International Journal of Engineering & Technology, 7 (3.27) (2018) 36-40

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Secure Online Data Sharing in Cloud by Private Transmit Cryptosystem Using Aggregate Keys

E. Amarnath Reddy[1*], M. Srinuvasa Reddy[2], Kompally Manisha[3], B. Mamatha[4]

[1]*Department of Computer Science and Engineering, MLR Institute of Technology, Dundigal, Hyderabad, India.*
[2]*Department of Computer Science and Engineering, MLR Institute of Technology, Dundigal, Hyderabad, India.*
[3]*Department of Computer Science and Engineering, MLR Institute of Technology, Dundigal, Hyderabad, India.*
[4]*Department of Computer Science and Engineering, MLR Institute of Technology, Dundigal, Hyderabad, India.*
**Corresponding author E-mail:amar.enumula@gmail.com*

Abstract

Cloud has become a crucial part of our day to day life because of its easy, effortless and straightforward nature of data storing and sharing. One of the important concerns for many users is data storing, we mustn't forget about data sharing. A convenient way of online data sharing is to look at its pros for simple access while preserving security are cons for any user. Thus, a better way of ensuring user's data is to implement data integrity with the KAC scheme. This scheme provides an efficient sharing method of decrypting multiple sets of data with the single key. It's one among many ways of quick and effective data retrieval in case of data loss or data alteration on the cloud. It also uses the broadcast algorithm to distribute data for a specific set of users. This scheme uses basic HMAC, one of the secure hash functions for the stability of data integrity. Therefore provides a protected environment where a user can share the bulk of data through integrity. Another advantage of using this scheme could reduce the burden of computation over the cloud.

Keywords: *Data integrity, broadcast, key-aggregate cryptosystem, cloud computing, data sharing.*

1. Introduction

Cloud computing is a shared pool of computing which provides access, sharing and storing of data over the web. It has emerged to be the best solution to easily manage a number of applications worldwide. These cloud applications are popular among government, private companies, healthcare, social networking and other integrated businesses. A cloud service varies in terms of reliability, cost, agility, security, management, and maintenance. Based on the type of investment user prefer demand of these services may differ. Most of the cloud applications concentrate upon privacy problem of storing data as it's considered the foremost aspect for any user or sector.

Our data revolves around secure online data sharing, the ability to share same data resource with multiple applications. These offer easy, free, anytime access to files and keep up the data securely over the cloud. Many online data sharing services that are present today satisfy the user needs. Some of the examples are MediaFire, Dropbox, RapidShare etc capable of transferring user's data efficiently. Though being able to securely protect data in the cloud, securing the data is still a huge concern.

The world is after protecting the precious resource in the cloud, and so we put through data integrity as a primary property for protection. Data integrity is the property of an information that remains unchanged when modified by an attacker or unauthorized access. The major data protection revolves around a huge number of users who shares their information online. An old methodology to ensure data privacy is to depend on the server to enforce access control mechanisms [1]. Well, it's the simplest way of data sharing and thus luring many users. Less maintenance and easy handling make the data users glued to these services. A study conducted by salesforce concluded, 94% of private sectors saw a tremendous improvement in security after switching to the cloud. Apart from having the advantages of online data sharing few issues needs to be addressed. These include various factors responsible for data corruption such as technical errors and malicious data breaches. As per the study, only 20% of cloud users claim disaster recovery in 4 hr or less and 9% of non-cloud users could claim the same by sales force. Hence even after continuous breaches of the network protecting user's data is our prime aspect. Key lists of the requirement for data integrity are given as availability, accountability, confidentiality and computational integrity. Thus, we consider data integrity as the most crucial part of the cryptographic computation. One of the cryptographic methods that a user relies upon is message authentication schemes. Some of the message authentication requirements:

1. *Traffic Up do's*: It defines the number and length of messages transmitted between the two users.
2. *Content Mismatch*: Any changes in the data while transferring from a user to the other user can be known if received data is modified or altered.
3. *Repudiation*: Denial of the validity of message by the source.

These few basic requirements are followed while authorizing a message to prevent any data dissimilarity. The security of data was provided by different cryptographic schemes implemented in a number of ways. Some of them are described below.

1.1. Cryptographic Keys for Hierarchical Encryption

By the use of [2], [3], [4], [5] we generally reduce the space for storing and maintaining secret keys. A tree-like structure used to grant access to each node with respect to their keys. In symmetric key cryptography a stream cipher [6], uses a fixed length key for

Copyright © 2018 Authors. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

producing a pseudo-random stream of bits. It tries to approximate a one-time-pad wherein a key is just as long as the message. There have been many researchers [7], [8], [9], [10] under hierarchical encryption for secure key storage. These hierarchical approaches solve the problem of sharing a group of files within a particular branch for the corresponding node. The number of keys increases as the number of branches leading to a higher key size for a set of users is a major concern. Integrity is restored as the keys correspond to a file can be retrieved anytime.

1.2. Identity-Based Encryption with Compact Key

Identity-based encryption (IBE) [11] as the name of the scheme suggests, an identity string set based on the user. It's a public encryption over a trusted private key generator that holds a master secret key and distributes a key among each user. Using user's Id with some public parameters, an encrypted message is given to each user and decrypt using a secret key. Compact key [12], [13] are limited to all keys that must be aggregated from different identity divisions. [14] considers fuzzy IBE allowing a single compact key to decrypt multiple cipher texts. Given data integrity can be solved if the original messages are stored at a different location before generating the encrypted messages.

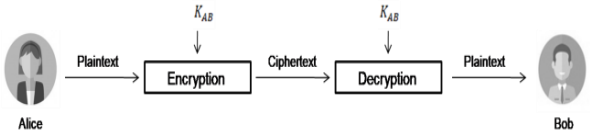

Figure 1: A simple view of symmetric key encryption

1.3. Attribute-Based Encryption Scheme

Attribute-based encryption (ABE) deals with a mixture of an attribute having a ciphertext. Encrypted files stored in the cloud, can only be decrypted if the user matches that particular set of attributes with the secret key. A slight modification in the attribute leads to revoking of the entire ciphertext. The ABE scheme is extended in [15] based on collision resistance, most important property for secure hash function.

1.4. Cryptographic Symmetric - Key Encryption Scheme

In a specific broadcast scenario [16] presents an encryption scheme for transmitting a large number of keys. The scheme is quite similar to the author's approach for symmetric key. Since the method requires a secret key to encrypt data. The idea is to generate secret value rather than a pair of public/secret keys. This scheme can't be implemented for public-key encryption scheme. A basic symmetric encryption scheme includes a same secret key used to encrypt and decrypt a message. Suppose Alice's encrypts a message by using her secret key, then sends to Bob with the shared key and that shared key can be used only by Alice and Bob. K_{AB} is the same secret shared among Alice and Bob as shown in Figure 1.

2. Related Work

Let's view a simple example to understand the importance of data integrity that we want to showcase in this paper. Consider a file that Alice wants to send Bob. Alice wants Bob to prove that it was unmodified and that file was sent by Alice. Before applying this example let's understand MAC [17], [18] as it's the startup idea for all cryptographic hash function for different data integrity algorithms.

2.1. Message Authentication Code

Message Authentication Code (MAC) [19] an idea to generate cryptographic hash function (MD5, SHA-1) over data the user wants to send and the secret key that the user wants to share. The goal is to create a hash that can only be verified by the key holder. For example, Alice wants to send Bob a file. The data gets appended with a shared secret key by Alice and the same hash function is generated. If the same hash result as Alice transmitted to Bob, then the data wasn't corrupted. The below equation shows a basic construction of the MAC,

$$MAC = H(key \parallel Message)$$

H denotes the cryptographic hash function, *key* denotes the secret key and *Message* denotes the data that Alice wants to send. Here, the data is XOR with the key later hashed to produce an outcome. Many assumptions were made in order to change the data in the message to have the same hash function. Figure 2 represents an attacker can change the message M, recomputed hash $H(M)$ is sent over the network. This is a naïve approach while this can't work for the real-time and creates checksum errors in many of the cases.

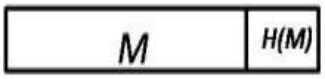

Figure 2: Appending a message with the hash

Now consider a case as in Figure 3, message M is padded with the key K then hashed with the original message. This is fairly a better solution when compared to the previous figure but leading to length extension attacks. An attacker who desires to change the message before sending to Bob must have same hash function value. By nature, a hash function has collisions such that multiple messages are hashed to the same value. The problem arises when an attacker modifies Alice message without knowing the key and transfer to Bob.



Figure 3: Padding original message with a key

Let's consider some of the scenario the researchers had tried to put through in various works,

$H(M \mid K)$

One of the worst ideas as the message gets appended to the key, easy for an attacker to change the message or even delete that message without letting the sender and receiver know about this alteration.

$H(K \mid M)$

A preferred solution to use a key placed at the beginning of the message.

$H(K \mid M \mid K)$

For a better result, we try to append key twice with the message.

$H(K \mid H(K \mid M))$

The best solution until now, but slower in many cases as we are appending hash twice.

2.2. Properties of a Secure Cryptography Hash Function

As these are mentioned by researchers to describe various states of hashing functions and behavior in different scenarios [20], [21],

Pre-Image Resistance

It's infeasible to determine M from $H(M)$. For example, Alice generates a hash, gives it to Bob then Bob is unable to convert that hash backward to find a message for which hash was generated. Given a hash, find a message with the same hash. Consider brute-force approach, the attacker has the hash output of the message. Picks up a random message, hash it and match the hash with the original message.

If they don't match again repeats the same procedure until he finds the match for the original message. How long will the attacker tries to break the hash is unknown. In the best case, the first attempt was correct.
In the worst case, attacker picks up every possible message and finally finds the desired hash for a message. For example suppose for 128 bits, at $(2^{128} - 1)$ attacker gets the original hash. Thus such scenario can't be implemented in real time as they consume a large amount of time and finally avoids the original message stolen from the attacker.

Second Pre-Image Resistance

Given M_1, infeasible to find M_2 such that $H(M_1) = H(M_2)$. If an attacker has hash function he/she is unable to find another message that has the same hash. The basic difference, in pre-image resistance a hash was taken and here message was considered. However, the scenario is same as the brute-force approach for pre-image resistance. So, breaking this property is difficult and consumes a lot of time.

Collision Resistance

It states that for any M_1, M_2 such that $H(M_1) = H(M_2)$ can't be found. Here, the attacker tries to find messages that have the same hash. But in second pre-image resistance the attacker had a specific M_1, he only needs to find another message with that hash. In collision resistance, the attacker has to find any two messages that have the same hash.
From the attacker perspective, he can break collision resistance as it can be broken with any two messages but still remains complicated. In pre-image resistance, the attacker had to find another message with the same hash for a specific one.

2.3. HMAC

All the problems faced in MAC are solved by HMAC [19], [22]. Hash-Based Message Authentication Code (HMAC) is only MACs based on hash function. HMAC provides TLS (Transport Layer Security) and is also known for a stronger pseudo-random function. These are the cryptographic hash functions that generally execute faster. Moreover, a major drawback of MACs i.e. length extension attacks is solved by HMAC. We utilize the same HMAC [24] to solve our data integrity issue.

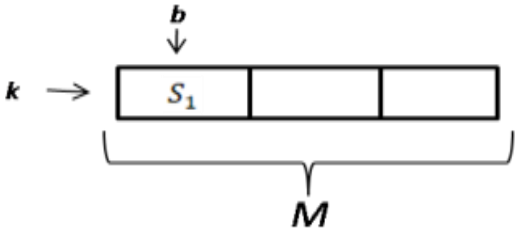

Figure 4: Representation of bits in multiple blocks for a message

HMAC invokes a hash function and a secret key k. The message M consists of multiple blocks of b bits as shown in Figure 4. For example, SHA 512 consists of 512 bits. Each block is 1024 bits so, b would be 1024. First, the key k would be padded to b bits. Suppose there are lesser bits then 1024, simply zeros are appended at the end of k. Then the padded key is XOR with ipad (inner pad), a constant design to eliminate any irregularities of the key. Herewith resulting in a b bit S_1. S_1 is pre-pended to the original message. The message M with S_1 and the original message hashed to produce n bit hash value. For example, if the hash function is SHA512, then n will be 512. Later the n bits hash value is again padded to b bits. Then the padded key, k is XOR with opad (outer pad). opad is used as the other constant designed to eliminate irregularities in the key. The result is a b bit value S_0 while the padded hash is then appended to S_0 and the entire message is hashed. Finally, n bit result is HMAC for the message with the key, k. Therefore HMAC uses an existing hash function and includes a secret key, k in the processing.

2.3.1. HMAC Security as a Major Priority

Security on a cloud or on the internet is a primary issue for any organization or an individual. It depends on the cryptographic strength of the underlying hash function [19], [22], [23]. It's also said that larger the hash function more difficult to break the code for an attacker. It's much harder to

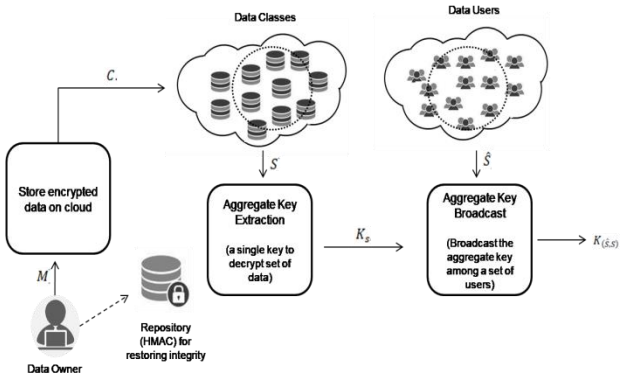

Figure 5: A representation of aggregate keys for secure online data sharing

launch successful collision attacks on HMAC because of the secret key. The secret key is hashed with the message content. As a result without knowing the secret key, an attacker can't compute the correct HMAC. For example, an attacker is able to obtain the HMAC of message M_1 and he has to get another message that has a collision with message M_1. That means for a different message M_2 that's not the same as M_1 but had the same HMAC value as M_1. But all attempts are worthless unless the attacker has a secret key as the correct HMAC can't be generated. That's why the attacker doesn't even know whether M_1 or M_2 will have a collision in HMAC. Because of the use of a secret key, HMAC is much more secure than a cryptography hash function. Thus we give more importance to provide security for HMAC and utilizes in our KAC (Key Aggregate Cryptosystem) scheme.

2.4. Extended KAC

An extended key Aggregate cryptosystem [25] which overcomes the disadvantages of a simple KAC [26] for a constant key size that can be efficiently broadcast data to multiple users. A secure construction could resist CPA and CCA using elliptic curves over a secure channel. This scheme allows decrypting multiple classes of data using a single key stored in an encrypted manner. Here Alice decrypts multiple classes of data with a single key of constant size. The data owner encrypts each class of data using the different public key but can decrypt a set of data with the single key. KAC with a part of broadcast encryption was derived by [27]. For comparison, Broadcast encryption relies more upon low overhead decrypting keys in contrast with KAC scheme. KAC scheme depends on low overhead aggregate keys using a single aggregate key for decryption. The author addresses various issues of cryptographic security for KAC and Broadcast encryption. The extended KAC framework solves the issue of CCA through a collision resistant environment. Moreover, extended KAC publicly broadcasts aggregate keys as they don't require a secure storage. In short, it reduces overhead for public parameters, ciphertexts, and aggregate keys.

3. System Architecture

The proposed method comprises of a data owner who uploads different data classes on the cloud. These multiple classes of data are grouped together for a specific set of users who wants to access the same set of data classes. Simultaneously a set of those files are stored in a repository in case of data loss or data modifications. Then the selected data classes are used to generate an aggregate key, a single key to decrypt multiple classes. This aggregate key is securely broadcast among multiple users for efficient data access[28].

Our system provides an efficient way of data retrieval when there are any modifications to the original data. Another advantage it uses a single key of fixed length throughout the decryption process. HMAC provides this fixed length key during whole encrypting and decrypting process. Figure 5 depicts an overview of our proposed system architecture in a best possible way.

4. Proposed System

We have studied that Key Aggregate Encryption scheme brings out the most efficient way of solving various security-related problems. This Extended Key Aggregate Cryptosystem Scheme [25] is one of the best schemes provided by the author's until now with Broadcast encryption. We also integrate on decrypting multiple classes of data using a single key. Well, we want to take up data integrity property in additional with extended KAC scheme. This property defines data owner who once outsourced their files have no physical copies of these files. We shed some light upon integrity as the user completely loses control over their personal files. It's not always important for a cloud server to report data loss incidents each time. Data integrity is one of the major reasons that we want to focus upon. Therefore, it's necessary for the data owners to frequently check if their outsourced data remains stored properly. Therefore adds an advantage for our construction to reduce a huge burden of computation over the cloud.

To carry out data integrity property, we use Hash-Based Message Authentication Code (HMAC). The HMAC was computed over every encrypted file to support that integrity for various users. When an encrypted file was stored over cloud it first passes through HMAC, the process of storing a copy of the original file in a secure database server. So as when the data owner verifies a certain change in the original data. He/she replaces that whole file with the original file on the cloud server. By doing so, we modify files for all the users at a single time. This basically happens when an attacker tries to access the cloud server. Though changes to unauthorized access by the attacker are nil. But we have to look at the worst case possible.

4.1. Framework for KAC with Data Integrity

A step by step guide to implementing data integrity into KAC as a better approach is presented below:

Step 1: $Setup(1^{\lambda}, n, m)$ – Given input number of data class n, number of uses m and the security constraint λ produces a public parameter $param$. This step takes $(1^{\lambda}, n, m)$ as basic restrictions.

Step 2: $OwnerKeyGen()$ – A data owner registering process, generates a public key pk with the master secret key msk and the broadcast secret key bsk. [29]

Step 3: $OwnerEncrypt(param, pk, i, M)$ – Input as data class $i \in n$ and the plaintext M produces output partially encrypted ciphertext C'. C' acts as undisclosed ciphertext for a secure data transfer.

Step 4:$Sign(C', K)$ – Using Step 2 credentials to login into the system. Then by restoring the modified contents of the data files using partial ciphertext C' and K a random number for generating hash code. This step utilizes two functions, $Verify()$ checks the validity of the original data and $Restore()$ if at all any changes in the original data.

Step 5:$SystemEncrypt(C', msk, bsk)$ – For input as partial ciphertext C', the master key msk and the broadcast key bsk produces final ciphertext C. C is accessible for a specific group of users on the cloud. [28]

Step 6:$UserKeyEncrypt(param, msk, id)$ – For input data user id $id \in m$ outputs a secret key K_{id}.

Step 7:$Extract(param, msk, S)$ – For input master secret key msk and a subset of data classes $S \subseteq n$. All the encrypted messages have K_s for data class S and pass input to broadcast algorithm for generating broadcast aggregate keys.

Step 8:$Broadcast(param, K_s, \hat{S}, pk, bsk)$ – For input K_s and a subset of users $\hat{S} \subseteq m$. Outputs a single broadcast aggregate key $K_{(\hat{S},S)}$ and the secret key K_{id} to decrypt a message.

Step 9:$Decrpyt(param, K_{(\hat{S},S)}, i, id, K_{id}, S, \hat{S})$ – For decrypting the ciphertext C and its data class i, a user id id. Then considers broadcast keys $K_{(\hat{S},S)}$ and the secret key K_{id}. Thus delivering the original data file to a set of specific users.

5. Evaluation and Results

The following figures depict various stages of our implementation during file encryption in Figure 6, Figure 7 and Figure 8 describe integrity for a certain class of data on the cloud.

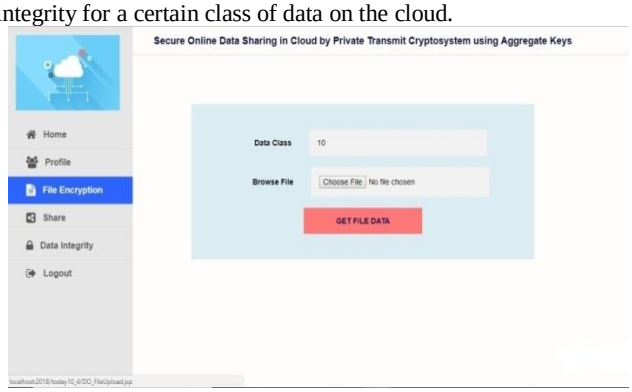

Figure 6: File encryption (Step 3)

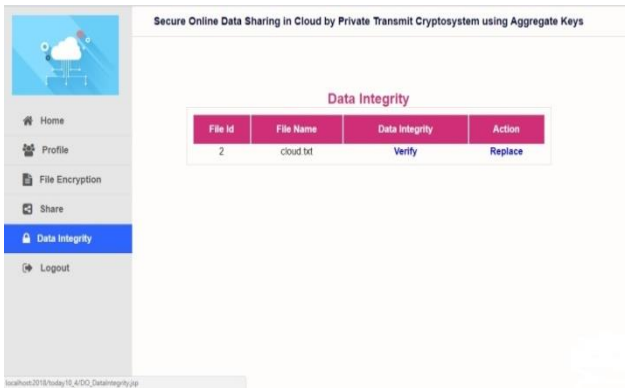

Figure 7: Data integrity for a specific class of data (Step 4)

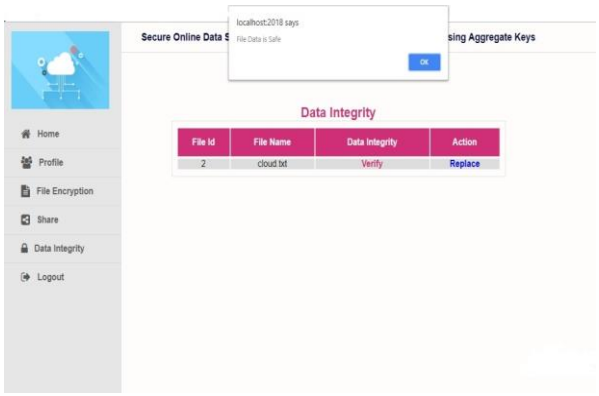

Figure 8: Verifying whether data altered or not (Step 4)

6. Conclusion

The overall idea of utilizing KAC scheme was to provide a more protected method for ensuring efficient data sharing. The issue of data integrity is resolved by using basic HMAC with the KAC while generating aggregate keys. This scheme not only solves issues related to data integrity but also reduces the burden of computation over the cloud. KAC method decrypts multiple classes of data with a single key and securely broadcasting among a set of specific users. This method provides an efficient way to restore modified data on the cloud. Thus our proposed scheme provides stability, efficiency, scalability and ensures data privacy throughout the network.

References

[1] Chow SS, He YJ, Hui LC & Yiu SM, "Spice simple privacy-preserving identity-management for cloud environment", *International Conference on Applied Cryptography and Network Security*, (2012), pp.526-543.

[2] Akl SG & Taylor PD, "Cryptographic solution to a problem of access control in a hierarchy", *ACM Transactions on Computer Systems (TOCS)*, Vol.1, No.3, (1983), pp.239-248.

[3] Chick GC & Tavares SE, "Flexible access control with master keys", *Conference on the Theory and Application of Cryptology*, (1989), 316-322.

[4] Tzeng WG, "A time-bound cryptographic key assignment scheme for access control in a hierarchy", *IEEE Transactions on Knowledge and Data Engineering*, Vol.14, No.1, (2002), pp.182-188.

[5] Ateniese G, De Santis A, Ferrara AL & Masucci B, "Provably-secure time-bound hierarchical key assignment schemes", *Journal of cryptology*, Vol.25, No.2, (2012), pp.243-270.

[6] Ravinderpal SS, "Cryptographic implementation of a tree hierarchy for access control", *Information Processing Letters*, (1988), pp.95– 98.

[7] Jeremy H & Ben L, "Toward hierarchical identity-based encryption", *Advances in Cryptology EUROCRYPT*, (2002), pp.466–481.

[8] Dan B, Xavier B & Eu JG, "Hierarchical identity based encryption with constant size ciphertext", *Advances in Cryptology–EUROCRYPT*, (2005), pp.440–456.

[9] Brent W, "Efficient identity-based encryption without random oracles", *Advances in Cryptology–EUROCRYPT*, (2005), pp.114-127.

[10] Xavier B & Brent W, "Anonymous hierarchical identity based encryption (without random oracles)", *Advances in Cryptology-CRYPTO*, (2006), pp.290–307.

[11] Adi S, "Identity-based cryptosystems and signature schemes", *Advances in Cryptology*, (1985), pp.47–53.

[12] Fuchun G, Yi M & Zhide C, "Identity-based encryption: how to decrypt multiple ciphertext using a single decryption key", In *Pairing-Based Cryptography–Pairing*, (2007), pp.392–406.

[13] Fuchun G, Yi M, Zhide C & Li X, "Multi-identity single key decryption without random oracles", *Information Security and Cryptology*, (2008), pp.384–398.

[14] Amit S & Brent W, "Fuzzy identity-based encryption", *Advances in Cryptology–EUROCRYPT*, (2005), pp.457–473.

[15] Ming L, Shucheng Y, Yao Z, Kui R & Wenjing L, "Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption", *IEEE Transactions Parallel and Distributed Systems*, (2013), pp.131–143.

[16] Benaloh J, Chase M, Horvitz E & Lauter K, "Patient Controlled Encryption: Ensuring Privacy of Electronic Medical Records", *ACM Workshop on Cloud Computing Security ACM*, (2009), pp.103–114.

[17] Goldreich O, *Foundations of cryptography I: Basic Tools, Cambridge*, Cambridge University Press, (2001).

[18] Mihir B, Joe K & Phillip R, "The Security of the Cipher Block Chaining Message Authentication Code", *Journal of Computer and System Sciences*, Vol.61, (2001), pp.362-399.

[19] Bellare M, Canetti R & Krawczyk H, "Keying hash functions for message authentication", *Annual International Cryptology Conference*, (1996), pp.1-15.

[20] Rogaway P & Shrimpton T, "Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance", *International workshop on fast software encryption*, (2004), pp.371-388.

[21] Schneier B, *Applied Cryptography*, John Wiley & Sons, (1996).

[22] Mihir B, "New Proofs for NMAC and HMAC: Security without Collision-Resistance", *Advances in Cryptology CRYPTO*, (2006).

[23] The Keyed-Hash Message Authentication Code (HMAC). *Federal Information Processing Standards Publication*, (2008).

[24] Sikhar P, Yash S & Debdeep M, "Provably Secure Key-Aggregate Cryptosystems with Broadcast Aggregate Keys for Online Data Sharing on the Cloud", *IEEE Transactions on Computers*, (2016).

[25] Cheng KC, Sherman SMC, Wen GT, Jianying Z & Robert HD, "Key-aggregate cryptosystem for scalable data sharing in cloud storage", *Parallel and Distributed Systems, IEEE Transactions on Computers*, (2014), pp.468–477.

[26] Dan B, Craig G & Brent W, "Collusion resistant broadcast encryption with short ciphertexts and private keys", *Advances in Cryptology-CRYPTO*, (2005), pp.258–275.

[27] Sai Prasad K, Chandra SRN, Rama B, Soujanya A & Ganesh D, "Analyzing and Predicting Academic Performance of Students Using Data Mining Techniques", *Journal of Advanced Research in Dynamical and Control Systems*, Vol.10, No.7, (2018), pp.259-266.

[28] B Kassimbekova, G Tulekova, V Korvyakov (2018). Problems of development of aesthetic culture at teenagers by means of the Kazakh decorative and applied arts. Opción, Año 33. 170-186

[29] M Pallarès Piquer and O Chiva Bartoll (2017). La teoría de la educación desde la filosofía de Xavier Zubiri. Opción, Año 33, No. 82 (2017): 91-113