Key Generating Cryptosystem

Ms. JemimaJayaKiruba S, Dr. Savithri.V, Ms. Mary Ivy Deepa

Abstract— Key Generating Cryptosystem helps to secure the files by generating the public or master key(secret key) pair randomly after account is created in the server by the owner in the Cloud database. Data owner will encrypt the data, public key and data index & then upload it in the Cloud Server. Data owner Generates Aggregate Decryption Key (ADK) using the master-secret key, Data owner can share the data to other users by sending its ADK through E-mail securely. Original Data can be downloaded by the user only after the Verification of ADK. Data Owner will encrypt the file, public key and index into an image called Steganography to the cloud. Image is Splitted using Merkle Hash Tree Algorithm in Cloud. User will send the Request to Data Owner; if Data Owner is interested to share the file then he will share the ADK & Public Key to the User. After the user receive that mail then he/she will give their User Name, password, user Public Key, Data Owner's Public Key & Aggregate Decryption Key to the Cloud to download the encrypted file.

Index Terms— Key Generating Cryptosystem, Key Generation, Cryptosystem, Aggregate Decryption Key, Steganography, Key Generation in Cloud, ADK, Securing the files, Encryption.

1 INTRODUCTION

In this paper, concrete materials are taken as three-phase composites consisting of mortar matrix, aggregate and bond between matrix and aggregate. Aggregate structures are randomly generated according to the concrete mix. Mesh of finite element is projected on a generated aggregate structure of concrete and different material properties are assigned to the respective elements according to element location in three phases. A failure criterion combining fracture toughness and strength is proposed. Whole processes of single edge notch specimen are simulated from damage to fracture under tension-displacement control condition using nonlinear finite element method. A new research method is presented for investigating fracture mechanism and developing computing strength of concrete[1]. A digital computer is generally believed to be an afficient universal computing device; that is, it is believed able to simulate any physical computing device with an increase in computation time by at most a polynomial factor. This may not be true when quantum mechanics is taken into consideration. This paper considers factoring integers and finding discrete logarithms, two problems which are generally thought to be hard on a classical computer nd which have been used as the basis of several prposed cryptosystem. Efficient randomized algorithms are given for these two problems on a hypothetical quantum computer. These algorithms take a number of steps polynomial in the input size, e.g., the number of digits of the integer to be factored[2].

2 METHODS & METHODOLOGIES

2.1 Proposed System

In this proposed system, Data owner will generate the public or master-secret key randomly after he creates account in the server. Data owner encrypts the data, public key and data index & then upload it in the Cloud Server. Data owner Generates Aggregate Decryption Key (ADK) using its master-secret key, Data owner can share the data to other Users by sending it's ADK. Original Data, Index and the Public key is downloaded only after Verification of ADK. In the modification process, Data Owner will encrypt the file, public key and index into an image called Steganography to cloud. Image is Splitted using Merkle Hash Tree Algorithm in Cloud.. Data User will Search in the cloud by Specifying the Keyword, Cloud will retrieve the Best Results based on the Keywords provided by the data user. Data User's request is forwarded to the Data owner. If data Owner is interested to share that Data to the

- *Author Ms. JemimaJayaKiruba S is currently pursuing masters degree program in computer science & technology in Women's Christian College, Chennai,TamilNadu, India, PH-9710820684. E-mail: jemima.18kiruba@gmail.com*
- *Co-Author Dr.Savithri.V is currently working as Assistant Professor in computer science & technology department in Women's Christian College, Chenna India,, PH-9841958932. E-mail:dr.savithri.v@gmail.com*
- *Co-Author Ms. Mary Ivy Deepa is currently working as Assistant Professor in computer science & technology department in Women's Christian College, Chenna, TamilNadu, India, PH-9444077051. E-mail:maryivydeepa@gmail.com*

data user, then forwards the ADK along with the Public Key to the data User. Data User will be giving his User Name, password, his Public Key along with the Data Owner's ADK & Public Key to the Cloud Server. Cloud Server will verify all those Credentials and then finally Shares the Encrypted Data. Now User has to give the Decryption Key to Extract the Original data. Advantages of proposed system are

1. It provide high security
2. Preserving data integrity and confidentiality
3. Data is Stored Securely in Cloud Server.

2.2 Cloud Server

Cloud servers are constructed with the files and the index information are maintained in the main cloud server. The data are added in each cloud servers, and network construction is made with the entire data index present in each cloud server. Query is given to the main cloud server, so that the main cloud server will verify the index information present in it & divert the query to the corresponding cloud servers.

2.3 Data User / Owner Registration

In this module we are going to create an User application by which the User is allowed to access the data from the Server of the Cloud Service Provider. Here first the User wants to create an account and then only they are allowed to access the Network. Once the User creates an account, they are to login into their account and request the Job from the Cloud Service Provider. Based on the User's request, the Cloud Service Provider will process the User requested Job and respond to them. All the User details will be stored in the Database of the Cloud Service Provider. In this Project, we will design the User Interface Frame to Communicate with the Cloud Server through Network Coding using the programming Languages like Java/ .Net. By sending the request to Cloud Server Provider, the User can access the requested data if they authenticated by the Cloud Service Provider.

2.4 Data Upload With Index Management

In this module while data owner uploading the file it is encrypted with AES algorithm and provided with public key. The index value means every files has to searched by using index value so while uploading the cloud owner has enter few index value for every files and then using the public key. So while uploading the file the cloud give th e index value and public key. And this index value, public key and files are encrypted by the AES algorithm and stored in the cloud server. And every upload of file a link will sent to the cloud owner it will be sent through the email

2.5 Steganography

Steganography is the art or practice of concealing a message, image, or file within another message, image, or file. Generally, the hidden messages will appear to be (or be part of) something else: images, articles, shopping lists, or some other *cover text*. In this module we encrypt the files, index value using the public key. The encrypt data is hidden in the image after it will be stored in the cloud server.

2.6 ADK Generation

In this module we are going to generate the ADK aggregate decryption key .This key will be generated every files uploaded by the cloud user .But this key is generated after validating the cloud owner by giving the master key every cloud owner has a master while they registered in the cloud. so using the master key the cloud owner generate the ADK key for every uploaded files.

2.7 User Authentication & Data Sharing

In this module we designed to the cloud user to interact with the cloud owner .so in this module the user will search the files that is he/she can search the files but he cannot see the file because they need to get permission from the cloud owner even thou the cloud user has is username, password and public key, he/she as get the permission from the cloud owner

then the cloud owner see the cloud user request and we send the ADK key and is private key to the cloud user through the email after retrieving the keys from the owner through the email the cloud user has to enter to see the files which he has make request.

2.8 Decrypt Phase

Decrypt(Primary key, Cipher text, Private key). The decryption algorithm takes as input the public parameters Primary key, a cipher text, which contains an access policy, and a private key, which is a private key for a set of attributes. If the set of attributes satisfies the access structure then the algorithm will decrypt the ciphertext and return a message.

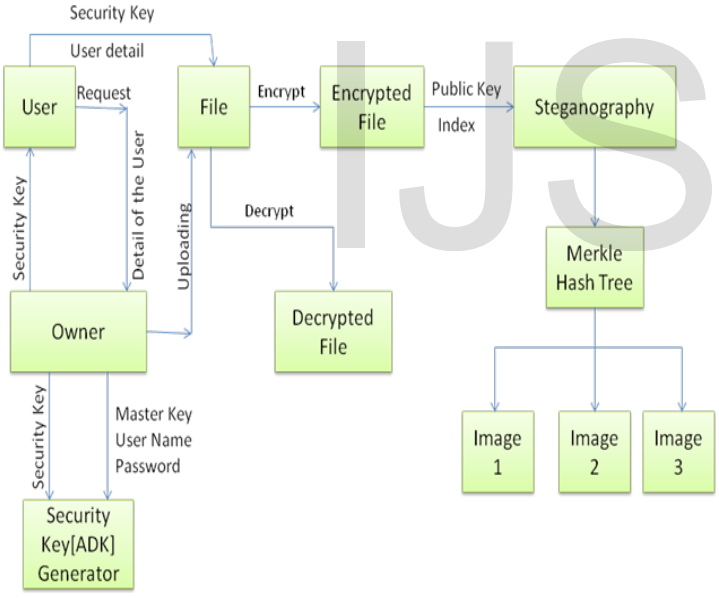

Fig. 1. Overview of Proposed System

3. CODING

```
log4j.rootLogger=INFO, file
log4j.appender.file=org.apache.log4j.RollingFileAppender
log4j.appender.file.File=D\:\\log.log
log4j.appender.file.MaxFileSize=1MB
log4j.appender.file.MaxBackupIndex=1
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L - %m%n

public static void main(String[] args)
{
    LoggerUtil log = new LoggerUtil();
    try
{
        // TODO code application logic here
        log.addLog("Program started");
        AuthenticateUser.main();
        new UserLogin().main();
 }
catch (Exception ex)
{
        log.addLog(ex.getLocalizedMessage()); Log-
        ger.getLogger(Steganography.class.getName()).log(
        Level.SEVERE, null, ex);
}

public class HibernateUtil
{
   private static final SessionFactory sessionFactory;
   private static LoggerUtil log = new LoggerUtil();
   static
   {
      try {
         // Create the SessionFactory from standard (hiber-
nate.cfg.xml)
         // config file.
      sessionFactory = new Configura-
      tion().configure("/hibernatecfg/hibernate.cfg.xml").bui
      ldSessionFactory();
```

```
}
catch (HibernateException ex)
{
        log.addLog("Hibernate exceptio=>"+ex);
       System.err.println("Initial SessionFactory creation failed." + ex);
        throw new ExceptionInInitializerError(ex);
}
```

4. CONCLUSION

To protect user's data privacy is a central question of cloud storage. With more mathematical tools, cryptographic schemes are getting more versatile and often involve multiple keys for a single application. This proposed system consider how to "compress" secret keys in public-key cryptosystems which support delegation of secret keys for different ciphertext classes in cloud storage. No matter which one among the power set of classes, the delegate can always get an aggregate key of constant size. Our approach is more flexible than hierarchical key assignment which can only save spaces if all key-holders share a similar set of privileges. A limitation in our work is the predefined bound of the number of maximum ciphertext classes. In cloud storage, the number of ciphertexts usually grows rapidly. So we have to reserve enough ciphertext classes for the future extension. Otherwise, we need to expand the public-key. Although the parameter can be downloaded with ciphertexts, it would be better if its size is independent of the maximum number of ciphertext classes. On the other hand, when one

carries the delegated keys around in a mobile device without using special trusted hardware, the key is prompt to leakage, designing a leakage-resilient cryptosystem, yet allows efficient and flexible key delegation is also an interesting direction.

ACKNOWLEDGMENT

The authors wish to thank I extend my deepest gratitude to our Principal Dr. Ridling Margaret Waller, our Dean Mrs. Margaret Alexander, our Head of Department Mrs. Mary Ivy Deepa, Dr.Savithri & my family. This work was supported in part by a grant from Women's Christian College, Chennai, TamilNadu, India.

REFERENCES

[1] S.S.M. Chow, Y.J. He, L.C.K. Hui, and S.-M. Yiu, "SPICE – Simple Privacy-Preserving Identity-Management for Cloud Environment,"
[2] L. Hardesty, "Secure Computers Aren't so Secure". MIT press.
[3] C. Wang, S.S.M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy- Preserving Public Auditing for Secure Cloud Storage,"
[4] B. Wang, S.S.M. Chow, M. Li, and H. Li, "Storing Shared Data on the Cloud via Security-Mediator,".
[5] S.S.M. Chow, C.-K. Chu, X. Huang, J. Zhou, and R.H. Deng, "Dynamic Secure Cloud Storage with Provenance," Cryptography and Security,
[6] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and Verifiably Encrypted Signatures from Bilinear Maps,"
[7] M.J. Atallah, M. Blanton, N. Fazio, and K.B. Frikken, "Dynamic and Efficient Key Management for Access Hierarchies,"
[8] J. Benaloh, M. Chase, E. Horvitz, and K. Lauter, "Patient Controlled Encryption: Ensuring Privacy of Electronic Medical Records,"
[9] F. Guo, Y. Mu, Z. Chen, and L. Xu, "Multi-Identity Single-Key Decryption without Random Oracles,"
[10] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data,"
[11] S.G. Akl and P.D. Taylor, "Cryptographic Solution to a Problem of Access Control in a Hierarchy,"
[12] G.C. Chick and S.E. Tavares, "Flexible Access Control with Master Keys,"
[13] W.-G. Tzeng, "A Time-Bound Cryptographic Key Assignment Scheme for Access Control in a Hierarchy,"
[14] G. Ateniese, A.D. Santis, A.L. Ferrara, and B. Masucci, "Provably- Secure Time-Bound Hierarchical Key Assignment Schemes,"
[15] R.S. Sandhu, "Cryptographic Implementation of a Tree Hierarchy for Access Control,"