International journal of Engineering Research-Online
A Peer Reviewed International Journal
Articles available online http://www.ijoer.in

Vol.2., Issue.5, 2014

RESEARCH ARTICLE

ISSN: 2321-7758

IDENTITY-BASED SECURE DISTRIBUTED DATA STORAGE SCHEMES

NAGANDLA SUMAN[1], P. KESAVARAO[2], K. PRASANTH KUMAR[3]
[1]M.Tech Post Graduate in JNTUH, Hyderabad ,
[2]Engineer and Research Scholar of IARE College, Hyderabad,
[3]Associate Professor in Computer Science and Engineering, JJIT College,Hyderabad

Article Received: 09/10/2014 | Article Revised on: 19/10/2014 | Article Accepted on:22/10/2014

NAGANDLA SUMAN

ABSTRACT

Identity based distributed data storage[1] Scheme divide the load of maintaining a large number of files from the owner to proxy servers. Currently, the only method for enforcing such policies is to employ a trusted server to store the data and mediate access control. However, if any server storing the data is compromised, then the confidentiality of the data will be compromised Proxy servers can convert encrypted files for the owner to encrypted files for the receiver without the necessity of knowing the content of the original files. In practice, the original files will be removed by the owner for the sake of space efficiency. Hence, the issues on confidentiality and integrity of the out-sourced data must be addressed carefully. The project contains two identity-based secure distributed data storage schemes. The scheme create private key and store on the server. For one query, a receiver can only access one file, instead of all files of the owner. Our schemes are secure against the collusion attacks, namely even if the receiver can compromise the proxy servers, he cannot obtain the owner's secret key. Although the first scheme is only secure against the chosen plaintext attacks (CPA), the second scheme is secure against the chosen cipher text attacks (CCA). It is the first schemes where access permissions is made by the owner for an exact file and collusion attacks can be protected in the standard model.

In an identity-based secure distributed data storage scheme, a user's identity can be an arbitrary string and two parties can communicate with each other without checking the public key certificates. At first, the file owner encrypts his files under his identity prior to outsourcing them to proxy servers. The File Creator transmit the cipher-texts to the proxy servers. The proxy servers can transfer a cipher-text encrypted under the identity of the owner to a cipher-text encrypted under the identity of the receiver after they have obtained an access permission (re-encryption key) from the owner.

Keywords: Distributed, IBSDDS

INTRODUCTION

Cloud computing provides users with a convenient mechanism to manage their personal files with the notion called database-as-a-service (DAS). In DAS schemes, a user can outsource his encrypted files to un-trusted proxy servers. Proxy servers can perform some functions on the outsourced cipher-texts[2] without knowing anything about the original files. Furthermore, how to guarantee that an authorized user can query the outsourced files from proxy servers is another concern as the proxy server only maintains the outsourced cipher texts. After out-sourcing the files to proxy servers, the user will remove them from his local machine. Therefore, how to guarantee the out-soured files are not accessed by the unauthorized users and not modified by proxy servers is an important problem that has been considered in the data storage research community.
The out-sourced files are not accessed by the unauthorized users and not modified by proxy servers is an important problem that has been considered in the data storage research community.

1. Data Storage Schemes

The Project is aimed two identity-based secure distributed data storage schemes in standard model where, for one query, the receiver can only access one of the owner's files, instead of all files.

A secret key algorithm (sometimes called a symmetric algorithm) is a cryptographic algorithm that uses the same key to encrypt and decrypt data. In other words, access permission (re-encryption key) is bound not only to the identity of the receiver but also the file. The access permission can be decided by the owner, instead of the trusted party (PKG). Furthermore, the schemes are secure against the collusion attacks.

2. Benefits of the System

It has two schemes of security, the first scheme is CPA secure, and the second scheme achieves CCA security[4].
To the best of our knowledge, it is the *first* IBSDDS schemes where access permission is made by the owner for an exact file and collusion attacks can be protected in the standard model.

- Unidirectional: For example after receiving access permission, the proxy server can transfer a cipher-text for A to a cipher-text for B while he cannot transfer a cipher-text for B to a cipher-text for A.
- Non-interactive: The access permission can be created by the file owner without any trusted third party and interaction with him.
- Key optimal: The size of the secret key of the receiver is constant and independent of the delegations which he accepts.
- Collusion-safe: The secret key of the file owner is secure even if the receiver can compromise the proxy server.
- Non-transitive: Receiving the access permissions computed by A for B and B for C, the proxy server cannot transfer a cipher-text for A to a cipher-text for C.
- File-based access: For one query, the receiver can only access one file. This can improve the security of the out-sourced files and is desirable to maintain the access record. Proxy invisibility discussed in n on-transitive is difficult to achieve as the length of the re-encrypted cipher-text is subject to be different from that of the original cipher-text.

To achieve a stronger security and implement file based access control, the owner must be online to authenticate requesters and also to generate access permissions for them. Therefore, the owner in our schemes needs do more computations than that in PRE schemes. Although PRE schemes can provide the similar functionalities of our schemes when the owner only has one file, these are not flexible and practical.

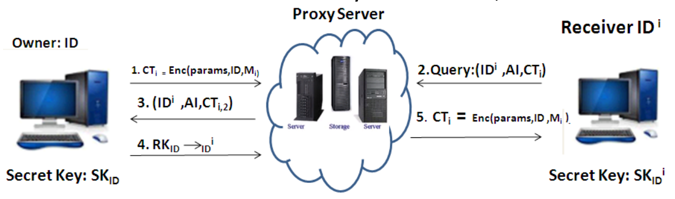

Proxy Server
Receiver ID^i
Owner: ID
1. CT_i = Enc(params,ID,M_i)
2.Query:(ID^i ,AI,CT_i)
3. (ID^i ,AI,$CT_{i,2}$)
4. $RK_{ID} \rightarrow_{ID^i}$
5. CT_i = Enc(params,ID ,M_i)
Server
Storage
Server
Secret Key: SK_{ID}
Secret Key: SK_{ID^i}

Fig 1. The Model of Identity-Based Secure Distributed Data Storage Scheme

3. ALGORITHMS[3] USED

The setup algorithm will choose a bilinear group G_0 of prime order p with generator g. Next it will choose two random exponents α, β ∈ Z_p. The public key is published as:
PK = G0, g, $h = g^{\beta}$, $f = g^{1/\beta}$, $e(g, g)^{\alpha}$ and the master key MK is (β, g^{α}). (Note that f is used only for delegation.)

Encrypt (PK,M, T) The encryption algorithm encrypts a message M under the tree access structure T . The algorithm first chooses a polynomial q_x for each node x (including the leaves) in the tree T. These polynomials are chosen in the following way in a top down manner, starting from the root node R. For each node x in the tree, set the degree d_x of the polynomial q_x to be one less than the threshold value k_x of that node, that is, dx = $k_x - 1$.
Starting with the root node R the algorithm chooses a random s ∈ Zp and sets qR(0) = s.
Then, it chooses dR other points of the polynomial qR randomly to define it completely. For any other node x, it sets $q_x(0)=q_{parent(x)}(index(x))$ and chooses dx other points randomly to completely define q_x.
Let, Y be the set of leaf nodes in T . The cipher text is then constructed by giving the tree access structure T and computing
CT = (T , ˜ C = $Me(g, g)^{\alpha s}$, C = h^s,
$\forall_y \in Y$: $C_y = g^{qy(0)}$), $C'_y = H(att(y))^{qy(0)}$).
KeyGen (MK, S). The key generation algorithm will take as input a set of attributes S and output a key that identifies with that set. The algorithm first chooses a random r ∈ Zp, and then random r_j∈ Zp for each attributes j ∈ S.
Then it computes the key as SK = (D = $g^{(\alpha+r)/\beta}$,
$\forall j \in S$: $D_j = g^r \cdot H(j)^{rj}$, $D'j = g^{rj}$). This algorithm takes as input the security parameter li and outputs a bilinear group $GG(1^e) \rightarrow (e,p, G_i , G_r)$ with prime order p where e: G X G->G_r . Let g, h, n, g and h be the generators of G, $u_0 \leftarrow G$ and $U=(u_1,u_2,...,u_n)$ where $u_i \overset{R}{\leftarrow} G$ for i=1,2,...,n . It chooses $\alpha \overset{R}{\leftarrow} z_p$ and sets $g_1=g^\alpha$ and $g_2= g^\alpha$. It generates an one time signature scheme $sG(1^e) \rightarrow$(SKeyGen,Sign,Verify), where $SKeyGen(1^e) \rightarrow (sk,vk)$. Let H: $vk \rightarrow Z_p$ be a hash function. The public parameters are (e,p,G,Gr,g,h,n,g,h,u0,g1,g2,U,H,Sign,Verify) and the master secret keys is η^α.
KeyGen: Let ID denote an identity which is an n bit string, ID_i be the ith bit of ID and Z be the set which consists of all the index i with ID_i = 1. This algorithm takes as input the master secret key η^α and the user's identity ID, and computes
$K_{ID,1}= \eta^\alpha{}_{(uo \frac{\pi}{i \varepsilon I}} u_i)^r{}_{ID}$,KID,2 $=g^{rID}$ and $K_{ID},3= g^{rID}$.
The secret key for the user is $SK_{ID,1}=(K_{ID,1}, K_{ID,2}, K_{ID,3})$. This secret key can be verified by
$e(K_{ID,1,g}) =e(\eta,g1).e(u_0 \frac{\pi}{i \varepsilon I} u_i),K_{ID,2})$ and $e(K_{ID,2,g}) \overset{?}{=} e(g,K_{ID,3})$.
Encryption: Suppose that there are k message M_i ε $\{M_1,M_2,...M_k\}$. To encrypt M_i the owner runs $SKeyGen(1^e) \rightarrow (sk,vk)$, chooses $s_i \overset{R}{\leftarrow} z_p$ and computes
$C_{i,1}=M_i.e(g_1, \eta)^{ni}, c_{i,2}=g^{ni}$, $C_{i,3} \overset{=}{} (u_0 \frac{\pi}{i \varepsilon I} u_i)^l$, $C_{i,4}=(g^{H(vk)}g)^{ni}$
And $б_i$ = Sign(sk, $C_{i,2}$ $C_{i,3}$, $C_{i,4}$)
for i=1,2,........,k. The cipher-text for the message M_i is CT_i =($C_{i,1}$,$C_{i,2}$,$C_{i,3}$, $C_{i,4}$, $б_i$,vk). the owner sends $\{CT_1,CT_2,.......,CT_k\}$ to the proxy servers. The proxy servers (PSs) validate the cipher-texts by checking
$б_i \overset{?}{=}$ Verify(vk, $C_{i,2}$ $C_{i,3}$, $C_{i,4}$), $e(u_0 \frac{\pi}{i \varepsilon I} u_i), C_{i,2}) \overset{?}{=} e(C_{i,2} ,g)$ and $e(g,C_{i,4}) \overset{?}{=} e(C_{i,2} ,g^{H(vk)}g))$ for i=1,2,........,k, If the equation holds, the proxy servers store the cipher-text $CT_i=(C_{i,1},C_{i,2},C_{i,3}, C_{i,4}, б_i,vk)$ for the owner. Otherwise, the proxy servers reject the cipher-texts.
Query: If a receiver R with identity ID^i wants to access CT_i he chooses $t \overset{R}{\leftarrow} Z_p$, and computes $K_{ID,1} = K_{ID,1}\eta e$ and $\acute{\Gamma}=g^t$ .He sends (ID,K_{ID} $K_{ID},_3,\acute{\Gamma}$) to the proxy server. Then, the proxy server redirect (ID' ' $K_{ID},_3$, $K_{ID},_3$,$\acute{\Gamma}$, $C_{I},_2$) to the owner
Permission: The owner checks whether the receiver has been verified by the PKG by
$e(K_{ID,1,g}) \overset{?}{=} e(\eta,g_2). e((u_0 \frac{\pi}{i \varepsilon I} u_i), K_{ID,3}) .e(\eta,\acute{\Gamma})$.
If it holds, the owner chooses β , $p \overset{R}{\leftarrow} Z_p$ and computes
$D_1 = \frac{K\ ID,1}{KID,1,T)}$. $(u_0 \frac{\pi}{i \varepsilon I} u_i)^{\beta}$, $D_2=e(C_{i,2}(u_0 \frac{\pi}{i \varepsilon I} u_i))^{\beta}$ and $D_3=g^p$.
The owner sends $(D_1,D_2,D_3,K_{ID,2})$ to the proxy server.

Re-encryption Receiving (D_1,D_2,D_3,$K_{ID,2}$) from the owner, the proxy server computes the re-encrypted Cipher-text as

$C^i_{i,1}$ =D_2 . $C^i_{i,1}$, $C^i_{i,2}$= $C_{i,2}$, $C^i_{i,3}$= $C_{i,3}$ $C^i_{i,4}$= D_1 , $C^i_{i,5}$= D_3 and $C^i_{i,6}$= $K_{ID,2.}$

The proxy server response the receiver with CT^i_i=($C^i_{i,1}$, $C^i_{i,2}$, $C^i_{i,3}$, $C^i_{i,4}$, $C^i_{i,5}$, $C^i_{i,6}$)

Decryption:

1) To decrypt the cipher-text CT_i =($C_{i,1}$,$C_{i,2}$,$C_{i,3}$, $C_{i,4}$, $б_i$,vk) the owner O computes

$$Mi=C_{i,1} . \frac{e(g,KID,2,Ci,3)}{e(g,KID,1,Ci,2)}$$

2) To decrypt the cipher-text CT^i_i=($C^i_{i,1}$, $C^i_{i,2}$, $C^i_{i,3}$, $C^i_{i,4}$, $C^i_{i,5}$, $C^i_{i,6}$), the receiver R computes K_1=$K^i_{ID,1}$. $c^i_{i,5}$. $c^i_{i,4}$.

Then, he can compute

$$M_i= c^i_{i,1} = \frac{e(c^i_{i,6},c^i_{i,3})}{e(K1,c^i_{i,2})}.$$

4. METHODOLOGY & TOOLS USED

The Java Technology is used for the development of the Scheme, because, Java is portable, secure and supports Distributed file system. The Java is the Object Oriented Programming Language developed by Games Goslings in the year 1991. It has its own virtual machine called JVM. The Java technology supports more security than any other software. The resources are accessed faster between Network servers. The Files can be easily encrypted and transferred through Network.

MySql Database is used as backend because it has the capacity to hold large data on a Network. The MySql has cluster architecture and automatic data partitioning. The technology has Just-in-Time (JIT) compiler.

The JRE consists of the Java Virtual Machine (JVM), The Swing components are light weight components.

The software generates Class files which are executable on any type of Operating System, because, operating Systems have its own JIT compilers. So, the class file generated in one operating system can be executed on any other operating systems like Windows, UNIX, LINUX, Android and Mac OS.

It has features like

- Simple
- Object oriented
- Distributed
- Multithreaded
- Dynamic
- Architecture neutral
- Portable
- High performance
- Robust

Our project uses Java Server Pages and Swings GUI Graphical User Interface for interacting with clients.
Windows XP
JDK 1.6 and above
Java Server Pages
Swing Frame work
Net beans IDE used for Development
Apache Web server
MySQL DataBase
The Application is compatible on Lower versions.
The software is executed on limited client nodes.

5. Cloud Usage

Cloud computing is internet-based computing in which large groups of remote servers are networked to allow the centralized data storage, and online access to computer services or resources. Cloud computing refers to computing with a pool of virtualized computer resources. A cloud can host different workloads, allows workloads to be deployed.

At its most basic level, a cloud storage system needs just one data server connected to the Internet. A client (e.g., a computer user subscribing to a cloud storage service) sends copies of files over the Internet to the data server, which then records the information. When the client wishes to retrieve the information, he or she accesses the data server through a Web-based interface. The server then either sends the files back to the client or allows the client to access and manipulate the files on the server itself.

In the cloud computing environment, whenever the server is accessed through internet, the servers process the uploaded files through web server.

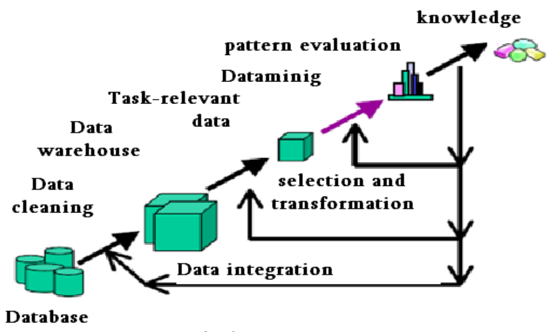


Fig.4 Knowledge Discovery Process

6. Processing

Cluster server: There are 3 cluster servers Cluster-server1 stores files of server1. Cluster-server2 stores files of server2 and Cluster-server3 stores files of server3.

Load server: Stores all files

Slip server cluster:

Browses the file
Selects the path
Download the fie

SIP user agent clients select file and location to download the file. To download the selected file server will send file to the SIP user agent.

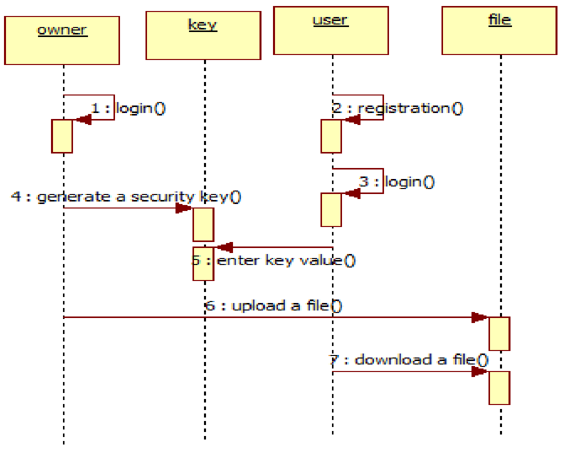
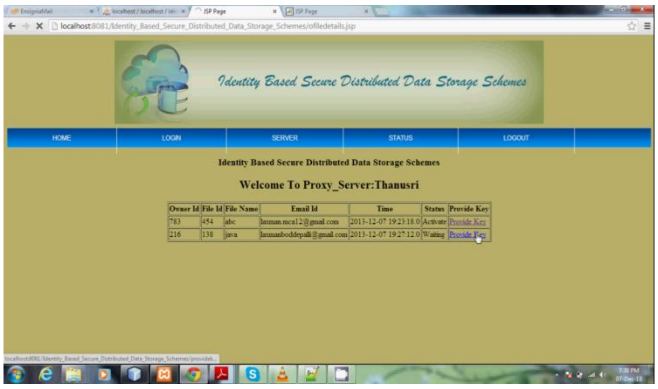


Fig.5 User Login



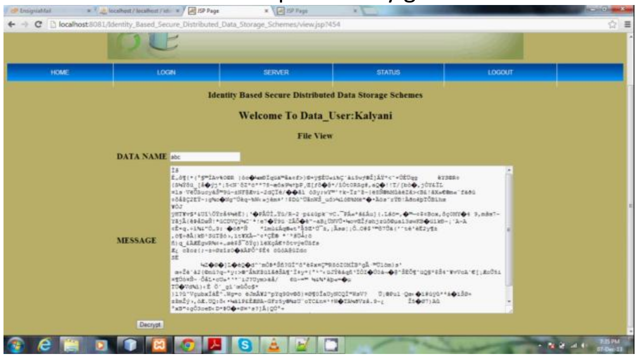
Identity Based Secure Distributed Data Storage Schemes

Welcome To Proxy_Server:Thanusri

Owner Id	File Id	File Name	Email Id	Time	Status	Provide Key
783	454	abc	laxman.mca12@gmail.com	2013-12-07 19:23:18.0	Activate	Provide Key
216	138	java	laxmanboddepalli@gmail.com	2013-12-07 19:27:12.0	Waiting	Provide Key

Fig. 6 The Proxy Server Details

(Figure.6 is the file details and its private key generated to access the user.)



Identity Based Secure Distributed Data Storage Schemes

Welcome To Data_User:Kalyani

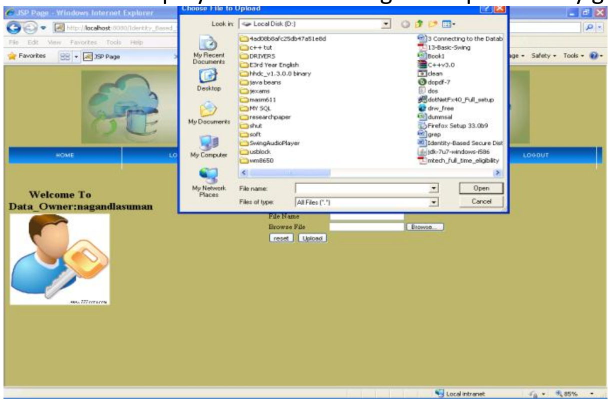
File View

DATA NAME abc

MESSAGE [illegible]

Fig.7 The File Encrypted

(Form Figure.7 is the file displayed when user login and private key given as input.)



Welcome To Data_Owner:nagandlasuman

File Name
Browse File

Fig.8 File Upload Process

(Figure.8 is the file upload process as Data Owner Login. The file is uploaded to mysql database.)

RESULTS

The Unit testing, Integration testing are successful. The Functional tests were conducted on windows based platform, but further tests on multiple platforms are in the process. The project is tested on Apache and Xampp web servers. Tests are not conducted on IIS web server. Because the IIS web server has to be configured to run .jsp files. While creation of database proper Data type has to be assigned. The file up loader data type is blob.

CONCLUSION

When the file is uploaded the file security is implemented using generated private key. One can have hierarchy of application of algorithms. The Encryption Algorithm applicability provides the flexibility in range and sequence to the user's choice because from the three of the Encryption Methods a user can apply all or omit any in any order. The owner has to be online to monitor the file access permissions. The implemented two identity-based secure distributed data storage schemes tested on intranet successfully.

REFERENCE

[1] Jinguang Han, Student Member, IEEE, Willy Susilo, Senior Member, IEEE, and Yi Mu, Senior Member, IEEE- "Identity-Based Secure Distributed Data Storage Schemes"-IEEE TRANSACTIONS ON COMPUTERS, 2013.

[2] Cipher text-Policy Attribute-Based Encryption , John Bethencourt Carnegie Mellon University, bethenco@cs.cmu.edu

[3] B. Lynn. The Pairing-Based Cryptography (PBC) library. http://crypto.stanford.edu/pbc.

[4] https://cloud.oracle.com/home