186.
10. Nayak SK, Mohapatra S, Majhi B (2012) An Improved Mutual Authentcation Framework for Cloud Computing. International Journal of Computer Applications 52(5): 36-41.
11. Venkataramana K, Dr Padmavathamma, M (2012) Agent Based approach for Authentication in Cloud. International Journal of Computer Science and Information Technology & Security.
12. Acar T, Belenkiy M, Alptekin Kupcu (2013) Single Password Authentication.
13. Moghaddam FF, Ghavam I, Varnosfaderani SD, Mobedi S, (2013) A Client-Based User Authentication and Encryption Algorithm for Secure Accessing to Cloud Servers. IEEE Student Conference on Research and Development.
14. Namasudra S, Roy P, (2016) A new secure authentication scheme for cloud computing environment. 29(20).
15. Al-Dubai YF, & Dr Khamitkar SD (2014) Kerberos: Secure Single Sign-on Authentication Protocol Framework for Cloud Access Control. Global Journals Inc 14(1).
16. Pravinbhai MP (2014) Implementation of Multi-tier Authentication Technique for Single-Sign On access of Cloud Services.
17. Madhu Babu BNV, Dr Rajasekhara Rao K (2017) An Attribute based Authentication protocol with Quan- tum key cryptography in cloud servers. International Journal of Computational Intelligence Research 13(5): 907-916.
18. Singh M, Singh S (2012) Design and Implementation of Multi-tier Authentication Scheme in Cloud. International Journal of Computer Science Issues 9(5).
19. Babaeizadeh M, Bakhtiari M, Maarof MA (2014) Authentication Method through Keystrokes Measurement of Mobile users in Cloud Environment. 6(3): 94-112.
20. Popescu DE, Lonea AM (2013) A Hybrid Text-Image Based Authentication for Cloud Services. 8 (1): 263-274.
21. Sasi E, Saranya Priyadarshini R (2015) Secured Biometric Authentication in Cloud Sharing System. 4(3): 572-573.
22. Paranjape V, Pandey V (2013) An Improved Authentication Technique with OTP in Cloud Computing. International Journal of Scientific Research in Computer Science and Engineering 1(6): 22-26.
23. Gujar GV, Sapkal S, Korade MV (2013) STEP 2 - User Authentication for Cloud Computing. International Journal of Engineering and Innovative Technology 2(10).

24. Wang P, Ku CC, Wang TC (2011) A New Finger- print Authentication Scheme Based on Secret-Splitting for Enhanced Cloud Security. Recent Application in Bio- metrics.

25. Modi KJ, Kapadia N (2019) Securing Healthcare In- formation over Cloud Using Hybrid Approach. In: Pan- igrahi C and Pujari A (Eds.), Progress in Advanced Computing and Intelligent Engineering. Advances in Intelligent Systems and Computing, volume 714.

DOI: 10.32474/CTCSA.2019.01.000121