

Capability-based Cryptographic Data Access Control in Cloud Computing

Chittaranjan Hota, Sunil Sanka

Department of Computer Science & Information Systems, BITS Pilani Hyderabad Campus, Hyderabad
Email: hota@bits-hyderabad.ac.in, sunil.grt@gmail.com

Muttukrishnan Rajarajan

Information Security Group, School of Engineering & Mathematical Sciences, City University, London
Email: r.muttukrishnan@city.ac.uk

Srijith K. Nair

Security Futures Practice, BT Innovate and Design, British Telecommunications, Suffolk IP5 3RE, UK
Email: srijith.nair@bt.com

ABSTRACT

Cloud computing has emerged as a popular model in computing world to support processing large volumetric data using clusters of commodity computers. It is the latest effort in delivering computing resources as a service. It is used to describe both a platform and a type of application. A cloud computing platform dynamically provisions, configures, and deprovisions servers as needed. Cloud computing also describes applications that are extended to be accessible through the Internet. Data security and access control is one of the most challenging ongoing research work in cloud computing, because of users outsourcing their sensitive data to cloud providers. Existing solutions that use pure cryptographic techniques to mitigate these security and access control problems suffer from heavy computational overhead on the data owner as well as the cloud service provider for key distribution and management. This paper addresses this challenging open problem using capability based access control technique that ensures only valid users will access the outsourced data. This work also proposes a modified Diffie-Hellman key exchange protocol 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 analysis shows that the proposed approach is highly efficient and secure under existing security models.

Keywords - Cloud computing, Cryptography, Access control, Capability, Security.

Date of Submission: July 14, 2011

Date of Acceptance: August 23, 2011

I. INTRODUCTION

Cloud computing has become a necessity now days when an enterprise plans to increase its' capacity or capabilities on the fly without investing on new infrastructure, training new personnel, buying new software licenses etc. It encompasses any subscription-based or pay-per-use service that extends the enterprise's existing IT capabilities, in real-time over the Internet. The market research and analysis firm IDC suggests that the market for cloud computing services was \$16bn in the year 2008 and will rise to \$42bn/year by 2012 [1]. The US National Institute of Standards and Technology (NIST) defines cloud computing as "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" [2]. Cloud computing can also be defined as "a type of parallel and distributed system consisting of a collection of interconnected and virtualized

computers that are dynamically provisioned and presented 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 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.1.

The model described in Fig.1 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 owners must 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.

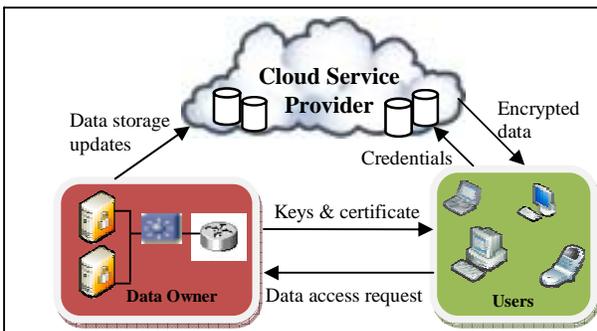


Figure 1. Example of a secure data access.

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 re-encryption, 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 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 alongwith 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.2 illustrates our scheme by an example. Here, the owner can be a university teacher who posts the students' grades into the cloud and user can be any university student registered into the same course who views his/her grades 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 alongwith 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.

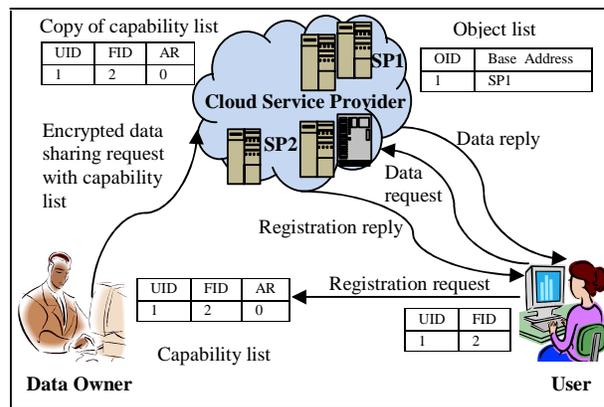


Figure 2. An exemplary case in education scenario.

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	Private Key of User
PUOWN	Public Key of Owner
PROWN	Private Key of Owner
DS _i	i _{th} Data Set
f _i	i _{th} file
D _i	i _{th} file Message Digest
ND _i	New Message Digest for i _{th} file
O _i	i _{th} Object
EO _i	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
<u>Diffie –Hellman Parameters</u>	
$X_{A/B}$	Chosen Secret Key
$Y_{A/B}$	Calculated Public Key
q	Prime number
p	Primitive root
K_S	Secret session key
mod	Modulus operation.

Figure 3. Notations used in the proposed scheme.

```

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

Figure 4. Algorithm for DO sending encrypted outsourced data items and capability list to CSP.

Fig.5 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 does not allow the CSP to know the actual data items as it does not know the K_o 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  $\leftarrow$  Receive  $(EO_i, EK_{PUSP}(EK_{PROWN}(CapList)))$ 
StorageArray[1]  $\leftarrow (EO_i)$  ;
Storage Array[2]  $\leftarrow (CapList)$  ;
Step 2: Updating the Object Table List
 $O_i \leftarrow DK_{PRSP}(DK_{PUOWN}(Storage Array[1]))$  ;
ObjTable  $\leftarrow (O_i, SP_i)$  ;
Step 3: Update Capability List
CapList  $\leftarrow DK_{PRSP}(DK_{PUOWN}(Storage Array[2]))$  ;
    
```

Figure 5. Algorithm for CSP receiving and storing both encrypted data 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 sends 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 EK_{PUUSR} . 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  $\leftarrow$  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  $(EK_{PUUSR}(EK_{PROWN}(EK_{PUUSR}(EK_O, MD5, N_1+1, TimeStamp))))$  ;
Step 5: Now the user decrypts the message and is aware of the symmetric key and hash functions used by the owner.
    
```

Figure 6. 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.7 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 O_i which is an encrypted version of file (f_i) and its' digest (D_i) 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. 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.8.

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 analysed first.

```

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 YA

3.1 Choose a private key say, XA and q, p.
YA ← pXA mod q.

Step 4: CSP sends the parameters to User without
encrypting

Send (YA , q, p, N2);

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

5.1 StorageArray ←Recv(YA, q, p, N2);
5.2 YA,q,p,N2 ← StorageArray;
5.3 Generates his private key XB and calculates YB,
and shared secreta key as follows
YB ← pXB mod q
Session Key KS ← YAXB mod q
5.4 User now sends the global parameter YB to CSP
encrypted
Send (EKPUSP (YB, N2+1));

Step 6: CSP calculates the shared secret key using YB
and XA

6.1 StorageArray ← Recv(EKPUSP (YB,N2+1) );
6.2 YB,N2+1 ←DKPRSP(StorageArray);
6.3 Session key KS←YBXA mod q

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

Step 8: CSP sends the encrypted data to the User
Send (EOi,N2+2);

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

```

Figure 7. Algorithm for secure data exchange between CSP and User using D-H key exchange.

```

Step 1: User receives the data items from the CSP
Storage Array ← Recv ( EOi ) ;
Step 2: User decrypts the data items using KS and KO.
2.1 Decrypt using Session Key KS
Oi←DKS {Storage Array} ;
2.2 Decrypt using Symmetric Key KO
(fi , Di)←DKO ( 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 ( NDi != Di )
  {Discard data;
  Send error report to Data Owner ;
  Goto step 5;}
Else
  { Process the data;
  Goto step 5;}
End if
Step 5: Stop

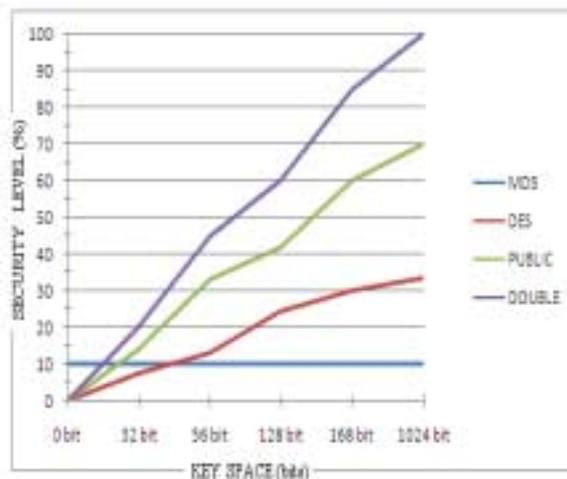
```

Figure 8. Algorithm for response check by the User

1) *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: $O_i \leftarrow EK_O (f_i, D_i)$, followed by $EO_i \leftarrow EK_S (O_i)$. The EK_O is one symmetric key that is known to DO and user, hence O_i is non-intelligible to CSP. Further, EK_S is the session key between CSP and user, hence no one else even knows O_i . By employing over encryption, the key length is increased and hence brute-force attack becomes difficult on the cipher. In Fig.9, 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.9 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.

2) *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.



3) *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

1) *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.2, 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 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.

2) *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. O_i is increased, complexity of sending it using public key is further increased to 99.3%. For the same case, by using D-H key exchange we can send the encrypted data (O_i) 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.

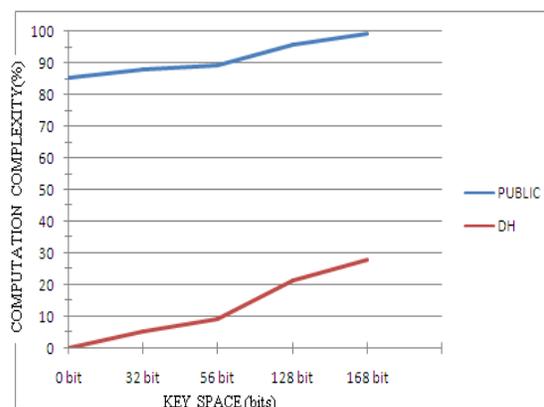


Figure 10. Computational complexities of public key and D-H key exchange ciphers

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 FlexiScale [25] 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 1.96 GB of RAM. The machine was running Microsoft Windows XP, Service Pack 3 operating system. Our experiments are carried out in FlexiScale 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 FlexiScale and

images increases, the data access time reduces to few milli-seconds as shown with 28 and 56 VMs with say, 9 number of disk images.

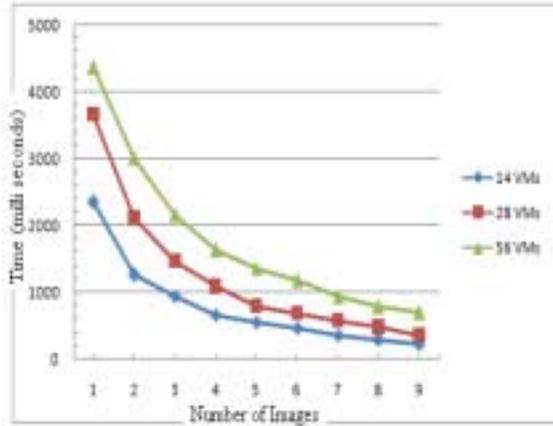


Figure 14: Disk images versus Data access time

As we access the servers located remotely via the Internet, speed of the network also plays an important role in increasing or decreasing the application access time. Initially we have experienced the communication overhead, as the network bandwidth lies around 100Mbps as shown in Fig.15. The cloning of VMs for better results depend on the network bandwidth. With the increase in bandwidth from 100 Mbps to 1Gbps, we get a reduction in the application access time from 360 milli-seconds to 80 milli-seconds as shown in both Fig. 15 and 16. Hence, we observed that by increasing the bandwidth, cloning of many servers at the Cloud also takes less time.

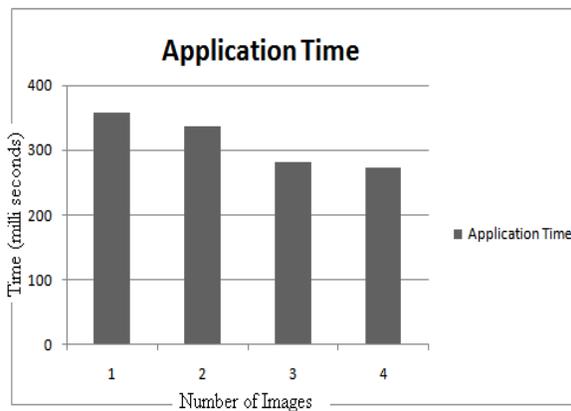


Figure 15: Illustration of varying large application time with Low-bandwidth (100Mb).

The network bandwidth will also affect the image distribution time at the cloud end. As we have experimented with the large number of disk images, access to those will become easier with high bandwidth. So the distribution time will also reduce with high network bandwidth, which in turn reduces the overall Cloud time.

We got varying access times with low network bandwidth as shown in Fig.15. But, as we increased the network bandwidth, the application access time decreased as shown in Fig.16.

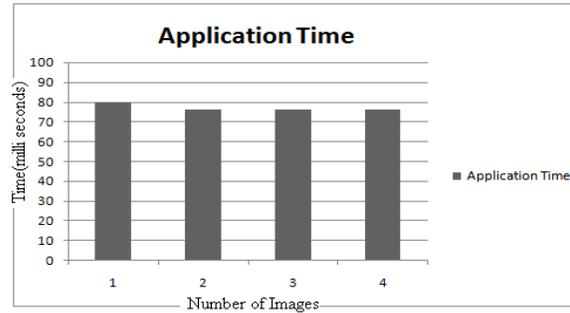


Figure 16: Illustration of the application access time with high network access time (1 GB).

VII. CONCLUSION

The work done in this paper is an extension of our earlier work on secure data access in Cloud computing that is published in IEEE IMSAA 2010 conference held at Bangalore. We extended our earlier work 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 Flexiscale. 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.

ACKNOWLEDGEMENTS

The authors would acknowledge the support from Flexiant Ltd, UK in providing a real cloud environment to demonstrate our access control mechanism in a seamless manner.

REFERENCES

- [1] Eoin Gleeson, Computing industry set for a shocking change, MoneyWeek, April 2009: from <http://www.moneyweek.com/investment-advice/computing-industry-set-for-a-shocking-change-43226.aspx>
- [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 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/crypto-cloud.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. Suci, 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, Niranjana G. Shivaratri, Advanced Concepts in Operating Systems, Tata McGraw-Hill Edition, 2001.

[25] Flexiant available from:
<http://www.flexiant.com/products/flexiscale/>

Authors Biography



Chittaranjan Hota is Associate Professor and Head, Department of Computer Science and Information Systems at Birla Institute of Technology and Science, Pilani Hyderabad Campus, Hyderabad. He has earned his Bachelors, Masters and PhD degrees in Computer Science and Engineering from Amravati University, TIET (deemed), and BITS, Pilani (deemed). He had visited UNSW, Sydney; City University, London; HIIT, Finland; and UNICA, Cagliari, Italy as visiting professor to carry out joint teaching and research over past several years. He has published extensively in International conferences and journals. He is a life member of ISTE, and an ACM member. His research interests are in the areas of Traffic Engineering in IP Networks, Security and Quality of Service issues over the Internet, Peer-to-Peer Overlays, Mobile Ad-Hoc Networks, Cloud Computing, and Machine Learning.



Sunil Sanka has completed his Masters degree in Computer Science & Engineering from BITS Pilani Hyderabad Campus, Hyderabad in July 2011. He has presented a paper on cloud computing in IEEE IMSAA 2010 at Bangalore. Currently he is working as a Software Engineer at Computer Associates, Hyderabad.



M Rajarajan received his BEng and PhD degrees from City University London in 1994 and 1999 respectively. He worked at Logica as a Telecommunication consultant in 2000, and then joined City University London as a Lecturer in January 2002 where he is now a Reader (Associate Professor) and also the Assistant Dean for E-Learning initiatives. At City he now leads the research activities of Information security group. He is also a member of Mobile Networks research group where he is investigating secure delivery of data over the air interface. He has organized several conferences and published in journals and conferences. He is a senior member of IEEE and a member of IET, UK. He is also in the editorial board of Springer wireless networks, Journal of information management and Computer security, and International journal of Healthcare.

Srijith K. Nair earned his PhD degree in Computer Science from Vrije University, Amsterdam, The Netherlands in the year 2010. Currently he is a Senior Researcher in the Security Futures Practice of BT I&D. He

joined BT in the beginning of 2009 and has been devoting much of his time on investigating strategic and architectural, in particular security, aspects of cloud computing. He has been part of the ENISA expert advisory group working on Cloud Security Risk Analysis and also part of the group that helped draft various sections of the Cloud Security Alliance's Guidance document. He has published in several peer reviewed conference and journals.