Public Data Auditing Scheme using RSA and Blockchain for Cloud Storage

A. Vamshi*, R. Eswari, and Shivam Dubey

Department of Computer Applications
National Institute of Technology Tiruchirappalli, India
vamshiadouth47@gmail.com*

Abstract. With the widespread cloud storage devices, most users store their data over cloud storage. A dishonest user might raise a false claim about the integrity of their data to get compensation. On the other hand, a corrupt Cloud Service Provider (CSP) might hide the fact from the user that their data has been compromised. So, ensuring the integrity of the data is of utmost importance in such cases. The integrity of user data is ensured through Third Party Auditor (TPA), who is supposed to be trusted and possess the computation resources to perform the auditing. Although this TPA reduces the burden of verifying the data, there's still a risk of TPA, who may not be honest. The TPA might collude with the CSP or not perform its task correctly. This paper proposes a Public Data Auditing Scheme using RSA and Blockchain (PDASRSAB) for cloud storage to bring security and transparency into the auditing process, which increases confidence in cloud storage, along with the experimental analysis for the proposed scheme.

Keywords: Blockchain · RSA · Data auditing · Cloud storage.

1 Introduction

Cloud computing is widely gaining importance from users and researchers. It has been one of the fastest adoptions into mainstream life compared to other technologies in this domain. Cloud computing is an efficient way to provide/manage information and communication resources to remote users. The importance of cloud computing in today's world stems from this process of solving several problems faced by the average user and large organizations. With such an exponential rate of development of cloud computing, we can now access its facilities much more quickly and efficiently. Cloud computing has different applications in today's world, like Big-Data Analytics, SaaS (i.e., Software as a Service) , SD-WAN (i.e., Software-Defined Wide Area Networking), and so on. However, there are a lot of challenges, like lack of privacy, quality, and safety protocols in existing methods. Cloud Service Providers (CSPs) around the world have faced many issues, such as the LinkedIn data breach in 2021, Amazon cloud's storage outage in 2021, Facebook data breach in 2019, Twitter data breach in 2018, Tencent cloud user's silent error in 2018, and Intuit's power failure in 2010. So,

it becomes essential in such a scenario to ensure the integrity of the data to increase the confidence of users and organizations in cloud services.

Now the question arises, "How can we verify the integrity of the data?". Neither the user, i.e., the Data owner (DO), nor the Cloud Service Provider (CSP) can blindly trust each other. A malicious CSP might hide an incident of data failing in integrity verification, or a malicious DO might claim false data integrity. Therefore, in such a case, we need a Third Party Auditor (TPA) who can verify the integrity of the data. There are many existing Data Auditing schemes where a TPA with sufficient computing power checks the data integrity when the DO suspects it or can't verify it on its own. The user or DO sends a request for the audit to a Third Party Auditor or TPA. The TPA then generates a challenge and sends it to the CSP (it asks the CSP for proof to verify the data integrity). In response, the CSP generates the proof by performing cryptographic operations on the data blocks and then provides the proof to the TPA. Once TPA receives proof generated by CSP, it verifies the data integrity and sends the Audit report to the DO.

The problem with such Auditing schemes is that the DO has to trust the TPA completely. It might be possible that the TPA might not be performing its task correctly and efficiently or that the TPA could collude with the malicious CSP to conceal that DO's data. Another possibility is that the TPA's system is hacked by some hacker, which again poses a considerable security risk. We can replace the TPA with a blockchain network to avoid the above risks. Introducing blockchain into the auditing scheme can eliminate the possibility of a malicious TPA. Since tampering with a blockchain is not a feasible task and also due to its transparency, a blockchain network is an ideal candidate for the job of a trusted TPA. The DO can send the auditing request to the Public Auditor (i.e., blockchain network), which can verify the integrity of the data by generating a challenge for the CSP. Based on the proof given by the CSP, it can tell whether the data's integrity is compromised.

1.1 Contribution

We propose a Public Data Auditing Scheme for Cloud Storage in this paper. The following are the contribution made in this paper.

1. We propose a Public Data Auditing Scheme using RSA and Blockchain for Cloud Storage, which is not dependent on a traditional Third Party Auditor (TPA). This prevents the auditing process from malicious auditors and makes it more secure.
2. We used an RSA-based key generation centre (KGC) to sign the data and store it in the cloud.
3. In experimental analysis, we test the proposed scheme for various file sizes, which shows that the proposed scheme is suitable for cloud-based devices.

2 Related work

A. Juels et al. (2007) first proposed PoRs: Proofs of Retrievability for large files, which checks the correctness of data stored [1]. Later Ateniese et al. (2007) proposed PDP: Provable Data Possession at Untrusted Stores. They used RSA-based homomorphic linear authenticators for data -auditing[2]. In 2011 Wang et al. came up with Privacy-Preserving Public Auditing for Secure Cloud Storage. In this approach, TPA verifies the correctness of the cloud data. [3]. Zhu et al. (2012) proposed a model of Cooperative provable data possession for verifying data integrity in multi-cloud storage. [4]. Yang et al. (2013) proposed a dynamic auditing protocol in which a TPA verifies the correctness of the cloud data by using verifiable homomorphic tags and data fragmentation [5]. Later, Yang et al. proposed a model that provides auditing for data over cloud storage which supports traceability and privacy. Its drawback is the burden of computational burden on the Group Manager (GM), and dependency on the TPA [6]. Debiao He et al. proposed a Public Auditing Scheme which utilises Certificateless cryptography. The scheme has four entities, a third-party auditor, a KGC, a user, and a cloud server and uses certificate-less public auditing (CLPA) [7]. P. Singh et al. proposed A Secure Data Dynamics and Public Auditing Scheme for Cloud Storage. This model consists of three entities: D.O, CS, and TPA. It uses AES256, RSA-15360, and SHA512 algorithms. The drawback is its dependency on TPA [8]. Although the various models and schemes proposed by different researchers[9–17] have some advantages, one of the major drawbacks of these schemes is, they make the assumption that CS or TPA has no motivations whatsoever, to collude in the auditing process.

3 Preliminaries

3.1 RSA Algorithm

RSA (Rivest, Shamir, Adleman) algorithm is an asymmetric cryptography algorithm [18].Ron Rivest, Adi Shamir, and Leonard Adleman proposed this algorithm in year 1977. Asymmetric cryptography works with two different keys known as Public Key and Private Key. In the proposed scheme, we use RSA algorithm to generate the keys and sign the data. As of now, there are no published work that claims to break the system as long as a very large key is used. The algorithms involved in RSA are Public-Private key generation, encryption, and decryption.

Key Generation:

1. Select any two different prime numbers p and q (preferably large prime numbers for stronger encryption)
2. Calculate the RSA modulo N, computed as N = p*q.
3. Calculate a derived number Z, such that: Z = (p-1)*(q-1)
4. Randomly choose an integer e in such a way that, the following conditions are meet:

$$e < Z. \tag{1}$$

$$gcd(e, Z) = 1 \tag{2}$$

5. Return the public-private key pair. Here, d is private key (e, N) and d is public key.

Encryption:

1. Input the public key (e, N) and message m
2. Generate the cipher-text c such that: $c = m^e \ mod \ N$

Decryption:

1. Input the private key d and cipher-text c.
2. Generate the corresponding message m by computing $m = c^d \ mod \ N$

3.2 Merkle Hash Tree (MHT)

Merkle Hash Tree (MHT) is tree data-structure where the leaf nodes are the contains the of the data-blocks. The non-leaf node contains the combined hash of it's child-node. The Merkle Hash Tree was patented by Ralph Merkle, in 1979. The structure of the MHT is such that, it allows a very competent mapping of large data. Even a small change results in the data-blocks can cause the hash-value in root to change drastically. The hash in the root acts as the fingerprint for whole data.

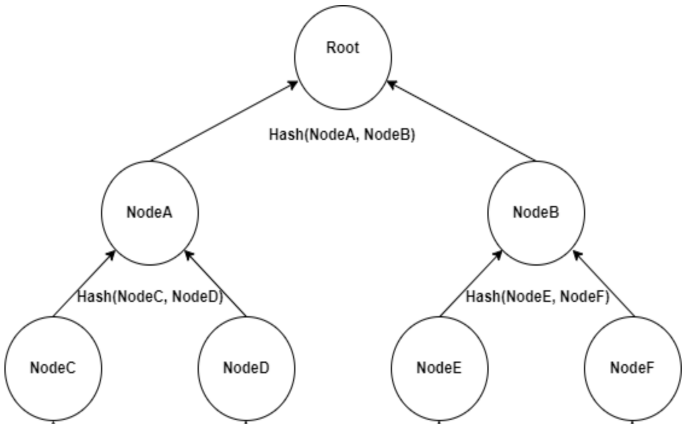

Fig. 1. Overview of MHT

3.3 Blockchain

Blockchain is a distributed database which is append-only by nature and is shared among the different nodes. It was first developed in 2008, by an anonymous person(s) by the name of Satoshi Nakamoto. As the name suggests, it's a chain of blocks that can be defined as a kind of DLT (Digital Ledger Technology), that consists of a growing list of records called blocks. These blocks contain a hash of transaction data, previous block, timestamp etc. Blockchain works on a P2P (peer-to-peer) network, in which all the nodes follow a consensus protocol for verifying/adding new transactions. The transactions cannot be altered in a blockchain once recorded. Blockchain was recently popularised because of its use in bitcoin (Cryptocurrency). The following are the 3 fundamental properties of a secure blockchain-systems:

1. X-chain consistency: During the mining process at a given time, the blockchains of two minners can only differ in last X blocks.
2. (l, X)-chain quality: The probability of any X consecutive blocks over a blockchain are produced by an opposing miner having hash-rate < 51 percent of entire network's mining hash-rate is negligible.
3. Chain growth: The height of a blockchain steadily increases w.r.t. long term as well as short term.

4 Proposed Scheme

4.1 System model

In the proposed scheme, user (Data owner) divides the file in blocks. These blocks are then encrypted by using the public key provided by the RSA based Key Generation Center. Once the file is divided into blocks, then for each block their is a hashtag generated by performing a hash operation on each one of the data blocks. Then the user broadcasts these hash-tags on the blockchain network. After that, user uploads the encrypted file on the cloud server. Figure.2. shows the system model.

Whenever a Data Owner (DO) needs to verify integrity of its file, it sends an auditing request the Public auditor(blockchain). Then public auditor then generates a challenge for the CSP. On receiving the challenge, the cloud server calculates the MHT root for the blocks of the file stored. Then the cloud server sends the MHT root as the proof to the public auditor. When the public auditor receives the proof it then compares this proof with the MHT root calculated by itself. If both matches it means the file's integrity is not compromised.

4.2 PDASRSAB-Auditing Framework

The following algorithms are used in the system model:

1. GenerateKey(): Used by the KGC to generate public-private keys for encryption-decryption.

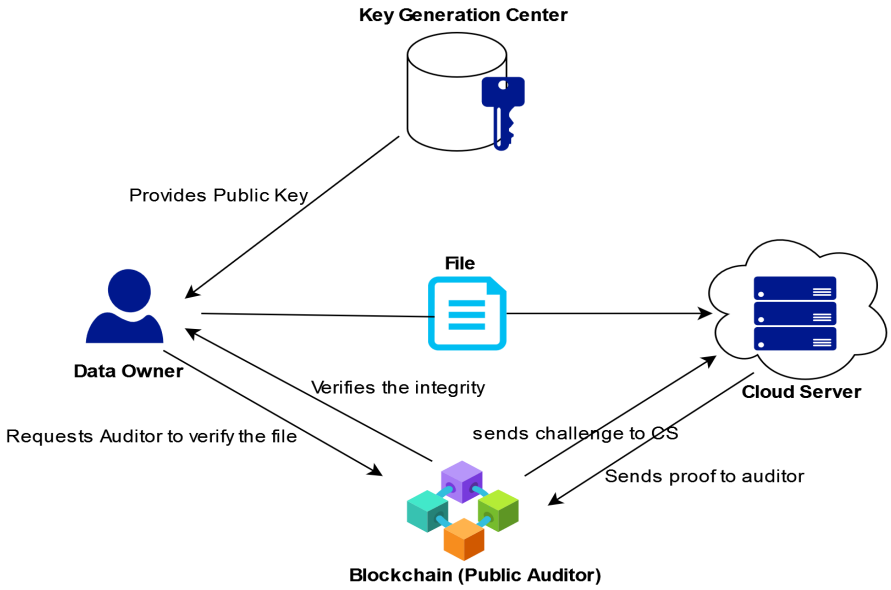

Fig. 2. System model

2. Encrypt(): Used by the Data Owner for encryption of file using public-key provided by KGC.
3. Decrypt(): Used by the CSP to decryption of file using the private key provided by KGC.
4. DivideAndGenHashTags(): Used by the Data Owner and CSP to divide the encrypted files into blocks and then generate hashtags for each corresponding block.
5. MHT_{root}(): Used by the Data Auditor (Blockchain) to calculate the MHT root for the blocks of the file.
6. GenerateProof(): Used by the CSP proof generation (MHT root) from the blocks of the files.
7. VerifyProof(): Used by the Data Auditor (Blockchain) to compare the MHT root that it has calculated with the one provided by the CSP as proof.

4.3 Notations

Table 1 shows the frequently used notations in proposed scheme.

Table 1. Notations

Description	Symbols
Key Generation Center	KGC
Data Owner	DO
Cloud Service Provider	CSP
Derived number	Z
Public key	(e,N)
ith data block	Bi
Hash of ith block	Hi
ith Encrypted Block	E_{Bi}
Message	m
Cipher text	c
Root of Merkle Hash Tree	MHT_{root}

4.4 PDASRSAB Construction

Setup: On the input of two parameters p and q (both prime numbers, preferably large prime numbers for stronger encryption), the algorithm returns the public-private key pair $(e, N), d$. Here (e, N) is a public-key and d is private key. First the algorithm calculates the RSA modulo N, which is computed as: $N = p * q$. Then a derived number Z is calculated as: $Z = (p - 1) * (q - 1)$ Once Z is computed, the algorithm Randomly chooses an integer e such that the following conditions are meet:

1. $e < Z$.
2. gcd $(e, Z) = 1, i.e.$, both should be co-prime.

Now for the corresponding value of e, another integer d is calculated in such a way that: $e * d = 1 \bmod Z$
Sign Generation On the input of public key i.e., (e, N) and message i.e., m, the Data Owner generates the cipher-text c such that: $c = m^e mod\ N$
Divide and Generate Hashtag On input of the cipher-text c (obtained after encryption), the algorithm first divides it into n blocks $(E_{B1}, E_{B2}, E_{B3}, ...E_{Bi})$. For each block a corresponding hashtags value is calculated as $(H1, H2, H3, ..Hi)$ where $Hi = Hash(E_{Bi})$. With the help of these hashtags, the MHT_{root} is calculated, later which helps in verifying the integrity of our data.
Calculate MHT root On input of the list of hashtags $(H1, H2, H3, ...Hi, ...)$ for the corresponding blocks $(E_{B1}, E_{B2}, E_{B3}, ...E_{Bi}, ...)$ of message m broadcasted by the user over the blockchain network, this algorithm computes the MHT_{root} as
$MHT_{root} = Hash(NodeA, NodeB)$
$NodeA = Hash(NodeC, NodeD)$
$NodeB = Hash(NodeE, NodeF)$... till the leaf node is reached.
Here, each non-leaf node is a hash of its two child nodes. So in this way, if there's a change even in a single block Hashtag Hi of the message m, the MHT_{root} will be completely different. Hence making it very easy to verify the integrity of the

message by simply comparing the MHT_{root}.
Proof Generation On input of the cipher text c received from the DO, CSP divides it into n blocks $(E_{B1}, E_{B2}, E_{B3}, ...E_{Bi}, ...)$. For each block, the value of a corresponding hashtag is calculated as $(H1, H2, H3, \ldots Hi, \ldots)$ where $Hi = Hash(EBi)$ The CSP then receives a challenge from the Public Auditor (Blockchain). In response to the challenge, CSP computes the MHT_{root} as
$MHT_{root} = Hash(NodeA, NodeB)$
$NodeA = Hash(NodeC, NodeD)$
$NodeB = Hash(NodeE, NodeF)$... till the leaf node is reached. Once the MHT_{root} is computed, the CSP sends it as the proof to the Public Auditor.
Verify On input of the proof generated by the CSP, (which is nothing but the MHT_{root} computed by the CSP from the corresponding Hashtags of the message m), the algorithm compares the proof and the MHT_{root} to verify the integrity of the data. If the value of the proof and MHT_{root} are found to be equal, then the integrity of the message m is conserved; otherwise, it is compromised.
if $ProofCSP = MHT_{root} => Integrity$ conserved
if $ProofCSP \neq MHT_{root} => Integrity$ compromised

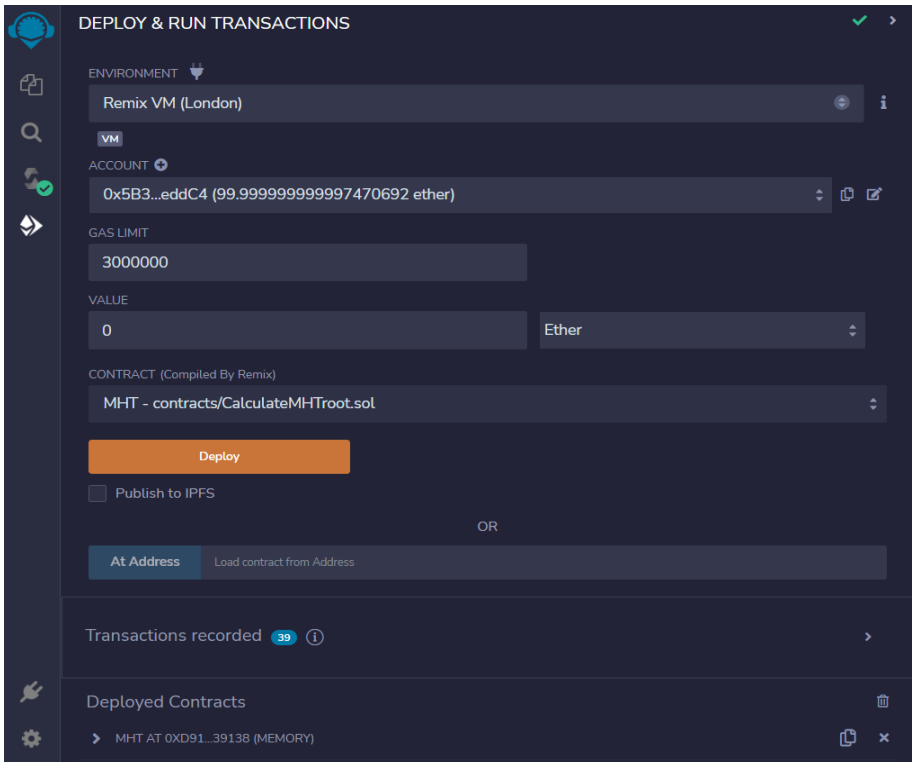

Fig. 3. Smarts contracts deployment

5 Experimental analysis

5.1 Cryptographic Setup

For the Cryptographic setup, we have used the RSA algorithm. This part was run on AMD Ryzen 7 3750H with Radeon Vega Mobile Gfx 2.30 GHz with 8.00 GB RAM, using C++ language and STL library.

5.2 Blockchain Setup

For the blockchain setup, we have used Remix - Ethereum IDE as shown in figure 3, which allows development, deployment, and administration of smart contracts for Ethereum-like blockchains. Protocol for verifying the integrity of the message is implemented on the blockchain by using smart contracts. Smart contract is basically a program that runs over a Ethereum-blockchain. This program resides at a particular address on the blockhain. For this setup smart contracts are written in solidity 0.8.3. The smart contract is implemented on Environment: Remix VM (London), used to connect to sandbox blockchain in browser. The Remix VM is its own "blockchain" . On each and every reload new blockchain is started and old one is deleted. Figure 3 shows the deployment of the smart contract. The details of smart contract deployed locally are as follow:
Account: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
Gas Limit : 3000000
Memory : 0xd9145CCE52D386f254917e481eB44e9943F39138
Unit : Ether

The following are the transaction and execution cost of various tasks/operations performed by the smart contract:

1. Broadcasting hashtag corresponding each data-blocks on blockchain network.
   transaction cost : 48834 gas
   execution cost : 48834 gas
2. Calculating MHT root.
   transaction cost : 157223 gas
   execution cost : 157223 gas
3. Checking integrity of data.
   transaction cost : 0 gas
   execution cost : 23624 gas

5.3 Experimental results

We evaluate the performance of our scheme by measuring the time taken in various operations like Sign generation, proof generation and verification for different file sizes. The results are given in below table 2. We have taken an average of readings with the same file size.

The graphs in figure 4, figure 5, and figure 6 shows relation between the size of file and the time taken in Sign Generation, Proof Generation, and Proof Verification respectively.

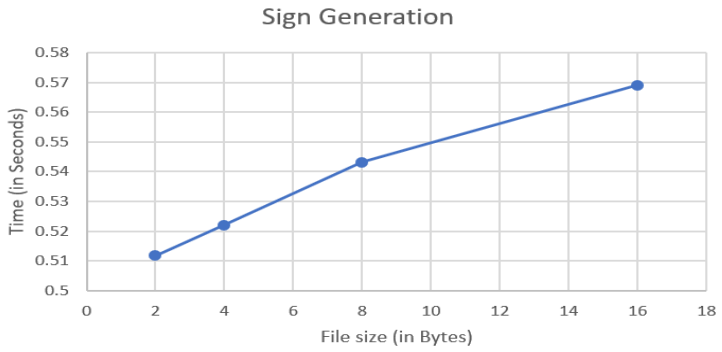

Fig. 4. Sign Generation

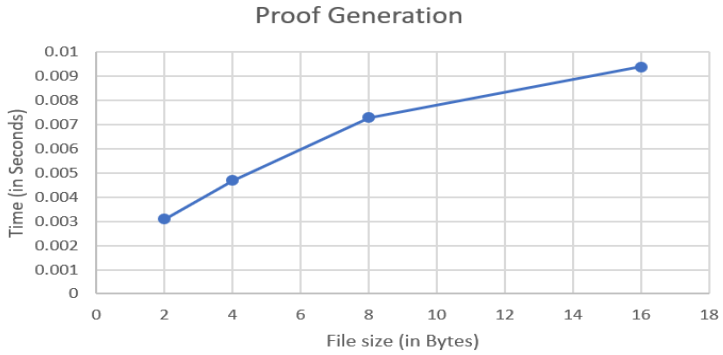

Fig. 5. Proof Generation

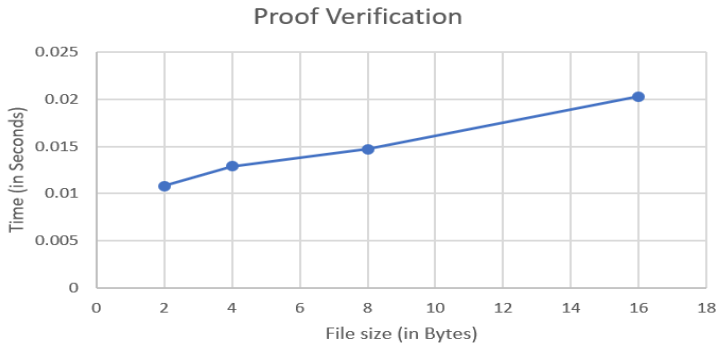

Fig. 6. Proof Verification

Table 2. Experimental results

S.no	File Size(bytes)	Sign Gen(sec)	ProofGen(sec)	Proof Verify(sec)
1	2	0.5118	0.0031	0.0108
2	4	0.5221	0.0047	0.0129
3	8	0.5332	0.0073	0.0147
4	16	0.569	0.0094	0.0203

6 Conclusion

We discussed the drawbacks and risks involved in the available public data auditing schemes that relies on a TPA. To overcome the drawbacks of TPA based auditing schemes, we proposed a PDASRSAB (Public Data Auditing Scheme using RSA and Blockchain)- Auditing framework, which uses RSA algorithm, for key generations, and Blockchain as a public auditor. We explained the proposed model in detail followed by the experimental analysis.

References

1. Ari Juels and Burton S Kaliski Jr. Pors: Proofs of retrievability for large files. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 584–597, 2007.
2. Giuseppe Ateniese, Randal Burns, Reza Curtmola, Joseph Herring, Lea Kissner, Zachary Peterson, and Dawn Song. Provable data possession at untrusted stores. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 598–609, 2007.
3. Qian Wang, Cong Wang, Kui Ren, Wenjing Lou, and Jin Li. Enabling public auditability and data dynamics for storage security in cloud computing. *IEEE transactions on parallel and distributed systems*, 22(5):847–859, 2010.
4. O Rahamathunisa Begam, T Manjula, T Bharath Manohar, and B Susrutha. Cooperative schedule data possession for integrity verification in multi-cloud storage. *Int J Modern Eng Res (IJMER)*, 3:2726–2741, 2012.
5. Kan Yang and Xiaohua Jia. An efficient and secure dynamic auditing protocol for data storage in cloud computing. *IEEE transactions on parallel and distributed systems*, 24(9):1717–1726, 2012.
6. Guangyang Yang, Jia Yu, Wenting Shen, Qianqian Su, Zhangjie Fu, and Rong Hao. Enabling public auditing for shared data in cloud storage supporting identity privacy and traceability. *Journal of Systems and Software*, 113:130–139, 2016.
7. Debiao He, Sherali Zeadally, and Libing Wu. Certificateless public auditing scheme for cloud-assisted wireless body area networks. *IEEE Systems Journal*, 12(1):64–73, 2015.
8. Premlata Singh and Sushil Kr Saroj. A secure data dynamics and public auditing scheme for cloud storage. In *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pages 695–700. IEEE, 2020.
9. Lunzhi Deng, Yixian Yang, Ronghai Gao, and Yuling Chen. Certificateless short signature scheme from pairing in the standard model. *International Journal of Communication Systems*, 31(17):e3796, 2018.

10. Lunzhi Deng, Yixian Yang, and Yuling Chen. Certificateless short aggregate signature scheme for mobile devices. *IEEE Access*, 7:87162–87168, 2019.
11. SK Hafizul Islam. A provably secure id-based mutual authentication and key agreement scheme for mobile multi-server environment without esl attack. *Wireless Personal Communications*, 79(3):1975–1991, 2014.
12. Michael Scott, Neil Costigan, and Wesam Abdulwahab. Implementing cryptographic pairings on smartcards. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 134–147. Springer, 2006.
13. Arijit Karati, SK Hafizul Islam, and GP Biswas. A pairing-free and provably secure certificateless signature scheme. *Information Sciences*, 450:378–391, 2018.
14. Debiao He, Baojun Huang, and Jianhua Chen. New certificateless short signature scheme. *IET Information Security*, 7(2):113–117, 2013.
15. Gaimei Gao, Hongxia Fei, and Zefeng Qin. An efficient certificateless public auditing scheme in cloud storage. *Concurrency and Computation: Practice and Experience*, 32(24):e5924, 2020.
16. Jiguo Li, Hao Yan, and Yichen Zhang. Certificateless public integrity checking of group shared data on cloud storage. *IEEE Transactions on Services Computing*, 14(1):71–81, 2018.
17. Rui Zhou, Mingxing He, and Zhimin Chen. Certificateless public auditing scheme with data privacy preserving for cloud storage. In *2021 IEEE 6th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA)*, pages 675–682. IEEE, 2021.
18. Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.