J. Appl. Environ. Biol. Sci., 5(8S)522-532, 2015

ISSN: 2090-4274
Journal of Applied Environmental and Biological Sciences
www.textroad.com

An Efficient Way to Protect the Security of Data in Cloud Computing

Sasan Azhdarpour

Department of Computer, ACECR Institute of higher education-kermanshah, Iran,
sasan.azhdarpour@gmail.com

Received: March 8, 2015
Accepted: May 10, 2015

ABSTRACT

The major factor in development limitations of cloud computing is an issue about the data security. The data security protects data versus deviation and unauthorized access. The creation acceptable level of the data security requires an attention of the principles of data security. The most important principles are confidentiality and integrity. Confidentiality, protects disclosure of data of unauthorized individuals and systems, integrity ensures authenticity and information accuracy. The data security problems in all types of cloud computing services is an issue, and requires to use of security principles and technical mechanisms safety, in order to solve user's concerns. So far, several methods have been proposed to control the security of cloud computing. But data security in the cloud computing is still a bottleneck. In this study, with the combination of the techniques, authentication APCC and check of data integrity DICM, a method is proposed for improving of the data security in cloud computing. The most important features of the proposed method are the protection of data security principles, public auditability, and supporting of dynamic data operations. The proposed method is implemented by using of the simulator Cloud Sim. Analyzing of the evaluation parameters derived from results of simulation experiments shows that, in one hand the process of authentication in proposed method in comparison with protocol SAP has a lower computational cost and communication, especially in client side. On the other hand, the process of check integrity has the lowest cost of communication in comparison with other designs, the consumption of network bandwidth is minimizing.

KEYWORDS: Cloud Computing, Authentication, Data Integrity, Public Auditability, Dynamic Data.

1. INTRODUCTION

By developing of parallel computing, distributed computing, grid computing, cloud computing model appeared. The most widely accepted current definition for cloud computing is what the National Institute of Standards and Technology (NIST) developed: Cloud Computing is a model for enabling convenient, 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 [1]. The factors that impel the occurring and development of cloud computing include: the development of grid computing, the appearance of high quality technology in storage and data transportation, and the appearance of Web2.0, especially the development of Virtualization [2]. Improving of productivity, reduction of costs and management facility of servers is one of the most important benefits of this technology [3].

The old dream of computing as a service was realized in the form of cloud computing by using this technology [4], [5]. Today, cloud computing services based on pay peruse model is built and now several large companies including Amazon, Google and Microsoft to provide this service pay. Cloud computing is offered Internet Cyber space But the Internet is not a controllable entirely place. According to researches which are conducted in [6], in comparison with the efficiency and accessibility, security is always a concern in open system architectures. As a virtual environment cloud computing has its special security threats and these threats are completely different from the threats in physical systems. So the old security policies cannot solve the new cloud computing security issues.

In this study first it reviews existing works in the field of data security, then we will summarize the concepts used in the structure of the proposed method. After that, the proposed method will be described in details. Finally, the performance of the proposed method compared with other existing methods.

2. RELATED WORK

In [6], Data security requirements are determined by analyzing architecture HDFS[1], And mathematical approach to data security as C2DSM to meet these security requirements are presented. In [8] by combining cloud computing architecture and service access control security or SACS[2], a method is provided for security. In [9], a security method based on the separation of security in cloud computing service models offered. Each layer security has an issue of confidentiality, integrity and availability, and responsibility for users and providers are different in resolving the issues. In [6] a security plan has been proposed three-levels defense structure and each level has itself duties to ensure data

[1] Hadoop Distributed File System
[2] Security Access Control System

* **Corresponding Author:** Sasan Azhdarpour, Department of Computer, ACECR Institute of higher education-kermanshah, Iran, sasan.azhdarpour@gmail.com

security. The first level is authentication by using OTP[3]. The second level is encryption automatic data and third level is user data retrieval.

The main factor limiting development cloud computing is an issue about data security [10]. Data security is data protection of diversion and unauthorized access. To establish an acceptable level of data security, consideration to the principles of data security is important, and the most important of these principles, are confidentiality and integrity.

Confidentiality is the term used to prevent the disclosure of information to unauthorized individuals or systems. Correct authentication is one of the most difficult things to guarantee in a cloud computing environment. Identity-based cryptography (IBC) is a public key technology that allows the use of a public identifier of a user as the user's public key. The majority of cloud computing systems provide digital id entity for users in order to access its services. Most cloud computing systems to provide data security and mutual authentication use Identity-based cryptography [11]. Identity-based cryptography has some interesting features that seems to match well with the requirements of cloud computing. Currently, SSL or TLS authentication protocol that called SAP, is the most widely used standard protocols to ensure the privacy of the web [12].By checking the process of SAP, We know that a heavy burden provides in both computing and communications for the user's cloud storage system [13]. One of the biggest issues with cloud data storage is that of data integrity verification at untrusted servers. For example, the storage service provider, which experiences Byzantine failures occasionally, may decide to hide the data errors from the clients for the benefit of their own. In order to solve the problem of data integrity checking, many schemes are proposed under different systems and security models. Considering the role of the verifier in the models, all the schemes presented before fall into two categories private auditability and public auditability. Although, schemes with private auditability can achieve higher scheme efficiency, public auditability allows anyone, not just the client (data owner), to challenge the cloud server for correctness of data storage while keeping no private information. Then, clients are able to delegate the evaluation of the service performance to an independent third party auditor (TPA), without devotion of their computation resources.

Ateniese et al [14], are the first to consider public auditability in their defined provable data possession model for ensuring possession of files on untrusted storages. In this scheme, they utilize RSA-based homomorphic tags for auditing outsourced data, thus public auditability is achieved. However, Ateniese et al. do not consider the case of dynamic data storage, and the direct extension of their scheme from static data storage to dynamic case may suffer design and security problems. Juels and Kaliski [15] describe a proof of retrievability model, where spot-checking and error-correcting codes are used to ensure both possession and retrievability of data files on archive service systems. Specifically, some special blocks called sentinels are randomly embedded into the data file F for detection purpose, and F is further encrypted to protect the positions of these special blocks. However, like [16] the number of queries a client can perform is also a fixed priori, and the introduction of precomputed sentinels prevents the development of realizing dynamic data updates. In addition, public auditability is not supported in their scheme. Still, the authors only consider static data files. Erway et al. [17], were the first to explore constructions for dynamic provable data possession. They extend the PDP model in [14] to support provable updates to stored data files using rank-based authenticated skip lists. This scheme is essentially a fully dynamic version of the PDP solution. To support updates, especially for block insertion, they eliminate the index information in the tag computation in Ateniese's PDP model [14] and employ authenticated skip list data structure to authenticate the tag information of challenged or updated blocks first before the verification procedure. However, the efficiency of their scheme remains unclear.

In this study, a method is provided for maintaining data security. This method has three phases and in each phase is used a technique for preserve data security. These techniques in comparison with existing techniques has higher performance.

3. THE PROPOSED METHOD DATA SECURITY

3.1. Definitions

3.1.1. Cloud data storage architecture

A representative network architecture for cloud data storage is illustrated in Fig.1. Three different network entities can be identified as follows [14].

Client: an entity, which has large data files to be stored in the cloud and relies on the cloud for data maintenance and computation, can be either individual consumers or organizations.

Cloud Storage Server (CSS): an entity, which is managed by Cloud Service Provider (CSP), has significant storage space and computation resource to maintain the client's data.

Third Party Auditor (TPA): an entity, which has expertise and capabilities that clients do not have, is trusted to assess and expose risk of cloud storage services on behalf of the clients upon request.

[3] On Time Password

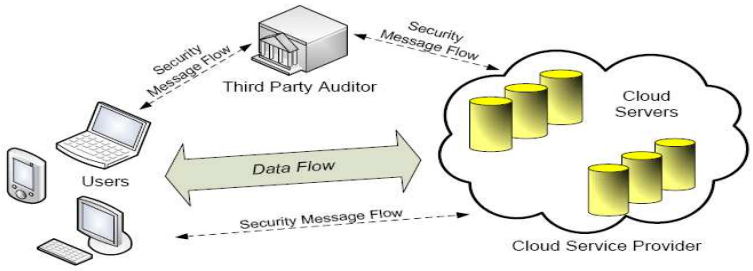

Fig.1. Cloud data storage architecture

3.1.2. Merkle hash tree

A merkle hash tree (MHT) is a well- studied authentication structure which is intended to efficiently and securely prove that a set of elements are undamaged and unaltered [12]. It is constructed as a binary tree where the leaves in the MHT are the hashes of authentic data values. Fig. 2 depicts an example of authentication.

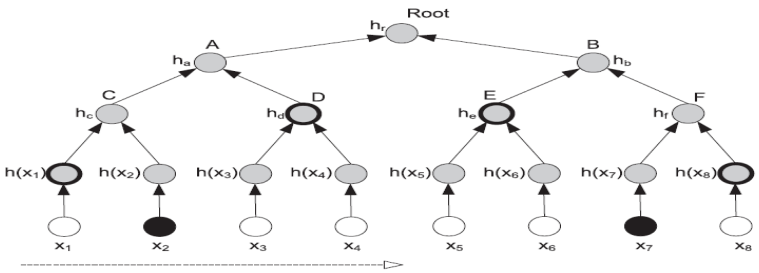

Fig.2. Merkle hash tree authentication of data elements

The verifier with the authentic hr requests for $\{x_2\ ,\ x_7\}$ and requires the authentication of the received blocks. The prover provides the verifier with the auxiliary authentication information (AAI) $\{\Omega_2 = h(\ h(x_1), h_d\ )$، $\Omega_7 = h(\ h(x_8), h_e\ )\ \}$.

The verifiercan then verify x_2 and x_7 by first computing$h(c) = h(\ h(x_1)\ ||\ h(x_2)), h(f) = h(\ h(x_7)\ ||\ h(x_8)), h(a) = h(h_c||\ h_d), h(b) = h(h_e||\ h_f), hr = h(h_a||\ h_b)$ and then checking if the calculated hr is the same as the authentic one.

3.2. Preliminaries

3.2.1. Identity-Based Encryption for HACC[4]

IBE is based on the above Root PKG setup, Lower-level setup and User-level setup algorithms. It is composed of encryption and decryption.

Encryption: Assume E_1 and E_2 are two users in the cloud computing. The identity of E_2 is $ID_{E2} = DN0||DN1||DN2$. To encrypt message m with ID_{E2} , E_1 acts as follows:

1) Compute the public key for E_2 $P_{E2} = H_1(ID_{E2})$
2) Convert the message in form of number M, so that the process is reversible and the number M is smaller than n. Obviously, if the message is larger than customary, it will be sent in form of package separately.
3) Compute the number C asciphertext $C = M^{E2}(mod\ n)$

However, C as an encrypted message is sent to entity E_2.

Decryption**:** after receiving the ciphertext C,$E_2$by using your private key decode $M = C^{SE2}\ (mod\ n)$.

3.2.2. Identity-Based Signature For HACC

IBS is also based on the above Root PKG setup, Lower- level setup and User-level setup algorithms. It incorporates two algorithms: signature and verification.

1) Using the H_2, the hash value message M obtains. $H_2: M \rightarrow M'$
2) Using private key S_{E1} to encrypt the message M' $C' = M'^{SE1}(mod\ n)$

signature's output is C' along with the ciphertext C which is sent to the entity of interest (for example E_2).

Verification: other entities can be confirmed E_1 Signature as follows.

1) The M obtain by IBE decryption algorithm is as input function $H_2$and obtain the hash value. $hashvalue = H_2(M)$
2) Using the public key E_1, C' is decoded. $m' = C'^{P_{E2}}(mod\ n)$

If $hash\ value = M'$ is established, the Signature's E_1 is valid.

[4] Hierarchical Architecture For Cloud Computing

3.3. The proposed method data security

The proposed method of data security in cloud computing is as Fig.3. The proposed method for maintaining data security principles has three phases.

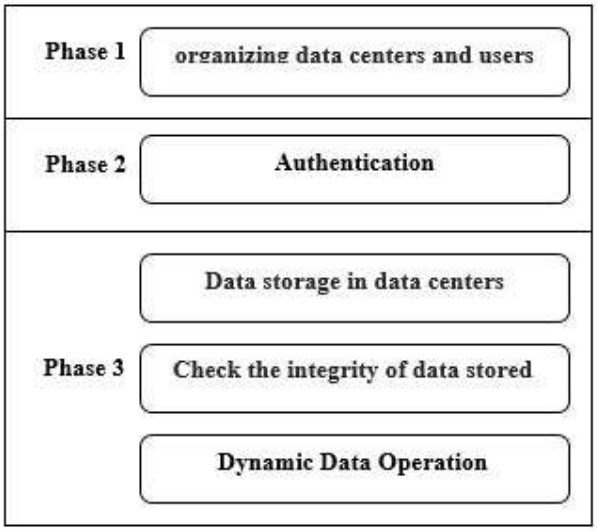

Fig.3. The structure of the proposed method

3.3.1. First phase: organizing data centers and users

As shown in Fig.4, Hierarchical Architecture for Cloud Computing (HACC) is composed of three levels [18]. The top level (level-0) is root PKG. The level-1 is sub-PKGs. Each node in level-1 corresponds to a data-center (such as a Cloud Storage Service Provider) in the cloud computing. The bottom level (level-2) are users in the cloud computing. In HACC, each node has a unique name. The name is the node's registered distinguished name (DN) when the node joins the cloud storage service. For example, in the Fig.4, DN of the root node is DN_0, DN of node M is DN_M and DN of node N is DN_N. We define the identity of node is the DN string from the root node to the current node itself. For example, the identity of entity N is $ID_N=DN_0||DN_M||DN_N$."||"denotes string concatenation. The deployment of HACC needs two modules: Root PKG setup and Lower- level setup (Users & Datacenters).

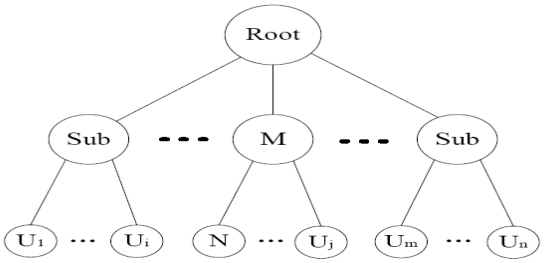

Fig. 4. Hierarchical architecture for cloud computing

HACC will require the establishment of two modules:

- Root PKG setup
- Set up the first and second levels (data centers and users)

Root PKG setup

1) Choose two distinct prime numbers p and q so be$p \neq q$
2) Compute $n = p \times q$
3) Compute$\varphi(n) = (p-1)(q-1)$, where φ is Euler's totient function. This value is kept private.
4) Choose two hash functions so that $H1$ function receives string identity entity X (ID_X) as input and returns Public keyPx equivalent $H_1: ID_x \rightarrow Px$ and H_2 is a standard function.

The values of $(n, \varphi(n), H_1, H_2)$ are as the general parameters of the system.

Set up the first and second levels

Assume there are m nodes in the level-1. For each node, the root PKG acts as follows (let X be an arbitrary node in the m nodes):

1) Compute public key of node X: (where $ID_x = DN_0||DN_x$) $Px = H_1(IDx)$ so that $1 < Px < \varphi(n)$, Px and $\varphi(n)$ are coprime.
2) Compute private key of node X, Sx: where $(Px)(Sx) \equiv 1 \ (mod\ \varphi(n))$.

After these two steps are done for each node x in the first level, All m nodes, are in the first level that the private keys Sx retain securely and public keys Px are delivered in general. Similarly, all nodes in the second level are doing the above steps.

3.3.2. Second phase: Authentication Protocol for Cloud Computing

In this section, based on the former IBE and IBS schemes, a secure Authentication Protocol for Cloud Computing (APCC) is proposed [18].

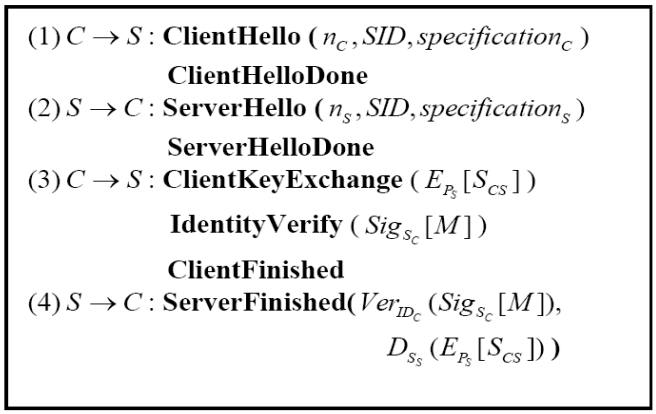

Fig.5. Authentication Protocol for Cloud Computing

Where
C: Client
S: Server
n_c , n_S: The fresh random number
SID: The session identifier
$Specification_C$: The cipher specification of C
$Specification_S$: The cipher specification of S
S_{CS}: A pre-master secret used to generate the shared key
E_{PS} [S_{CS}]: Encrypt $S_{CS}$with the public key P_S of S using the encryption algorithm of IBE
M: All handshake messages since the ClientHello message
Sig_{SC}([M]): Sign M with the private key S_C of C using the signature algorithm of IBS
Ver_{pc} (Sig_{Sc}([M])): Verify the signature Sig_{Sc}([M] with the help of ID_C using the verification algorithm of IBS
D_{Ss} ($E_{ps}[S_{cs}]$): Decrypt the $E_{ps}[S_{cs}]$ with the private key S_S using the decryption algorithm of IBE.

As shown in Fig.5 in step (1), the client C sends the server S a ClientHellomessage. The message contains a fresh random number n_c, session identifier ID_S and $Specification_C$. $Specification_C$ extends from TLS to handle the IBE and IBS schemes. For example, $Specification_C$ could be the form TLS- IBE- IBS- WITH –SHA- AES. IBE and IBS are used as secure transporting and authentication. SHA is the hash function. AES is the symmetric encryption algorithm. The Client Hello Done message means the step (1) finishes. In step (2), the server S responds with a Server Hello message which contains a new fresh random numbern_S , the session identifier SID and the cipher specification $Specification_S$. The $Specification_S$ is the suie of ciphers supported by S. The ServerHelloDone message means the step (2) is over. In step (3), C chooses a pre-master secret S_{CS} and encrypts it with the public key P_S of S using the encryption algorithm of IBE. The ciphertext is transmitted to S as ClientKeyExchange message. Then C generates a signature Sig_{SC}([M]) as the IdentityVerify message and forwards it to S. Finally, The ClientFinished message means the step (3) finishes. In step (4), S firstly verifies the signature Sig_{SC}([M]) with the help of ID_C. C can pass the verification only if it is the valid owner of ID_C. This completes the authentication of C by S .Then S decrypts the $E_{ps}[S_{cs}]$ with its private key S_S. Because of the fresh S_{CS}, the correct decryption indicates S is the valid owner of ID_S. This step authenticates the validity of$S$. The ServerFinished message means the step (4) finishes. Eventually, a shared secret key between C and S is calculated by $K_{CS} = PRF$ (S_{CS}, n_C , n_S), where PRF is a Pseudo-Random Function.

3.3.3. Third phase

3.3.3.1. Data storage in data centers

Assume that the outsourced data file F (Potentially encrypted) consists of a finite ordered set of blocks $F = m_1, m_2, \dots, m_n$ (assuming the blocks are separated and a systematic code is used to encode them). Client to store file F is operating in this way, first runs algorithm $keyGen(.)$ and Acquires public key PK, private key SK and public parameter N ($N = pq$). Then, client runs algorithm $SigGen(.)$, this algorithm by using PK and N for each block m_i ($i = 1,2, \dots, n$) computes a tag $T(m_i)$ in this form $T(m_i) = Pk^{m_i} mod N$. The set of tags are shown $\Phi = \{T(i)\}_{1 \le i \le n}$. In addition to, this algorithm, the set of tags Φ for the blocks of the file F, placed in the leaf nodes MHT from the left-to-right sequence and so the R root of MHT structure is computed. Then by using private key SK is signed. Now client sends $\{F, t, \Phi\}$ to server for saving and removes $\{F, \Phi\}$ from local storage and keeps $\{t, sig_{Sk}(T(R))$. (t is tag of F file, $t = name||n||Sig_{Sk}(name||n)$).

3.3.3.2. Check the integrity of data stored

Client gives public key Pk, $sig_{Sk}(T(R))$ and t, to TPA. Now TPA can check integrity file F.Fig.6 shows the process of check data integrity in the proposed method DICM. In step(1), To generate the message "$chal$" the TPA (verifier) chooses a random c-element subset $I = \{s_1, s_2, \dots, s_c\}_{s_1 \le \dots \le s_c}$ of set $[1, n]$ (n=counts of blocks file$F$), For each $i \in I$ the TPA chooses a random element v_i. The message "$chal$" specifies the positions of the blocks to be checked in this challenge phase. The TPA sends the $Chal\ \{(i, v_i)\}_{s_1 \le i \le s_c}$ to the prover (server). In step (2), After receiving the challenge, server runs algorithm $GenProof(.)$. This algorithm receives File F, the set of tags Φ and message $chal$. Then computes two values μ and $\mathcal{T}$.

$$\mu = \sum_{j=1}^{C} v_j\, mi_j \qquad\qquad \mathcal{T} = \prod_{j=1}^{C} T(mi_j)^{v_j} mod N$$

Here $1 \le i_1, \dots, i_C \le n$ refers to the blocks indexes challenged mi_j , C number of blocks challenged and n the total number of blocks fileF. $v_j$is also a random value is sent by the client or TPA in message $Chal$. In μ and $\mathcal{T}$data blocks challenged and labels which belong to them, have been collected in a single block. The size of the two combined blocks$\mu, \mathcal{T}$ isabout the size of a single block. So there is a very small overhead to send blockC. Moreover, in this algorithm a small amount auxiliary authentication information$\{\Omega_i\}_{s_1 \le i \le s_c}$computedto each block of the set C blocks challenged. Ω_i includes sibling nodes on the path from the leaf $\{h(T(m_i))\}_{s_1 \le i \le s_c}$to the R rootnode intheMHT. At the end of this step, the server sends proof P to TPA in form of $P = \{\mu, \mathcal{T}, \{T(m_i), \Omega_i\}_{s_1 \le i \le s_c}\}$ .In step (3), TPA after receiving P fromthe server, runs the algorithm $VerifyProof(.)$. In this algorithm, first by using $\{T(m_i), \Omega_i\}_{s_1 \le i \le s_c}$ computes root R in MHT, then is compared with metadata $sig_{Sk}(T(R))$ that exists in client's local storage, if it does not equal the output is false, otherwise in step (4) based on the values μ and $\mathcal{T}$, check $\mathcal{T} \stackrel{?}{=} Pk^{\mu}\ mod\ N$ that μ and $\mathcal{T}$ exist in proof P.If so, output is True, otherwise False.

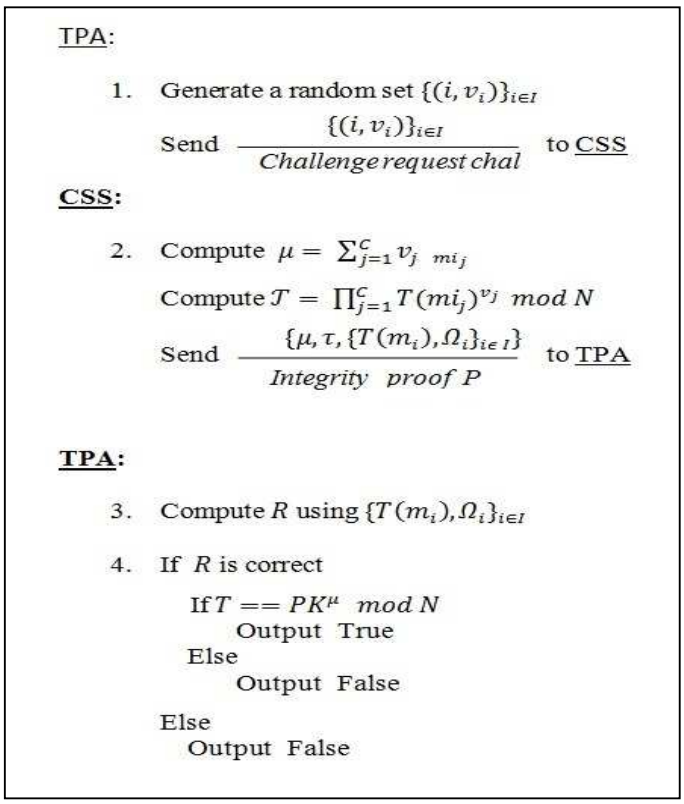

Fig.6. The method of DICM

3.3.3.3. Dynamic Data Operation with Integrity Assurance

Now we show how our scheme can explicitly and efficiently handle fully dynamic data operations including data modification (M), data insertion (I), and data deletion (D) for cloud data storage. Note that in the following descriptions, we assume that the file F and the hash values Φ have already been generated and properly stored at server. R root created by the client in structure MHT and then signed by the client is stored in local memory .So that the TPA can challenge the correctness of data storage with client's public key and metadata R.

Data Modification: We start from data modification, which is one of the most frequently used operations in cloud data storage. A basic data modification operation refers to the replacement of specified blocks with new ones. Suppose the client wants to modify the ith block m_i tom'_i. This process in DICM is described in Fig.7.
in step (1), based on the new block m'_i, the client generates the corresponding tag $T(m'_i) = Pk^{m'i} \bmod N$, Then, he constructs an update request message $update = \left(M, i, m'_i, T(m'_i)\right)$ and sends to the server, where M denotes the modification operation. In step (2), Upon receiving the request, the server runs algorithm $ExecUpdate(.)$.This algorithm performs the following actions:

1) replaces the block m_i with m'_i and outputs F'
2) replaces the $T(m'_i)$ with $T(m_i)$ and outputs the set of tag Φ'
3) replaces $T(m'_i)$ with $T(m_i)$ in the MHT construction and generates the new root R'

Finally, the server responses the client with a proof for this operation, $P_{update} = (\Omega_i, T(m_i), R')$ where Ω_i is the AAI for authentication of m_i. In step (3), After receiving the proof $P_{update}$for modification operation from server, the client runs algorithm $Verifyupdate(.)$.In this algorithm, first using $\left(\Omega_i, T(m_i)\right)$ generates root R in MHT then this value with metadata R (available in local memory client) is compared. If it does not equal the algorithm returns False. Otherwise, in step (4), it is Checked whether the server has done data modification or not? For this purpose, the root R_{new} of MHT base on modified file by using $\left(\Omega_i, T(m'_i)\right)$ is calculated, then checks $R_{\text{new}} \stackrel{?}{=} R'$ if so, the client signed R' in P_{update} and replaces with metadata R (The unmodified version File) that saved in client's local storage. Finally, client runs method data integrity (fig.6), if the output is True then removesm'_i and P_{update} in local storage.

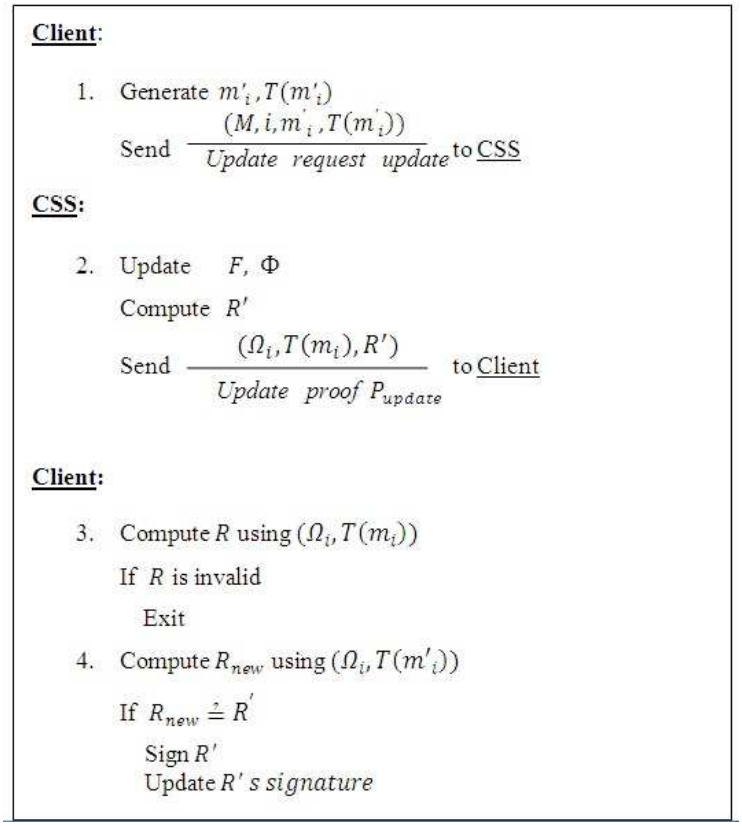

Fig.7. Dynamic data operation

Data Insertion: Compared to data modification, which does not change the logic structure of client's data file, another general form of data operation, data insertion, refers to inserting new blocks after some specified positions in the data file F. Suppose the client wants to insert block m^* after the ith block m_i . The method procedures are similar to the data modification case (see Fig.7, now m'_i can be seen as m^*). The difference is that in step (2), The server after receiving a request $update$, runs algorithm $ExecUpdate(.)$.This algorithm performs the following actions:

1) Stores block m^* and adds leaf node $h(T(m^*))$ after that leaf $h(T(m_i))$ in MHT and offers new file F' in the output.
2) Adds $T(m^*)$ to set of tag Φ'
3) Produced new root R' based MHT updated

Continuation of the process data insertion is similar to data modification process.

Data Deletion: data deletion is just the opposite operation of data insertion. For single block deletion, it refers to deleting the specified block and moving all the latter blocks one block forward. Suppose the server receives the update request for deleting block m_i, it will delete m_i from its storage space, delete the leaf node $h(T(m_i))$ in the MHT and generate the new root metadata R′. The details of the method procedures are similar to that of data modification and insertion, which are thus omitted here.

4. SIMULATION EXPERIMENTS

The platform of simulation experiment is CloudSim 3.0 which is a simulation platform based on Java. Special users and resources can be generated by rewriting these interfaces. This aligns well with various users and resources of cloud computing. This aligns well with randomness of cloud computing entity action. Therefore, it is feasible to simulate our proposed method by CloudSim. In this method encryption standard algorithms used which consists of Public key RSA, Symmetric encryption AES and hashing algorithm MD5. To implement these algorithms in simulation of proposed method is used OpenSSL0.9.8 library. To precisely simulate the network delay, there is 25~45ms waiting time before a message is sent. The simulation environment is composed of two Entities (data center and client) which Configuration of these entities attributes are asTable1.

Table 1: simulation entities attributes

Entity	Processor Architecture	Operating System	RAM (GB)	Storage(GB)
Data Center	Intel Core 2 3.0 GHz	Linux	16	500
Client	P4 3.0 GHz	Win XP	4	20

4.1. Evaluation simulation results of APCC

According to parameters values which is derived from the outputs of the simulation experiments (ten tests), the average evaluation parameters for two authentication schemes APCC and SAP on the table 2 is calculated.

Table 2: Comparing the average parameters values of evaluation authentication schemes

Authentication schemes	Computation time of client (ms)	Computation time of server (ms)	Communication cost (KB)	Authentication time (ms)
APCC	37	193	1588	514
SAP	220	276	5252	879

As Table 2 shows the average values for all parameters evaluated in the APCC is less than the SAP protocol.
Computation time of client**:** the average computation time of client for APCC is approximately 37 ms while that for SAP is 220ms. That is to say, the computation time of client for APCC is 17% of that for SAP. In the APCC, client is calculated only a sign and an encrypted ciphertext. Furthermore, in SAP authentication server must also be done. For this reason in APCC client computing is much less than in comparison to SAP.
Computation time of server**:** the computation time of server for APCC is approximately 193 ms while that for SAP is276ms. Therefore, the computation time of server for APCC is 70% of that for SAP. In the APCC, server is calculated only a sign a true verification and a decrypt ciphertext. Furthermore in SAP, authentication client must also be done. For this reason in APCC server computing is less comparison to SAP.
Communication cost**:** the communication cost of APCC is approximately 1588 bytes while that of SAP is 5252 bytes. That is to say, the communication cost of APCC is 30% of that of SAP.Note that only dominant communication is considered, i.e. certificates signed or encrypted messages, which may have the greatest consumptions of the network bandwidth. The communication cost of SAP is two public key certificates and two RSA signatures. However, in the APCC, the communication cost is only one IBS signature and one IBE ciphertext. So, the APCC is exchanged less data that reduces communications costs and thus reduces the consumption of network bandwidth.
Authentication time**:** the authentication time of APCC is approximately 514 ms while that of SAP is 879ms. That is, the authentication time of APCC is 58% of that of SAP.

The computations time in SAP are shown in Fig.8. Accordingly the average time of client is 220ms while that for server is 272ms. Therefore, in SAP client as much as 80% computation of server is incurred computational overhead, while limited client resources.

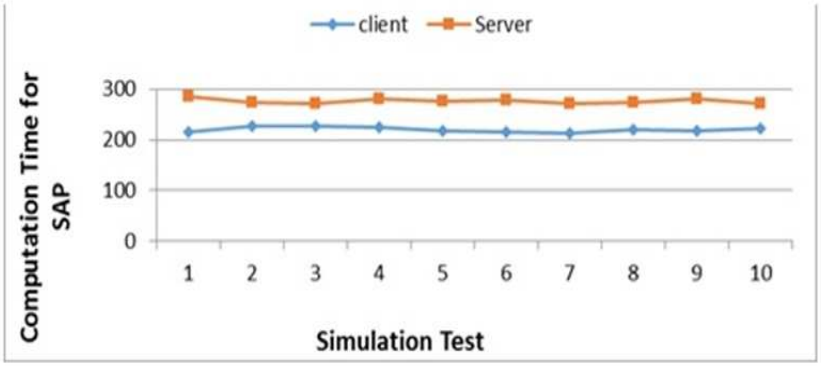

Fig.8. Computation time for SAP

The computations time in APCC are shown in Fig.9. Accordingly the average time of client is 37ms while that for server is 193ms. Therefore in APCC, client as much as 20% computation of server is incurred computational overhead. This aligns well with the idea of cloud computing which allows the user with a platform of limited performance to outsource its computational tasks to some more powerful servers. As a result, the more lightweight user side can connect more servers and contribute to a larger scalability.

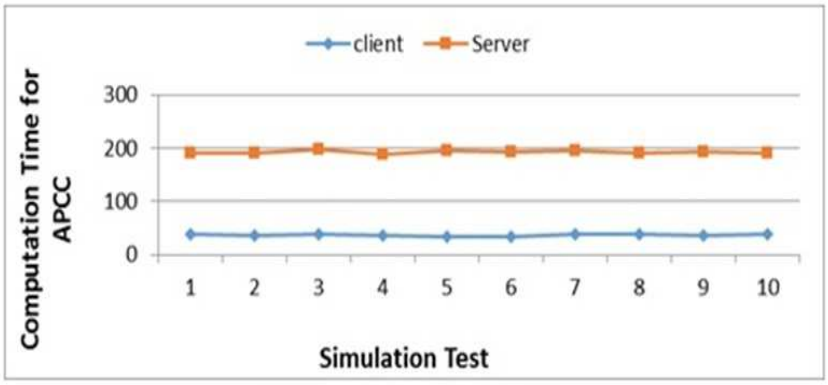

Fig.9. Computation time for APCC

4.2. Evaluation simulation results of DICM

In Table 3, Data integrity checking schemes are compared with the DICM (in the proposed method).

Table 3: Comparisons of different remote data integrity checking schemes

Metric \ Scheme	PDP	POR	DPDP	DICM
Data dynamics	No	No	Yes	Yes
Public auditability	Yes	Yes	No	Yes
Server comp. complexity	O(1)	O(1)	O (log n)	O (log n)
Client comp. complexity	O(1)	O(1)	O (log n)	O (log n)
Comm. Complexity	O(1)	O(1)	O (log n)	O (log n)
Probability of detection	$1-(1-f)^c$	$1-(1-f)^c$	$1-(1-f)^c$	$1-(1-f)^c$

Public auditability**:** in all data Integrity Checking Schemes except DPDP, Client to challenge the cloud servers for correctness of data storage does not need to explicit retrieval of data blocks. The DICM, can check the integrity of data without retrieving the data blocks. In this way obtained Public auditability.
Data dynamics**:** POR and PDP schemes, Consider only static data. These schemes by using information index i create block tag. So, after inserting or delete the block m_i also other blocks index Changes. Therefore, to insert or delete a block, all other blocks tags will be calculated based on new indices and this is unacceptable over head calculation. So, these schemes do not support dynamic data operation. But, in DPDP dynamic data is considered for the first time. For this purpose, in DICM similar to DPDP information indexes have been removed to create tags and for the block m_i are used $T(m_i)$ as tag. Whereas in PDP and POR, $T(name||i)$ and $T(V||i)$ is used. Accordingly, in DICM and DPDP operations on each data block will not affect to the blocks.

Parameters complexity**:** These parameters are computation server, computation client and communication cost. POR and PDP schemes, Consider only static data. Therefore, constant time achieve for parameters complexity. But, DPDP and DICM are considered to dynamic data. So the time complexity of Parameters is $O(log\ n)$.
Probability of detection**:** this parameter shows the probability of detecting server misbehavior and this parameter value is $1-(1-f)^C$ for all schemes **(**where F is the ratio of corrupted blocks and C is the ratio of challenged blocks**).**
In table 2, the results of the simulation tests the data integrity check schemes are shown. Accordingly, the cost of communications PDP is significantly less than other schemes. But it is considerable that the PDP unlike other schemes does not support the dynamics data operation. Increasing the cost of communications in DICM and DPDP is solely for supporting the dynamics data operation. Whereas in both of DPDP and DICM, the client and server computational overhead compared with the PDP has not increased substantially.

Table 4: Comparing the average parameters values of evaluation data integrity checking schemes

Data Integrity Checking Schemes	computation time of client (ms)	computation time of server (ms)	communication cost (KB)
PDP	1251	537	207
DPDP	1825	578	583
DICM	1630	554	516

According to Fig.10, In all schemes the cost of communication almost grows with increasing the block size linearly that mainly due increase in combined block size of verification $\mu = \sum_{j=1}^{C} v_j\, mi_j$. As seen in Fig.10, PDP has the lowest cost communication among other schemes. But it is considerable that the PDP does not support the dynamics data operation. On the other hand, DPDP has the most cost communications. But The cost of communication in our DICM scheme, is more than the PDP and it is less than the DPDP.

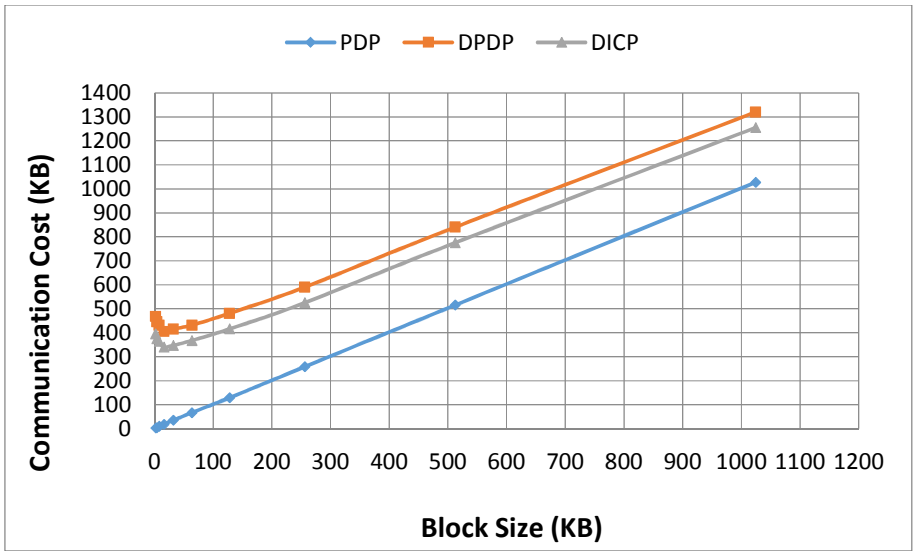

Fig.10. The cost of communications in data integrity check

The results in Fig.10, shows that when the block size of 16KB is selected, both DICM and DPDP schemes can achieve optimal point that it has minimum communication cost. In this case, the lowest communication cost in DICM is 340KB and in DPDP is 405KB.

5. CONCLUSION

The proposed method presents two schemes APCC and DICM in order to, protect the confidentiality and integrity, which the most important principles of data security are considered. Performance analysis indicated that the authentication process with using APCC is more efficient and lightweight than SAP, especially the more lightweight client side. DICM is efficient way to check the integrity of outsourced data in data centers in the cloud computing system. On the one hand, with the Public auditability, it is possible that the client is not always concerned about the resources constraints and overhead of stored data integrity checking and the efficiency of storage service assess to at third party in dependent auditor. On the other hand, in DICM the public forms data dynamic operations (modify, delete and insert) is considered. This means that, the index information has been removed to create tags. Accordingly, in DICM operations on each data block will not affect other block. Therefore, does not increase the computation cost. Moreover, DICM has lowest cost communications in comparison with other schemes. Reduce the cost of communications represents less network bandwidth consumption and increase the speed of transfer information in outsourced data integrity checking. Both APCC and DICM schemes reduce the computational overhead on the client side. This aligns well with the idea of

cloud computing which allows the user with a platform of limited performance to outsource its computational tasks to some more powerful server.

For future work on the third phase of the proposed method can improve security by providing methods or using standard encryption algorithms.

REFERENCES

[1] Mell, P., Grance, T. (2009). *"The NIST definition of cloud computing, National Institute of Standards and Technology"*, Information Technology Laboratory, Vol. 15, No. 10, pp. 107-347.

[2] Zhang, S., Zhang, S., Chen, X., &Huo, X. (2010)., In*" Cloud computing research and development trend"* proceedings of the 2nd international conference on future networks, Sanya, China, , pp. 93-97

[3] Koh, Y., Knauerhase, R., Brett, P., Bowman, M., Wen, Z., & Pu, C. (2007). "*An analysis of performance interference effects in virtual environments"*, In the IEEE International Symposium on Performance Analysis of Systems & Software, San Jose, CA, USA, pp. 200-209.

[4] Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., &Brandic, I. (2009). *"Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility"*, Future Generation computer systems, Vol. 25, No. 6, pp. 599-616.

[5] Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., ...&Stoica, I. (2009). *"Above the clouds: A Berkeley view of cloud computing"*, Department Electrical Engineer and Computer Sciences, University of California, Berkeley, Vol. 28.

[6] M Mohamed, E., S Abdelkader, H., & El-Etriby, S. (2013). "Data Security Model for Cloud Computing". In proceedings of the 12th International Conference on Networks, Seville, Spain, pp. 66-74.

[7] Dai Yuefa, W. B., Yaqiang, G., Quan, Z., &Chaojing, T. (2009). "Data Security Model for Cloud Computing". In Proceedings of the International Workshop on Information Security and Application, Qingdao, China, pp. 52-58.

[8] Jing, X., & Jian-Jun, Z. (2010). "A brief survey on the security model of cloud computing". In proceedings of the 9th International Symposium on Distributed Computing and Applications to Business Engineering and Science, Hong Kong, China, pp. 475-478.

[9] Ajoudanian, S., & Ahmadi, M. R. (2012). "A Novel Data Security Model for Cloud Computing". International Journal of Engineering and Technology, Vol. 4, No. 3, pp. 230-233.

[10] Kotulic, A. G., & Clark, J. G., (2009). *"Why there aren't more information security research studies"*, Information & Management, Vol. 41, No. 5, pp. 597-607.

[11] Yan, L., Rong, C., & Zhao, G. (2009). "Strengthen cloud computing security with federal identity management using hierarchical identity-based cryptography". In Proceedings of the 1st International Conference on Cloud Computing, Beijing, China, pp. 167-177.

[12] Kwon, E. K., Cho, Y. G., &Chae, K. J. (2001). "Integrated transport layer security: end-to-end security model between WTLS and TLS". In Proceedings of 15th International Conference on Information Networking, Beppu City, Oita, pp. 65-71.

[13] Soghoian, C., &Stamm, S. (2012). "Certified lies: Detecting and defeating government interception attacks against ssl" .In Proceedings of 15th International Conference *on F*inancial Cryptography and Data Security, (pp. 250-259).

[14] Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., & Song, D. (2007). *"Provable data possession at untrusted stores"*, In Proceedings of the 14th ACM conference on Computer and communications security, , Alexandria, Virginia, USA, pp. 598-609.

[15] Juels, A., &Kaliski Jr, B. S. (2007). *"PORs: Proofs of retrievability for large files"*, In Proceedings of the 14th ACM conference on Computer and communications security, , Alexandria, Virginia, USA, pp. 584-597.

[16] Ateniese, G., Di Pietro, R., Mancini, L. V., &Tsudik, G. (2009). *"Scalable and efficient provable data possession"*, In Proceedings of the 4th international conference on Security and privacy in communication networks, Istanbul, Turkey, pp. 118-129.

[17] Erway, C., Küpçü, A., Papamanthou, C., &Tamassia, R. (2009). *"Dynamic provable data possession"*, In Proceedings of the 16th ACM conference on Computer and communications security, Chicago, Illinois, USA, pp. 213-222 .

[18] Li, H., Dai, Y., & Yang, B. (2011). "Identity-Based Cryptography for Cloud Security". In Proceedings of the 1st International Conference on Cloud Computing, Beijing, China, pp. 157-166.