ISSN No. 0976-5697

Volume 4, No. 11, Nov-Dec 2013

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Security Enhancement through Fine Grained Access Control in Cloud Computing

S. Rama Krishna[1], B. Padmaja Rani[2]
Associate Professor, Department of Computer Science, VYCET, Chirala.
Professor & Head of the Department, Department of Computer Science & Engineering, JNTUCEH, Hyderabad
Ramakrishna.ss@gmail.com, padmaja_jntuh@yahoo.co.in

Abstract: Cloud computing has showed up as a popular design in managing world to back up managing large volumetric details using cluster of commodity computer systems. It is the newest effort in offering and managing computing as a service. Either program or Application, it is used to describe both. A cloud computing paradigm dynamically assigns, configures, relocates and de provisions these computing resources as needed. it also describes applications that are to be extended accessible through the Internet. Data security and availability management is one of the most complex ongoing studies in cloud managing, because of clients outsourcing their sensitive details to cloud service providers. Current alternatives that use genuine cryptographic techniques to reduce these security and availability management problems suffer from heavy computational cost on both data owner as well as the cloud service provider for key distribution and management. This paper capability based access control addresses this challenging problem to ensure only valid users will access the outsourced data. It reduces burden over the data owner for key management it is proposed to maintain key management in the cloud it self. This work also proposes some modifications in Diffie-Hellman key exchange protocol to thwart from man in middle attack between cloud service provider and the user for secretly sharing a symmetric key for secure data access that alleviates the problem of key distribution and management at cloud service provider. The simulation run and research reveals that the recommended strategy is highly efficient and secured under current security designs.

I. INTRODUCTION

Cloud computing has become a necessity today when the company plans to increase capacity "or capabilities on the fly without getting to invest new infrastructure, training new individual purchase new license application, etc. based service encompasses any subscription or pay per use which extends the existing IT capabilities of the company, current time through Online.

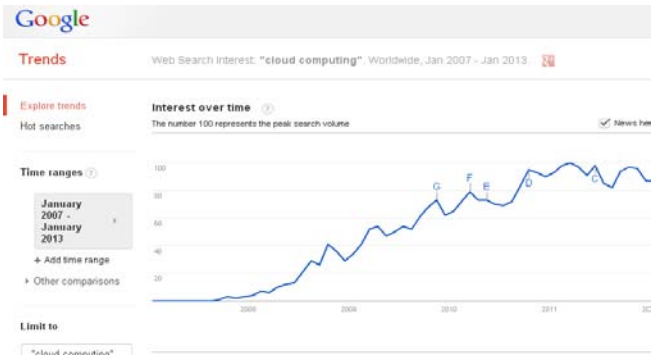

Figure1: Trend showing interest towards cloud [27]

Figure 2:As per Gartener survey februvery 2013 Global spending on public cloud services is expected to grow 18.6% in 2012 to $110.3B, achieving a CAGR of 17.7% from 2011 through 2016. The total market is expected to grow from $76.9B in 2010 to $210B in 2016 [1].

The U.S. National Institute of Standards and Technology (NIST) to determine cloud computing as "a model for enabling convenient access and on demand network to a shared pool of configurable computing resources (eg, networks, servers, storage, applications and services) that can need to acquire rapidly and released with minimal management effort or service provider interaction "[2].Cloud computing can also be defined as "a type of parallel and distributed system consisting of a collection of interconnected and virtualized as one or more unified computing resources based on service-level agreements established through negotiation between the service provider and consumers" [3]. In recent past, various commercial models are developed that are described by "X as a Service (XaaS)" where X could be hardware, software or storage etc [4]. Successful examples of emerging cloud computing infrastructures are Microsoft Azure [5], Amazon's EC2 and S3 [6], and Google App Engine [7] etc.

Cloud computing also faces the data security challenges as that of any other communication models. As data owners store their data on external servers, there have been increasing demands and concerns for data confidentiality, authentication and access control [8]. Besides confidentiality and privacy breaks, the external servers could also use part of the data or whole for their financial gain and hence tarnishing the data owners market or even bringing economic losses to the data owner. These concerns originate from the fact that cloud servers are usually operated by commercial providers which are very likely to be outside of the trusted domain of users [9]. The work done in [8][10-11] propose cryptographic access control model as shown in Fig.1 which we have also considered as the system model in our work. The model depicted in Fig.1 has three participants Data Owner (DO), Cloud Service Provider (CSP), and the User. The DO places the data on the CSP which the user wants to access. As the CSP is un-trusted, DO places encrypted data on CSP. Upon receiving a data access request from the user, DO sends required keys and a

© 2010, IJARCS All Rights Reserved

certificate to the user. User then presents the certificate to CSP and gets the encrypted data upon successful verification by CSP as shown in Fig.3.

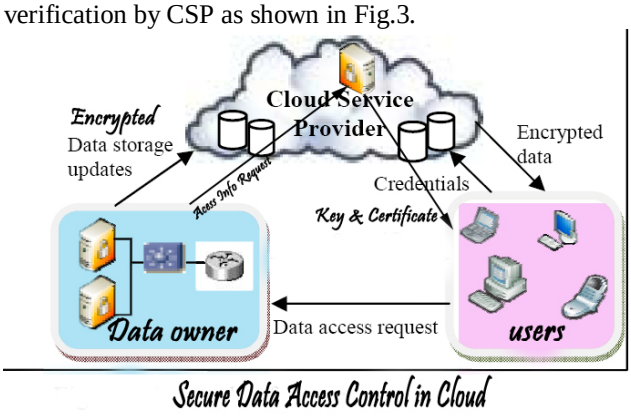

Figure 3.

The model described in Fig.3 guarantees confidentiality, integrity and authentication, but the problem with this model is that the owner should be always online when the user wants to access the data [12]. The key management between all the communicating parties is also cumbersome. In some situations, an owner with poor computing capabilities becomes a bottleneck. Traditional access control architectures usually assume the data owner and the servers storing the data are in the same trusted domain [9], where the storage servers are modelled as an omniscient reference monitor [13] entrusted to define and enforce access control policies. An assumption like this does not hold true for cloud computing as the data ownersmust ensure the trust worthiness of the cloud servers which is very difficult in practice.

The general principle of cryptography has also been used with Access Control Lists (ACLs) for ensuring access control and confidentiality to the data storage on un-trusted servers [14-15]. Use of ACLs or filegroups reduces the complexity of data encryption and key management. However, ACLs or filegroups still lack scalability, and fine-grainedness for confidentiality and access control in cloud computing [9].Access control policies based on data attributes and encryption as suggested in [9] also becomes cumbersome as it is computationally challenging to derive a unique logical expression for every user in the cloud.

In this paper, we address this open issue of access control and propose a secure, scalable, and efficient data access control mechanism using capability based access control [24] and over encryption for cloud computing paradigm. Data owner encrypts the outsourced data with a symmetric key which is shared only with the user. The CSP and user generate a symmetric key using a modified DH key exchange protocol for the purpose of secure communication between them that relieves the CSP from key management burden as needed in public key cryptography. The proposed work guarantees secure access to outsourced data and at the same time it relieves the DO from worrying about every data access request made by the user except the initial one. Hence, DO will not be a bottleneck and rather will increase efficiency as it does not remain in scene for all future data access requests and responses.

The remainder of the paper is organized as follows. Section II reviews the related research. Section III discusses models and assumptions. Section IV presents our proposed scheme. In Section V, we analyse our proposed scheme in terms of performance and strength. Section VI gives details about the simulation run. Finally, Section VII concludes the paper and presents future research directions.

II. RELATED WORK

A few research efforts have directly tackled the issues of access control in cloud computing model. Yu et al. [9] proposed a scheme to achieve fine-grained, secure, and scalable access control in cloud computing by combining techniques of attribute-based encryption (ABE), proxy reencryption, and lazy re-encryption. A set of attributes are associated to a file that are meaningful in the context of interest. The access structure of each user is defined as a logical expression over these attributes, which reflects the scope of data file that the user is allowed to access. A public key component is defined for each attribute.

Data files are encrypted using the public keys corresponding to their attributes. User secret keys are defined matching their access structures so that a user is able to decrypt a ciphertext if and only if the data file attributes satisfy his access structure. The main issue with this scheme is that as the cloud servers store a vast amount of data, deriving a unique logical expression for every user using the attributes of every file will become computationally complex. Also, re-encryption becomes a problem as updating the user secret for all the users except the revoked one is a challenging process when the number of users is high. Ateniese et al. [14] proposed a secure distributed storage scheme based on proxy re-encryption. The data owner encrypts blocks of content with symmetric content keys. The content keys are all encrypted with a master public key. The data owner uses his master private key and user's public key to generate proxy re-encryption keys, using which the semi-trusted server can then convert the ciphertext into plaintext for a specific user. The issue with this scheme is that collaboration between a malicious server and any single malicious user would expose decryption keys of all the encrypted data and compromise data security of the system. Miklau et al. [16] presented a framework for access control on published XML documents by using different cryptographic keys over different portions of XML tree. They also introduced special metadata nodes in the structure to enforce access control. The complexity of this approach is XML tree generation and key management.

Vimercati et al. [17] proposed a solution for securing data storage on untrusted servers. Each file is encrypted with a symmetric key and each user is assigned a secret key. The data owner creates corresponding public tokens from which, together with his secret key, the user derives decryption keys. The data owner sends these public tokens to the semi-trusted server and also delegates the responsibility of distribution. Given these public tokens, the server is not able to derive the decryption key of any file. This approach introduced a minimal number of secret key per user and a minimal number of encryption key for each file. The issue with this scheme is that the complexity of operations of file creation and user grant or user revocation requests is linear to the number of users, because of which the scheme becomes non-scalable. Naor et al [18] proposed application of symmetric key primitives in an untrusted storage environment to ensure data confidentiality and access control. The scheme is based on pre-key distribution mechanisms using Blom [19] scheme that can reduce public

© 2010, IJARCS All Rights Reserved

key cryptography in the storage-as-a-service model. The issue in this work is that they have not evaluated the performance of their schemes and also, do not provide an expressive access control model.

III. MODEL AND ASSUMPTIONS

Similar to [8][10][20], we assume that the system is composed of a Data owner, many Data consumers called as Users, and a Cloud service provider. The authentic users get the data file that is stored on the CSP by the DO in a confidential manner. We also assume that neither the DO nor the User will be always online as is done in [9]. DO comes online when a new user is to be registered or when the capability list is to be updated at CSP. CSP is a conglomeration of several Service providers like Amazon, Google, and Microsoft which has very large storage and computation capacity. CSP is always online. We also assume that the DO can also execute a binary application code at the CSP for managing his data files in addition to storing those in encrypted form as is done in [9][21]. Communication between CSP and user or between user and DO is made secure using cryptographic primitives like SSL/TLS. In our model, users cannot access other's data files as there will be no capability granted by DO for these users. For the purpose of simplifying the secure communication between DO and CSP, DO and user, we assume that each party is preloaded with others public keys hence, we do not need any PKI for distributing public keys of each other involved in secure communication. Fig.3 shows the notations used in our scheme.

IV. PROPOSED SCHEME

In this section, we provide an example of our approach along with data structures and notations used in the algorithm. We also present the pseudo-code of our algorithms. In order to achieve secure and efficient data access control in cloud computing, we uniquely combine capability based access control technique with cryptography.

Fig.4 illustrates our scheme by an example. Here, the owner can be a doctor who posts the patients' reports into the cloud and user can be any Hospital patient registered into the same decease who views his/her health reports from the cloud. The data owner computes a message digest using MD5 for every file belonging to the data set available with it. We have used a 128-bit MD5 hash over any other like SHA-1 (160-bit) for data integrity because we are encapsulating this digest along with the file using a symmetric key.

This in turn gives cryptographic strength much more than using the later one i.e. SHA-1. This ensures data confidentiality and integrity between owner and user. DO then updates the capability list with a new entry for every user and the data item that can be accessed by the user. It inserts access rights (0 for read, 1 for write or 2 for both read and write) into the AR field of the tuple (UID, FID, AR). DO then sends everything encrypted using its private key first and then using public key of the CSP for the purpose of authentication and confidentiality between CSP and DO. This procedure is described in Fig.4.

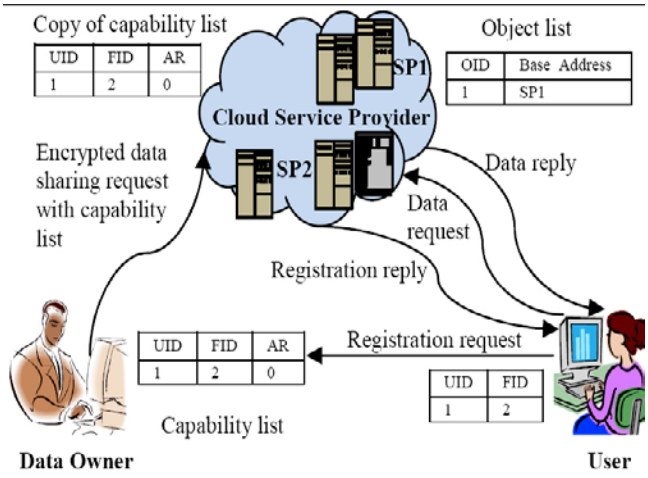

Figure 4: example operations Private health record

Notations	Description
PU	Public Key
PR	Private Key
PUSP	Public Key of Service Provider
PRSP	Private Key of Service Provider
PUUSR	Public Key of User
PRUSR	Public Key of Service Provider
PUOWN	Public Key of Owner
PROWN	Private Key of Owner
DS_i	i_{th} Data Set
fi	i_{th} file
Di	i_{th} file Message Digest
ND_i	New Message Digest for i_{th} file
Oi	i_{th} Object
EOi	Encrypted form of i_{th} object
EK	Encryption
DK	Decryption
K_O	Symmetric key of owner
MD5	Hash Algorithm
CapList	Capability List
StorageArray	Array that stores capability and data files
AR	Access Rights
Diffie –Hellman Parameters	
$X_{A/B}$	Chosen Secret Key
$Y_{A/B}$	Calculated Public Key
q	Prime number
p	Primitive root
K_S	Secret session key
mod	Modulus operation.

Figure 5: Notations used throughout this paper

Step 1: Preparing encrypted data items
1.1 for each file $f_i \in DS_i$
$Di \leftarrow MD5(f_i)$;
$Oi \leftarrow EK_O(fi, Di)$;
1.2 for each Object Oi
$EOi \leftarrow EK_{PUSP}(EK_{PROWN}(Oi))$;
Step 2: Updating the capability list
CapList ← (UID,FID,AR)
Step 3: Send data items & capability list to CSP
Send $(EO_i, EK_{PUSP}(EK_{PROWN}(CapList)))$;

Figure 6. Algorithm for DO sending encrypted outsourced Data items and capability list to CSP.

Fig.7 illustrates the procedure that the CSP will adopt when it receives encrypted data files and capability lists from the DO. It uses its' own private key and the public key of DO to decrypt the message and store the encrypted data files and capability list in its' storage. However, our model

© 2010, IJARCS All Rights Reserved

does not allow the CSP to know the actual data items as it does not know the Ko i.e. the shared symmetric key between user and DO. This achieves one of our design objectives where the data files only should be visible to the user and DO, not to the CSP as it is available over an untrusted domain.

Step 1: Storing data items & capability list
StorageArray ←Receive (EO_i , EK_{PUSP} (EK_{PROWN} (CapList))
StorageArray[1] ← (EO_i) ;
Storage Array[2] ← (CapList) ;
Step 2: Updating the Object Table List
O_i ←DK_{PRSP} (DK_{PUOWN} (Storage Array[1])) ;
ObjTable ← (O_i , SP_i) ;
Step 3:Update Capability List
CapList ← DK_{PRSP} (DK $_{PUOWN}$ (Storage Array[2])) ;

Figure 7 Algorithm for CSP receiving and storing both encrypted files and capability list

When a new user is to be added, the user needs to send a registration request with UID, FID, Nonce, Timestamp and access rights required for the data file to the data owner. Fig.6 describes the pseudo code for this procedure. Here, after receiving a request, data owner adds an entry into the capability list if it is a valid request. For simplicity we assume that the DO has a separate procedure for verifying the genuineness of the client request.

DO now send the capability list and an encrypted message intended for user with all the key parameters needed at user for decrypting the data files to CSP. CSP now updates its' capability list and sends a registration reply to user using over encryption i.e. encrypting twice using EKPUUSR. This meets our critical design goal that is the key parameters required for decryption are still confidential to user (although these have come via CSP). The nonce and timestamps in the request and reply message serve the purpose of replay and man-in-the-middle attack avoidance.

Step 1: User sends an encrypted registration request to owner
Send (EK_{PUOWN} (EK_{PRUSR} (UID, FID ,N_1,TimeStamp, AR))) ;
Step 2: Owner updates capability list if the users request is valid
Caplist ← add(CapList,(UID,FID,AR));
Step 3:Owner encrypts updated CapList and sends it to the CSP
Send (EK_{PUSP} (CapList, (EK_{PROWN} (EK_{PUUSR}(EK_O ,MD5,N_1+1,TimeStamp))))) ;
Step 4: Service Provider Updates its copy of the capability list and sends the message to the user which is intended for him.
Send (EK_{PUUSR} (EK_{PROWN} (EK_{PUUSR} (EK_O ,MD_5,N_1+1,TimeStamp)))) ;
Step 5: Now the user decrypts the message and is aware of the symmetric key and hash functions used by the owner.

Figure 8.Algorithm for registering a new user

After the data files are available at the cloud in an encrypted form and keys are made available to the user, now that the actual data access request goes from a user to the CSP. If request is valid, D-H is initiated by CSP.

This satisfies our design criteria of not keeping the data owner always online. Fig.9 describes the use of modified DH key exchange protocol to acquire a shared session key for the purpose of confidential communication between CSP and user. In this algorithm, we have attempted to solve the man-in-the-middle attack that is prominent on D-H key exchange.

This is achieved by encrypting the D-H parameters using the public key of one side and using nonce in each direction. CSP encrypts the object Oi which is an encrypted version of file (fi) and its' digest (Di) using the shared session key generated from the D-H exchange. This over encryption ensures the confidentiality of the message between CSP and user and also CSP is unable to read the contents of the data file. Our assumption here is that the session key generated between CSP and user remains valid for a predefined period.

Step 1: User sends a data access request to CSP
Send(UID,FID,AR);

Step 2: CSP checks if User has a capability in the capability list and if the request is an element of the AR set then it proceeds

StorageArray ←Recv(UID,FID,AR);
If (StorageArray [1] == CapList (UID)
  If (StorageArray[3] ⊂ CapList (AR)
    Goto step 3
  Else
    Goto step 9

Step 3: Cloud Service Provider generates the D-H Parameters and calculates the Public key Y_A

3.1 Choose a private key say, X_A and q, p.
$Y_A \leftarrow p^{XA} \bmod q$.

Step 4: CSP sends the parameters to User without encrypting

Send (Y_A , q, p, N_2);

Step 5: User receives the message and generates his or her public and private keys

5.1 StorageArray ←Recv(Y_A, q, p, N_2);
5.2 Y_A,q,p,N_2 ← StorageArray;
5.3 Generates his private key X_B and calculates Y_B, and shared secrete key as follows
$Y_B \leftarrow p^{XB} \bmod q$
Session Key $K_S \leftarrow Y_A^{XB} \bmod q$
5.4 User now sends the global parameter Y_B to CSP encrypted
Send (EK_{PUSP} (Y_B, N_2+1));

Step 6: CSP calculates the shared secret key using Y_B and X_A

6.1 StorageArray ← Recv(EK_{PUSP} (Y_B,N_2+1)) ;
6.2 Y_B,N_2+1 ←DK_{PRSP}(StorageArray);
6.3 Session key $K_S \leftarrow Y_B^{XA} \bmod q$

Step 7:CSP encrypts the data using shared session key
EO_i ← EK_S (Oi) ;

Step 8: CSP sends the encrypted data to the User
Send (EO_i ,N_2+2);

Step 9: Send a rejection message to the User
Send ("User's request cannot be granted);

Figure 9. Algorithm for secure data exchange between CSP and User using D-H key exchange.

© 2010, IJARCS All Rights Reserved

This is to avoid the use of D-H key exchange for every data access request. The user upon receiving an encrypted response from the CSP again calculates the digest by using the hash function. The newly calculated digest is then compared with the digest that is attached with the message to check the integrity of the message. This is described in Fig.10.

V. ANALYSIS OF PROPOSED SCHEME

A. Security Analysis:

In this section, we analyse security properties of our proposed scheme and also, the performance in terms of scalability and strength of cryptographic primitives. The following properties are analyzed first.

Step 1: User receives the data items from the CSP
  Storage Array ← Recv (EOi) ;
Step 2: User decrypts the data items using K_S and K_O.
  2.1 Decrypt using Session Key K_S
    Oi←DK_S {Storage Array} ;
  2.2 Decrypt using Symmetric Key K_O
    (fi, Di)←DK_O (Oi);
Step 3: User calculates message digest of original file
    NDi ← MD5 (fi);
Step 4: Compare the new digest NDi, and old digest, Di that is attached with the file.
  If (ND_i != D_i)
    {Discard data;
     Send error report to Data Owner ;
     Goto step 5;}
  Else
    { Process the data;
     Goto step 5;}
  End if
Step 5: Stop

Figure 10. Algorithm for response check by the user

a. Data confidentiality: We analyse data confidentiality of our proposed scheme by comparing it with standard encryption algorithms like Data Encryption Standard or Advanced Encryption Standard that use symmetric keys. As described in [22], how does a data owner merge cloud security data with its' own security metrics and policies? We attempt to give an answer to this using our proposed over encryption scheme. The CSP is not able to know the owners data and also the digest due to the fact that both are encrypted and the key is only shared between data owner and user.

The over encryption (double encryption) is described by the following in our scheme:Oi->EKO (fi , Di) , followed by EOi -> EKS (Oi). The EKO is one symmetric key that is known to DO and user, hence Oi is non-intelligible to CSP. Further, EKS is the session key between CSP and user, hence no one else even knows Oi. By employing over encryption, the key length is increased and hence brute-force attack becomes difficult on the cipher. In Fig.11, we plot a graph showing the strength of our double encryption scheme over other standard ciphers like symmetric, asymmetric etc. Although DES and Public key algorithms like RSA, DSS etc are used for different purposes, because of large key sizes, we have shown in Fig.11. that the Public key ciphers are stronger than DES. Our proposed scheme only discloses the capability list (CapList) to the cloud service provider using which the access control is guaranteed.

b. Authentication and Integrity: The communications from DO to CSP is authenticated by encrypting the scrambled data files and capability list using the private key of owner. This is described in Steps 1.2, and 3 in Fig.4. At the time of adding a new user, user is authenticated at owner by signing with his private key, and also data owner is authenticated at CSP by signing with his private key. This is shown in Fig.6. Integrity of the data file is ensured by using MD5 as the hashing algorithm. User computes a new hash and compares it with the one created by owner and stored in CSP. If both do not match, then integrity violation is reported and a message is sent to the data owner.

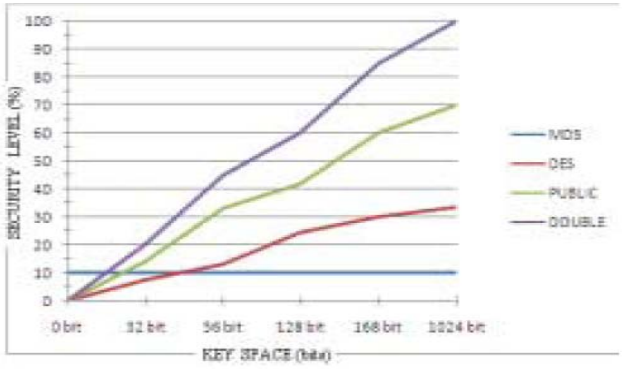

Figure 11.

c. Capability-based Access Control: In our proposed scheme, only the data owner is able to create, modify or delete an appropriate capability from the CapList for a user to access a data file. Earlier schemes [14-15] have used Access Control Lists (ACLs) for access control where as in our work, we propose Capabilities as the data structure for controlling access to data files.

Capabilities are row decomposition of a Access Matrix (AM), and hence are more appropriate to individual users in the application scenario as against the column decomposition of AM done in ACLs case. In a cloud computing environment, creating an ACL for an object for the purpose of access control may not be practicable as we may find in most probability the data files accessed by one user may not be needed by another. This is the reason that why we selected capabilities for access control rather than ACLs or *filegroups* as is done in literature.

B. Performance Analysis:

a. Capabilities as Addresses in the Cloud: As reported in [23], in addition to access control, capabilities can also be used as addressing mechanism. There is a substantial advantage in using capabilities as a basic component of the address of every data file that is stored at the servers in CSP. The CSP uses capabilities to index into Object list or table as shown in Fig.4, where it maintains a pointer to the Service provider (one out of many in the conglomeration) who actually stores the data file UID which is unique for every capability is used as an index to search the CapList which points to an Object table that stores length of

© 2010, IJARCS All Rights Reserved

the object and an base address of the object that points to a specific service providers' domain.

This enhances the addressability or identification of the encrypted data file in the Cloud.

b. ***Efficient Data Access:*** In our approach, the data owner need not be online always. Our approach is flexible in the sense that it can create, add, and delete capabilities as and when required. We plot the statistics of computation complexity for Public key encryption and D-H key exchange protocols, which is used in our scheme to send the data to user by a CSP. As the public key encryption uses a key space of 1024 bits, the computation complexity is high. When the key space of encrypted data i.e. Oi is increased, complexity of sending it using public key is further increased to 99.3%. For the same case, by using DH key exchange we can send the encrypted data (Oi) to the user with less complexity i.e. 27.98% at max. We plot the graph on 1200 point scale and then reduced it to 100 point scale. Also, the user can get the original file and the digest only by decrypting the data items with secret session key that it shares with CSP using D-H key exchange. This in turn helps both CSP and the user to send and receive multiple data files for a certain amount of period that can be agreed upon by both the parties priori.

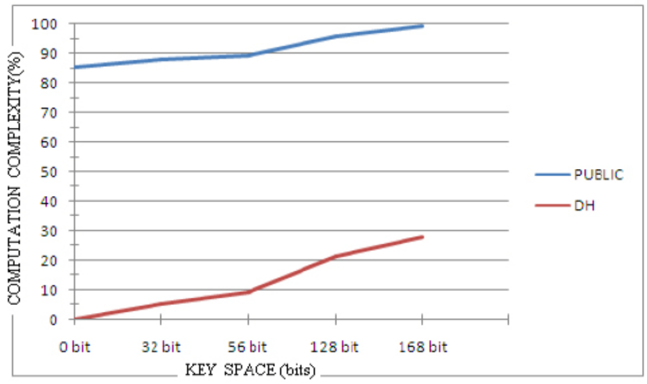

Figure 12.Computational complexities of public key and D-H key exchange ciphers

VI. SIMULATION

In this section, we describe our simulation run. All the servers and clients are created using Java RMI. We set up the Cloud Server and Data Owner Server at Amazon Web Service [25][26] cloud and Clients at various places on separate IBM Lenovo machines with Intel[R] Core 2 Duo CPU with the speed of 2.90GHz having 2 GB of RAM. The machine as running Microsoft Windows XP, Service Pack 2 operating system. Our experiments are carried out in Aamazon ec2 Server with a single CPU, 512 MB RAM, 20 GB Hard Disk, and 64-bit Windows 2008 Operating system. We used Java RMI because of the fact that the methods of remote Java objects can be invoked from other Java virtual machines, possibly on different hosts. We deployed the Cloud Server and Data Owner server at Amazon web server and Data Owner client processes can access the

Data Owner Server to get the updates, Users can make new request to Data Owner Server and also Data Access request to Cloud Server. The Data Owner has the flexibility to come online at any time and check the updated capability list. Data Owner can make changes to capability list depending on the availability of the resources. We ran all our processes as per the scheme defined in Section IV and then we intercepted the messages that went between DO, CSP, and user. Few of our screen shots are as given below.

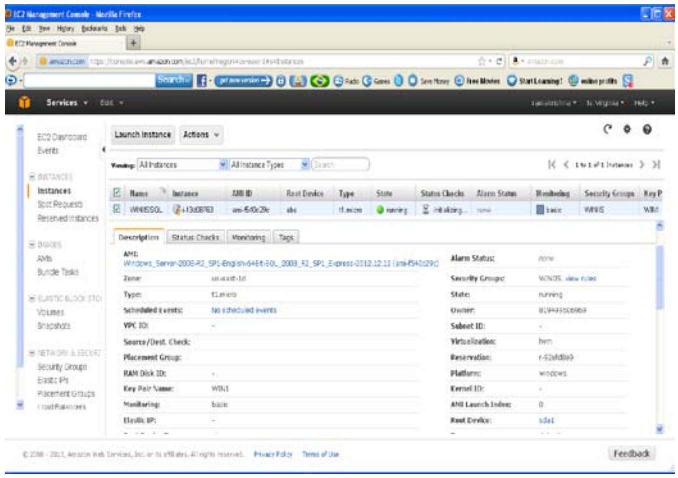

Figure 13:Setup of server in Amazon Web services.

We deployed our Cloud server code at Amazon ec2, and then ran the Owner server also at Amazon cloud. Fig.13 shows the run of both these servers at FlexiScale cloud. We started several clients at three of the laboratories at VYCET,Chirala Campus in India to access the servers running at Amazon web services cloud whose data centre was situated at US.EAST. The servers were running at IP address 109.233.76.202. Upon receiving the new request the Data owner sent the updated Capability list to the Cloud server running at Amazon Web services. As per our protocol, the client then sent a data request to Cloud server and received an encrypted response from Cloud server after access control is verified through the capability list. Fig.12 shows the encrypted response.

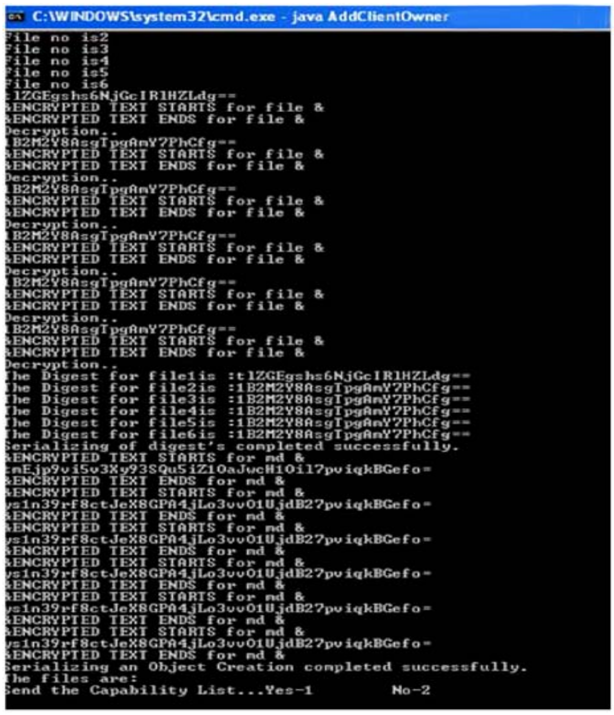

Figure 14: Illustration of Remote client getting data from Cloud Server.

© 2010, IJARCS All Rights Reserved

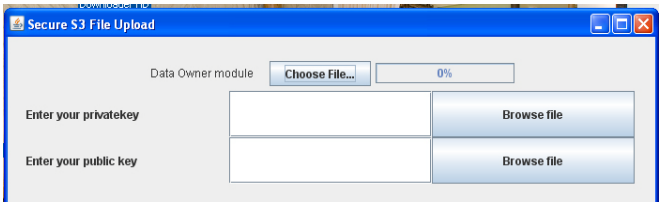

Figure 15: Illustration of Security APIs at Data owner.

To verify the communication between Data owner and Cloud server running at Amazon Web service, we took a snapshot at Data owner as shown in Fig.15.

Figure 16: Network watch on instance

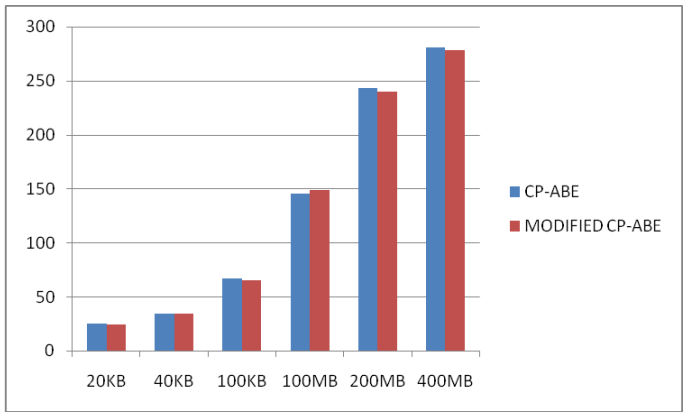

Figure 17 Performance comparison

Figure 17 shows the performance comparison with existing CP-ABE to modified CP-ABE time complexities when normalized with a factor of multiplication modified CP ABE shows the better results in implementation.

VII. CONCLUSION

The work done in this paper by adding user access part to the data stored in Cloud servers, and also deploying our applications over a real-time cloud environment given by Amazon web services. In this paper, we presented a set of security protocols to secure the data files of a data owner in the cloud infrastructure. In our proposed scheme, the combined approach of access control and cryptography is used to protect the outsourced data.

We use the capability based model for access control mechanism along with public key encryption. A D-H key exchange model is proposed for the users to access the outsourced data efficiently and securely from cloud service providers' infrastructure. The D-H protocol fits better as we have assumed that the CSP does not have the public key of user which is otherwise valid in a cloud set up where the number of users normally handled by providers is very large and key management becomes a complex issue in this scenario. The public key, hash, and private key ciphers that are proposed between cloud service provider, data owner, and user ensure an isolated and secure execution environment at the cloud. This paper also presented a proof of concept implementation of the cryptographic algorithms in a Cloud computing environment using Java RMI.

Our proposed scheme empowers the data owner to outsource the security enforcement process on the outsourced data files without losing control over the process. Moreover, our scheme can also delegate most of the computation overhead to Cloud servers. Future extensions will include enhancement in design decisions like inclusion of a trusted third party auditor which will have capabilities of assessing and exposing Cloud service risks, key management and distribution scenarios, and formal security proofs of our security protocols.

VIII. REFERENCES

[1]. Gartner survey feb 2013: from http://www.slideshare.net/GaldeMerkline/pwc-cloudenabledtelcoopportunitiespdf

[2]. Peter Mell, and Tim Grance, Draft NIST Working Definition of Cloud Computing, 2009: from http://csrc.nist.gov/groups/SNS/cloud-computing/

[3]. R. Buyya, C. S. Yeo, and S. Venugopal, Market oriented cloud computing: vision, hype, and reality, for delivering IT services as computing utilities, *Proc*. 10th IEEE International Conference on High Performance Computing and Communications, Dalian, China, Sept 2008.

[4]. M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, Above the clouds: A berkeley view of cloud computing, University of California, Berkeley, Tech Rep USB-EECS-2009-28, Feb 2009.

[5]. David Chappell, Introducing the Azure Service Platform, White paper, Oct 2008.

[6]. Amazon EC2 and S3, Online at http://aws.amazon.com/

[7]. Google App Engine, Online at : http://code.google.com/appengine/

[8]. S. D. C. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, A Data Outsourcing Architecture Combining Cryptography and Access Control, *Proc*. ACM Workshop on Computer Security Architecture (CSAW'07), Nov 2007, USA.

[9]. S. Yu, C. Wang, K. Ren, and W. Lou, Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing, *Proc*. IEEE INFOCOM 2010, San Diego, CA, pp. 1-9.

[10]. W. Wang, Z. Li, R. Owens, and B. Bhargava, Secure and efficient access to outsourced data, *Proc*. ACM Cloud

© 2010, IJARCS All Rights Reserved

Computing Security Workshop 2009, Chicago, Illinois, USA, 2009, pp. 55-65.

[11]. S. Kamara, and K. Lauter, Cryptographic Cloud Storage, *Proc*. Financial Cryptography: Workshop on real life cryptographic protocols and standardization, 2010: from http://research.microsoft.com/pubs/112576/cryptocloud.pdf

[12]. Z. Dai, and Q. Zhou, A PKI-based Mechanism for Secure and Efficient Access to Outsourced Data, *Proc*. International Conference on Networking and Digital Society, Wenzhou, China, 2010, pp. 640-643.

[13]. J. Anderson, Computer Security Technology Planning Study, Air Force Electronic Systems Division, report ESD-TR-73-51, 1972: from http://seclab.cs.ucdavis.edu/projects/history/

[14]. G. Ateniese, K. Fu, M. Green, and S. Hohenberger, Improved proxy re-encryption schemes with applications to secure distributed storage, ACM Transactions on Information and System Security, Vol. 9, No. 1, Feb 2006, pp. 1-30.

[15]. S. D. C. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, Over-encryption: Management of access control evolution on outsourced data, *Proc*. 33rd International Conference on Very Large Databases (VLDB'07), Vienna, Austria, 2007, pp. 123-134.

[16]. G. Miklau, and D. Suciu, Controlling access to published data using cryptography, *Proc*. 29th VLDB, Germany, Sept 2003, pp. 898-909.

[17]. E. Goh, H. Shacham, N. Modadugu, and D. Boneh, Sirius: Securing remote untrusted storage, *Proc*. Network and Distributed Systems Security Symposium (NDSS'03), San Diego, California, USA, 2003, pp.131-145.

[18]. Dalit Naor, A. Shenhav, and A. Wool, Toward securing untrusted storage without public-key operations, *Proc*. 2005 ACM Workshop on Storage Security and Survivability (StorageSS), Virginia, USA, Nov 2005, pp. 51-56.

[19]. Rolf Blom, An optimal class of symmetric key generation systems, *Proc*. EUROCRYPT 84 workshop on Advances in cryptology: theory and application of cryptographic techniques, Springer Verlag, NY, USA, 1985, pp. 335-338.

[20]. L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, A Break in the Clouds: Towards a Cloud Definition, *Proc*. ACM SIGCOMM Computer Communication Review, 39(1), Jan 2009, pp. 50-55.

[21]. L. Youseff, M. Butrico, and D. D. Silva, Toward a Unified Ontology of Cloud Computing, *Proc*. Grid Computing Environments Workshop 2008 (GCE'08), Austin, TX, USA, Nov 2008, pp. 1-10.

[22]. R. Chow, P. Golle, M. Jakobsson, E. Shi, J. Staddon, R. Masuoka, and J. Molina, Controlling data in the cloud: outsourcing computation without outsourcing control, *Proc*. ACM Cloud Computing Security Workshop 2009 (CCSW'09), Chicago, USA, 2009, pp. 85-90.

[23]. R. S. Fabry, Capability-Based Addressing, Communications of the ACM, 17(7), July 1974, pp. 403-412.

[24]. Mukesh Singhal, Niranjan G.Shivaratri, Advanced Concepts in Operating Systems, Tata McGraw-Hill Edition, 2001.

[25]. Amazon Elastic Compute Cloud (Amazon EC2) [Online]. Available: http://aws.amazon.com/ec2/

[26]. Amazon Web Services (AWS) [Online]. Available: https://s3.amazonaws.com/

[27]. Google Trends.Available: http://www.google.com/trends/

Short Bio Data for the Authors

S.Rama Krishna M.Tech is working as associate professor in the Department of computer Science & Engineering**,** VYCET,chirala.

Dr. B Padmaja Rani M.Tech,Ph.D is Working as Professor & Head of department in Computer Science & Engineering at JNTUH College of Engineering Hyderabad (Autonomous). She has done extensive research in the areas like Information Retrieval Embedded Systems. Official Email: padmaja_jntuh@jntuh.ac.in

© 2010, IJARCS All Rights Reserved