JOURNAL OF BASIC SCIENCE AND ENGINEERING

An Effective Auditing Framework to Confirm Data Confidentiality and Integrity in Cloud Data Storage

Mr. Shivarajkumar Hiremath[1] and Dr. Prashantha G. R[2]

[1] Research scholar, Department of IS&E, SKSVMACET, Lakshmeshwar, Visveswaraya Technological University (VTU), Belagavi, India.
[2]Professor, Department of CS&E. JIT, Davanagere, Visveswaraya Technological University (VTU), Belagavi, India.

DOI: https://doie.org/10.0618/Jbse.2024168991

Abstract

Cloud computing involves delivering computing services like servers, databases, storage, and networking over the Internet. Cloud service providers (CSPs) are responsible for overseeing the data stored on the cloud. Users benefit from time and memory savings when they utilize cloud storage. However, once data is uploaded to the cloud, it becomes inaccessible to the user. Hence, safeguarding user data in the cloud necessitates tackling several critical security challenges. In our study, we employed a Third-Party Auditor (TPA) to develop a secure auditing system. We utilized the Secure Hash Algorithm (SHA-2) for computing the message digest and applied the Advanced Encryption Standard (AES-256) for encrypting user data. The system operates by launching a Windows Server instance on the Amazon EC2 cloud platform. The results demonstrate that our proposed method is secure and requires a significant amount of time to verify the accuracy of files stored in the cloud.

Keywords: Cloud-based Data, Public Verification, Data Integrity assurance, Data Privacy.

1. Introduction

The practice of utilizing remote services via the internet is termed as cloud computing. It provides network, server, storage, and application access on demand. Utilizing cloud computing services enhances data scalability, availability, and agility. It allows consumers and organizations to store enormous amounts of data in the cloud. Still, many individuals remain cautious about storing their data in the cloud[1] because of security apprehensions. In order to safeguard their reputations, certain cloud service providers may opt to conceal data breaches; others might make more room by deleting less often requested or unnecessary data [2].

The locations of service providers' cloud servers vary across various regions and nations. Consequently, the data is dispersed throughout the network in data centres. Concerns over the safety of user data are raised by this. These are maintained through the use of cryptography and auditing techniques. One useful technique to verify the accuracy of cloud data is auditing. It is the process of having data verified either directly by the customer or by a third-party auditor (TPA).

The Trusted Third Party audit (TPA) is responsible for scrutinizing data stored in the cloud and possesses superior expertise and capabilities compared to users. Clients are not involved in the auditing process by TPA. Because of this, users will find it easier to maintain

their data because TPAs tend to be less burdensome. TPA shouldn't be aware of data kept in the cloud, nor should it charge cloud customers more for communications [3].
An independent entity TPA can validate the correctness of data stored remotely due to its public auditability. Nonetheless, the TPA might access user data content through the information collected during the auditing process.

This serious flaw has a significant impact on these protocols' security in cloud computing. Therefore, in order to safeguard cloud data, a public auditing system that respects privacy must exist. In the proposed study, we use established cryptography protocols to ensure a public auditing scheme for data safety while protecting privacy.

The suggested work would prioritize achieving data privacy, data integrity, and data secrecy. Throughout the auditing process, the TPA don’t get data about the data content kept on the cloud. Thus, users no longer have to worry about their outsourced data leaking, and it stores their data securely, relieving them of the tiresome and costly auditing chore. The primary goal of the planned research is to create a safe cloud data storage system.

2. Literature Survey

The rank of cloud data storage is impacted by various aspects, including data dynamics, privacy, confidentiality, integrity, and integrity. We discovered three workable solutions, Proof of Retrievability (POR), Proof of Ownership (POW), and Provable Data Possession (PDP), are utilized to handle integrity verifications in cloud storage.

Ateniese et al. [4] introduced the Provable Data Possession (PDP) model, advocating for the idea of public auditability to evaluate remotely stored data. They utilized homomorphic linear authenticators (HLA) for this purpose. This system generates proofs of ownership by randomly selecting unspecified block sets, effectively decreasing input/output costs. The data owner employs metadata, particularly hash functions, to verify the proof. The challenge protocol facilitates communication with minimal data exchange, thus reducing network overhead. Despite their method involving the linear arrangement of sampled blocks visible to an external auditor, it does not entirely ensure user privacy, potentially exposing user data to the auditor.

M. A. Shah et al. [5] introduced a PDP model based on MAC (Message Authentication Code) to ensure the integrity of files stored in cloud storage. The owner employs a series of confidential keys to ascertain the file's MAC. Prior to cloud storage, these keys are securely stored locally. The file is transmitted from the local system to the Cloud Service Provider (CSP), which retains only the MAC value. When the client requests file retrieval from the CSP to verify content integrity, the client cross-references the stored MAC with the one provided by the CSP. However, the proposed system may not be suitable for handling large files.

A Merkle hash tree-based deduplication approach was proposed by Shai Halevi et al. [6]. Under this scheme, the owner of a file F can demonstrate that he owns the file to the server, but an external rival who does not have the complete file cannot do so. A user can use an erasure code to encrypt the whole contents of a file F and then build a Merkle tree using the encoded file blocks to demonstrate file ownership. As a proof of ownership of F, the user

will then submit the sibling-paths of every node that the server requests. Because the Merkle Tree in this technique is constructed using every file block, it can become very huge and take a long time to build, which lowers efficiency.

Pietro and Sorniotti [7] proposed an effective "POW" system to increase POW's efficiency. The technique is divided into two phases: the first occurs when a file is received by the server for the first time, and it uses that file to precompute the answers to several POW challenges. POW challenges for a specific file are computed when a request to upload a file that is not yet on the server is received and when the available supply of previously computed challenges and responses is exhausted. This plan is quite effective.

The client initiates the second stage when it provides the server with a file's unique identification that it wants to verify. From the pre-computed challenges for that file, the server selects one that is not in use and delivers it to the client, who uses what it knows about the file to determine the response and sends it back to the server. The server then verifies if the response from the client and the one that was precomputed match. The plan outperforms the work suggested by Shai Halevi et al. [6] and is provably safe, but it falls short in addressing concerns about confidentiality and privacy.

The original Proof of Retrievability (POR) concept was formulated by Jules and Kaliski in 2007 [8]. Their suggested framework utilizes a sentinel-oriented strategy coupled with error-correcting codes to guarantee data integrity and retrievability. Nevertheless, the incorporation of sentinels and error-correcting codes leads to higher storage overheads on the server end. The scheme allows for a restricted number of challenge algorithm executions, bound by the precalculated sentinels included within the encoded file. Furthermore, their system lacks support for public auditing.

To tackle the constraint of limited executions, Shacham and Waters [9] introduced an upgraded version of the POR scheme named Compact POR. Their method integrates two pivotal elements: effective homomorphic verifiers, derived from pseudorandom functions (PRFs), and BLS signatures. PRFs bolster the security of the PoR scheme within the standard model, while BLS signatures facilitate public verifiability. Unlike the earlier Jules and Kaliski [8] scheme, which permits only a restricted number of challenge algorithm runs, Compact POR enables an infinite number of queries, thus diminishing communication overhead. However, despite these enhancements, this revised scheme also lacks public auditing integration.

Cong Wang et al. [10] introduced a secure cloud storage system that facilitates privacy-conscious public auditing to bolster confidentiality during the audit phase. Their solution employs a randomized masking method alongside homomorphic linear authenticators (HLA) to ensure that the Third-Party Auditor (TPA) remains uninformed about the data content stored on the cloud server. This strategy eases the audit burden for cloud users, diminishing both the time and expense involved, while addressing their anxieties regarding the security of their outsourced data. Moreover, they expand the privacy-preserving auditing protocol to a multiuser setting, enabling the TPA to efficiently conduct numerous audit tasks in a single batch. Despite its efficacy in safeguarding privacy and enabling public audits, this approach still raises certain security apprehensions

Vol. 21, No. 1, (2024)
ISSN: 1005-0930

Table 1: Summary of Literature Survey

Schemes	Techniques	Benifits	Shortcomings
Provable Data Possession	Key Generation Algorithm	• This technique gives a robust proof of data integrity. • Allows public verifiability.	• Lack of privacy preservation.
Provable Data Possession	Message Authentication Code (MAC)	• Simple & Secure Technique. • Gives robust proof of Data Integrity.	• Not appropriate for big file. • Public auditability is not reinforced
Proofs of Ownership in Remote Storage Systems	MHT based deduplication	• Time Saving • Identify attacks • Saving bandwidth	• Less Efficient
Secure Proof of Ownership	Cryptographic hash function	• Better performance • Provably Secure	• Communication Overhead
Implementing Proof of Retrievability for large files	Sentinel-based approach and error-correction code	• Ensures the retrievability and possession of files on archival service systems	• Storage overhead at the server
Compact POR	HAs & BLS	• Supports an unlimited queries • Requires minimal communiqué in the clouds	• Applicable only to fixed data
Third Party Audit	• Implemented with Multiple servers • Achieves data integrity.	• Achieves data integrity	• Computational costs are impacted by the size of data blocks.

3. Proposed Methodology

The suggested approach comprises three components constituting the proposed public auditing system

Cloud server: It offers efficient tools for users to create, store, modify, and request data retrieval. Additionally, it has the infrastructure and expertise to meet external storage requirements.

User(s): Individuals who store data on the cloud delegate IT-related responsibilities to experts while managing their own activities.

Vol. 21, No. 1, (2024)
ISSN: 1005-0930

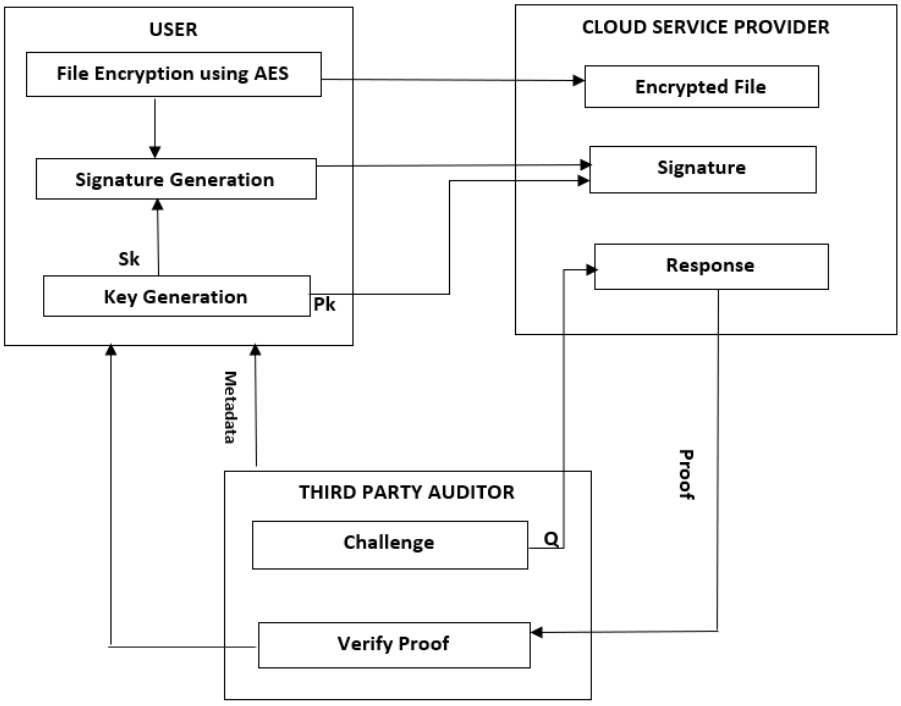

Fig 1: Proposed Model

TPAs: When necessary, consumers can rely on an extra party with greater qualifications and experience to assess the legitimacy and dependability of cloud storage. The ideas of Baidaa et al. [11] are applied in the development of this system. The process is split into two stages.

1. Pre-processing phase
2. Verification Phase.

During the initial processing stage, the Client divides the provided file Fi into a series of data blocks bn. These blocks are then encrypted using the Advanced Encryption Standard (AES) with a 256-bit key. After encryption, the resulting encoded data file is created. Subsequently, the client generates public keys, private keys, and signatures for each encrypted block. The encrypted file, along with its associated signatures, is then sent to the cloud server. Since the cloud stores the file in encrypted form, the Cloud Service Provider (CSP) cannot access the user's data. The outlined framework is shown in figure 1.user's data. The outlined framework is shown in figure 1.

3.1 Pre-processing phase

The initial processing stage includes 3 algorithms namely Data Protection, Key Generation, and Signature Generation.

3.1.1 Data Protection

In this procedure, the data file Fi undergoes division into data blocks (X1, X2, X3, … Xn). Subsequently, these chunks are encoded by the AES algorithm, resulting in the encrypted data file Ɛ = (a1, a2, … an).

Algorithm for Data Protection

Vol. 21, No. 1, (2024)
ISSN: 1005-0930

Input: File F
Output: Encrypted File Ɛ
1. Splits Fi into block (X1, X2, X3, … Xn).
2. Apply AES 256 algorithm to (X1, X2, X3, … Xn) to obtain Ɛ = (a1, a2, … an).
3. Return Ɛ

3.1.2 Key Generation

Implemented at the user's end, this process produces a public key Bk and a super key Uk. The algorithm utilizes a arbitrary variable value r as a secret key Uk ∈ Yp and generates a public key Bk for each block. The user retains Uk while disclosing Hsk, representing the public key ∈ H.

Algorithm for Key Generation
Input: Arbitrary variable r
Output: Uk (Secret Key)and Bk (Public Key)
1. Selects a arbitrary variable value r in the range 0 to k-1
2. Find the public key as Bk=Hsk
3. Return Uk and Bk

3.1.3 Signature Generation Algorithm

In this algorithm, user generates a signature Ui for each data block by using Uk, Bk, a random value z and encrypted data Ɛ. Here Ui is an verification identifier for each block.

$$Si=[G(Xi).H^{a_i}]]^{Uk_i} \quad \text{(Eq. 1)}$$

Where G is a hash function, Xi represents block i, Uki and is secret key for corresponding data block Xi and $Si=H^{a_i}$ is produced by operator for every chunk in encrypted file Ɛ. Later the metadata M is created that constitutes (Si,Xi). Following this, the client transmits the encoded information chunks, along with their respective monograms and public keys, to the bank of cloud service provider.

Algorithm for signature generation algorithm.
Input: Uk, Bk, random value y and Ɛ = (a1, a2, … an).
Output: Signatures Ui
1. Compute hash value for each block Xi using SHA-256
2. Calculate (Vi) using $Vi=.H^{a_i}$
3. Compute Ui using equation 1
4. Generate metadata M. (M=(Vi,Xi)
5. Upload file to cloud (Ɛ, Ui, Bki)
6. Return Ui.

3.2 Verification Phase

This phase is used to validate the information kept in bank of cloud storage is intact or not. This phase includes TPA challenge generation to cloud service provider ($GenerateChallenge_{TPA}$), Response from cloud service provider ($Response_{csp}$) and Verification of Proof by TPA($VerifyProof_{TPA}$).

3.2.1 Generate Challenge to Cloud Service Provider

Vol. 21, No. 1, (2024)
ISSN: 1005-0930

Whenever a user wishes to substantiate the reliability of data kept at the Bank of cloud Service Provider (CSP), they can enlist the assistance of a Third Party Auditor (TPA) to conduct an audit. The user forwards metadata M to the TPA for auditing purposes. Upon getting the audit demand after the user, the TPA engages with the CSP by issuing a challenge. This challenge is generated by the TPA using a random subset Q of size k from the set [1, n], denoted as Ai ∈ Q, alongside a randomly chosen qi ∈ Zp. Each element in Q is represented as (i, qi), where i ∈ Q. Subsequently, the TPA transmits Q to the CSP.

Algorithm for Generate ChallengeTPA

Input: Metada M

Output: Challenge Q

1. User triggers TPA
2. User transmits metadata M to TPA
3. TPA generates challenge Q = {i, qi}, where i denotes the challenge block number, and prompts CSP with Q.
4. Return Q

3.2.2 Response form CSP

Upon receiving the challenge Q from the TPA, the CSP relays the query to the user to verify the legitimacy of the TPA. If the user validates the query, the CSP proceeds to accept the challenge Q from the TPA; otherwise, the challenge is disregarded. After the CSP accepts the challenge Q, it generates proof U for the given challenge. Using BLS, multiple signatures for different blocks can be merged into a single signature as detailed in Equation (2). Finally, the CSP transmits proof U to the TPA.

$$U = \sum_{i=0}^{k} (Ui) \quad \text{(Eq.2)}$$

Algorithm for ResponseCSP

I/P: Query, Public Key Bk, Response Q, Super Key U,
Encrypted Data E

O/P: Proof B

1. CSP requests user's approval for the challenge.
2. If user confirms, proceed to step 3.
3. Upon user confirmation, CSP:
4. Constructs U using Equation (2).
5. Generates the Proof P = {U, Bki}.
6. Transmits B to the TPA.
7. Return B as output.

1.2.1 Verify Proof by TPA

Once TPA receives proof from CSP, TPA validate the proof from CSP. If proof is intact then TPA sends success message to the user.

Algorithm for Verify Proof

Input: Pk, P, Q, metadata M

Output: Pass/Fail

- The TPA audits proof from CSP

Vol. 21, No. 1, (2024)
ISSN: 1005-0930

- Conveys the audit outcome to the handler.
- Returns either attainment or failure.

4. Results and Discussions

The projected methodology is established and implemented on an Amazon EC2 instance using Py programming. A Microsoft Windows Server 2016 Base instance, featuring a 2.5GHz Xeon processor and 1GB of RAM, is deployed. The front-end is designed using CSS3 and HTML. AES-256 encryption is utilized for securing data. Message digest is generated using the SHA-256 algorithm, yielding a unique 32-byte hash for a given text. Below are some hash values obtained during the implementation of our suggested approach.

1) 7w7e3ffg4gha6jajd6j12cb49n01vzjdyeydfk692c147ckkalc71l38gakc08375va9
2) 3ta49sfsg4iaf881fxcsoet 4926c926659zn6cgxk294762rcnkdtavxspqi028gvjsh3

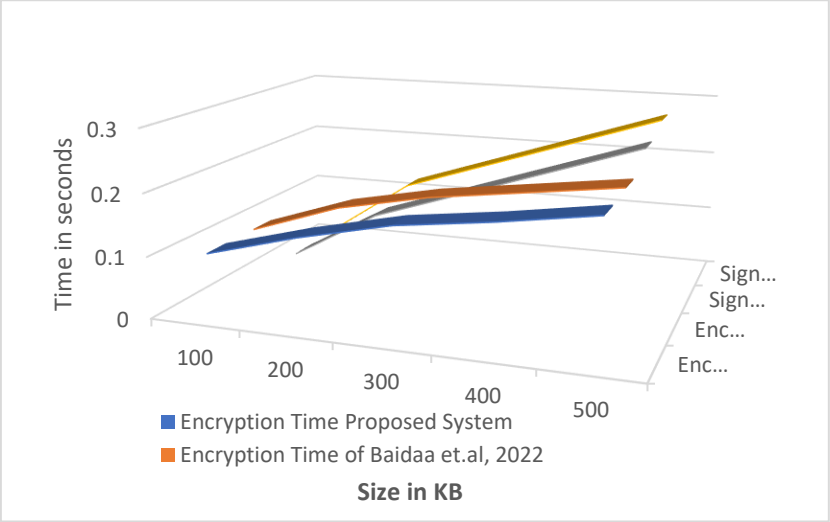

Fig 2: Computation cost of Encryption Time and Signature Generation Time

An investigational structure of the projected scheme is as revealed as follows.

• Programming : Python
• Front End : HTML5 and CSS3
• Virtual Instance: Amazon EC2's Ubuntu Server 16.04
• Input - Files ranging from 100KB to 500KB
• Output – Auditing time, Proof generation time etc

We evaluated the performance in terms of encryption time, time for generating keys and signatures, proof generation time by CSP, and proof verification time by the TPA. We compared our results with Baidaa A J et.al [11]. The results of this experimental analysis are, our proposed scheme considers almost constant time in all aspects and the results shown are proven. The comparison of results is shown in following figures.

Vol. 21, No. 1, (2024)
ISSN: 1005-0930

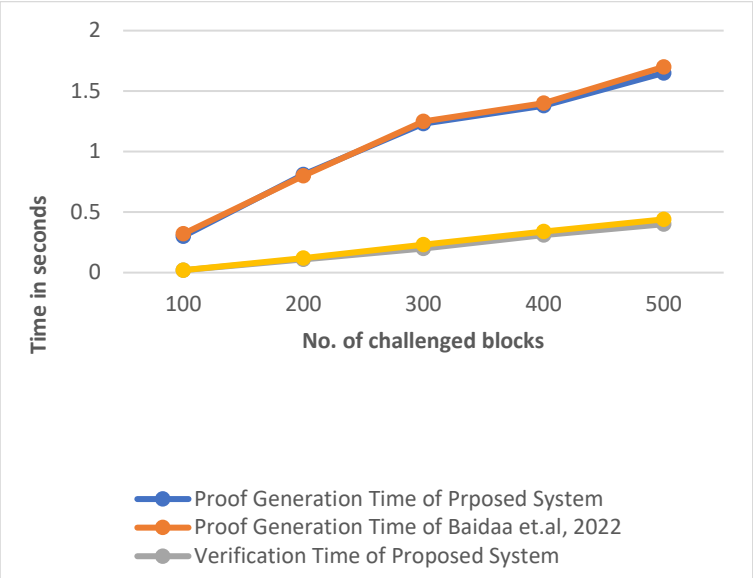

Fig 3: Computation cost of Proof Generation and Verification Time

5. Conclusion

In order to protect privacy when storing data in the cloud, this article suggested a third-party auditing mechanism. Data integrity and secrecy were both achieved by the suggested solution. Without obtaining a cloud user's data copy, TPA conducts audits. To assess the suggested system's performance, we compared it to correlated solutions. The outcomes demonstrate that the suggested solution is secure and ensures data integrity when stored in the cloud. Data recovery methods and data dynamics must be completed in the future.

References

[1]. W. A. Sultan Aldossary, "Data Security, Privacy, Availability and Integrity in Cloud Computing", International Journal of Advanced Computer Science and Applications,Volume 7, Issue 4, pp. 485-498, 2016.

[2]. N. K. Yang and X. Jia, "Data Storage Auditing Service in Cloud Computing: Challenges, Methods and Opportunities", World Wide Web, Volume 15, Issue 4, pp. 409–428, 2012.

[3]. Cong Wang , Sherman S M Chow, Qian Wang, Kui Ren, and Wen jing Lou. "Privacy Preserving Public Auditing for Secure Cloud Storage." IEEE Transactions on Computers, Volume 62, Issue 2, February 2013.

[4]. Ateniese, G., Burns, R.C., Curtmola, R., Herring, J., Kissner, L., Peterson, Z.N.J., Song, D.X., "Provable Data Possession at Untrusted Stores", In proceedings of ACM Conference on Computer and Communications Security, CCS 2007, pp. 598–609, 2007. DOI:10.1145/1315245.1315318.

[5]. M. A. Shah, M. Baker, J. C. Mogul, and R. Swaminathan, "Auditing to Keep Online Storage Services Honest", In proceedings of 11th workshop on Hot Topics in Operating Systems (HotOS'07). Berkeley, CA, USA: USENIX Association, pp. 1-6, 2007. DOI: 10.1.1.532.2861.

[6]. Shai Halevi, Danny Harnik, Benny Pinkas, and Alexandra Shulman-Peleg, “Proofs of Ownership in Remote Storage Systems”, in Proceedings of the 18th ACM conference on Computer and communication security, ser. CCS’ 11, 2011. DOI: 10.1145/2046707.2046765.
[7]. Roberto Di Pietro, Alessandro Sorniotti, “Boosting Efficiency and Security in Proof of Ownership for Deduplication”, ASIACCS ’12, May 2–4, Seoul, Korea, 2012. DOI: 10.1145/2414456.2414504.
[8]. Juels Jr., A., Kaliski, B.S., “POR’S: Proofs of Retrievability for Large Files”, In proceedings of the ACM Conference on Computer and Communications Security, CCS 2007, pp. 584–597, 2007. DOI: 10.1145/1315245.1315317.
[9]. Shacham, H., Waters, B., “Compact Proofs of Retrievability”, In Journal of Advances in Cryptography, Volume 26, Issue 3, pp. 442-483, July 2013. DOI: 10.1007/s00145-01 2-9129-2.
[10].Wang, C., Wang, Q., Ren, K., Lou, W., “Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing”, In proceedings of INFOCOM, 2010 Proceedings IEEE, pp. 1– 9, March 2010. DOI: 10.1109/INFCOM.2010.5462173.
[11]. Baidaa Abdulrahman Jalil, Taha Mohammed Hasan, Ghassan Sabeeh Mahmood, Hazim Noman Abed, “A secure and efficient public auditing system of cloud storage based on BLS signature and automatic blocker protocol”, Journal of King Saud University - Computer and Information Sciences, Volume 34, Issue 7, 2022, Pages 4008-4021, ISSN 1319-1578, DOI: 10.1016/j.jksuci.2021.04.001.