International Journal of Application or Innovation in Engineering & Management (IJAIEM)
Web Site: www.ijaiem.org Email: editor@ijaiem.org
Volume 3, Issue 3, March 2014 ISSN 2319 - 4847

A Review on Third Party Auditing by using KERBEROS System for Secure Cloud Storage

Mr.Ved M. Kshirsagar[1], Prof. V.S.Gulhane[2]

[1]ME (I.T) Department of Information Technology, Sipna College Of Engineering and Technology, Amravati, India. Sant Gadgebaba Amravati University, Amravati,Maharashtra,India - 444602

[2] Professor, Department of (I.T), Sipna College Of Engineering and Technology, Amravati, India
Sant Gadgebaba Amravati University, Amravati, Maharashtra,India - 444602

Abstract

Cloud computing is an environment which enables convenient, efficient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. Cloud is kind of centralized database where many organizations/clients store their data, retrieve data and possibly modify data. Using cloud storage, users can remotely store their data and enjoy the on-demand high-quality applications and services from a shared pool of configurable computing resources, without the burden of local data storage and maintenance. Data stored and retrieved in such a way may not be fully trustworthy so here concept of TPA (Third Party Auditor) is used. Thus, enabling public auditability for cloud storage is of critical importance so that users can resort to a third-party auditor (TPA) to check the integrity of outsourced data and be worry free. To securely introduce an effective TPA, the auditing process should bring in no new vulnerabilities toward user data privacy, and introduce no additional online burden to user. It will be our attempt to further extend the result to enable the TPA to perform audits for multiple users simultaneously and efficiently. Extensive security by applying various encryption algorithms and Kerberos as a third party authentication system shows the proposed schemes are provably secure and highly efficient.

Keywords: Public Auditing, Cloud Computing, Third Party Auditor.

1. Introduction

CLOUD computing has been envisioned as the next generation information technology (IT) architecture for enterprises, due to its long list of unprecedented advantages in the IT history: on-demand self-service, ubiquitous network access, location independent resource pooling, rapid resource elasticity, usage-based pricing and transference of risk. While cloud computing makes these advantages more appealing than ever, it also brings new and challenging security threats toward user's outsourced data. Since cloud service providers (CSP) are separate administrative entities, data outsourcing is actually relinquishing user's ultimate control over the fate of their data. As a result, the correctness of the data in the cloud is being put at risk due to the following reasons [4],[5]. CSP might reclaim storage for monetary reasons by discarding data that have not been or are rarely accessed, or even hide data loss incidents to maintain a reputation. In short, although outsourcing data to the cloud is economically attractive for long-term large-scale storage, it does not immediately offer any guarantee on data integrity and availability. Simply downloading all the data for its integrity verification is not a practical solution [3].It is desirable that cloud only entertains verification request from a single designated party. To fully ensure the data integrity and save the cloud user's computation resources as well as online burden, it is of critical importance to enable public auditing service for cloud data storage, so that users may resort to an independent third-party auditor (TPA) who has expertise and capable to audit the outsourced data when needed. Public auditability allows an external party, in addition to the user himself, to verify the correctness of remotely stored data [6], [12]. This severe drawback greatly affects the security of these protocols in cloud computing. It is an attempt to show the security by applying various techniques and justify the performance of proposed schemes through concrete experiments and comparisons. It is our attempt to provide security to the cloud by just simply using Kerberos systems for public auditability. Specifically, proposed scheme achieves batch auditing where multiple delegated auditing tasks from different users can be performed simultaneously by the TPA in a privacy-preserving manner.

2. LITERATURE REVIEW AND RELATED WORK

G. Ateniese et al. are the first to consider public auditability in their "provable data possession" (PDP) model for ensuring possession of data files on untrusted storages. They utilize the RSA-based homo-morphic linear authenticators for auditing outsourced data and suggest randomly sampling a few blocks of the file. When used directly, their protocol is not provably privacy preserving, and thus may leak user data information to the external auditor [7]. *Juels and B. Kaliski et al.* Describe a "proof of retrievability" (PoR) model, where spot-checking and error-correcting codes are used to ensure both "possession" and "retrievability" of data files on remote archive service systems [9]. Later, *Y. Dodis* ***et al***. also give a study on different variants of PoR with private auditability [13]. *H. Shacham and B. Waters* design an improved PoR

scheme built from BLS signatures with proofs of security in the security model defined. Similar to the construction in, they use publicly verifiable homo-morphic linear authenticators that are built from provably secure BLS signatures [8]. *C. Wang et al.* consider a similar support for partially dynamic data storage in a distributed scenario with additional feature of data error localization [3], [10]. *C. Erway et al.* develop a skip list based scheme to also enable provable data possession with full dynamics support. However, the verification in both protocols requires the linear combination of sampled blocks as an input, like the designs, and thus does not support privacy-preserving auditing [2]. *Schwarz and Miller* propose the first study of checking the integrity of the remotely stored data across multiple distributed servers. Their approach is based on erasure-correcting code and efficient algebraic signatures, which also have the similar aggregation property as the homo-morphic authenticator utilized in our approach. *R. Curtmola et al.* aim to ensure data possession of multiple replicas across the distributed storage system. They extend the PDP scheme in to cover multiple replicas without encoding each replica separately, providing guarantee that multiple copies of data are actually maintained [7].

Privacy Preserving Public Auditing Proposed by Cong Wang

Public auditing allows TPA along with user to check the integrity of the outsourced data stored on a cloud & Privacy Preserving allows TPA to do auditing without requesting for local copy of the data. It contains 4 algorithms as:

1) Key generation: It is a key generation algorithm used by the user to setup the scheme.
2) Sin generation: It is used by the user to generate verification metadata which may include digital signature.
3) Generation Proof: It is used by CS to generate a proof of data storage correctness.
4) Verify proof: Used by TPA to audit the proofs It is divided into two parts as setup phase and audit phase [3],[4],[5].

Using EAP

S. Marium proposed use of Extensible authentication protocol (EAP) through three ways of hand shake with RSA.
They provide an authentication protocol for cloud computing, lightweight and efficient as compared to SSL protocol. Challenge-handshake authentication protocol (CHAP) is used for authentication [11].

2.1 System and threat model

It is considered that a cloud data storage service involving three different entities, as illustrated in Fig.1: the cloud user, who has large amount of data files to be stored in the cloud; the cloud server, which is managed by the cloud service provider to provide data storage service and has significant storage space and computation resources (we will not differentiate CS and CSP hereafter); the third-party auditor, who has expertise and capabilities that cloud users do not have and is trusted to access the cloud storage service reliability on behalf of the user upon request.

Some more problems related with the cloud storage are as follows :

- Correctness of the data is being put at a risk.
- Not offer any guarantee on data integrity and availability.
- Also threat of identity spoofing attack.
- Data tempering attack, repudation attack, Information Disclosure on upload/download attack.
- Denial of service attack.

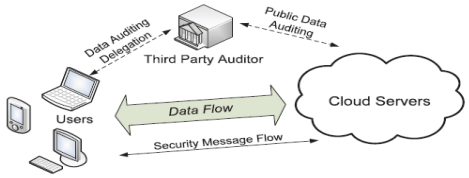


Fig. 1. The architecture of cloud data storage service.

3. PRPOSED WORK

While applying this third party auditor i.e. Kerberos system the network connection is must for exchange or retrieval of the data by the user. To maintain the data integrity of data stored on the cloud some proposed algorithms and encryption techniques are being considered. By applying various authentication methods, the users are authenticated properly and security will be achieved this is the main objective of our proposed work, to make the cloud storage secure.
During the implementation of this proposed work our objectives will be as follows:

- Detection probability against the data modification.
- Authentication of user by using third party authentication.
- Availability of the data.
- Correctness of the data.
- No information leakage of the data.
- No data loss.

With the best level efforts, above one or more tasks or objectives may be tried to be implemented. For these objectives to be achieved we considered the following authentication system and some encryption/decryption protocols explained below.

3.1.1 Kerberos As A Trusted Third Party Auditing / Authentication System protocol

Kerberos is a computer network authentication protocol which works on the basis of 'tickets' to allow nodes communicating over a non-secure network to prove their identity to one another in a secure manner. Its designers aimed it primarily at a client–server model and it provides mutual authentication—both the user and the server verify each other's identity. Kerberos protocol messages are protected against eavesdropping and replay attacks. Kerberos builds on symmetric key cryptography and requires a trusted third party, and optionally may use public-key cryptography during certain phases of authentication.

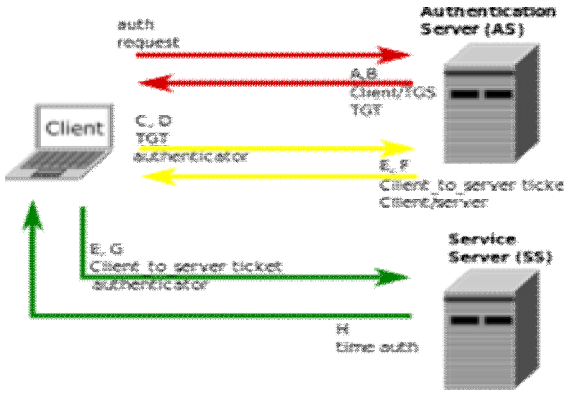


Fig : Working of Third Party Auditing / Authentication (Kerberos)

The protocol is described in detail below.

a) User Client-based Logon

A user enters a username and password on the client machines. Other credential mechanisms allow the use of public keys in place of a password. The client transforms the password into the key of a symmetric cipher. This either uses the built in key scheduling or a one-way hash depending on the cipher-suite used.

Client Authentication

The client sends a clear text message of the user ID to the AS requesting services on behalf of the user. (Note: Neither the secret key nor the password is sent to the AS.) The AS generates the secret key by hashing the password of the user found at the database (e.g. Active Directory in Windows Server).The AS checks to see if the client is in its database. If it is, the AS sends back the following two messages to the client:

- Message A: Client/TGS Session Key encrypted using the secret key of the client/user.
- Message B: Ticket-Granting-Ticket (which includes the client ID, client network address, ticket validity period, and the client/TGS session key) encrypted using the secret key of the TGS.

Once the client receives messages A and B, it attempts to decrypt message A with the secret key generated from the password entered by the user. If the user entered password does not match the password in the AS database, the client's secret key will be different and thus unable to decrypt message A. With a valid password and secret key the client decrypts message A to obtain the Client/TGS Session Key. This session key is used for further communications with the TGS. (Note: The client cannot decrypt Message B, as it is encrypted using TGS's secret key.) At this point, the client has enough information to authenticate itself to the TGS.

b) Client Service Authorization

When requesting services, the client sends the following two messages to the TGS:

- Message C: Composed of the TGT from message B and the ID of the requested service.
- Message D: Authenticator (which is composed of the client ID and the timestamp), encrypted using the *Client/TGS Session Key*.

Upon receiving messages C and D, the TGS retrieves message B out of message C. It decrypts message B using the TGS secret key. This gives it the "client/TGS session key". Using this key, the TGS decrypts message D (Authenticator) and sends the following two messages to the client:

- Message E: *Client-to-server ticket* (which includes the client ID, client network address, validity period and *Client/Server Session Key*) encrypted using the service's secret key.
- Message F: *Client/Server Session Key* encrypted with the *Client/TGS Session Key*.

c) Client Service Request
Upon receiving messages E and F from TGS, the client has enough information to authenticate itself to the SS. The client connects to the SS and sends the following two messages:

- Message E from the previous step (the *client-to-server ticket*, encrypted using service's secret key).
- Message G: a new Authenticator, which includes the client ID, timestamp and is encrypted using *Client/Server Session Key*.
- The SS decrypts the ticket using its own secret key to retrieve the *Client/Server Session Key*. Using the sessions key, SS decrypts the Authenticator and sends the following message to the client to confirm its true identity and willingness to serve the client:
- Message H: the timestamp found in client's Authenticator plus 1, encrypted using the *Client/Server Session Key*.

The client decrypts the confirmation using the *Client/Server Session Key* and checks whether the timestamp is correctly updated. If so, then the client can trust the server and can start issuing service requests to the server.
The server provides the requested services to the client.[14]

3.1.2 RSA Algorithm
RSA is an algorithm for public-key cryptography that is based on the presumed difficulty of factoring large integers, the factoring problem. RSA stands for Ron Rivest, Adi Shamir and Leonard Adleman, who first publicly described the algorithm in 1977. Clifford Cocks, an English mathematician, had developed an equivalent system in 1973, but it wasn't declassified until 1997.

The RSA algorithm involves key generation, encryption and decryption.

RSA involves a **public key** and a **private key.** The public key can be known by everyone and is used for encrypting messages. Messages encrypted with the public key can only be decrypted in a reasonable amount of time using the private key. The keys for the RSA algorithm are generated the following way:

1. Choose two distinct prime numbers p and q.
2. For security purposes, the integer's p and q should be chosen at random, and are of similar bit-length. Prime integers can be efficiently found using a primarily test.
3. Compute $n = p*q$.
4. n is used as the modulus for both the public and private keys. Its length, usually expressed in bits, is the key length
5. Compute $\varphi(n) = \varphi(p)\varphi(q) = (p-1)(q-1)$, where φ is Euler's totient function.
6. Choose an integer e such that $1 < e < \varphi(n)$ and gcd $(e, \varphi(n)) = 1$; **i.e.** e and $\varphi(n)$ are coprime.
- e is released as the public key exponent.
- e having a short bit-length and small Hamming weight results in more efficient encryption – most commonly $2^{16} + 1 = 65{,}537$. However, much smaller values of e (such as 3) have been shown to be less secure in some settings.

7. Determine d as $d^{-1} \equiv e \pmod{\varphi(n)}$, i.e., d is the multiplicative inverse of e (modulo $\varphi(n)$).
- This is more clearly stated as solve for d given $d \cdot e \equiv 1 \pmod{\varphi(n)}$
- This is often computed using the extended Euclidean algorithm.
- d is kept as the private key exponent.

The **public key** consists of the modulus n and the public (or encryption) exponent e. The **private key** consists of the modulus n and the private (or decryption) exponent d, which must be kept secret. p, q, and $\varphi(n)$ must also be kept secret because they can be used to calculate d.[15]

2) A working example
Here is an example of RSA encryption and decryption.
- Choose two distinct prime numbers, such as

$p = 61$ and $q = 53$.

- Compute $n = p*q$ giving

$n = 61 \times 53 = 3233$.

- Compute the totient of the product as $\varphi(n) = (p-1)(q-1)$ giving

$\varphi(3233) = (61-1)(53-1) = 3120.$

- Choose any number $1 < e < 3120$ that is coprime to 3120. Choosing a prime number for e leaves us only to check that e is not a divisor of 3120.

Let $e = 17$.

- Compute d, the modular multiplicative inverse of e (mod $\varphi(n)$) yielding

$d = 2753.$

- The **public key** is (n = 3233, e = 17). For a padded plain text message m, the encryption function is

$c(m) = m^{17} \pmod{3233}.$

- The **private key** is (n = 3233, d = 2753). For an encrypted cipher text c, the decryption function is $c^{2753} \pmod{3233}$.

$m(c) = c^{2753} \pmod{3233}.$

- For instance, in order to encrypt m = 65, we calculate

$c \equiv 65^{17} \pmod{3233} \equiv 2790$

- To decrypt c = 2790, we calculate

$m \equiv 2790^{2753} \pmod{3233} \equiv 65.$

3.1.3 Advanced Encryption Standard (AES) Algorithm

The **Advanced Encryption Standard** (**AES**) is a specification for the encryption of electronic data established by the U.S. National Institute of Standards and Technology (NIST) in 2001. It is based on the Rijndael cipher developed by two Belgian cryptographers, Joan Daemen and Vincent Rijmen, who submitted a proposal to NIST during the AES selection process. Rijndael is a family of ciphers with different key and block sizes. For AES, NIST selected three members of the Rijndael family, each with a block size of 128 bits, but three different key lengths: 128, 192 and 256 bits.

AES has been adopted by the U.S. government and is now used world wide. It supersedes the Data Encryption Standard (DES), which was published in 1977. The algorithm described by AES is a symmetric-key algorithm, meaning the same key is used for both encrypting and decrypting the data.

AES is based on a design principle known as a substitution-permutation network, and is fast in both software and hardware. Unlike its predecessor DES, AES does not use a Feistel network. AES is a variant of Rijndael which has a fixed block size of 128 bits, and a key size of 128, 192, or 256 bits.

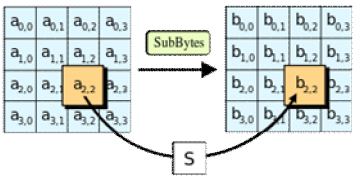
The key size used for an AES cipher specifies the number of repetitions of transformation rounds that convert the input, called the plaintext, into the final output, called the cipher text. The number of cycles of repetition are as follows:

- 10 cycles of repetition for 128-bit keys.
- 12 cycles of repetition for 192-bit keys.
- 14 cycles of repetition for 256-bit keys.

Each round consists of several processing steps, each containing five similar but different stages, including one that depends on the encryption key itself. A set of reverse rounds are applied to transform ciphertext back into the original plaintext using the same encryption key.

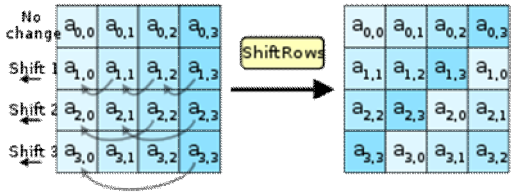
- **Key Expansion—round keys are derived from the cipher key using Rijndael's key schedule. AES requires a separate 128-bit round key block for each round plus one more.**
- Initial Round
- `Add Round Key`—each byte of the state is combined with a block of the round key using bitwise XOR.
- Rounds
  1. `Sub Bytes`—a non-linear substitution step where each byte is replaced with another according to a lookup table.
  2. `Shift Rows`—a transposition step where each row of the state is shifted cyclically a certain number of steps.
  3. `Mix Columns`—a mixing operation which operates on the columns of the state, combining the four bytes in each column.
  4. `Add Round Key`
- Final Round (no `Mix Columns`)
  1. `Sub Bytes`
  2. `Shift Rows`
  3. `Add Round Key`.

3) The Sub Bytes step



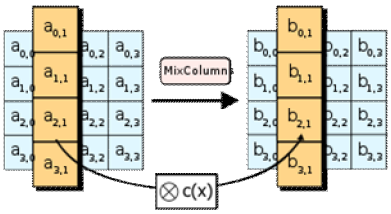
In the `Sub Bytes` step, each byte in the state is replaced with its entry in a fixed 8-bit lookup table, S; $b_{ij} = S(a_{ij})$.

4) The Shift Rows step



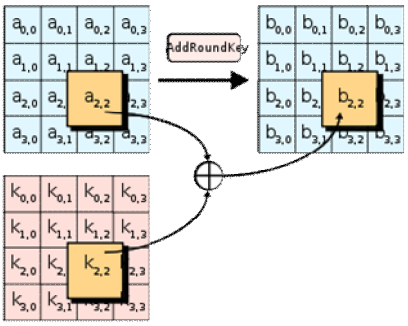
In the `Shift Rows` step, bytes in each row of the state are shifted cyclically to the left. The number of places each byte is shifted differs for each row.

5) The `Mix Columns` step



In the `Mix Columns` step, each column of the state is multiplied with a fixed polynomial $c(x)$.
The `Mix Columns` function takes four bytes as input and outputs four bytes, where each input byte affects all four output bytes. Together with `Shift Rows`, `Mix Columns` provides diffusion in the cipher.

6) The `Add Round Key` step



In the `Add Round Key` step, each byte of the state is combined with a byte of the round sub key using the XOR operation (⊕) [16].
This third party security will be applicable to any organization who wants to secure their data storage and deliberately want to restrict unwanted users to access the data and also provide layering to the database so that layer 1 users only access the data from layer 1 at the data storage. Our attempt is to make such a security policy by which the database is accessible to the proper users of that particular layers.

CONCLUSION

This paper is an attempt to implement such a system that will provide a complete to the cloud storage by applying KERBEROS type of authentication third party and various algorithms for communication between clients, cloud and third party.

References

[1] A. Mohta, Lalit Kumar Awasti,"Cloud Data Security while using Third Party Auditor", International Journal of Scientific & Engineering Research, Volume 3,Issue 6, ISSN 2229-8 June 2012.

[2] C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in Proceedings of the 16th ACM conference on Computer and communications security, ser. CCS '09. New York, NY, USA: ACM, 2009, pp. 213–222

[3] C. Wang, Q. Wang and K. Ren, "Ensuring Data Storage security in Cloud Computing", IEEE Conference Publication, 17th International Workshop on Quality of Service (IWQoS), 2009

[4] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving Public auditing for storage security in cloud computing," in Proc.of IEEE INFOCOM'10, March 2010.

[5] C. Wang, Sherman S. M. Chow, Q. Wang, K. Ren and W. Lou, "Privacy-Preserving Public Auditing for Secure Cloud Storage", IEEE Transaction on Computers I, vol. 62, no. 2, pp.362-375 , February 2013.
[6] D. Shrinivas, "Privacy-Preserving Public Auditing in Cloud Storage security", International Journal of computer science and Information Technologies, vol 2, no. 6, pp.2691-2693, ISSN: 0975-9646, 2011
[7] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in Proceedings of the 14th ACM conference on Computer and communications security, ser. CCS '07. New York, NY, USA: ACM, 2007, pp. 598–609.
[8] H. Shacham and B. Waters, "Compact proofs of retrievability," in Proc. of Asiacrypt 2008, vol. 5350, Dec 2008
[9] Juels, B. Kaliski. "Pors: proofs of retrievability for large files[C]", Proceedings of CCS 2007. Alexandria, VA,USA, 2007. 584-597.
[10] Q. Wang, C. Wang, K. Ren, W. Lou and Jin Li "Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing", IEEE Transaction on Parallel and Distributed System, vol. 22, no. 5, pp. 847 – 859,2011.
[11] S. Marium, Q. Nazir, A. Ahmed, S. Ahthasham and Aamir M. Mirza, "Implementation of EAP with RSA for Enhancing The Security of Cloud Computing", International Journal of Basic and Applied Science, vol 1, no. 3, pp. 177-183, 2012.
[12] Wang Shao-hu, Chen Dan-we, Wang Zhi-weiP, Chang Su-qin, "Public auditing for ensuring cloud data storage security with zero knowledge Privacy" College of Computer, Nanjing University of Posts and Telecommunications, China, 2009
[13] Y. Dodis, S. Vadhan, and D. Wichs, "Proofs of retrievability via hardness amplification," in Proceedings of the 6th Theory of Cryptography Conference on Theory of Cryptography, ser. TCC '09, Berlin, Heidelberg, 2009, pp. 109–127.
[14] http//www.google.co.in/Wikipedia/Kerberos.
[15] http//www.google.co.in/Wikipedia/RSA algorithm.
[16] http//www.google.co.in/Wikipedia/AESalgorpthm.

AUTHORS

Mr. Ved M. Kshirsagar, ME (I.T.) Department of Information Technology, Sipna College of Engineering and Techonlogy,Amravati,Sant Gadge Baba Amravati University,Amarvati,Maharashtra,India - 444602

Prof. V.S. Gulhane, Phd pursuing from Sant Gadgebaba Amravati, ME (I.T.) Department of Information Technology, Sipna College of Engineering and Techonlogy, Amravati,Sant Gadge Baba Amravati University,Amarvati,Maharashtra,India - 444602