I J C T A, 9(15), 2016, pp. 7627-7632
© International Science Press

Secure Data Communication in Cloud Using Fully Homomorphic Encryption

Mebin K. Babu*, Deedish Udayan and Sumesh S.*****

ABSTRACT

Cloud computing ensures a convenient way of trading and sharing of content, where user can access the content easily and the distributor (content provider) outsources the content within the secure channel. Always the confidentiality of content is the main problem faced by the distributor and due to this reason they are afraid to outsource the content through cloud computing. As a solution they use the Digital Right Management (DRM) scheme which uses the concept of keyserver, License server and Content Encryption Key for data protection and also it uses Additive Homomorphic Probabilistic Public Key Encryption and Proxy Re-encryption rules for encrypting and decrypting the keys and contents. Homomorphic is simply adding of two keys, which is easy to the intruder to hack it and use the content. Instead of this simple Homomorphic Encrytpion we try to use Fully Homomorphic Encryption to give more security for data and also we try to reduce the number of transactions in the base system.

Keywords: Full Homomorphic Encryption, Re-Encryption, Content Encryption Key, Multiplication Key.

1. INTRODUCTION

In the research scheme [1] Secure and Privacy-preserving DRM scheme using Homomorphic Encryption (HE) in Cloud computing, the author proposes a DRM scheme which uses two types of methods like Proxy Re-encryption and Additive Homomorphic Probabilistic Public Key Encryption (AHPPE). The Proxy re-encryption (PRE) is widely used to protect the content in semi-trusted cloud environment and it generates a Content Main Key for user which is only re-encrypted by the license server. Additive Homomorphic Encryption (AHE) allows performers to compute correct operations over encrypted values without being aware of their content. It generates a Content Encryption Key (CEK) for user using Content Main Key (CMK) of the owner and Assistant Key (AK) of the key server.

But the problem is that, the Additive Homomorphic Encryption (AHE) in the proposed scheme just adds the two keys like CMK and Assistant Main Key together and generate a CEK, which is used by the user to encrypt the main content and that encrypted content sent to content server.

In this paper [2][7] state that, Homomorphic Encryption (HE) is classified into three types Partially Homomorphic Encryption (PHE), Somewhat Homomorphic Encryption, and Fully Homomorphic Encryption (FHE). The Additive HE includes the category of PHE. This encryption performs one operation on encrypted data, such as multiplication or addition but not both. Here, the proposed scheme uses addition on keys i.e. CMK and AK.

In the case of FHE, it is a cryptosystem which sustains both addition and multiplication and can compute any function. When we use FHE on proposed DRM scheme of Huang Qin-Lang[1], it sustains both addition

Department of CS & IT, Amrita School of Arts and Sciences, Kochi, Amrita Vishwa Vidyapeetham, India

* MCA Student, *Email: mebinova@gmail.com*

** MCA Student, *Email: deedish.ud@gmail.com*

*** Assistant Professor, *Email: ssumesh@hotmail.com*

and multiplication on the keys and for this purpose we introduce two more keys like multiplication key and two AK for generating a FH DRM system. Therefore it gives more security for the key rather than the AHE.

2. RELATED WORKS

DRM is an important technology of taking a series of measures to protect the digital contents from being abused and make sure that digital contents are fair to use. In the base paper [1], methods like PRE and AHPPE are used to protect the contents through keys. Here [1], they follow four steps for securing contents, key generation, content encryption, license acquisition and content consumption using with the keys of CMK, CEK and AK to full fill the above four steps. The CEK is the collection of CMK and AK. The basic diagram of existing system is shown below.

Homomorphic encryption [2] uses specific types of computations to be carried out in cipher text which produce an encrypted result which is also a cipher text. Earlier homomorphic encryption used a specific algebraic operation performed on a plain text which is equivalent to another algebraic operation performed on its cipher text. Mainly three types of Homomorphic encryption are used. First one is Partially Homomorphic which performs one operation on encrypted data. Second one is Somewhat Homomorphic, which uses more than one operation but can only support a limited number of addition and multiplication processes. Fully Homomorphic is the last one which performs addition and multiplication simultaneously in any function.

Some versions of DRM [3] try to solve the interoperability problem by suggesting that both content providers and customers need to open parts of their security properties without the assurance of a beneficial outcome. Without the participation of both DRM technology and content providers, DRM interoperability schemes are hard to achieve and they [3] used DRM interoperability agent (DIA) and DRM interoperability server (DIS) to solve the problem. .

A multiparty multilevel architecture [4] contains different eves of distributors who help content provider and consumer for sharing information from one end to another. In this paper, Usage Logs are used to find out unauthorized usage of data from the consumer side and analyses this usage in Log Collection center. Also this scheme uses Global encryption key (GEK), which is unique for a particular content or owner and Local encryption key (LEK), which is different for different owners and different distributors and different contents.

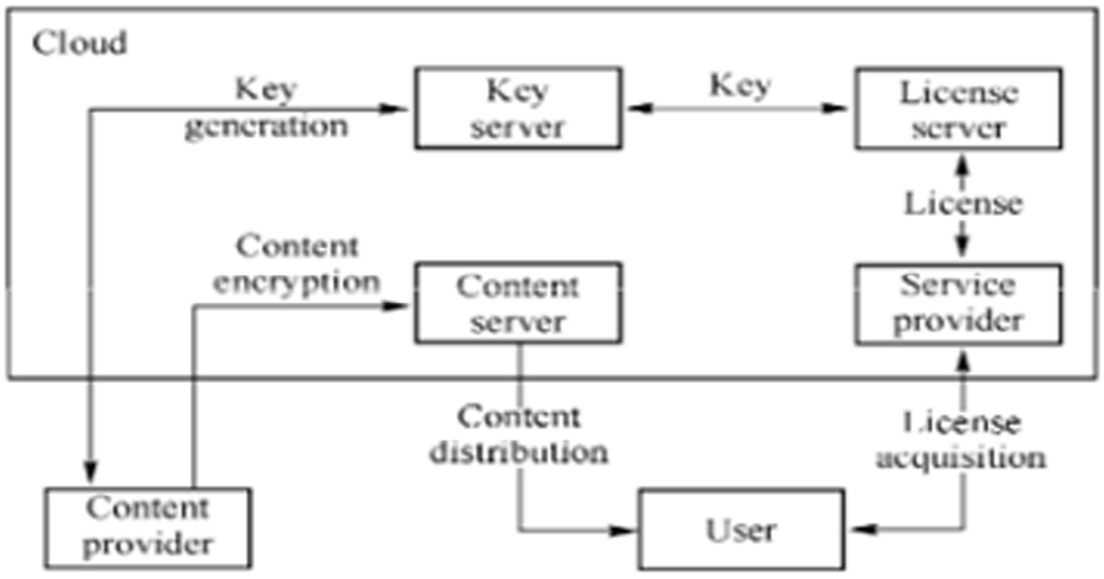

Figure 1: DRM System

3. PROPOSED SYSTEM

In the scheme [1] DRM system gives more effort to the Content Provider, where it generates the re-encryption key (combination of CP's and User's Public Keys) which is mainly used to identify the user. But in Multi User, CP wants to check each user and generate key for each one. So we proposed the re-encryption key generated on the key server side, and then the Content Provider has less effort on the basis of key and reduces one transaction of keys in DRM. Thus the content provider just worried about its content and not about its re-encryption key. Secondly, Key Server generates private- public keys for CP and User and sends them into an open channel. Here a middle man attack is possible and if intruder gets the both keys the system is totally unsecured. In our scheme, also the Key Server generate private-public keys for CP and User but only private key is send to them where we reduced the chance of getting two keys also we change the combination of re-encryption key, now it is a combination of public key of both CP and the User. Thirdly we add a time stamp in content server side to avoid the replay attack from the unauthorized user[8]. And lastly we introduce Fully Homomorphic Encryption[5][6] instead of additive homomorphic encryption for this purpose we use one more AK and Multiplication Key which is the combination of CMK and one of the AK. The following are the different steps involved in the proposed system.

3.1. Key Generation

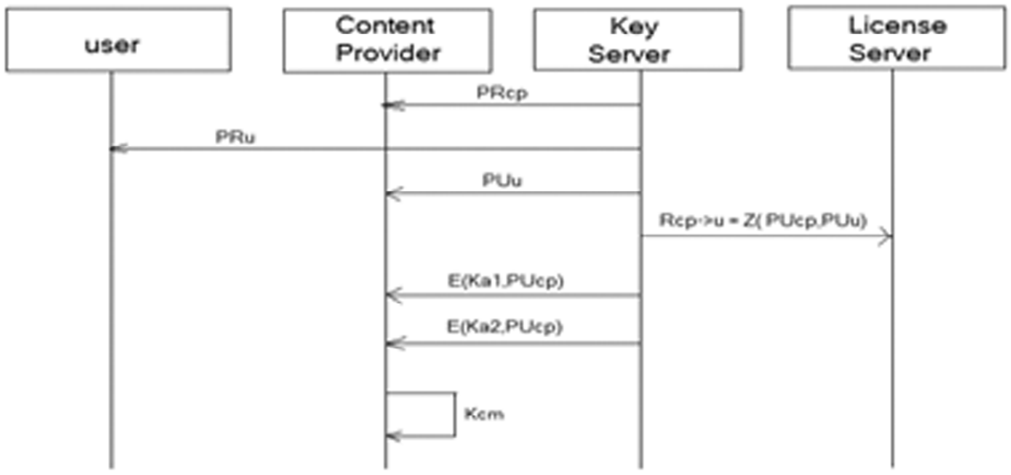

Figure 2: Sequence diagram of Key Generation

Step 1: KS $\rightarrow$ CP: Key (PR_{cp})

Step 2: KS $\rightarrow$ U: Key (PR_u)

Step 3: KS $\rightarrow$ CP: Key (PU_u)

KS $\rightarrow$ LS: $R_{cp \rightarrow u} = Z(PU_{cp}, PU_u)$

Step 4: KS $\rightarrow$ CP: E (K_{a1},PU_{cp})

KS $\rightarrow$ CP: E (K_{a2},PU_{cp})

KS	:	Key Server
CP	:	Content Provider
PR_{cp}	:	Private key of Content Provider
PR_u	:	Private key of User
PU_u	:	Public key of User
PU_{cp}	:	Public key of Content Provider
LS	:	License Server
U	:	User

The Key server creates a private/public key (PR_{cp} , PU_{cp}) for content provider and sends the private to the content provider. Then in second step key server creates a private key PR_u for unknown user and sends it to that unknown user from this way key server cannot be able to get the user's personal information. In third step key server sends a copy of user public key to the content provider and Key server generates a re-encryption key R_{cp}->$_u$ for the user using PU_{cp} and PU_u, sends it to license server. And in last step the Key server randomly generates two assistant key K_{a1}, K_{a2} and encrypts it with content provider's public key PU_{cp}, and then sends it to content provider. After this process content provider generates K_{cm} randomly.

3.2. Content Encryption

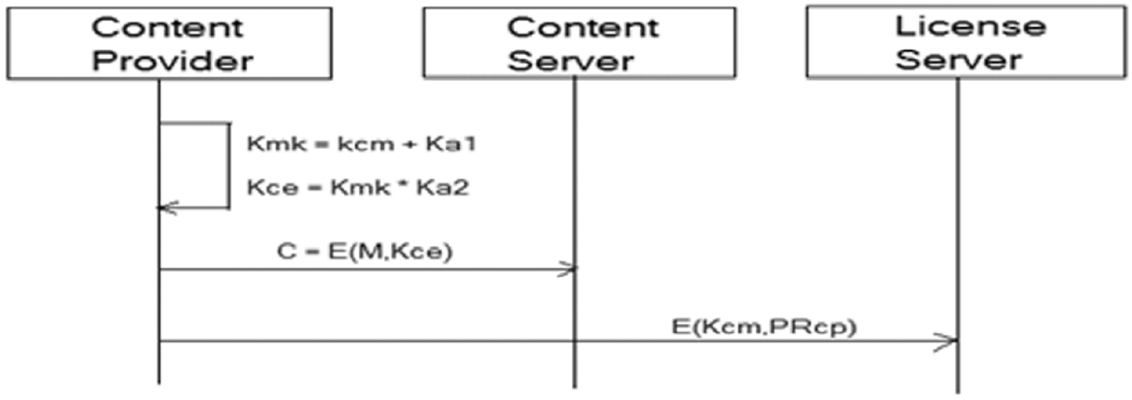

Figure 3: Sequence diagram of Content Encryption

Step 1: CP: Kmk = Kcm + Ka 1	K_{a1},K_{a2}	:	Assistant Keys
: $K_{ce} = K_{mk} * K_{a2}$	K_{mk}	:	Multiplication Key
Step 2: CP $\rightarrow$ CS : C = E (M, K_{ce})	K_{cm}	:	Content Main Key
Step 3: CP $\rightarrow$ LS : E (K_{cm}, PR_{cp})	K_{ce}	:	Content Encryption Key
	M	:	Plain Content text
	CS	:	Content Server

The content provider decrypts the K_{a1} and K_{a2} with the private key of content provider PR_{cp}, and then computes K_{ce} . In second step content provider encrypts the plain content with the K_{ce} and sends the encrypted content to the content server. And in last again content provider encrypts the K_{cm} with the private key of content provider and sends it to license server.

3.3. License Acquisition

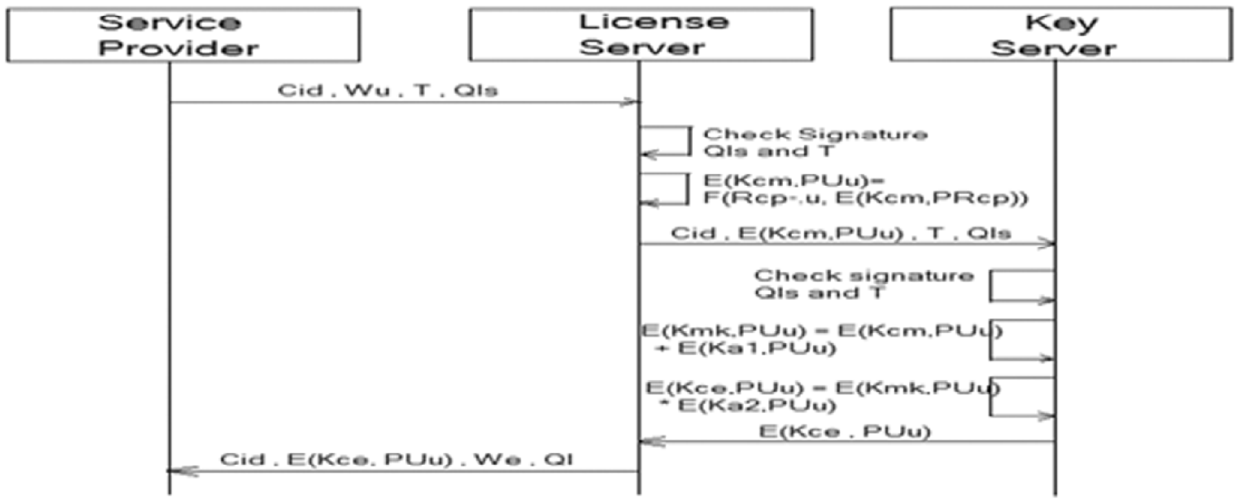
Service Provider
License Server
Key Server
Cid . Wu . T . Qls
Check Signature Qls and T
E(Kcm,PUu)= F(Rcp-.u, E(Kcm,PRcp))
Cid . E(Kcm,PUu) . T . Qls
Check signature Qls and T
E(Kmk,PUu) = E(Kcm,PUu) + E(Ka1,PUu)
E(Kce,PUu) = E(Kmk,PUu) * E(Ka2,PUu)
E(Kce . PUu)
Cid . E(Kce. PUu) . We . Ql

Figure 4: Sequence diagram of License Acquisition

Step 1: SP $\rightarrow$ LS: C_{id}, W_u, T, Q_{ls}

Step 2: LS : E (K_{cm}, PU_u) = F($R_{cp \rightarrow u}$, E(K_{cm}, PR_{cp}))

Step 3: LS $\rightarrow$ KS C_{id}, E((K_{cm}, PU_u), T, Q_{ks}

Step 4: KS : E(K_{mk}, PU_u) = E(K_{cm}, PU_u) + E(K_{al}, PU_u)

: E(K_{ce}, PU_u) = E(K_{mk}, PU_u) * E(K_{a2}, PU_u)

Step 5: KS $\rightarrow$ LS : E(K_{ce}, PU_u)

Step 6: LS $\rightarrow$ SP : C_{id}, E(K_{ce}, PU_u), W_e, Q_l

SP	:	Service Provider
R	:	Re-Encryption Key
C_{id}	:	Content identity
W_u	:	User Rights
T	:	Time Stamp
W_e	:	Right expression

Service provider sends an acquisition request to license server. In step two the license server checks the Q_{ls} with T, then re-encrypt the E (K_{cm}, PR_{cp}) with the re-encryption key and license server sends the key acquisition request to the key server. Then in step four Key server generates the key K_{ce} using the equation K_{mk}, K_{cm} + K_{a1}, and K_{ce}= K_{mk} * K_{a2}, and sends the encrypted Kce to the license server. And in last step license server generates the right expression W_e from the W_u, and license signature for user verification, and send the E(K_{ce}, PU_u) to the service provider.

3.4. Content Consumption

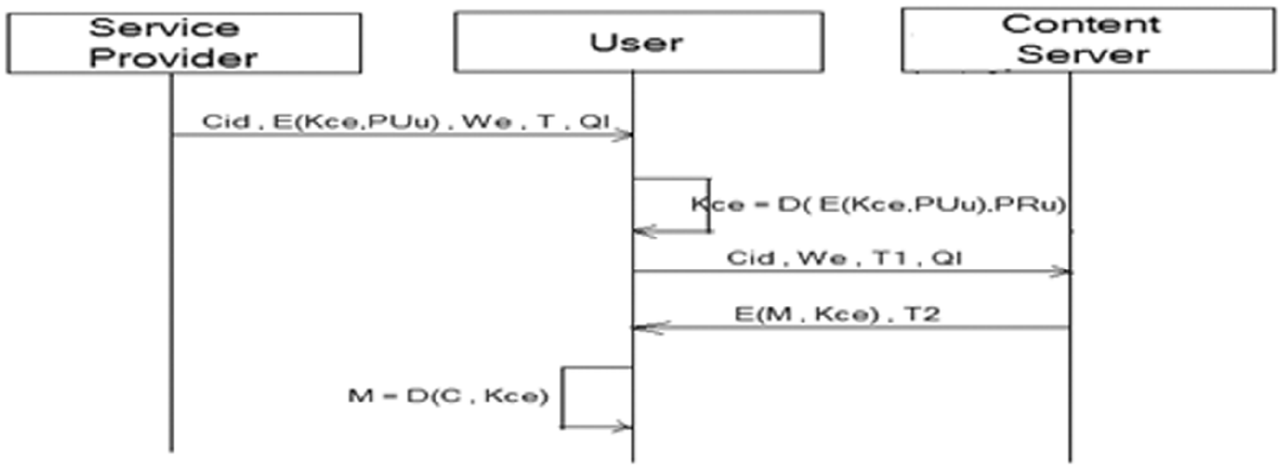

Figure 5: Sequence diagram of Content consumption

Step 1: SP $\rightarrow$ U: C_{id}, E(K_{ce}, PU_u), W_e, Q_l, T

Step 2: U: K_{ce}= D (E(K_{ce}, PU_u), PR_u)

U $\rightarrow$ CS: C_{id}, W_e, Q_l, T_1

Step 3: CS $\rightarrow$ U: T_2, C

U: M = D(C, K_{ce})

Q_l	:	License Signature
Q_{ks}	:	Key Acquisition Request Signature
Q_{ls}	:	License Acquisition Request Signature

First Service provider sends the encrypted K_{ce} to the user with user right expression, license signature and time stamp to prevent the replay attack. Then user decrypt the K_{ce} using its own private key PR_u, and send a request to the content server including C_{id}, W_e, Q_l and time stamp for Encrypted content data. After checking the right expression and license signature, content server outsources the encrypted content to the Authorized user. And at last user decrypt the encrypted content using K_{ce}.

4. CONCLUSIONS

In this scheme [1] the DRM system gives an authority to the CP to generate a Re-encryption Key for a user but the problem of this scheme is that, when more than one user uses the DRM system, CP want to create

Re-encryption Key for each user which is give more effort to the CP. In our scheme the Re-encryption Key is generated by key server which have given little effort to the CP and never bothered about multi-user. The proposed work is little complex in terms of timing but give high secure to the data comparing to the early scheme like Simple Homomorhic. We are now planning to simulate the proposed system in Amazon EC2 cloud environment. Where we used Credit card to select a linux based environment to work out our proposed scheme and it ensure high data security.

REFERENCES

[1] Huang Qin-Long, MA Zhao, YANG Yi-xian, FU Jing-yi,NIU Xin-xin "Secure and privacy-preserving DRM scheme using Homomorphic Encryption in Cloud computing" National Engineering Laboratory for Disaster Backup and Recovery, Beijing University of Posts and Telecommunications, Beijing 100876, China.

[2] Monique Ogburn, Monique Ogburn, Pushkar Dahalc "Homomorphic Encryption" The Journal of China Universities of Posts and Telecommunications, Information Security Center, Beijing University of Posts and Telecommunications, Beijing 100876, China 3. Beijing National Security Science and Technology Co. Ltd, Beijing 100086, China.

[3] Sangho Lee, Heejin Park, Jong Kim "A Secure and Mutual-Profitable DRM Interoperability Schem" Computers and Communications (ISCC), 2010 IEEE Symposium, pp. 75-80.

[4] Amit Sachan a, Sabu Emmanuel a, Amitabha Das a, Mohan S Kankanhalli "Privacy Preserving Multiparty Multilevel DRM Architecture" Proceeding CCNC'09 Proceedings of the 6th IEEE Conference on Consumer Communications and Networking Conference, pp. 1128-1132.

[5] Ryan Hayward, Chia-Chu Chiang "Parallelizing Fully Homomorphic encryption for a cloud environment". Journal of Applied research and technology 13(2015), pp. 245-252.

[6] Feng Chao, Xin Yang, "Fast key generation for Gentry-Style homomorphic encryption", National Engineering laboratory for disaster Backup and recovery, Beijing University of posts and telecommunications, Beijing 100876, China, December 2014, 21(6): 37-44, www.sciencedirect.com /Science/journal/10058885

[7] Hossein Rahmani, Elankovan Sundarajan, Zulkarnain Md. Ali, Abdullah Mohd Zin, "Encryption as a Service (EaaS) as a solution of cryptography in cloud", Faculty of Information Science and technology, Universiti Kebangsaan Malaysia, The 4th International Conference on Electrical Engineering and Informatics (ICEEI 2013)

[8] Jung Hee Cheon, Hyunsook Hong, Moon Sung Lee, Hansol Ryu, "The polynomial approximate common divisor problem and its application to the fully homomorphic encryption", Department of Mathematical Sciences, Seoul National University, 1 Gwanak-ro, Gwanak-gu, Seoul, pp. 151-742.