International Journal of Engineering &Technology, 7 (4) (2018) 6453-6457

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET
doi: 10.14419/ijet.v7i4.23293
Research paper

Data integrity proof and secure computation based on elgamal algorithm and iris extraction in cloud storage

Salah H. Abbdal Refish *

Imam Jaafar AL- Sadiq University, Najaf-Iraq Computer Engineering Department
**Corresponding author E-mail: manatheraa@yahoo.com*

Abstract

Storage servers may not be fully trusted in cloud storage. So, it is of critical importance for users to check whether the data stored are kept intact or not. In this paper, an efficient and secure integrity checking method based on iris feature extraction and Elgamal algorithm is presented. This method gives the cloud users more confidence in detecting any block that has been modified. Additionally, the proposed scheme supports data dynamics by employing Merkle Hash Tree (MHT), which is used to store the location of each data operation. Data dynamics include such data operations as block modification, insertion, and deletion. With the proposed method, provable data possession and remote integrity checking under secure computation are provided. The performance and security analysis show that the scheme is secure and can be practically used for cloud environments.

Keywords: *Cloud Storage; Data Integrity Checking; Iris Feature Extraction; Data Dynamics; MHT.*

1. Introduction

Cloud storage offers many benefits, including greater accessibility and reliability, rapid deployment, and strong protection for data backup over the Internet. Nowadays, a growing number of organizations and individuals prefer to remotely outsource their data to cloud storage to enjoy its benefits [1]. However, once a data owner uploads data and deletes the local copies of the files, the owner no longer possesses physical control over the outsourced data. Thus, protecting the integrity of data in the cloud is essential. Moreover, the data stored in the cloud are not only accessed but also frequently updated by users, with such actions as insertion, deletion, modification, and so on. Thus, findings ways to support the dynamic features of cloud storage has become an important research topic. The storage server is assumed to be untrustworthy in terms of both security and reliability. Most papers have introduced methods to manage user possession and integrity of data in cloud storage [2-4]. Whether the outsourced data are intact or not is one of the issues that users must deal with when using cloud storage. Gradually, the proposed schemes have attempted to address the dynamic data operations required in cases when users frequently update their data in the cloud. At the block level, the main operations are block insertion, block modification, and block deletion. Thus, another desirable feature of cloud storage is data dynamics. In this paper, I benefit from [5] to achieve my work, and I present a method for ensuring the possession and integrity of data as well as data dynamics. The proposed method is based on biometric technology, which is considered one of the modern approaches in the security field. Generally, biometrics employs physiological or behavioral characteristics to precisely identify a subject. Commonly used biometric features include the face, fingerprints, voice, iris, retina, gait, palm print, hand geometry, dental radiograph, and so on. This work involves the iris. Iris recognition has been considered an effective approach in carrying out individual identification during the last decade [6]. In this proposed system, iris features used that are taken from the cloud user. All data blocks and their meta-data are stored at cloud storage in the setup phase. When the cloud user challenges the storage server by selecting random data blocks to be verified, the server returns the proof of this challenge to the cloud user. The cloud user then computes his values of these data blocks and verifies whether they are equal. The Merkle hash tree (MHT) is introduced to support data dynamics when it records the location for each data operation.

The rest of this paper is organized as follows: Section 2 illustrates the related work, design issues are presented in Section 3, Section 4 describes the proposed scheme in detail, Section 5 addresses support data dynamics, Section 6 discusses security analysis and performance of this work, and finally, Section 7 concludes the paper.

2. Related work

Ateniese et al. [2] presented the first formally defined provable data possession (PDP) protocol, to verify the integrity of stored data in the cloud without downloading the entire data. They used homomorphic verifiable tags based on public key cryptography to produce a single tag by combining the block tags. Their method also allows a server to construct a proof and permits the client to check whether the cloud server has intact blocks, even if the client may not have access to the blocks. However, their scheme has the followingg drawbacks: it incurs high computation and communication overhead on the server; it cannot provide fully secure data possession, and it cannot prevent data leakage. In a later work, Ateniese et al. [7] proposed a new PDP, in which they improved the scalability and efficiency of the previous method [2]. In the new scheme, they used symmetric key cryptography to overcome the problem of data dynamics; however, the method is unable to completely solve the problem and also requires high computation-

al overhead on the data owner. Chen proposed a scheme using homomorphic hashing-based PDP, which provides unlimited number of queries [8]. However, the drawbacks included data dynamics and proof of retrievability (POR). Juels and Kaliski introduced the concept of POR and proposed a formal POR protocol definition and accompanying security definitions [3]. They used sample blocks, called sentinels, which are hidden among regular file blocks that the server cannot differentiate from encrypted blocks. Unfortunately, this scheme suffers from drawbacks. First, the number of challenges are limited; second, it requires computation and storage overheads, which arise from the error recovery and data encryption processes. Shacham and Waters [9] proposed another POR scheme, which is built from BLS signatures. However, because it has the shortest query and response with public verifiability, it is only applicable for private auditing. Dodis et al. [10] first formally defined the POR code.

The constructions either improved and generalized the prior POR constructions, or equipped existing POR schemes with the required properties. The main insight of their scheme comes from a simple connection between POR schemes and the concept of hardness amplification, which has been extensively studied in complexity theory. Wang et al. [11] first studied the problem of ensuring the integrity of data storage in cloud computing. They introduced a TPA to audit the cloud data storage and used a public key-based homomorphic authenticator with random mask technique to achieve a privacy preserving public auditing system. However, this public auditing cannot resist against existential forgery using a known message attack. Moreover, the protocol is vulnerable to attacks by a malicious cloud server and an outside attacker through four specific attacking schemes. In another study, Zhu et al. [12] proposed an interactive POR scheme that aimed to prevent the fraudulence of prover and the leakage of verified data. They gave full proofs of soundness and zero-knowledge properties by constructing a polynomial time rewindable knowledge extractor under the computational Diffie–Hellman assumption. In a later study, Wang [13] proposed a scheme-based bilinear pairing technique, which has a major drawback, that is, it cannot prevent information leakage. Yuan et al. [14] designed a scheme that aimed to support efficient and secure data integrity auditing with storage deduplication for cloud storage. Their proposed scheme is based on several techniques, including polynomial-based authentication tags and homomorphic linear authenticators; it also allows the eduplication of both files and their corresponding authentication tags. Data integrity auditing and storage deduplication are achieved simultaneously. However, their scheme does not support privacy and only permits static updates.

3. Design issues

3.1. Problem statement

We consider a cloud storage system consisting of three parts. The first one is the cloud user (CU) who possesses the data files that are to be stored in the cloud. The CU stores data on the cloud server and does not keep a local copy. Hence, the CU should be able to verify the integrity of the data stored in the remote non-trusted server. The second one is the cloud server, which is controlled by a third component that knows the cloud service provider (CSP) that will provide data storage service and has important storage space and computation resources. The CSP must ensure that all significant data are covered and only formal users have access to the data in its entirety. The CSP has the ability to ensure that applications available as a service over the cloud are secure from adversaries. Assume a general cloud computing model involving n cloud servers, which may be monitored by one or more CSP. The CU delegates his data to the cloud servers and employs the cloud servers as data storage, thereby submitting some functions for computation. Additionally, we must refer to an important component called the third party auditor (TPA). The TPA has skills and capability to evaluate the security of cloud storage service instead of the CU upon request. The CU depends on the CSP to save and preserve his data. The CU may also automatically cooperate with the CSP to arrive and update his stored data for different application purposes. Sometimes, CU relies on the TPA to guarantee the storage security of his data, while wishing to preserve his data private from the TPA.

3.2. Merkle hash tree (MHT)

The overall idea of Merkle hash tree is to compose a tree based on a one-way cryptographic hash function h (.) [15]. Then, each leaf node can be verified through its authentication path information (API). Since only the hash functions are computed, the computation cost of verification is too low. We explain the construction and application of the Merkle hash tree through a sample example (see Fig.1).

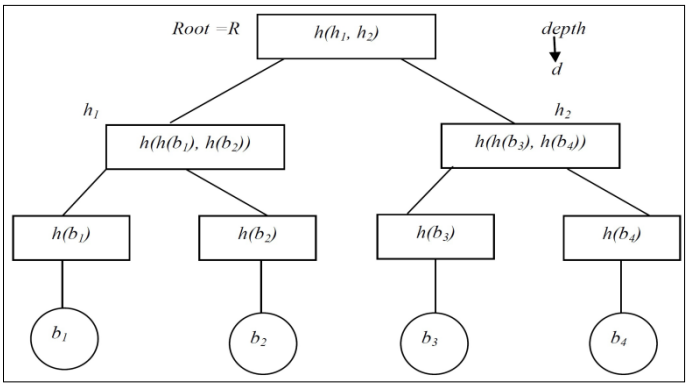

Fig. 1: Sample of MHT with Four Data Blocks.

The values of the four leaf nodes are the message hashes, i.e., $h_i=h(b_i)$, (i=1, 2…,4), respectively. The values of internal nodes are derived from their child nodes. For instance, the value of the node h_1 is $h_{1,2}=h(h(b_1),h(b_2))$, and the value of the node h_2 is $h_{3,4}=h(h(b_3),h(b_4))$. Each leaf node can be verified with R and the corresponding API. For instance, the node h1 can be authenticated by the server who stores R as follows: h_1 sends b_1 and the corresponding API=$(h(b_2),h_2)$ to the server. Then, the server can check the authenticity of node h_1 by first computing $h_1=h(h(b_1), h(b_2))$, $h_2=h(h(b_3), h(b_4))$, $R=h(h_1, h_2)$. And then, the server checks whether the computed R' is the same as the existing R. The server accepts h_1, only if the two values are equal.

4. The proposed scheme

In this section, I explain the scheme in details which aims to present a robust method for data integrity and support data dynamics. The scheme has three components, namely, the cloud user (CU), the cloud server provider (CSP), and the iris features (fr) of the cloud user. The proposed scheme will describe as follows.

4.1. Configuration process

First Stage: The CU generates his iris features. Fig.2 shows the main difference between traditional way (A) and proposed method (B) for obtaining iris features. As can be seen, the traditional way (A) is very costly, because needs hardware and software to obtain the iris features. The term “pay as you go” is a great solution in the cloud, and we can rely on the CSPs, such as Google, Amazon and Microsoft, which provide cost-effective services for both communication and computation. Meanwhile, (B) shows the mechanism for generating iris features based on the CSP; the data are stored in the CU’s USB for use in the proposed scheme (Fig 3).

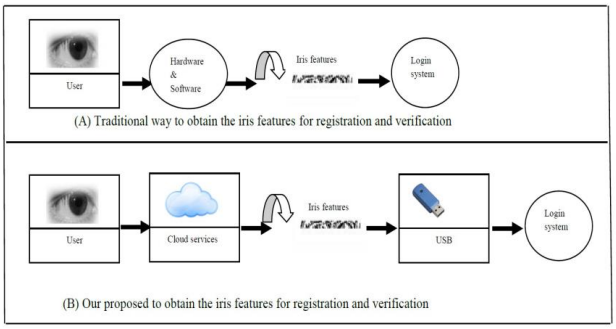

Fig. 2: Main Difference between Traditional Way (A) and Our Prposed (B) for Obtaining Iris Features.

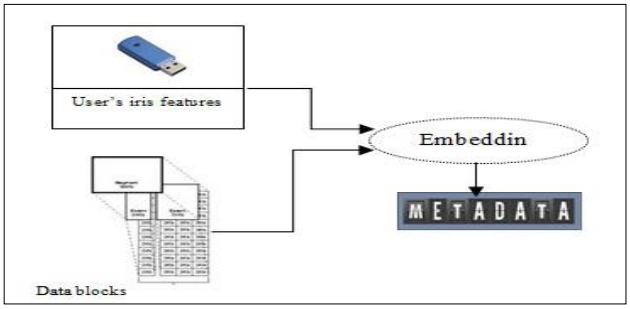

Fig. 3: Mechanism for Generating Metadata.

Second Stage: The CU splits his data file F into n blocks {b_1, b_2, …, b_n}, each containing m bytes as {d_1, d_2,…,d_m}. For every data block in F and use MHT to store metadata and original data into the cloud server. Fig. 4 describes how the metadata is generated and stored at cloud storage. In my scheme the metadata is generated as the following based iris extraction: With each two leafs foe example b_1, b_2: h_1=h(h(b_1), h(b_2),h(FI)), and so on for the rest. Where the FI is the extraction of iris.

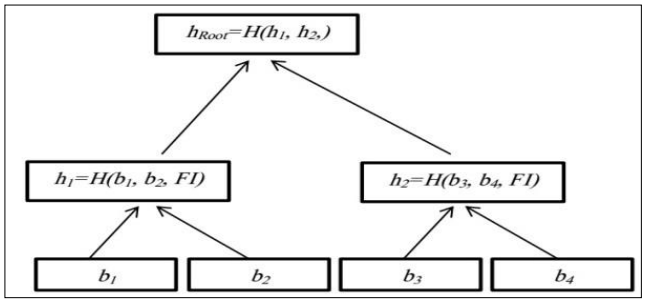

Fig. 4: Sample of Metadata which Stored at Cloud Storage.

4.2. Data integrity prove

In this stage, several important processes are followed to complete my checking. These are described below.

- The CU challenges the cloud server by specifying the block number i and the byte number j. So the verifier sends challenge (i, j) to the cloud server.
- The cloud user assumes that he has the following parameters:
- α, P, d
- He computes $\beta=\alpha^d \bmod P$
- Then, he sends β, α, P to the Cloud server
- The cloud server generates some values as follows:
- Assumes that the cloud server has t random number So, he computes $K_1= \alpha^t \bmod P$
- Then, calculates:
- $K_2=\beta^t \bmod P$
- $Y_1=b(i,j)* K_2 \bmod P$
- $Y_2=H(Y_1)$
- $E_i=Enc_{K2}(b(i, j))$, $b(i, j)$: value of original-data at the jth byte in the ith block.
- Next, the cloud server sends a response to the CU, which contains important elements: K_1, Y_1, Y_2 , E_i, $sig_{K1}(R)$
- Then, the CU follows the following procedures:
- The first thing is to generate $\bar{K}_2 = K_1^d \bmod P$
- Then, computes $b(i,j)=Y_1 * \bar{K}_2^{-1} \bmod P$
- Finally, the CU concludes b (i, j) using the $Dec_{K2}b(i, j)$, and compares Y_2 with the $b(i, j)* \bar{K}_2 \bmod P$ which are sent. If so, the authentication is applied. additionally, the CU uses original MHT according to the data block which sent by CS, to verify sig_{K1} (R)= sig_{K2} ((R), if so, the data integrity is proved, then the data are not modified; otherwise, if not equal, the data are modified.

4.3. Computation verification

Elgamal algorithm is used in this scheme for secure authentication [16]. The CU performs verification by choosing a random subset S={c_1, c_2, …., c_n} from the domain [1, n], and sends this challenge requests to the cloud server. For each $c_i \in S$, the cloud server finds in the Merkle hash tree a path of c_i, from the leaf to the root. For each node on this path c_i, then the cloud server sends the sibling sets and sig_{K2}(R) to the cloud user. The cloud user gets the values from the cloud server and generates the signature $sign_{\bar{K}_2}$ (R) using the result and the sibling value set sent by the cloud server. If the signature $sign_{\bar{K}_2}$ (R) matches with the sig_{K2}(R), the cloud user confirm that the computations are done correctly. This work is performed as shown in Fig. 5 bellow.

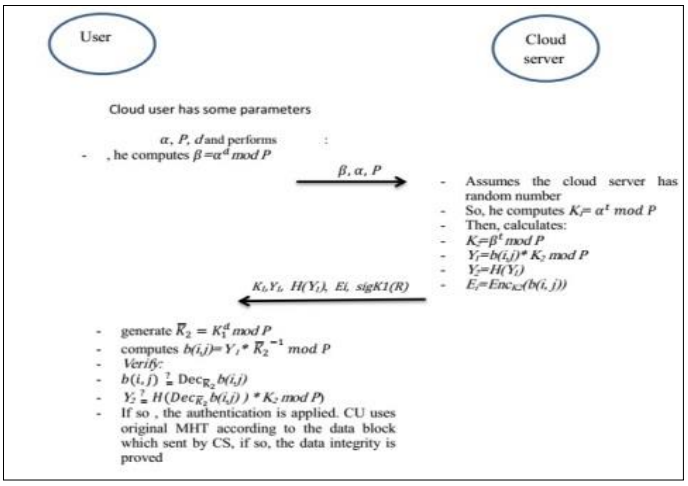

Fig. 5: System Flow Diagram.

5. Providing data dynamic

In this section, we show how the MHT based scheme can handle fully dynamic data operations including data modification (M), data insertion (I) and data deletion (D) for cloud data storage. We assume that the file F have already been generated and properly stored at cloud server.

Data modification: is one of the most frequently used operations in cloud data storage. A basic data modification operation refers to the replacement of specified blocks with new ones. Suppose the cloud user wants to modify the ith block b_i to b_i'. At the beginning, based on the new blockb_i', the cloud user generates the corresponding meta-data f'(i, j). Then he constructs an update request message (M, i,b_i', f'(i, j)) and sends it to the cloud server, where M denotes the modification operation. Upon receiving the request, the cloud server runs ExecUpdate(F, f(i, j), update): it replaces b_i with b_i' and outputs F'; it replaces f(i, j) with f'(i, j) in the MHT construction and produces the new root R'. Then, the cloud server responses the cloud user with a proof for this operation, P_{update}={Ω_i,b_i, (h(R)), R'}, where Ω_i is the auxiliary authentication information (AAI) for the authentication of b_i. After receiving the

proof for modification operation from cloud server, the cloud user generates the root R'' using (Ω_i, b_i) and authenticates R'' to R by checking whether (h(R'')) is equal to (h(R)). If it is not true, output FALSE, otherwise it checks whether the cloud server has performed the modification as required or not, by computing the new R value using (Ω_i', b_i') and comparing it with R'. If it is not true, output FALSE, otherwise output TRUE. Finally, the cloud user executes the default integrity verification. If the output is TRUE, delete (R', P_{update}, b_i') from its local storage.

Data insertion: it refers to inserting new blocks after some specified positions in the data file F. Suppose the cloud user wants to insert block b^* after the ith block b_i. It is similar to the data modification case, where b_i' can be viewed as b^*. At start, based on b^* the cloud user generates the corresponding $f^*(i,j)$ and then he constructs an update request message (I, i, b^*, $f^*(i,j)$) and sends to the cloud server, where I denotes the insertion operation. Upon receiving the request, the cloud server runs ExecUpdate(F, Ω_i, update): it stores and adds b^* after leaf b_i in MHT and outputs F', it adds f^*(i, j) into the tag set and outputs Ω_i'; it generates the new root R' based on the updated MHT. Then the cloud server responses the cloud user with a proof for this operation, P_{update}={Ω_i, b_i, (h(R)), R'}, where Ω_i is the AAI for the authentication of b_i. After receiving the proof for modification operation from cloud server, the cloud user generates the root R'' using (Ω_i, b_i) and authenticates $R''$to R by checking whether (h(R'')) is equal to (h(R)). If it is not true, output FALSE, otherwise it checks whether the cloud server has performed the insertion as required or not, by computing the new R value using (Ω_i', b^*) and comparing it with R'. If it is not true, output FALSE, otherwise output TRUE. Finally, the cloud user executes the default integrity verification. If the output is TRUE, delete (R', P_{update}, b^*) from its local storage.

Data deletion: This is the opposite operation of data insertion. It indicates removing the particular block and moving all the following blocks one block forward. For instance, the cloud server receives the update request for deleting block b_i, it will delete b_i from its storage space, delete the leaf node h(b_i) in the MHT and generate the new hash tree root R'. The detail of the procedure is similar to that of data modification and insertion operations.

6. Security analysis and experimental results

6.1. Security analysis

Theorem 1: *Our scheme provides key management.*

Proof. First, the CU generates α, *P*, *d* parameters and computes $\beta = \alpha^d \bmod P$ then he sends β, α, *P* to the cloud server. The cloud server then calculates $K_1 = \alpha^t \bmod P$ and then computes $K_2 = \beta^t \bmod P$, $Y = b(i,j) * K_2 \bmod P$. $E_i = Enc_{K2}(b(i,j))$. Finally, the cloud user sends (K_1, *Y*, E_i) to the cloud user to regenerates $\bar{K}_2 = K_1^d \bmod P$, $b(i,j) = Y * \bar{K}_2^{-1} \bmod P$. Thus, the proposed scheme can provide key management.

Theorem 2: *The proposed work can supply the security of iris features.*

Proof. Noticed that the communication messages only include (i, j, M(i, j), b(i, j)) and do not contain any information about the iris features extraction. The iris feature extractions and verification messages are completely individualistic. In addition, the cloud server does not include file iris features, thereby decreasing the processing time of our proposed scheme and reduces malicious attacks. Thus, our work ensures the security of the iris features extraction.

Theorem 3: *Proposed scheme withstands replay attack.*

Proof. Generally, replay attack is a form of attack in which valid data is fraudulently repeated or delayed. An attacker performs a replay attack by eavesdropping at auditing message $E_i = Enc_{K2}(b(i,j))$, which is transmitted by the CSP to the cloud user. After the interchange between the CSP and the cloud user, the attacker uses key K_2 to impersonate the valid CSP when the cloud user asks the CSP to audit the file again. In our proposed scheme, auditing key K_2 is changed after each auditing process. Moreover, the key K_1 is a one-time key and any attacker cannot computes $\bar{K}_2 = K_1^d \bmod P$ without getting K_1. Therefore, an attacker cannot pass any replayed key for an auditing file. Consequently, the attacker will fail to perform this type of attack because the proposed scheme has deflect edit.

Theorem 4: *This work can provide known-key security.*

Proof. In this scheme, the Diffie–Hellman key exchange is used to provide more secure session key to be shared key between the cloud user and cloud server. The shared key is started from the cloud user when he computes $\beta = \alpha^d \bmod P$, then the cloud server generates k_1, k_2 according to the parameters β, α, P which sent by cloud user and it's finished to the verifier for each verification phase. Therefore, the proposed scheme can gain known-key security.

Theorem 5: *The proposed scheme can provide recoverability.*

Proof. The scheme addresses illegal updating for outsourcing data block. I notice that in verification phase when the result of sig_{K1}(R $\neq^?$ sig_{K_2}(R) is not equal. So the data block has modified. Therefore, the cloud user should returns original data block to the cloud server by sends $\bar{E}_i = Enc_{K_2}\, \bar{b}$(i,j), and regenerates the MHT again according to the sibling set. So, the proposed scheme can gain recoverability.

Theorem 6: *Proposed scheme can resist Man-In-The-Middle (MITM) attack.*

Proof. Generally, the MITM able to intercept message between the cloud user and cloud server. He uses that message when the cloud user sign out the cloud server. In the verification phase, some factors encrypted and securely send it between cloud user and cloud server since the Diffie-hellmam is used between them. These factors become useless in the sign-off phase since they use these factors only one time for each verification. So, the scheme can resist MITM attack.

6.2. Efficiency analysis

The CU generates and encrypts the metadata and then appends such data to the original data before storing them at the cloud server. The iris feature extraction ensures efficiency, high performance, and security. The efficiency of this work has been tested by measuring the response time of the CS. Our work has been executed and tested on a database containing iris features of many users. These iris features (fr) were acquired randomly from the CU. Figure 6 shows that the response time is increased linearly with the number of users. Furthermore, the average time for the data integrity prove stage of this work is equal to 0.02220 seconds for each user which indicates the high speed of my solution.

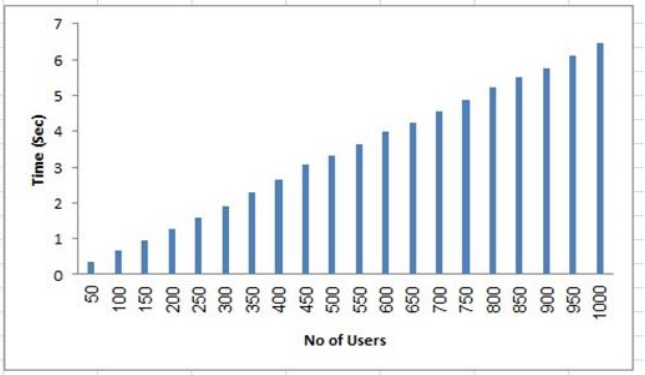

Fig. 6: Average Time of Data Integrity Prove for Proposed Scheme.

7. Conclusion

An efficient and secure scheme to verify the integrity of data stored in cloud storage is introduced. The method employs the iris feature extraction properties of the outsourced data blocks to remotely check the integrity of the files, thus minimizing the computational and communication costs on the CU and cloud server. The proposed work supports data dynamics especially block insertion, which is not provided by most existing schemes. To enable data dynamics, the MHT is presented to record the location for each data updating operation. The CU can perform unlimited number of verification. Proposed method provides PDP and integrity protection.

References

[1] M. Sookhak, A. Gani, M. K.Khan, and R. Buyya, "Dynamic remote data auditing for securing big data in cloud computing," *Journal of Information Sciences*, Elseveir, Sept, pp. 1-16,

[2] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable Data Possession at Untrusted Stores," Proceedings of the 14th Conference on Computer and Communication Security (CCS'07). Alexandria, USA, ACM, pp. 598-609, 2007. https://doi.org/10.1145/1315245.1315318.

[3] A. Juels and B.S. Kaliski Jr., "Pors: Proofs of Retrievability for Large Files," Proceedings of the 14th Conference on Computer and Communication Security (CCS'07), Alexandria, USA, ACM, pp. 584-597, 2007. https://doi.org/10.1145/1315245.1315317.

[4] KD. Bowers, A. Juels, and A. Oprea, "HAIL: A High-Availability and Integrity Layer for Cloud Storage," Proceedings of the 16th Conference on Computer and Communications Security (CCS'09), Chicago, IL, USA, ACM, pp. 187-198, 2009. https://doi.org/10.1145/1653662.1653686.

[5] A. A. Yassin, H. Z. Neima, and H. SH. Hashim, "Security and Integrity in Cloud computing Based on Feature Extraction of Handwriting Signature," International Journal of Cyber-Security and Digital Forensics (IJCSDF), Vol. 3, No. 2, pp. 93-105, 2014. https://doi.org/10.17781/P001299.

[6] P. S. Patil, S. R. Kolhe, and R. V. Patil, "The Comparison of Iris Recognition using Principal Component Analysis, Log Gabor and Gabor Wavelets," International Journal of Computer Applications, vol. 43, no.1, April, pp. 29-33, 2012. https://doi.org/10.5120/6070-8229.

[7] G. Ateniese, R.D. Pietro, L.V. Mancini, and G. Tsudik, "Scalable and Efficient Provable Data Possession," Proceedings of the 4th International Conference on Security and Privacy in Communication Networks (SecureComm'08), Istanbul, Turkey, pp. 1-10, 2008. https://doi.org/10.1145/1460877.1460889.

[8] C. Lan-xiang, "A homomorphic hashing based provable data possession," Journal of Electronics and Information Technology, JEIT, Vol. 33, No. 9, pp. 2199-2204, 2011. https://doi.org/10.3724/SP.J.1146.2011.00001.

[9] H. Shacham and B. Waters, "Compact Proofs of Retrievability," Proceedings of the 14th Annual International Conference on the Theory and Application of Cryptology and Information Security, Advances in Cryptology (ASIACRYPT'08), Melbourne, Australia, pp. 90-107, 2008. https://doi.org/10.1007/978-3-540-89255-7_7.

[10] Y. Dodis, S. Vadhan, and D. Wichs, "Proofs of retrievability via hardness amplification," International Association for Cryptologic Research, Springer, pp. 109-127, 2009. https://doi.org/10.1007/978-3-642-00457-5_8.

[11] C.Wang, Q. Wang, K. Ren and W. Lou, "Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing," Proceedings of the 29th Conference on Computer Communications (INFOCOM'10), San Diego, USA, IEEE, pp. 1-9, 2010. https://doi.org/10.1109/INFCOM.2010.5462173.

[12] Z. Yan, W. Huaixi, Z. Hu, A. Gail-Joon, and Hu. Hongxin, "Zero-knowledge proofs of retrievability," Science China Information Sciences, Vol. 54, No. 8, pp. 1608-1617, 2011. https://doi.org/10.1007/s11432-011-4293-9.

[13] H.Wang, "Proxy provable data possession in public clouds," IEEE Transactions on Services Computing, pp. 551-559, 2013. https://doi.org/10.1109/TSC.2012.35.

[14] J. Yuan and Yu. Shucheng, "Secure and constant cost public cloud storage auditing with deduplication," Proceedings of International on Communications and Network Security (CNS), pp. 145-153, 2013. https://doi.org/10.1109/CNS.2013.6682702.

[15] R. C. Merkle, "Protocols for public key cryptosystems," Proceedings of the IEEE Symposium on Security and Privacy, pp. 122-134, 1980. https://doi.org/10.1109/SP.1980.10006.

[16] Z. Wu, Di Su, & Gang Ding (2014).ElGamal Algorithm for Encryption of Data Transmission. 2014 International Conference on Mechatronics and Control (ICMC). July 3 - 5, 2014, Jinzhou, China. 978-1-4799-2538-4/14/$31.00 ©2014 IEEE. https://doi.org/10.1109/ICMC.2014.7231798.