***I.J. Education and Management Engineering*, 2015, 4, 21-26**
Published Online September 2015 in MECS (http://www.mecs-press.net)
DOI: 10.5815/ijeme.2015.04.03

Available online at http://www.mecs-press.net/ijeme

A Practical Scheme for Implementing Client Side Encryptions and Decryptions

Lija Mohan[a], Sudheep Elayidom M.[b]

[a,b] *Division of Computer Science, Cochin University of Science & Technology, Kerala, India.*

Abstract

The amount of data that is generated each day is rapidly increasing so that the traditional ways of data storage seems to be inadequate. Cloud Computing has helped a lot to solve the storage issues but fear of data leakage hinders its widespread acceptance. Even though the public clouds like Amazon, Google etc store the data in encrypted form, data loss can still occur during transmission. The only practical solution is to upload encrypted data to cloud. Hence users should have an additional software or application to encrypt the data before uploading and decrypt the data after retrieval. This article analyzes different solutions for implementing such applications. The article also describes about the cryptographic primitives that can be added for enforcing security.

Index Terms: Cloud computing, information security, client side encryption, Java applet.

1. Introduction

Data is evolving day by day. Traditional client server architecture cannot handle storage and retrieval of such large amount of data. Cloud computing clusters come to rescue here. A cloud system is a cluster of large amount of connected computers having high storage capacity and immense processing power. Cloud systems allow internet based access also. Scalability, efficiency, pay as you use pricing model, cost effective, etc are some advantages of cloud system. Irrespective of all these advantages, people are still reluctant to avail cloud facilities. Security and confidentiality of stored data, is one of the major factor that hinders the wide spread usage of cloud systems.

Users do not wish to disclose their sensitive information to a third party sink like cloud. Hence there should be some mechanism to ensure secrecy before uploading the data to the cloud systems. Encryptions [2] come to rescue here. That is before uploading any data to cloud, it should be encrypted, and at the user side, it should be decrypted to retrieve original data. Here comes the need for encryptions and decryptions at client machines.

* Corresponding author
Email:

Usually the first solution that anyone thinks of is to implement this part as a desktop application because web applications are always prone to threats and all the processing takes place in server. But this article analyzes another efficient applet programming based technique. Since applets can be embedded as part of a web application there has to be some security measures to protect these applets. SSL certificates [4] are utilized to enforce applet security. Also by combining symmetric and asymmetric encryptions data security is enforced.

The article is organized as follows. Section 2 defines the problem; Section 3 compares the possible solutions to solve the problem. Section 4 deals with cryptographic primitives that can be applied. Section 5 discusses the implementation constructs and finally session 6 concludes the article.

2. Problem Statement

Bob needs to upload some files to Cloud server. But Bob is feared of data leakage since he is storing the data in a third party system. Hence he decides to encrypt the data and upload that encrypted version to cloud. For a user Alice to access Bob's data, she should decrypt the data and get the actual content. Hence encryptions and decryptions should be done at client side for better security. In this article we illustrate possible practical techniques for implementing client side encryptions and decryptions.

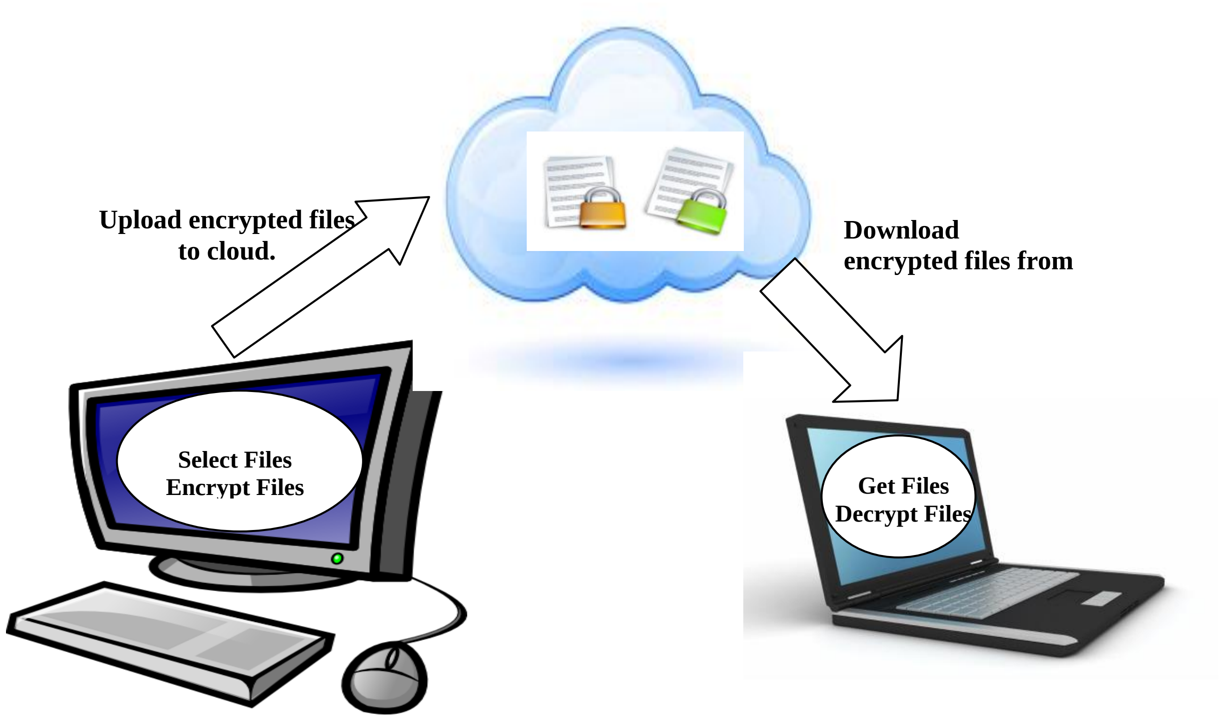
Upload encrypted files to cloud.
Download encrypted files from
Select Files Encrypt Files
Get Files Decrypt Files

Fig.1. Basic Architecture

3. Possible Solutions For Client Side Encryptions & Decryptions

Our aim is to implement data security by converting the data to an unreadable format in the client machine itself. Similarly at the receiving side, reverse conversion should be applied to retrieve actual content. For better security implementation we can adopt some encryption/decryption techniques for data conversion to unreadable formats. Hence we need to add some encryption/decryption modules to client machines. There are two possible solutions to add this module to the client machine.

Solution 1: Developing a Desktop Application

Develop a desktop application to select the files and encrypting/decrypting it at the client side. Then we should connect this desktop application with web application. For that we can create an http-endpoint in the desktop application. For example, we can embed a simple servlet container like jetty [5]. Once we create a servlet container, we can use something like Jersey to write a REST API/Servlet API [6]. But in today's world, web applications are gaining so much popularity compared to desktop applications. Also we need to install the application separately for each user. Update to the software also becomes difficult. However since desktop applications are standalone applications the threat rate will be less.

Solution 2: Developing a Web Application

From the above section it is clear that web applications have a lot of advantages compared to desktop applications. Hence it will always be a better option to implement our application as a web application. But the crucial factor here is security. The data should be encrypted before passing through the network. So here the only solution is making use of applet programming concepts. Java applet is a small application written in Java language. Ultimately they are converted to byte code format. A user can launch the Java applet from a web page, and the applet will be executed within a Java Virtual Machine (JVM) in a process separate from the web browser. Or in other words we can say that applets actually run in the client machines. They can access the client machine resources for processing if the client grants permission to do so.

Table 1. Comparison between Solution1 and Solution2

Features Vs Solution	**Solution 1** **Using a Desktop Application to implement Client Side Security**	**Solution 2** **Using a Web Application to implement Client Side Security**
Maintenance	Software Update is difficult Need to be installed in each machine seperately	Software updates are easy Need to be installed only once
Ease of Use	Users should have the system where application is installed. Otherwise cannot access.	Can be accessed from anywhere in the world through internet.
Internet Connectivity	No need of internet connectivity Speed and performance depends only on system configurations	Needs internet Connectivity Speed and performance also depends on characteristics of internet connectivity
Security	Programmer has total control over the desktop applications and hence it is safe from various vulnerabilities.	Prone to more security risks, but since we are making use of applet programming only encrypted data will reach the network.

4. Enforcing Security Using Cryptographic Primitives

From Section 3 it is clear that the client side processing of files can be handled by either a desktop application implementation or as a web application. In both of these methods the files that need to be stored in cloud should be encrypted before passing through the network. For encrypting the files two possible methods are symmetric encryption scheme and asymmetric encryption scheme.

In the case of symmetric encryption scheme [7], same key is used for encryption and decryption. Here the encryption/decryption time needed is very less and key length will also be small. But the number of keys needed will be on the order of O(n^2), because each user has to share its key with each previous user. Also the key distribution is a bottleneck.

Asymmetric encryption scheme make use of two keys, one public key and other private key which are paired in some way. Data will be encrypted using public key and decrypted using private key. Here the public key will be disclosed to every user in the system whereas private key should be kept secretly by each user. Private key cannot be derived from public key and vice versa. Hence the key distribution in this scenario is pretty much easy and the number of keys to be generated will be in the range O(n) (each user should have 2 keys i.e. 2n keys). But the keys should be large primes and encryption/decryption time is much greater which makes this unsuitable for encrypting large amount of data.

Table 2. Comparison of using Symmetric Encryption and Asymmetric Encryption

Symmetric Encryption	**Asymmetric Encryptions**
Same key should be shared by the sender and receiver	Each person creates and keeps his own secret key.
Substitution and permutation techniques are utilized	Based on dicrete mathematical concepts, primes, exponentiation and logarithms
Single key is used for encryption and decryption	Related pair of keys are used for encryption and decryption
Computational Time is less hence faster execution	Consumes more computational power and slower process.
Number of keys needed is of the order of O(n^2)	Number of keys needed is of the order of O(n)
For large amount of data, symmetric encryptions are the best.	Practical for small amount of data.

Table 2 gives a comparison of Symmetric and Asymmetric encryption scheme. From this comparsion it is clear that asymmetric schemes impose more security for the data but is not practical for large amount of data. Symmetric schemes fail to implement strong cryptographic security but it is the only practical solution for large amount of data. In contexts similar to ours, the solution is to combine symmetric and asymmetric encryption schemes. We can encrypt the files using symmetric encryption scheme like AES and the AES [8] key is encrypted using asymmetric encryption scheme like RSA [9]. More specifically saying, a file will be encrypted using an AES key. This AES key is encrypted using RSA public key generated by owner. All the authorized users will possess the public key of owner. This encrypted key is transferred to users. Using their private key they could decrypt and regenerate the AES key. Only if they are successful in recovering the AES key they will be able to decrypt the file.

5. Implementation Details

We implemented the client side encryption/decryption schemes using Java programming language. Java is an object oriented programming language and is platform independent. Java is suitable for developing web application and desktop applications. It provides support for applet programming also.

We implemented the work as a web application. For processing client side data we used applets. Applets are able to access the data stored in client machines and they can process the data at client side itself. For security, we used an additional feature in java called applet signing. Signing is the process of securing your java application with a SSL certificate. The purpose of SSL certificates is to distribute the public key and verify the identity of sender. Some browsers will allow the applets to run only if they are signed. In Java, an applet can be selfsigned using Keytool [10].

The method adapted to self sign a Java applet is described below.

1. Create an applet program in Java using any IDE
2. Test the applet in IDE
3. Generate a Jar file of applet
4. Generate the certificate with the following shell commands

```
jre_path>keytool -genkey -keyalg rsa -alias myKeyName
jre_path>keytool -export -alias myKeyName -file myCertName.crt
```

Using jarsigner capability sign the jar file.

```
jre_path> jarsigner “path_to_my_jar/myJar.jar" myKeyName
```

Create an html file to test the applet on browser

```
<html>
  <body>
<applet code="path_to _applet_inside_jar/myApplet.class" archive=".path_to_my_jar/myJar.jar"/>
  </body>
</html>
```

For encryption to work properly the key size should be correct. AES supports a key size of 128,192 and 256 bits. When we implement RSA, the minimum key size should be 1024, but we set keysize as 2048 bits since our application need higher security. When we double the key size the decryption time needed will increase by 6-7%. Since we are encrypting the AES key using RSA, there arises a need for converting that key size to 2048 bits. For that we can make use of the concept of hashing. We can set the hashing algorithm such that whatever be the input size, result can be set to 2048 bits. Other simple solution is to use padding feature in java if the key generated is small or take the first 2048 bits if the key generated is larger.

6. Conclusion

Today large number of users depends on cloud especially for storage of data. The main threat here is security. To prevent leakage of information, encryption can be made use of. But the data should be encrypted before it reaches the network, similarly decryption can take place only at client machine. This article explains some techniques to be considered while implementing such client side encryptions and decryptions. Always web applications are preferred over desktop applications but security attacks occur more in such applications. To eliminate that we suggested an applet based scheme where security is implemented using SSL certificates. The article also dealt with the encryption techniques that can be applied which ensures security with better processing time. RSA over AES is proposed and key size incompatibilities are eradicated using hashing or padding.

Acknowledgement

The authors gratefully acknowledge the INSPIRE fellowship provided by DST, India, without which this work would not have been completed successfully**.**

References

[1] Won Kim, Sungkyunkwan University, Suwon, S. Korea, "Cloud Computing: Today and Tomorrow", Journal of Object Technology, Vol. 8, No. 1, January-February 2009.
[2] E. Cronin, M. Sherr, and M. Blaze, "The Eavesdropper's Dilemma", Technical Report MS-CIS-05-24. University of Pennsylvania. 2005.
[3] Pankaj Kamthan, "Java Applets in Education", Indian Institute of Technology, Bombay, http://ekalavya.iitb.ac.in/documents/java.pdf.
[4] J.K.Haris,"Understanding SSL/TLS", https://computing.ece.vt.edu/~jkh/Understanding_SSL_TLS.pdf
[5] Jos Dirksen, "Jetty a Lightweight, Open-Source Web Server and Servlet Container", DZone technical articles, December, 2012.
[6] "Web API Design Crafting Interfaces that Developers Love", https://pages.apigee.com/rs/apigee/images/api-design-ebook-2012-03.pdf.
[7] Gusustav J. Simmons, "Symmetric and Asymmetric Encryption", Computing Surveys, Vol. 11, No. 4, December 1979.
[8] Alan Kaminsky, Michael Kurdziel, Stanisław Radziszowski, "An Overview of Cryptanalysis Research for the Advanced Encryption Standard ", Springer, December 2009.
[9] Somani, U.; Lakhani, K.; Mundra, M., "Implementing digital signature with RSA encryption algorithm to enhance the Data Security of cloud in Cloud Computing", IEEE International Conference onParallel Distributed and Grid Computing (PDGC), October, 2010.
[10] https://www.digitalocean.com/community/tutorials/java-keytool-essentials-working-with-java-keystores.

Author(s) Profiles

Lija Mohan (born April 4, 1988) is an Academician and Researcher who is very much interested in High Performance Computing Domain. She is more interested in Big Data Analytics and have done several projects based on that. She is a University rank holder for both Masters in Computer Science and Bachelors in Computer Science Engineering. She has published several research works and handles sessions during Big Data workshops. She has got sufficient experience as a software engineer and teacher. She has won several fellowships and research grants including Windows Azure Research Grant.

Dr. Sudheep Elayidom M. is working as Associate Professor in Computer Science Division of School of Engineering for the past fifteen years. He secured 1st Rank in the University for both his Masters Degree and Bachelor's Degree. He has several journal and conference publications to his credit and he is the author of the book titled 'Data Mining and Warehousing' published by Cengage publishers. He obtained Ph.D degree from CUSAT and he is a pioneer in Data Mining. He is actively involved in latest research trends including Big Data Analytics, Twitter Analytics, etc.