Reinforcing the Reliability of Cloud Based Data Sharing with Access Revocation using Identity-Based Encryption

Ajay P. Sharma [1], Vidit V. Save [2], Md. Abuzar Shaikh [3], Satish Y. Ket [4]

U.G. Student, Department of Computer Engineering, Rajiv Gandhi Institute of Technology, Mumbai, Maharashtra, India[1]

U.G. Student, Department of Computer Engineering, Rajiv Gandhi Institute of Technology, Mumbai, Maharashtra, India[2]

U.G. Student, Department of Computer Engineering, Rajiv Gandhi Institute of Technology, Mumbai, Maharashtra, India[3]

Professor, Department of Computer Engineering, Rajiv Gandhi Institute of Technology, Mumbai, Maharashtra, India[4]

ABSTRACT: Cloud computing has gained immense popularity for providing on-demand services for users as well as corporate data centers. These Cloud technologies allow the data centers to operate like the Internet, for administering shared computing re-sources, virtually accessible in a secure and scalable manner. Thus, it is necessary to secure the data or files uploaded by the user while providing an access control mechanism. Identity-based encryption (IBE) is an efficient way to achieve a crypto-graphically secure data transfer over the cloud server. However, we must understand that access control is static in the case of IBE. Hence, we must provide a mechanism to prohibit a user from accessing the system when his/her authorization code is expired or revoked. To overcome the issue, we follow the concept of revocable storage identity-based encryption (RS-IBE) for providing both forward and back-ward secrecy of data. We are presenting a specific implementation of RS-IBE and would evaluate its security in the defined security model. On analyzing the security and performance aspects of the proposed RS-IBE scheme, we found advantages in terms of functionality and efficiency for data-sharing applications. Finally, we provide the implemented results and demonstrate RS-IBE practically.

KEYWORDS:Cloud Computing, Data Security, Hashing, Identity-Based Encryption, Multi-key file access.

I.INTRODUCTION

Cloud computing is a platform where we store and access data/programs over the internet instead of our computer's hard drive. Hence we are not restricted to get services in a time-bound manner. This enables users to get services anytime, anywhere and also across various platforms like mobile phones, desktop computers, tablets, etc. Therefore it has become a very convenient way for users to share and access the data.

Cloud storage services like Google Drive [1], OneDrive [2], iCloud [3], etc. offer a flexible and easy way to share our data over the internet in a comfortable manner. However, cloud services are vulnerable to various threats.

Initially, when users put their secret data on the cloud server, it implied that it was out of his/her control. The risk of a data breach is not unique to cloud computing but it is consistently increasing day by day. Secondly, as the cloud is an open-source environment, the storage is often outsourced to many users which may increase the probability of an attack. Even worse, there is a possibility that the cloud service provider itself may disclose the user's data for an illegal profit. Thirdly, data sharing is static. That is, when a user's time gets expired, he/she will still be able to access the previously shared data. Therefore, when users put their secret data on a cloud server, they also want to control access to these data so that only authorized users can view the data. A reasonable solution to suppress the following problem is to use cryptographically enforced access control such as identity-based encryption (IBE) [4]. Hence, to reduce the above security threats, identity-based access control must be implanted on the shared data to meet the following security goals:

- **Data Confidentiality:** The aim of data confidentiality is that only the sender and his intended receiver should be allowed to access the documents/data. Hence we encrypt the data stored over the cloud server using Identity-based Encryption. The implemented automated key authority generate and distributes keys that can decipher the documents. Thus any unauthorized people cannot interpret the file contents even if they access the document directly.

- **Data Integrity:** It means that the contents of the message must not be modified until it reaches the authorized person. Data Integrity assures us that the data is received flawlessly without any modifications or deletion. To provide an integrity check, we use a hash validation which authorizes the document accessing entity along with the data transferred.
- **Timestamp based Revocation**: In timestamp-based expiry, we use the server's clock/timestamp. When the data provider accepts a file access request, the auto-mated key authority generates a timestamp. While accessing the file, this times-tamp is used for comparison with the current timestamp. If the timestamp sur-passes the one provided by the key authority the user's access is denied. Hence he/she cannot view the requested files. To further access the file, the user must again log in and make a request.

II.LITERATURE REVIEW

Identity-based encryption(IBE) is the type of public-key encryption. Generally, the public key is unique information of owners' identity eg. Name of user, email address, employID, etc. Central authority generates the private key based on their unique identification. The central authority generates both master public key and master private key , where a master public key is shared with all users who need it to encrypt their message and master private key is shared with only particular user hence it avoids direct communication overhead between two users and make them secure for message exchange.

There are many researches have been done for security related issues for data sharing over a cloud. The initialization for IBE is done by Mr.Adi Shamir [5] in 1984 .Also RS-IBE concepts has been deployed by various researchers before, they were using Deffie-Hellman exchange algorithm,Ku-nodes algorithm,Eigamal etc. These algorithms are secure but not very easy to implement also having complexities in its time complexity calculation. In 2001 Boneh and Franklin [6] have suggested

identity-based encryption. In this algorithm, the PKG generates the master key which is used to design the private key of users by some mathematical function like:

PKG(k)= P and Km

where k= security parameter, the binary length of key material.

km= private key of the master

P= set of system parameters including master (M), cipher(C),etc.Using this algorithm any two users can exchange their message securely as well as without their direct contacts.The most efficient identity-based encryption schemes are currently based on bilinear pairings on elliptic curves, such as the Weil or Tate pairings [7],[8]. The first of these schemes was developed by Dan Boneh and Matthew K. Franklin (2001), and performs probabilistic encryption of arbitrary ciphertexts using an Elgamal-like approach. Though the Boneh-Franklin scheme is provably secure, the security proof rests on relatively new assumptions about the hardness of problems in certain elliptic curve groups.

The limitations for existing algorithms are as follows:

1) In the existing IBE mechanism once the key is distributed by the central authority to every finite number of user then its secret key can be destroyed. this is the main issue that arises during key revocation.

2)Because the Private Key Generator (PKG) generates private keys for users, it may decrypt and/or sign any message without authorization. This implies that IBS systems cannot be used for non-repudiation. This may not be an issue for organizations that host their own PKG and are willing to trust their system administrators and do not require non-repudiation.

3) A complex mechanism needs to design and handle the central authority.

4) Any user has to login every time without any dynamic access options such as QR-Code.

III.PROPOSED ALGORITHM

Over the years, RS-IBE has been implemented by many developers . The general implementation consists of three major components: The Data Provider, the User (i.e. A client who has access privileges to the shared data), and the Key Authority. The key authority, shown in the figure, can be implemented as a manual or an automatic server. In our proposed system, we have automated the key sharing mechanism. Its working can be explained as follows:

1. Initially, the data provider uploads the data/files to the cloud server. To provide data security, we encrypt the uploaded data using the AES encryption algorithm using a hash key generated.
2. After encrypting, the user has an option to select the recipients of his/her files. A different key is generated for each of the users who are provided the privilege to access a file.

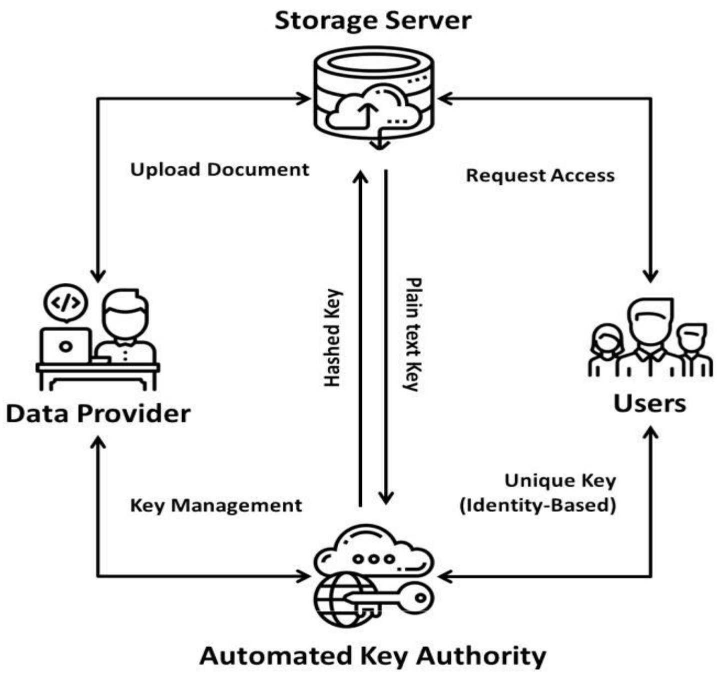

Fig.1. Proposed Mechanism

The working of our proposed RS-IBE system can be explained using the following mentioned steps:
Step 1: Suppose ’D’ is the data provider, ’U’ is the user who needs to access the data and ’K’ is the key-Authority.
Step 2: Firstly, ’D’ has to sign-up/login to upload his/her files on the cloud server. The files are encrypted and stored to be accessed by privileged users.
Step 3: The ’U’, once logged-in, needs to request access privileges for the files uploaded by ’D’.
Step 4: There will be a database on the server where ‘D’ can verify the user ’U’ who has requested for the file. After verification, if ’D’ finds ’U’ to be an intended recipient, he can accept the request and send a one-time-key via an automated key authority.
Step 5: The key can be generated in such a way that ’U’ would be allowed to access the file only for a particular interval after which he/she will be automatically revoked. The procedure of the key generation is explained in further detail. The key point in our RS-IBE system is that the client can only view data/files over the cloud and is not allowed to access the file offline. If the client is revoked, he/she has to make another request using the previously mentioned steps.

A.USER PASSWORD HASHING

Hashing is the transformation of a string of characters into a usually shorter fixed-length key or value that represents the original string. When a user enters his/her passwords, the text is first hashed using our encryption algorithm. Based on the actions, for sign-up, the hash is stored in the database, or for login, it’s compared to the one present there [9]. The diagram below shows our encryption mechanism:
Step 1: Initially, the input taken is the user password. If the entered password is less than 32 characters, we append the 0character to make its length equal to 32 characters.
Step 2: Now, we append a randomly generated salt, of size 32 characters, to the password.
Step 3: For the initial round, we take the phrase ‘ANewHashAlgorithmForOurRSIBEMajorProject’. These 40character long phrase is broken into five equal parts named A, B, C, D and E respectively.
Step 4: These blocks are passed as an input to the hash function’s first round as shown in the figure Fig2.
Step 5: As the rounds proceed, 4 functions are applied consecutively as follows:

- **Function 1:** [(A AND C) OR (B AND D)]
- **Function 2:** [B XOR C XOR D XOR E]
- **Function 3:** [(B AND C) OR (D AND E) OR (C AND D)]
- **Function 4:** [A XOR B XOR D]

Step 6: To provide an improved avalanche effect, we use the bit-shifting function to shift the block A, B, and C by 4, 5 and 6 respectively.

Step 7: The blocks obtained are then accordingly evaluated as shown in the above diagram. The resulting elements are divided equally and stored again in the blocks A, B, C, D, and E.
Step 8: Finally, we repeat the algorithm from Step 5 to Step 7, 32 times to obtain the final hashed output.

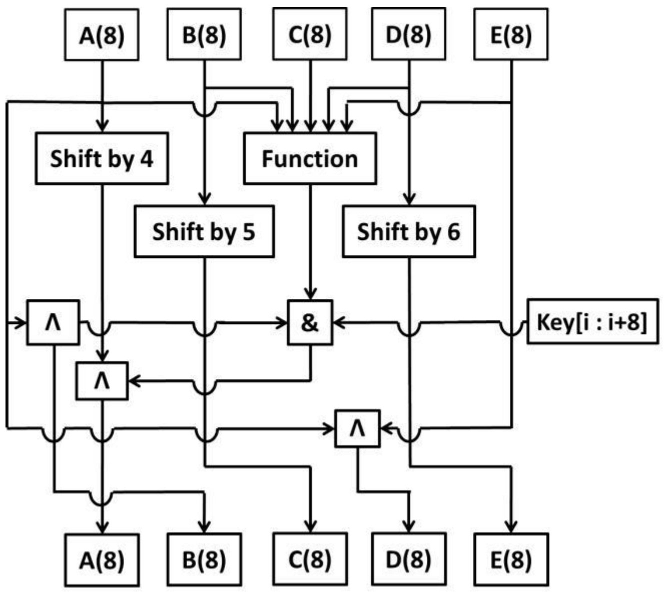

Fig.2. Hashing Algorithm

B.FILE UPLOAD AND ENCRYPTION

The simulation studies involve the deterministic small network topology wit nodes as shown in Fig.1. The proposed For encrypting the file on a server we are using an AES (Advanced Encryption Standard) algorithm. The original name of the AES algorithm is Rijndael. AES is a specification for the encryption of electronic data established by the U.S. National Institute of Standards and Technology (NIST) in 2001. AES algorithm can be operated on various key lengths such as 128,192 and 256 bits. In our file encryption, we are taking 72 character key length generated from hashing algorithm as shown in fig.2. In order to meet the proper key length required for AES input just append the 72 character key with a proper number of bits. In general, we can say that file is encrypted on the server with an input key generated by a hash mechanism. The output length may vary with different versions of the AES algorithm. In our implementation, we have used PHP openssl-encrypt AES-256-CBC [10].

C.IDENTITY BASED MULTI-KEY GENERATION

This phase encompasses the identity-based access granting, revocation mechanism, timestamp expiry, and file decryption. The process starts when the Data Provider logs into his/her account and uploads a file to the server. Once the file is uploaded, other users can view the file details and then send a request to access it. The data provider has the choice to either accept or reject the request upon inspecting the details of the requester.

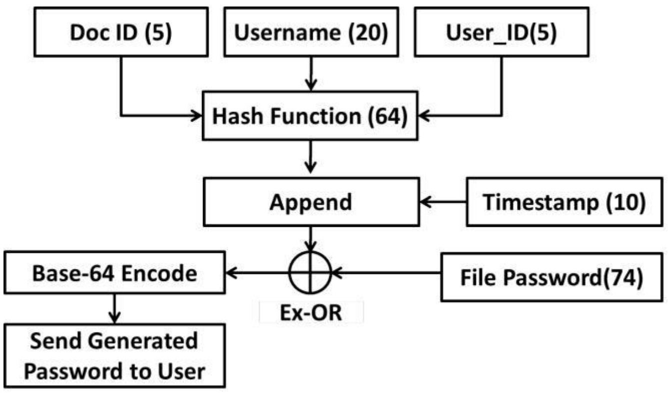

Fig.3. Identity based Multi-key Generation

On granting access to the file, the Key Authority creates a new key that can be used by the intended recipient to access the specific file. The key generation utilizes three basic parameters, which are the Document ID, User Name and User ID. The concatenation of the above parameters, on padding to the desired length, is passed to the SHA hash function to generate a 64 character output. Now this generated hash is appended with the timestamp, which has a size of 10 characters. The 74 character output is then logically EX-ORed with the password used to encrypt the file (72 + 2 character padding). Finally, the output is encoded in the Base-64 format for it to be stored and transmitted easily. Now the Data Provider has the option to send the encoded key via mail or a QR code. The link, once received by the receiver, redirects to the server page to view the file. The decryption procedure follows the reverse steps performed during encryption. The embedded data in the URL is extracted to obtain the User's Key and the Document ID. The key, after Base-64 decoding, is EX-ORed with the appropriately padded file password. Now the generated output is divided into two parts, i.e., Hashed Identity of the User (64 Characters) and Timestamp (10 Characters).To verify the user's credentials, we generate a new 64 characters hashed output using the Document ID from the link, Username, and User ID from the ongoing session. The hashed identity from the URL and the newly generated data must match to proceed further. Next, a new timestamp is generated and is compared with the URL's timestamp which specifies the access expiry. If the current timestamp is found to be lower than the one specified in the URL, the file is decrypted and can be viewed. For all other cases, the user is not considered as the intended recipient and thus isn't allowed to view the file.

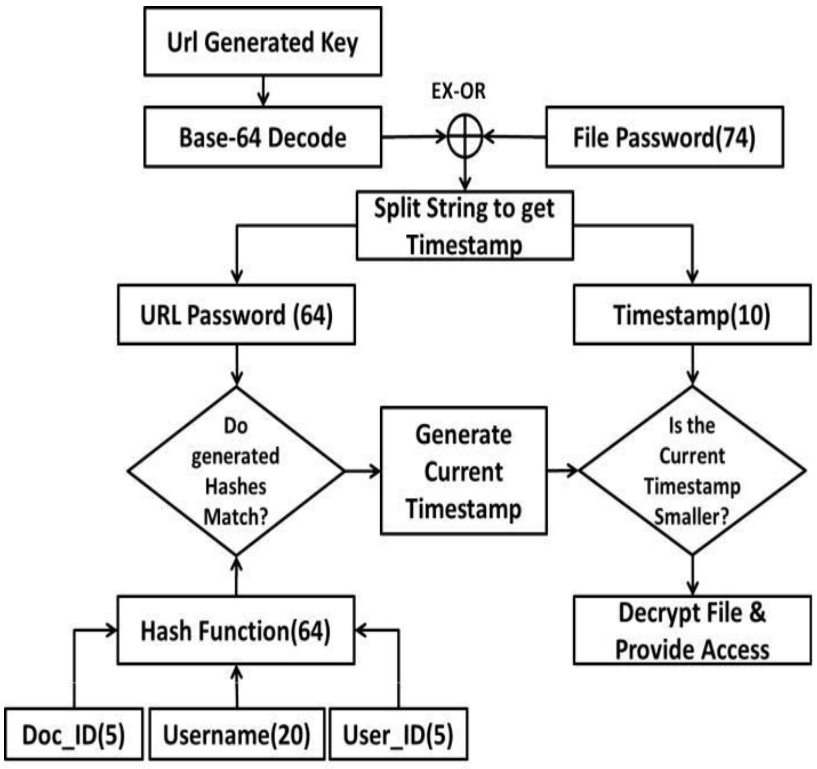
Url Generated Key
Base-64 Decode
EX-OR
File Password(74)
Split String to get Timestamp
URL Password (64)
Timestamp(10)
Do generated Hashes Match?
Generate Current Timestamp
Is the Current Timestamp Smaller?
Hash Function(64)
Decrypt File & Provide Access
Doc_ID(5)
Username(20)
User_ID(5)

Fig.4. Identity based Multi-key Validation

IV.OUR CONTRIBUTION

In this paper, we have re-introduced the term called revocable-storage identity-based encryption (RS-IBE) for implementing a very flexible data sharing method that fulfills the required security goals such as confidentiality, integrity, and authenticity. We provide the forward and backward secrecy by introducing a timestamp constraint to access the data in our implementation of RS-IBE. More precisely, the following attainments are captured in this paper:

- We provide an easy way to encrypt and decrypt the files over a server.
- The main attainment is we have automated the key authority part which was not there in previous versions of RS-IBE.
- In previous versions, the key is needed to be generated by manually performing an action and this generated key is required for decrypting the file. Whereas in our paper we have provided the self -generation of unique key and automatic decryption of the file over a server.
- Also, the new functionality that is involved in our implementation is to access the file over multiple devices by scanning the QR-code.
- We have used the AES algorithm for file encryption and SHA hashing for storing user passwords and also in the generation of keys for clients/users.

V.RESULTS

1. Time complexity for the Hashing Algorithm:

Hashing takes the input of 40 characters to initialize the vectors. These 40 characters are then divided into 5 equal parts (8 characters each i.e A, B, C, D and E). There are 32 rounds in our hashing function.

$h(A, B,C,D,E)=h(64bit,64bit,64bit,64bit)$

Let a, b, c be the constant time delay for AND, OR and XOR operations.

delay for function1 = 2a+c	(in 1st round)
delay for function2 = 3c	(in 2nd round)
delay for function3 = 3a+2b	(in 3rd round)
delay for function4 = 2c	(in 4th round)

Since we have a of total 32 rounds, the delay for function is represented by α is:

$\alpha = 8*(2a+c+3c+3a+2b+2c)$

$\alpha = 8*(5a+2b+6c)$

We have used 3 bit-shifting operation, for A, B, and D respectively, in each round. Considering the delay for one bit shifting operation as t, the shifting delay represented by β is:

$\beta = 32*(4t+5t+6t)$

$\beta = 32*(15t)$

Again 3 XOR and 1 AND operation is performed, hence total delay for 32 rounds over here represented by γ is:

$\gamma = 32*(a+4c)$

Hence total time delay for performing hashing = $\alpha + \beta + \gamma$

2. Complexity for Identity based Multi-key Generation and Validation:

The time complexity for Identity based Multi-key Generation nearly equal to Complexity for Identity based Multi-key Validation. The only new operation performed during validation is the splitting and timestamp checking. So if we ignore this very small time, the overall complexity to decrypt the file over a server would nearly the same as encrypting a file.

Let us consider the following:

Base-64 encoding/Decoding takes constant time = k1

SHA hashing takes constant time = k2

Time complexity for XOR operation = c

Hence overall complexity for generating and validating the password for a user = k1 + k2 + c

Sr.no	Operations	Estimated time complexity
1.	Hashing for user password	α+β+γ
2.	Identity based Multi-key Generation	k1+k2+c
3.	Identity based Multi-key Validation	k1+k2+c

Table .1.Time complexity chart

VI.CONCLUSION

Cloud computing provides us a flexible and convenient way to store and access the uploaded data. We provide data security by storing the data in an encrypted format to avoid any misuse. Hence we categorize our project under Software as a Service (SaaS). Today, data is considered the most crucial asset. Hence, we have designed our encryption methodology to serve all three major security goals (Confidentiality, Integrity, and Authenticity). Our designed encryption mechanism improves security while providing the user with more simplicity. We included the QR code scanning option, for key sharing, to allow the intended users to access files without opening their e-mail key. As a measure of efficiency, we have shown the response time and uniqueness of the algorithms we have used.

REFERENCES

1. Google Drive. (2012) Cloud storage solution. [Online]. Available: https://drive.google.com/
2. OneDrive. (2007) Microsoft storage service. [Online]. Available: https://onedrive.live.com/
3. iCloud. (2011) Apple storage service. [Online]. Available: https://www.icloud.com/
4. Wei Jianghong, Liu Wenfen and Hu Xuexian, “Secure Data Sharing in Cloud Computing Using Revocable-Storage Identity-Based Encryption”, pp. 1-1, March 2016 [IEEE Transactions on Cloud Computing].
5. A. Shamir, “Identity-based cryptosystems and signature schemes,”in Advances in cryptology. Springer, 1985, pp. 47–53.
6. Dan Boneh, Matthew K. Franklin, Identity-Based Encryption from the Weil Pairing Advances in Cryptology - Proceedings of CRYPTO 2001.
7. D. Boneh and M. Franklin, “Identity-based encryption from theweil pairing,” SIAM Journal on Computing, vol. 32, no. 3, pp. 586–615, 2003.
8. Sakai, Ryuichi; Kasahara, Masao (2003). "ID Based cryptosystems with pairing on elliptic curve". Cryptography ePrint Archive. 2003/054.
9. A. A. Alkandari, I. F. Al-Shaikhli and M. A. Alahmad, "Cryptographic Hash Function: A High Level View," 2013 International Conference on Informatics and Creative Multimedia, Kuala Lumpur, 2013, pp. 128-134.
10. F. J. D'souza and D. Panchal, "Advanced encryption standard (AES) security enhancement using hybrid approach," 2017 International Conference on Computing, Communication and Automation (ICCCA), Greater Noida, 2017, pp. 647-652.