Journal of Ambient Intelligence and Humanized Computing (2022) 13:3787–3805
https://doi.org/10.1007/s12652-021-03321-7

ORIGINAL RESEARCH

LDuAP: lightweight dual auditing protocol to verify data integrity in cloud storage servers

Mohamed Sirajudeen Yoosuf[1] · **R. Anitha**[1]

Received: 2 April 2020 / Accepted: 3 June 2021 / Published online: 13 June 2021

Abstract
Cloud Service Providers (CSP) allow the users to store their data in the cloud storage servers. However, outsourcing the confidential data increases the security vulnerabilities as the data owner loses the physical on-premise control over the data. In order to verify the integrity of the outsourced data, auditing has to be performed frequently. Existing public auditing schemes completely depend on the third party auditor (TPA) to verify the integrity. If the trustworthiness of the TPA is compromised, then the TPA might send the fraudulent integrity result to the data owners. Existing schemes does not possess cross verification procedures to overcome the trust issues associated with the TPA. In addition, most of the existing public auditing schemes use RSA and BLS signatures to verify the authenticity of the data owner. Due to large key size, the computation time to perform auditing remains high. To overcome these issues, an LDuAP (lightweight dual auditing protocol) based on the Cramer-Shoup cryptosystem has been proposed. It combines both public and private auditing schemes to improve the authenticity of the integrity results. Initially, a lightweight public auditing is performed for all the data blocks stored in the cloud. Later, to cross-verify the integrity results generated by the TPA, private auditing is performed. The proposed scheme reduces the size of the signature by 50% and subsequently reduces the overhead of the entire auditing scheme. The extensive implementation assessments and security analysis exhibit the legitimacy and efficiency of the proposed scheme.

Keywords Integrity verification · Dual auditing · Cramer-Shoup cryptosystem · Cloud security · Third party auditor

1 Introduction

Cloud computing is a technology, which provides a wide range of on-demand services to the cloud users over the internet. Cloud computing encompasses different properties viz., rapid elasticity, fault tolerance, measured service and resource pooling has made it as a promising technology. In recent years, cloud computing has obtained a large variety of users from IT companies, enterprises, educational institutes and healthcare sectors. According to the recent report "2020 Cloud Computing Survey" of IDG Communications Inc., around 92% of information technology based organizations have adopted the cloud technology and outsourced their data to the cloud servers and only 8% of organizations are maintaining their data in the local on-premise servers.

Data storage is one of the essential service offered by the cloud service provider (CSP). Data owners (DO) can outsource the data to cloud storage server and can access it from anywhere. Once the data are uploaded to the cloud storage, data owner loses the physical control over the data. It increases the data vulnerability. Challenges like data tampering, deletion and data crash are more common in the cloud environment. CSP may delete less accessed user data from the cloud to improve the storage efficiency without the knowledge of the data owners (Liu et al. 2018; Tabrizchi and Kuchaki 2020). Moreover, to maintain the reliability and reputation of the company, CSP may hide the information pertaining to security attacks, crashing and tampering of data from the data owners (Khan et al. 2018).

To overcome these challenges, the data owner has to be convinced that the outsourced data are kept safe in the cloud storage servers. In case, if the outsourced data face threats from the notorious cyber-attackers or internal entities, then the system has to notify the data owner immediately. It can

✉ Mohamed Sirajudeen Yoosuf
ducksirajsmilz@gmail.com

R. Anitha
ranitha@svce.ac.in

[1] DST Cloud Research Lab, Department of Computer Science and Engineering, Sri Venkateswara College of Engineering, Tamilnadu, India

be achieved only through an efficient data integrity verification mechanism.

Data integrity is a process, which helps to maintain the accuracy, validity and consistency of the user's data over its entire lifecycle (Ateniese et al. 2007). Any unintended or unauthorized changes identified over the data, it is reported immediately to the data owners. The two major categories of data integrity verifications are private and public integrity verification. Private auditing allows the data owners or information proprietor to directly contact the CSP to verify the integrity of the cipher text stored in the cloud servers. Data owner holds metadata information about the encrypted data blocks and their corresponding storage locations. From the metadata information, at a regular interval, the data owner challenges the CSP to prove the integrity of the cipher text. Private auditing schemes are more secure and accurate. But, it puts an extra workload on the data owners by letting them to challenge the cloud service provider directly. As a result, private auditing increases the computation and communication overhead on the data owners, which is dreadful on a service oriented applications. (Ateniese et al. 2009; Wang 2013).

The second type of integrity verification scheme is public auditing, which is very popular in the service oriented cloud applications. It reduces the communication and computation workload on the data owner by introducing a TPA. It acts as a bridge between the data owners and the cloud service providers. TPA helps the data owners to verify the integrity of the data blocks stored in the cloud servers. Data owner shares a limited information about the data chunks to the TPA to challenge the CSP. Incorporating TPA reduces the communication, computation and metadata storage cost at the data owner's side. Introducing a third parties between the data owners and cloud service providers results in increased data vulnerability and trust issues (Kang et al. 2017).

Completely trusting and relying on Third Party Auditor (TPA) for integrity verification is not advisable as it may produce inaccurate results to the data owners in two circumstances. (1) If the trustworthiness of the TPA is compromised, then the TPA may produce fraudulent integrity results to the data owners. (2) TPA maintains a dedicated index table to perform integrity verification tasks. Due to false positive errors in the index table, the TPA may produce inaccurate results. (Jiang et al. 2013, 2018).

In addition, most of the existing public auditing schemes use an RSA signature (Venkatesh et al. 2012; Yu et al. 2016; Xu et al. 2017) and BLS (Benoh–Lynn–Shacham) signature (Wang et al. 2011; Mukundan et al. 2014; Liu et al. 2015; Xiling et al. 2018; Zhang et al. 2019) to verify the authenticity of the data owners in the cloud environment. But, the size of the keys used in the RSA and BLS signature schemes is large. So, the computation time to generate signature, proof and verifying the authenticity remains high. To manage and verify the integrity of high velocity data in the cloud environment, the authenticity verification procedure should take a low computational cost to achieve maximum result.

1.1 Drawbacks of existing auditing schemes

The drawbacks of existing auditing schemes are (1) data owner faces high computation and communication overhead while performing private auditing. (2) Existing public auditing schemes completely rely on third party auditors and trusts the integrity results without any cross-verification. (3) TPA produces erroneous results due to false positive errors in the index table. (iv) Authenticity verification of data owners using RSA and BLS signature results in increased computation cost on the public auditing schemes.

1.2 Contributions

To overcome the drawbacks of existing auditing schemes and considering the necessity of lightweight, secure integrity verification scheme in the cloud environment, this research work proposes an LDuAP (lightweight dual auditing protocol). It combines both public and private auditing schemes to verify the integrity of the cipher text stored in the cloud servers. Initially, for all the data chunks stored in the cloud, the proposed LDuAP scheme verifies the integrity using a third party auditor. Later, depending upon the computational power and importance of the data, the private auditing is performed for a random number of data blocks.

The contributions of the proposed LDuAP scheme are,

- Introduces a lightweight secured public auditing scheme based on the Cramer Shoup cryptosystem to reduce the computation and communication overhead.
- Performs private auditing for random data blocks and cross verifies the integrity results generated by the Third Party Auditor. It potentially increases the authenticity of the auditing results.
- To overcome the false positive errors in the TPA's index table, a scalable Bloom Matrix Hash Table (BMHT) is introduced.
- Lightweight signature is introduced to verify the authenticity of the data owner.

The rest of the paper is organized as follows: Sect. 2 summarizes the related works that are carried out in the data integrity verification schemes. Section 3 explains the problem statement. Section 4 describes the system architecture model and discusses about the design of key generation, encryption, public and private auditing of the proposed model. Section 5 explains the structure of the proposed scalable Bloom Matrix Hash Table (BMHT). The advancement that are made in the proposed LDuAP scheme are discussed

in Sect. 6. The security analysis of the proposed LDuAP scheme is discussed in Sect. 7. The performance evaluation of the proposed LDuAP integrity scheme and BMHT is given in Sect. 8 and Sect. 9 concludes and discusses about the future works.

2 Related works

In this section, the recent research works that have been carried out on integrity verification schemes and the drawbacks of using RSA and BLS signatures for authenticity verification in public auditing are summarized.

Data integrity demands accuracy and completeness of the user data in cloud storage servers. Data owners always expects the data to be stored, managed in a trustworthy environment. Ensuring the integrity remains challenging as the data are stored in a different geological location. Depending upon the working nature of the data integrity schemes, it can be categorized into two difference schemes, such as private and public auditing.

2.1 Private auditing schemes

Private auditing schemes allows the data owner to directly verify the integrity of the data blocks stored in the cloud servers. Private auditing schemes can be further sub classified into two types, such as provable data possession (PDP) and proof of retrievability (POR).

PDP is a probabilistic integrity scheme, which allows the data owner's to directly verify whether the cloud storage server possesses the original data by sampling a set of random data blocks. PDP schemes can effectively identify the corruption or alteration in the cipher text. But, PDP schemes does not support recovery of corrupted data. In POR schemes, at a frequent intervals, the CSP generates a proof to intimate the data owners that their data blocks are safe and recoverable at all time.

Ateniese et al. (2007), proposed the first probabilistic proof of possession scheme by sampling random set of data blocks from the server. It allows the server to access a small portion of the file to generate the proof. To perform authentication, RSA algorithm based homomorphic verifiable tags are used. While uploading the data blocks, data owner precomputes the homomorphic tags for each data block and stores it in the server. Later, the data owner challenges the server to prove the integrity of the data block using the generated homomorphic tags. The server generates the proof of possession for the requested data block and send it back to the data owner. However, in this scheme, the data owners has to stay active at all time, which is not practically possible. Sending frequent challenges to CSP puts an extra workload on the data owners and it increases the communication cost and computation overhead.

Later, Ateniese et al. (2011) proposed a remote PDP data integrity verification scheme based on spot-checking. In this scheme, the server uses a forward error checking (FEC) mechanism to prove the data possession. Initially, the files are encoded with FEC by the data owners and then the encoded files are used by the cloud servers to prove the possession of the data blocks. FEC methods supports the data recovery for minor attacks and detects the major attacks effectively. Subsequently, Wang (2013, 2015), He et al. (2017) and Li et al. (2017) proposed PDP based schemes. However, all these schemes were unable to perform data recoverability.

POR schemes are introduced to support data recoverability. Shacham and Waters (2008), proposed the first POR method to verify the integrity of the user data in the distributed storage systems. It uses BLS signature for short queries and pseudorandom functions for long queries. It uses blockless verification method but fails to prevent data leakage. Later, Bowers et al. (2009), proposed a HAIL (high availability and integrity layer) architecture to maintain integrity of the data. HAIL uses integrity protected error correcting code (IP-ECC) and universal hash functions (UHFs) and pseudo random function for corruption resilient. Recently, Yuhan et al. (2017), proposed an IPOR scheme, which supports both integrity verification and recoverability. But, the signature used in the scheme is too large and result in high computation cost on data owner.

As a summary, both PDP and POR private auditing schemes are accurate in verifying the integrity of the cipher text in the cloud environment. Allowing the data owners to verify the integrity increases the computation and communication overhead on the data owner. Because, the cipher text stored in the cloud storage servers is transferred to the premises of the data owner and decrypted before every private integrity verification. In addition, letting the cipher text to flow between the CSP and data owner for private verification increases the possibility of network attacks.

2.2 Public auditing schemes

Public auditing reduces the communication overhead of the integrity verification by employing a TPA between the data owners and the cloud service provider. Data owner shares metadata information about the data block and ownership signature with the TPA to perform public auditing. Most of the public auditing schemes uses RSA and BLS signatures to verify the authenticity of the data owners.

Yu et al. (2016), proposed an Identity based cloud data integrity checking (ID-CDIC) using an RSA encryption algorithm to support public auditing. It reduced the complexity of the certificate management in the PDP and PoR

and supports the variable sized blocks. The security of the ID-CDIC method highly depends on the RSA assumptions with large public exponents. However, RSA with the key size of 512 bits is already proved to be breakable. Using a larger key size for each data block will increase the computation overhead. In addition, ID-CDIC method fails against recovery attacks and the results of the integrity verification highly depend upon the performance of the Third Party Auditor.

To improve the security against recovery attacks in Yu et al. (2016) and Xu et al. (2017), proposed a modified Identity based integrity verification algorithm using RSA signature. It tightens the security against recovery attacks. But, the computation overhead to generate the signature and verifying the proof becomes higher than the Yu et al. (2016) scheme. Moreover, both schemes are infeasible to perform dynamic and batch auditing. Later, Walid et al. (2019), proposed an RSA based cryptographic accumulator integrity verification scheme to provide security for highly sensitive data. It allows the data owners to perform an unlimited number of integrity check to ensure the authenticity of the data.

BLS (Benoh–Lynn–Shacham) signature is used as an alternative to the RSA signature. Since, the size of the RSA signature is high, (to achieve security level $\lambda = 128$ bits, the required size of the RSA signature should be O $(\lambda)^3$ (i.e.) signature size = 2048 bits) BLS signature scheme was introduced. It reduces the required signature size to 2λ from O $(\lambda)^3$. The required signature size for BLS to achieve security level $\lambda = 128$ bits is 256 bits (twice as the size of the λ). Liu et al. (2015), proposed a public auditing scheme called, MuR-DPA (multi-replica dynamic public auditing). It uses BLS signature and Merkle Hash Tree (MHT) to verify the integrity. Here, MHT is used to store all the replicas of each data block in a same sub-tree. It supports both dynamic and batch auditing schemes. But MuR-DPA uses an additional key server to generate a private key for the users to encrypt the data. Later, Mukundan et al. (2014), proposed a dynamic multi replica provable data possession (DMR-PDP) method to perform integrity verification for replicated data on cloud storages based on BLS signature and homomorphic encryption. It tightens the security of user data stored in the cloud server. But, the time taken to perform homomorphic operations on the encrypted data remains high. Implementing homomorphic encryptions to verify integrity of the data needs a large computation power. Later, Xiling et al. (2018), proposed a privacy preserving EoCo architecture to improve the efficiency of the integrity verification. It reduces the communication overhead between the TPA and CSP. Later, Zhang et al. (2019), proposed a fuzzy auditing protocol to allow multiple cloud users to modify the data while ensuring the data integrity. It uses BLS signature to verify the authenticity of the data owner. The required size of the BLS signature to achieve the 128 bit security level is 2λ (i.e.) 256bits.

Moreover, all the existing public auditing schemes (Yu et al. 2016; Liu et al. 2015; Wang et al. 2011, 2018; Xiling et al. 2018; Zhang et al. 2019, 2020) completely trusts and accepts the integrity results generated by the TPA without any cross-verification. However, if the trustworthiness of the TPA is compromised, then the TPA might send fraudulent integrity results to the data owners. Completely relying and trusting the TPA increases the security vulnerabilities in many aspects.

2.3 False positive errors in index tables and its consequences in public auditing

In public auditing schemes, index table or hash based tree structure is used by the TPA to perform auditing tasks. However, the rate of false positive errors in the index table increases as the number of data blocks grows in the cloud storage. It subsequently reduces the accuracy of the integrity results and increases the trust issues between the data owners and the CSPs.

Aditya et al. (2011), proposed an auditing scheme using Bloom filter. Each block of data stored in the cloud server is verified using its corresponding hash values in the bloom filter. It improves the space efficiency for computation and metadata management. If a single user file has 1000 data blocks and the rate of false positive error in the Bloom filter is 0.001, then the reliability of the verification probability for one file is positive for all blocks. If it is not found, the reliability is only about 90%. In addition, in this scheme, during ownership verification, the data owner has to upload the entire file. It increases the vulnerability of the data. Subsequently, Nianmin et al. (2014), Xiang et al. (2016), Zhang et al. (2017) and Yan et al. (2018), have proposed several variants of bloom filter based integrity verification schemes in the cloud storages. However, the rate of false positive error in the bloom filter is not controlled to work with a large velocity of data (i.e.) big data.

As an alternative to Bloom filter, Zhu et al. (2013), proposed the public auditing scheme using the Index Hash Table (IHT). It uses a probabilistic query and verification to improve the performance of the auditing scheme. IHT uses serial number, block number, version number to maintain the information. IHT helps in achieving maximum accuracy but it also incurs a large computational overhead while performing inserting and deletion in the hash table. Later, Tian et al. (2017), proposed public auditing scheme using Dynamic Hash Table (DHT), which is a two-dimensional data structure located at a TPA to record the data property information. It reduces the computational overhead in Zhu et al. (2013). However, the search operations become inefficient while performing the verification and updation process.

In addition, both the IHT and DHT are exposed to false positive errors.

3 Problem statement

As referred in the related works, the existing auditing schemes face four major issues, such as, (1) allowing the data owners to contact the cloud storage servers directly and perform the private integrity verification which increases the computation and communication overhead on the data owner, (2) employing third party auditor between the cloud service provider and data owners increases the data vulnerability and trust issues. (iii) Existing public auditing scheme uses RSA and BLS signature to perform ownership verification. However, the size of the keys used in these signature schemes is large and results in increased computation overhead. (4) False positive errors in the index table reduces the accuracy of the public auditing.

In this research work, an LDuAP is proposed to overcome the issues associated with the existing auditing schemes. It combines both the public and private auditing methods to verify the integrity of the cipher text stored in the cloud servers. LDuAP scheme uses IND-CCA2 (Indistinguishable Adaptive Chosen Cipher text Attacks) secured Cramer Shoup cryptosystem to perform the auditing task. To reduce the false positive error from the hash table, a two-dimensional scalable Bloom Matrix Hash Table is proposed.

4 System model

The proposed LDuAP scheme creates an efficient framework to verify the integrity of the data stored in the cloud environment. The major entities involved in the proposed scheme are, (1) data owner, (2) third party auditor and (3) cloud service provider. The relationships and communications between the entities are shown in Fig. 1.

Data owners (DO) can be either individuals or a group of peoples who have full rights to access, retrieve and modify the data from the cloud at any point of time. Data owner generates keys, signature and metadata for each data block. The generated keys are used to encrypt the data block using Cramer Shoup cryptosystem. After encrypting, the cipher texts are uploaded to the cloud storage and the metadata information about the data blocks are shared with the Third Party Auditor to perform public auditing. In addition, DO are responsible for cross-verifying the integrity results generated by the TPA or Public auditing.

Third party auditors (TPA) act as a bridge between the data owners and the cloud service provider. TPA manages a scalable Bloom Matrix Hash Table (BMHT) to perform auditing. At a frequent interval, TPA performs two major tasks, such as (1) challenges the CSP to prove the integrity of the data block stored in the cloud storage server and (2) proof verification.

Cloud service providers (CSP) is an untrusted entity of hardware and software resource cluster, which have an

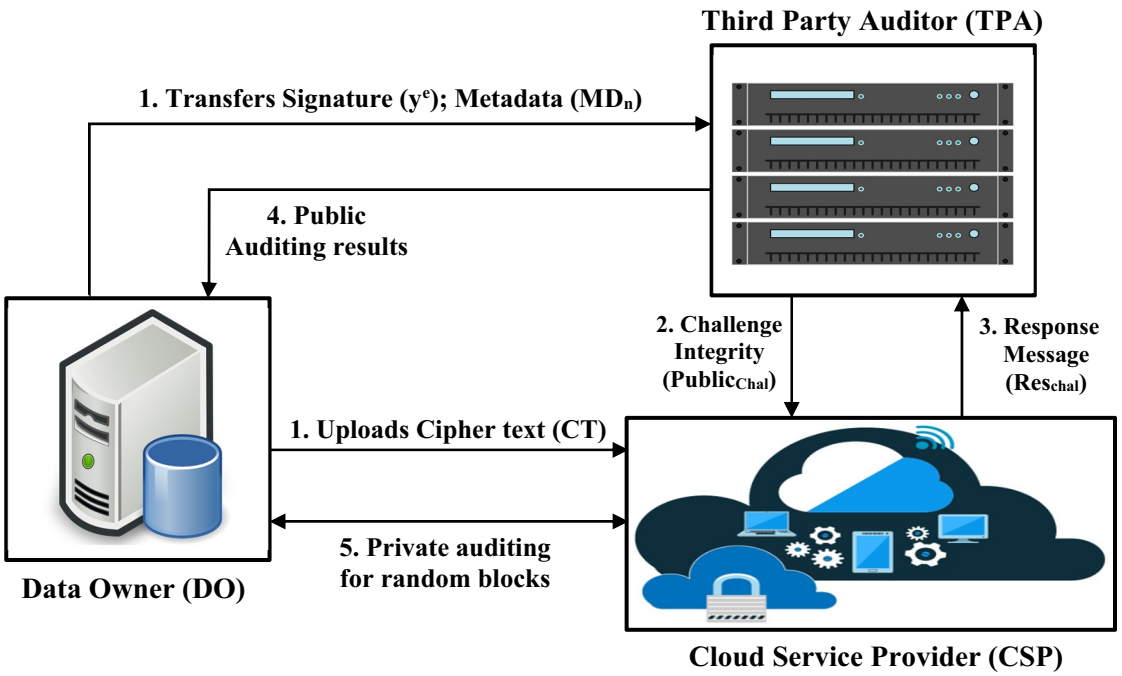

Fig. 1 Overall architecture of the proposed LDuAP scheme

unlimited capacity to store the cipher text uploaded by the data owners. In the proposed integrity verification scheme, cloud service provider performs two major tasks such as, (1) verifies the ownership of the challenged data block. (2) Generates proof and acknowledges the challenge raised by the TPA.

The operations that are performed by the data owners viz., key generation, signature generation, metadata generation and data encryption are executed only once. However, to ensure the integrity of the data block, operations performed at TPA and CSP, such as, integrity challenge, proof generation and proof verifications are performed unlimited number of times.

4.1 Data security

Data owner generates asymmetric keys, signature and metadata for each block of data and encrypts it using Cramer Shoup cryptosystem. Algorithm 1 explains the key generation and encryption of the data block and Algorithm 2 explains the creation of metadata and signature for the data block.

For each data block, data owner creates public as well as the private keys. Here, 'c', 'd' and 'h' are used as public key and 'x_1', 'x_2', 'y_1', 'y_2', 'z' are used as private keys. The calculated values such as, 'U_1', 'U_2'and 'Enc', 'v' are the cipher text of the data block 'R'.

Later, to create the signature for data block 'R', it is hashed using a collision resistance one-way hash function (HF). It produces 'L' bit hash values. A prime number 'e' is chosen from $\left(\frac{(L+1)}{2}\right)$ hash bits and a random string 'β' is chosen from $\left(\frac{L}{2}\right)$ bits hash value. Here, 'e' and β are used to compute the signature for the data block 'R'. Instead of choosing a prime number 'e' from 'L' bits, the proposed method chooses the value of 'e' from $\left(\frac{(L+1)}{2}\right)$ bits, which reduce the size of the signature by 50% without compromising the security. Algorithm 2 explains the signature generation and metadata creation for data block 'R' in detail.

Algorithm 1: Data Owner - Key generation and encryption

Input: Data block 'R'
Output: Cipher text of the data block

Begin

1. Data owner creates two random distinct generators (g_1, g_2), which belong to a cyclic group G of order 'q'.
2. Generates five random variables, such as, x_1, x_2, y_1, y_2, z using Pseudo Random Number Generator (PRNG) from {0, 1, 2… (q-1)}. Here, x_1, x_2, y_1, y_2, z are kept as a secret key.
3. Computes, $c = (g_1)^{x_1}.(g_2)^{x_2}$; $d = (g_1)^{y_1}.(g_2)^{y_2}$; $h = (g_1)^{z}$
4. The values of 'c', 'd' and 'h' are used a public key and x_1, x_2, y_1, y_2, z are used as private keys.
5. Publishes the public key with its descriptors (G, q, g_1, g_2) and keeps the private keys x_1, x_2, y_1, y_2, z.
6. Converts the data block 'R' in to an element of G. Such that, $R \in G$
7. Choses a random variable '*k*' from {0, 1, 2… (q-1)} to derive the cipher text for the data block.
8. Calculates, $U_1 = (g_1)^{k;}$ $U_2 = (g_2)^{k}$ and $Enc = (h)^{k}$. R
9. Compute, $v = (c)^{k}.(d)^{k\,\alpha}$; Here, 'v' is used to verify the integrity in direct verification (i.e.) private auditing.
10. Uploads the cipher text (U_1, U_2 and Enc) of the data block 'R' to the cloud storage.

End

Algorithm 2: Data Owner - Signature and metadata creation

Input: Data block 'R'; Cipher text U_1, U_2, and Enc
Output: Signature and metadata of the data block 'R'

Begin

Begin - Signature generation

1. Choose two prime numbers, 'p' and 'q', where, p = 2a + 1 and q = 2b + 1.
2. Calculate n = p * q and choose four quadratic residues such as, h_1, h_2, h_3 and $x \in QR_n$
3. Here, n, h_1, h_2, h_3 and x are denoted as public key and p, q are private key for signature generation.
4. Hash the input data 'R' and produce 'L' bit hash value. (i.e.) HF(R):= L bits hash value.
5. Split the 'L' bit hash values into two halves, such that, h_1 = L/2 bits and h_2 = L/2 bits
6. Choose a random prime 'e' from $\left(\frac{(L+1)}{2}\right)$ bit and random string 'β' from $\left(\frac{L}{2}\right)$ bit hash value.
7. Compute signature, $y^e = (x * h_1^{\beta} * h_2^{\beta \oplus H1(R)} * h_3^{\beta \oplus H2(R)}) \bmod n$; Computing the e^{th} root of 'y' is easy, as it the factorization of 'n'.
8. Sends the signature (y^e) to the TPA.

End - Signature generation

Begin – Metadata creation

1. The cipher text Enc, U_1 and U_2 are given as input to 'n' hash functions $\alpha_n = HF_n(U_1, U_2, Enc)$; Here, HF represents hash function such as MD5, SHA-1 and SHA-2.
2. As a result, it produces 'n' number of hash values (α_n).
3. The results of the hash values (α_n) are stored in data owners.
4. For each data block 'R', the metadata MD_n is created. Metadata consists of message digest (α_n), Time (T), Block id (B_{id}) and user id (U_{id}). (i.e.) MD_n = {α_n, T, B_{id} and U_{id}}
5. Sends the metadata to the Third Party Auditor (TPA).

End – Metadata creation

End

After the completing the data encryption, signature generation and metadata creation, the cipher text (U_1, U_2, Enc and v) of data block 'R' is sent to the cloud storage server and the corresponding metadata $MD_n = \{\alpha_n, T, B_{id}$ and $U_{id}\}$ and signature ($y^e_{)}$ is sent to the Third Party Auditor. In metadata, α_n represents the hash value of the data block, 'T' represent the time of data encryption, B_{id} represents the data block identification number and U_{id} represents the user identification number.

4.2 Dynamic public auditing

Upon receiving the metadata $MD_n = \{\alpha_n, T, B_{id}$ and $U_{id}\}$ and signature (y^e) from the data owners, Third Party Auditor (TPA) calculates the corresponding hash bits and stores it in the Bloom Matrix Hash Table. Hash value (α_n) in the metadata may consist of two or more message digests. Consider, if the data owners chooses three hash function (n = 3), then it will produce three unidentical message digest of different length for single data block 'R'. From these message digest, TPA will derive the hash bits and stores it in the Bloom Matrix Hash Table. The values in the BMHT will either be

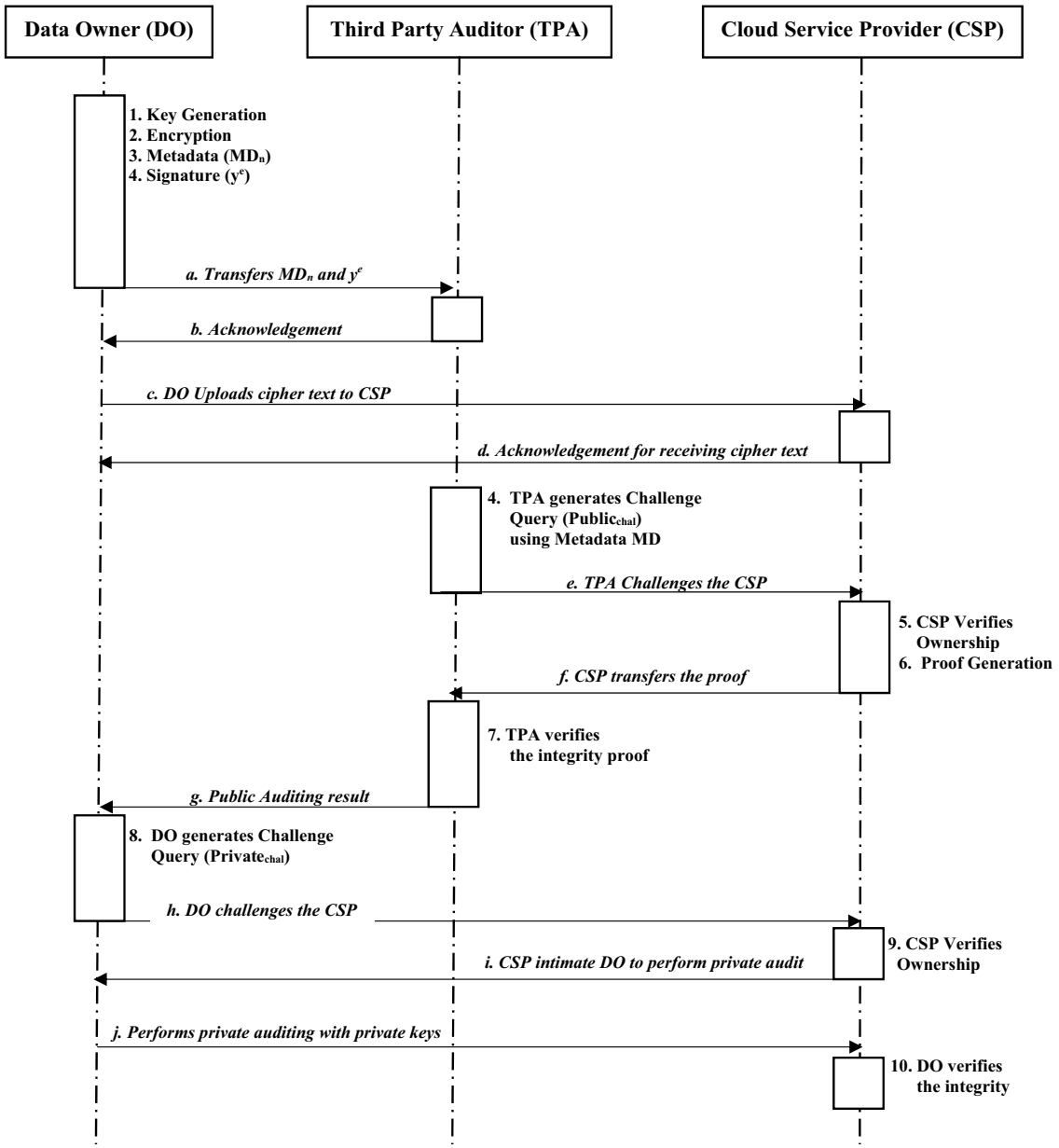

Fig. 2 Workflow of the proposed LDuAP scheme

0 or 1. If the value in BHT is 0, then it means that no hash values has accommodated in that particular location. If the bit value is 1, then it means that a hash value has accommodated location. The structure and scalability feature of the proposed Bloom Matrix Hash Table is explained in detail in Sect. 5.

At a frequent interval, TPA challenges (Chal) the cloud service provider to prove the integrity of the data blocks stored in the cloud storage. (CSP). Algorithm 3, explains the proposed dynamic public auditing method. Here, timestamp (Ts) in Challenge query is used to maintain the sequential order and U_{id} is used for user identification. The workflow of the proposed LDuAP scheme is shown in Fig. 2.

Cloud service provider performs two major tasks such as, (1) verifies the ownership of the challenged data block and (2) generates proof and acknowledges the challenge raised by the TPA.

Verifying the data owner's signature is important in cloud storage to maintain the legitimacy. It protects the cloud storage from unauthorized access. After verifying the signature of the data owner, acknowledging the challenge helps the TPA to ensure the integrity. Algorithm 4, explains the signature verification and integrity proof generation in detail.

Algorithm 3: Third Party Auditor – Dynamic public auditing

Input: $MD_n = \{\alpha_n, T, B_{id} \text{ and } U_{id}\}$
Output: Result of public auditing

Begin

1. TPA receives $MD_n = \{\alpha_n, T, B_{id} \text{ and } U_{id}\}$ from the data owner and creates Bloom Matrix Hash Table (BMHT).
2. At frequent interval, TPA creates a challenge query ($Chal_{public\text{-}id}$) with Time stamp and metadata information for each block of data (i.e.) Chal $_{public\text{-}id} = \{ T_s, B_{id} \text{ and } U_{id,} \}$
3. Send the challenge query Chal $_{public\text{-}id} = \{T_s, B_{id} \text{ and } U_{id}\}$ to the CSP.
4. TPA receives the hash values (α_n) of the challenged data block from the CSP and computes the corresponding hash bits and perform proof verification for the challenged data blocks.
5. Proof verification
   If (hash bits received from Client = = hash bits received from CSP)
   then
   Proof is valid. No data is tampered or changed
   Send the result to the data owner.
   else
   Data blocks are tampered.
   Intimate the data owner and suggest to conduct private integrity verification.

End

Algorithm 4: Cloud Service Provider – Signature verification and proof generation

Input: Ciphertext: (U_1, U_2, Enc and v) and signature of the data owner
Output: Result of signature verification and integrity proof verification

Begin

Begin – Signature verification

1. CSP receives the signature y^e from the third party auditor.
2. Checks whether the value of prime number 'e' is odd and derived from $\left(\frac{(L+1)}{2}\right)$ bits.
3. Checks whether the β' is $\left(\frac{L}{2}\right)$ bit long.
4. Computes $y^e = (x * h_1^{\beta} * h_2^{\beta \oplus H1(R)} * h_3^{\beta \oplus H2(R)}) \bmod n$; where $n = (p * q)$ Here, h_1, h_2, h_3, x are public keys and 'p', 'q' are private keys for signature verification.
5. Signature verification
   If (both the values of y^e same),
   then
   Signature of the sender for sensor value R is legitimate
   else
   Confirm as forged sender and reject cloud storage service

End – Signature verification

Start – Proof generation

1. Receive challenge query Chal $_{public\text{-}id} = \{T_s, B_{id} \text{ and } U_{id}\}$ from the TPA
2. Identify the cipher text of the corresponding request data block from the B_{id}
3. Derive 'n' hash value for the requested data block (i.e.) $\alpha_n = HF_n (U_1, U_2, Enc)$.
4. Send α_n hash value to the TPA along with its Timestamp(T_s)

End – Proof generation

End

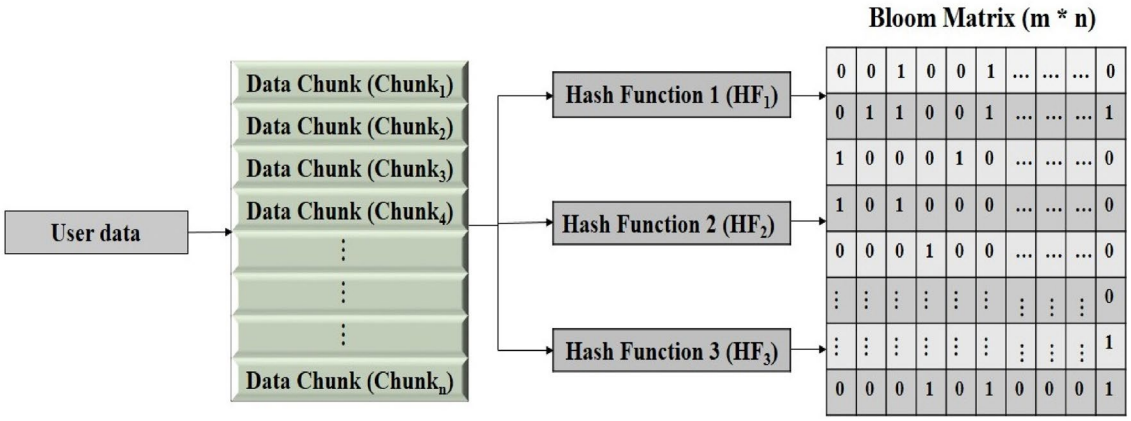

Fig. 3 Creation of Bloom Matrix Hash Table

The proposed public auditing is efficient and secure. However, if the trustworthiness of the TPA is compromised, then the TPA might hide the original auditing results generated by the proposed scheme and may send a fraudulent result to the data owners continuously. To overcome this issue, a cross verification mechanism is introduced using private or direct auditing method. The proposed LDuAP scheme allows the data owner to cross-verify the integrity results produced by the Third Party Auditor.

In existing private auditing schemes, the integrity of the data block can be verified only after decrypting the cipher text. It is a time consuming task in both computational and communication aspects. However, the proposed scheme allows the data owners to verify the integrity without decrypting the entire file. Integrity of the data block can be directly verified using the following equation,

$$\left(U_1\right)^{x_{1+}\, y_{1\alpha}} + \left(U_2\right)^{x_{2+}\, y_{2\alpha}} = v. \tag{1}$$

The process of verifying the integrity using direct verification is explained in Algorithm 5.

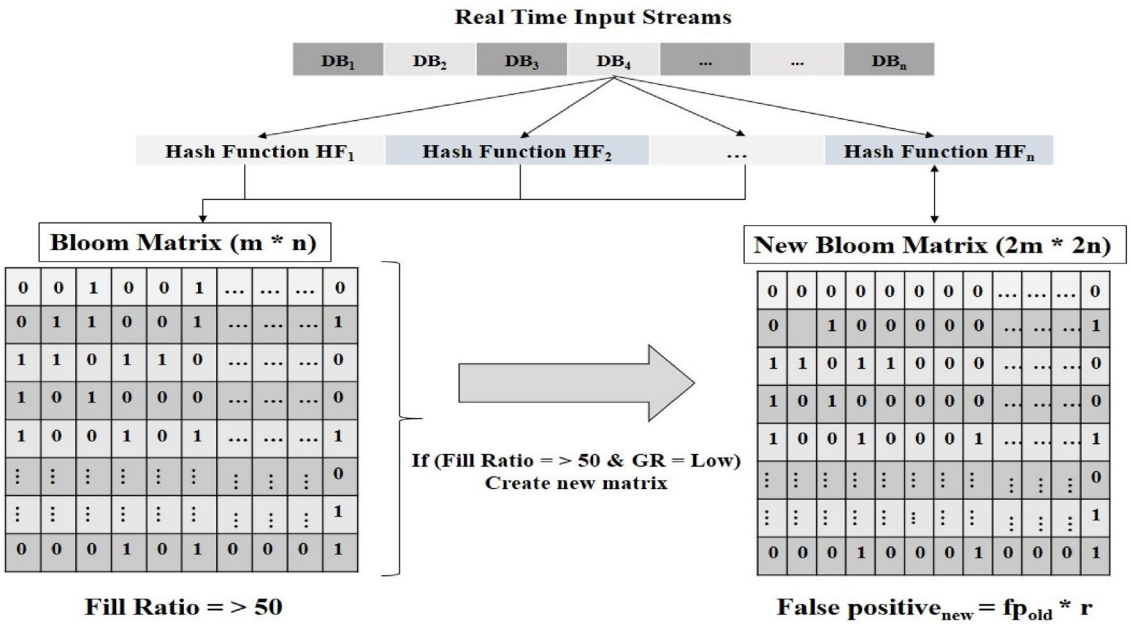

Fig. 4 Scalability on Bloom Matrix Hash Table

Algorithm 5: Data owner – Private integrity verification

Input: Random data blocks

Output: Results of the private integrity verification

Begin

1. Data owner receives the tamper alert from TPA or chooses a random data block.
2. Creates a private challenge query ($Chal_{private\text{-}id}$) with time stamp (T_s) along with metadata information. (i.e.) $Chal_{private\text{-}id} = \{T_s, B_{id}$ and $U_{id}\}$
3. CSP verifies the ownership of the request data blocks.

   If (Ownership == verified)

   then

   a. CSP allows the data owner to access the data block.

   b. Data owner verifies the integrity of the data block by, $(U_1)^{x1+y1\alpha} + (U_2)^{x2+y2\alpha} = v$; Here, x_1, x_2, y_1 and y_2 are private keys (only known to the authorized persons)

   If (values are equal)

   then

   File is safe. No tampering happened.

   else

   Tampering is confirmed. Contact CSP immediately.

   else (Ownership == not verified)

   a. CSP denies access.

End

Introducing private auditing in the LDuAP scheme slightly increases the computation overhead of the data owners. However, the IND-CCA2 secured Cramer Shoup cryptosystem allows the data owners to perform private auditing with decrypting the data block. In addition it also reduce the trust issues associated with the TPA.

5 Structure of the proposed scalable Bloom matrix Hash Table (BMHT)

Bloom filter is a space efficient probabilistic data structure which is used to test whether an element is a member of a set. Existing standard Bloom filter uses a one-dimensional array data structure to map the elements to the bits arrays. However, standard Bloom filters are not scalable. The size of the array used in the Bloom filter has to be predetermined. As far as the cloud is concerned, predetermining the number of data blocks to be stored in the cloud storage server is impractical.

Using standard non-scalable bloom filter in the TPA increases the bottleneck while addressing the high velocity of input data. It drastically reduces the processing speed of the TPA and results in increased false positive error rates.

To overcome the issues in the standard Bloom filter, the proposed LDuAP auditing scheme creates a two dimensional BMHT. It helps the TPA to perform pubic auditing effectively.

Instead of using array data structure, the proposed BMHT uses a two dimensional scalable zero-matrix with the size of $m * n$, whereas *'m'* and *'n'* are always same. The initial size 'm' of the BMHT is calculated using the following formula,

$$m = \text{Sqrt}\left(\frac{-\,Total\ number\ of\ data\ chunks \times \ln(False\ positive\ rate)}{\ln(2)^2}\right). \tag{2}$$

Table 1 Storage requirement of Bloom Matrix Hash Table

Total number of data blocks	Input data size	Size of the BMHT (m×n)	Storage requirement in the TPA
1024	1 GB	99×99	20 KB
2048	2 GB	140×140	39 KB
3072	3 GB	171×171	58 KB
4096	4 GB	198×198	77 KB
5120	5 GB	221×221	96 KB
10,240	10 GB	313×313	215 KB
102,400	100 GB	990×990	1.9 MB
204,800	200 GB	1400×1400	4.254 MB
512,000	500 GB	2215×2215	10.065 MB
1,024,000	1000 GB	3132×3132	19.163 MB
2,048,000	2000 GB	4430×4430	38.594 MB

Table 2 Probability of False Positive error in BMHT

Total number of data blocks	Number of hash function used (k)	Total bits in BMHT (m)	Probability of false positive
1024	3	9801	0.003319
2048	3	19,600	0.003319
3072	3	29,241	0.002969
4096	3	39,204	0.003319
5120	3	48,841	0.003058
10,240	3	97,969	0.00319
102,400	3	980,100	0.00321
204,800	3	1,960,000	0.00325
512,000	3	4,906,225	0.00312
1,024,000	3	9,809,424	0.00317
2,048,000	3	19,624,900	0.00319

Table 3 Probability of false positive error of newly created BMHT

Total number of data blocks	Total bits in previous BMHT matrix (m_{prev})	Prob. of false positive (fp_{prev}) (previous matrix)	Total bits in newly created matrix (m_{new})		Prob. of false positive fp_{new} (New matrix) $fp_{new} = fp_{prev} \times r$	
			Min (GR)	Max (GR)	Min (r = 0.8)	Max (r = 0.9)
1024	9801	0.003319	19,602	39,204	2.6×10^{-3}	2.9×10^{-3}
2048	19,600	0.003319	39,200	78,400	2.6×10^{-3}	2.9×10^{-3}
3072	29,241	0.002969	58,482	116,964	2.3×10^{-3}	2.6×10^{-3}
4096	39,204	0.003319	78,408	156,816	2.6×10^{-3}	2.9×10^{-3}
5120	48,841	0.003058	97,682	195,364	2.4×10^{-3}	2.7×10^{-3}
10,240	97,969	0.00319	195,938	391,876	2.5×10^{-3}	2.8×10^{-3}
102,400	980,100	0.00321	1,960,200	3,920,400	2.5×10^{-3}	2.8×10^{-3}
204,800	1,960,000	0.00325	3,920,000	7,840,000	2.6×10^{-3}	2.9×10^{-3}
512,000	4,906,225	0.00312	9,812,450	19,624,900	2.5×10^{-3}	2.8×10^{-3}
1,024,000	9,809,424	0.00317	19,618,848	39,237,693	2.5×10^{-3}	2.8×10^{-3}
2,048,000	19,624,900	0.00319	39,249,800	78,499,600	2.5×10^{-3}	2.8×10^{-3}

While generating the metadata, the data owner hashes the data block 'R' using 'n' number of one-way hash function. The hash functions in the proposed scheme are MD5, SHA1 and SHA2. The creation of the Bloom Matrix Hash Table in TPA is shown in Fig. 3.

Upon receiving the metadata (α_n) from the data owner, TPA calculates the hash bits and stores it in the corresponding bit array in BMHT. The values in the BMHT will either be 0 or 1. If the value in BHT is 0, then it means that no hash values has accommodated in that particular location. If the bit value is 1, then it means that a hash value has accommodated location.

Scalability on BMHT To reduce the false positive errors in the public auditing, the proposed BMHT is designed in a manner to increases its size from 'm' to '2*m'. Third Party Auditor monitors the remaining free slots (i.e.) the number of bit arrays with 0 values in BMHT continuously. If the free slots in BMHT reaches a certain threshold limit, then a new and larger BMHT is created. If the growth rate of the input data is low, then the newly created BMHT will be two times bigger than the old BMHT and four times bigger if the growth rate is high. Since, the new matrix have more slots, the false positive rate of new matrix is always lesser than the old matrix.

When the filter reaches a certain fill ratio threshold value, a new matrix is created with a tightened error probability. The tightening ratio '*r*' controls the growth of the new matrixes. The tightening ratio is always maintained between 0.8 and 0.9. Regarding to the fullness criteria of a BMHT matrix, the maximum number of items stored in the given matrix for fill ratio 50% can be found by the following formula, $n = m \times \frac{\ln 2^2}{|\ln P|}$ where '*m*' is the number of bits and *P* is the probability of the false positive error. Algorithm 6, explains the creation and scalability of BMHT. Figure 4, diagrammatically represents the scalability feature of the proposed BMHT.

Algorithm 6: Third Party Auditor – Creation and Scalability of BMHT

Input: Growth rate, BMHT size, false positive of rate and number of hash function used
Output: Creating of new BMHT with increased size

Begin

1. Find fill ratio of previous BMHT matrix, $n = m * \frac{ln2^2}{|\ln P|}$
2. If (fill ratio > = 50)

Then

If (growth rate == High)

Then

Create BMHT new matrix with a size of 200% of m*n
Insert stream bits to the new matrix

Else

Create BMHT new matrix with a size of 100% of m*n
Insert stream bits to the new matrix

Else

Return null and do not create a new matrix

End

5.1 Hash collision

The important issues that has to be considered while creating an index table is hash collision and false positive errors. If the hash function maps two or more different elements (i.e.) data blocks to the same bit array in the hash table, then the collision is said to occur.

To avoid hash collision and pigeonhole problem, the size of the index table should be always greater than the total number of data blocks stored in the cloud. Let us consider, at time t', the data owner wants to upload a dataset with the size of 211 GB to the cloud storage. Initially, to reduce the computational complexity, the input dataset will be chunked into multiple small sized data blocks. Here, the given dataset is chunked into 2,16,553 data blocks each of 1 MB size. The required size of the BMHT to store the hash bits of 2,16,553 input data blocks is calculated from the Eq. (2),

Table 4 Case processing summary

	Cases					
	Valid		Missing		Total	
	Number of test	Percent (%)	Number of test	Percent (%)	Number of test	Percent (%)
Test result reality	750	100	0	0	750	100

Table 5 Cross-Tabulation of the proposed BMHT table

			Reality	
			Positive	Negative
Test result	Positive	Count % with reality	746 (True Positive) 99.47%	True negative and False negative are not permitted in the proposed Bloom Matrix Hash Table
	Negative	Count % with reality	8 (False Positive) 0.53%	
Total		Count % with reality	750 (Total Result) 100%	

$$m = \text{Sqrt}\left(\frac{-216553 \times \ln\ (0.01)}{\ln(2)^2}\right) = Sqrt(2,075,686)\ bits = 1440.$$

So, the required size of BMHT is 1440×1440. Table 1, represents the BMHT matrix size and storage requirement of BMHT for different input data size.

In the proposed LDuAP scheme, data owners uses three hash functions such as, MD5, SHA-1 and SHA-2 to hash the input data block 'R'. MD5 hash function produces a 128bit message digest and SHA-1 and SHA-2 produces 160 bit message digest. The probability of collision of these hash function can be calculated as follows, $(Collision\ of\ MD5) = \frac{1}{2^{128}}$; $P(Collision\ of\ \ SHA1) = \frac{1}{2^{160}}$; $P(Collision\ of\ SHA2) = \frac{1}{2^{160}}$. So the Probability of total collision is, $P\ (Collision) = \left(\frac{1}{2^{128}} \times \frac{1}{2^{160}} \times \frac{1}{2^{160}}\right) = 1.375 \times 10^{-137}$; which is very low and considerably negligible. Hence, the probability of collision for two independent input data block is negligible in the proposed BMHT.

5.2 Probability of false positive errors in proposed BMHT

If an element refers to the same bit array as that of another element in the BMHT, then the false positive error is said to occur. The probability of false positive is calculated from,

$$P(false\ Positive) = \left(1 - e^{\frac{k*n}{m}}\right)^k. \tag{3}$$

The probability of the false positive of the proposed BMHT is derived for various input size. Collision in BMHT plays a major role in increasing the probability of the false positive errors. Let's consider that the BMHT has a '*Cf*' number of collision, then the probability of false positive, $P\ (false\ positive) = (1 - Cf)$ and the expected amount of false positive due to collision can be calculated by,

$$E(false\ positive)\ =\ \sum_{i=0}^{n-1} Cf\ (i)\ . \tag{4}$$

where '*i*' is the factor of collision. From the above-mentioned formula, it is clear that if collision increases in BMHT then there will be an increase in false positive rate. As the proposed model uses independent hashing techniques with the negligible collision factor, it produces a very low false positive error. Table 2, represent the probability of false positive in BMHT.

The acceptable false positive rate for an index table is 0.01. However, the probability of false positive errors in the proposed scalable BMHT ranges between 0.002 and 0.003, which is significantly low compared to the standard bloom filter. Table 3, shows the probability of the false positive errors in the newly created BMHT.

The probability of false positive error in the newly created BMHT is always lesser than the older BMHT. So, sudden increase in the velocity of input data will not affect the performance of the TPA.

6 Advantages of the proposed LDuAP scheme

In this section, the advantage of using Cramer Shoup cryptosystem, BMHT and dual auditing in the proposed LDuAP scheme are discussed in detail.

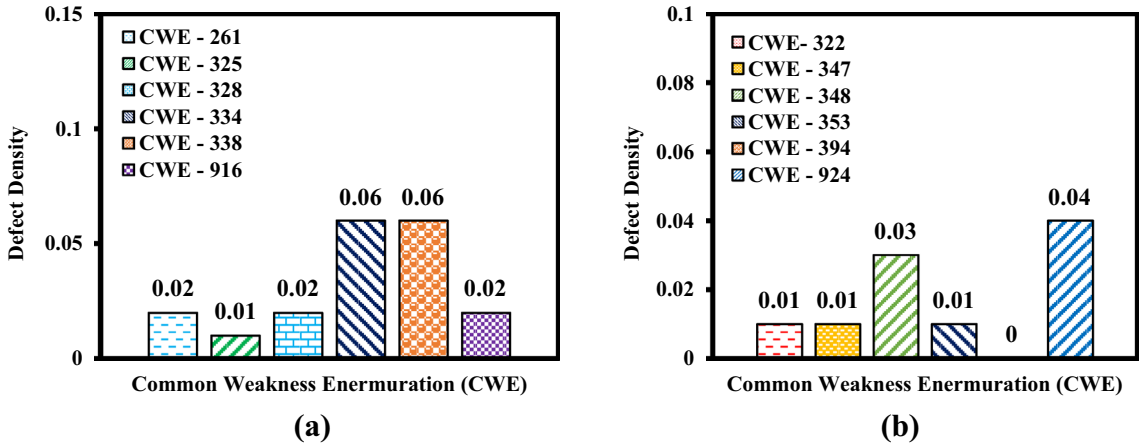

Fig. 5 **a** Vulnerability assessment of the proposed cryptographic model in LDuAP scheme. **b** Vulnerability assessment of the data integrity process in the proposed LDuAP scheme

6.1 Why Cramer-Shoup cryptosystem is preferred over other encryption algorithm in the proposed LDuAP scheme?

Popularly known asymmetric encryption algorithm such as, RSA, Paillier and Elgamal are malleable (i.e.) it is possible for an adversary to transform a cipher text into another cipher text which is decrypted to a related plaintext. Cramer Shoup cryptosystem is an extension of Elgamal algorithm is proven to be secure against indistinguishable adaptive chosen cipher text attack (IND-CCA2) and it ensures non-malleability even against a resourceful attacker. To ensure the non-malleability of the cipher text stored in the cloud storage, the Cramer-Shoup cryptosystem performs direct integrity verification by equating $(U_1)^{x1+y1\alpha} + (U_2)^{x2+y2\alpha}$ with 'v" before every decryption. The proposed LDuAP scheme has incorporated the non-malleability feature of the Cramer Shoup cryptosystem to perform direct integrity verification.

6.2 How does the Bloom Matrix Hash Table (BMHT) improves the performance of public auditing?

Existing standard Bloom filter uses a non-scalable one-dimensional array data structure to map the elements. Since, it is non-scalable, size of the bloom filter has to be predetermined based on the total number of input data blocks. However, in a cloud environment, calculating the number of real-time input data blocks is impossible. Usage of standard non-scalable bloom filter in the TPA increases the bottleneck in cloud environment. It may drastically reduce the performance of the TPA and results in increased false positive error rates.

To overcome these issues, a scalable two dimensional BMHT has been proposed in the LDuAP scheme. It effectively reduces the hash collision and false positive errors in the TPA. It helps in increasing the trust between the data owners and third party auditor. The rate of false positive errors in the proposed BMHT is shown in Tables 2 and 3.

In addition, the false positive rate of the BMHT is tested with IBM SPSS testing tool. To store 1 GB of user data in the cloud storage server, a two-dimension matrix with the size of 99×99 is created by the TPA (i.e.) it can store up to 9801 bits. The user data is chunked into 1024 data blocks (each consist of 1024 KB). The data blocks are encrypted using Cramer Shoup cryptosystem and uploaded to the cloud storage. Later, the public auditing is performed by the TPA to ensure the integrity of the data blocks. From 1024 data blocks, 750 public integrity results were collected, in which 742 results matched the actual value and remaining 4 results befallen into false positive category. Table 4 and Table 5 represents the results derived from the IBM SPSS testing tool.

The IBM SPSS analyzed results shows that the false positive rate of the proposed Bloom Matrix Hash Table is very less. In addition, since the proposed BMHT supports scalability, the false positive rate can be reduced to greater extent.

6.3 Importance of dual auditing scheme

In public auditing schemes, the proof verification or integrity verification is performed only on the premises of the TPA. It maintains a Bloom Matrix Hash Table (BMHT) to map the elements received from the data owners to the corresponding bit arrays. At any time 't', the integrity of the data blocks stored in the cloud is verified only if the hash bit received from the data owner and CSP are same. Subsequently, the public integrity result is communicated to the corresponding data owners. If the trustworthiness of the TPA is compromised, then the TPA may send a fraudulent public integrity to the data owners continuously. Existing public auditing schemes (Yu et al. 2016; Liu et al. 2015; Wang et al. 2011,

Table 6 Required size of the signature to achieve 128 bit security

Schemes	Size of the signature (λ)	Required size of the signature to achieve 128 bit security level
RSA	O (λ) ^{3}	2048 bits
ECDSA	4λ	512 bits
Schnorr	3λ	384 bits
BLS	2λ	256 bits
Proposed CS signature	λ	128 bits

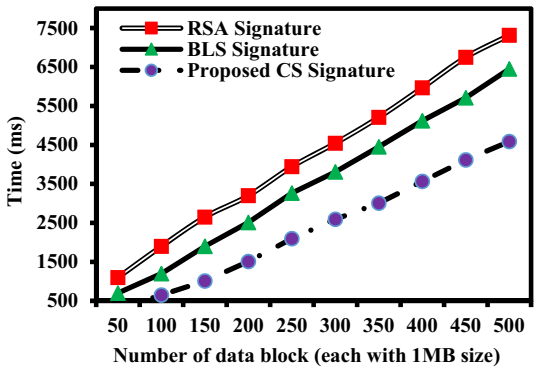

Fig. 6 Time taken to generate signatures for 500 data blocks

2018; Xiling et al. 2018; Zhang et al. 2019, 2020) does not allow the data owners to perform cross verification. However, to reduce the trust issues between the data owners and the TPA, the proposed LDuAP scheme has introduced a dual auditing scheme. It allows the data owners to perform cross verification to ensure the correctness of the public auditing.

7 Analysis of the proposed LDuAP scheme

The primary goals of the proposed LDuAP auditing scheme are (a) privacy protection and data security in cloud storages. (b) Cross verification of the integrity results generated by the TPA and (c) signature size reduction or lightweight auditing.

7.1 Privacy protection and vulnerability assessment

Privacy protection and data security are the two important features that has to be considered while creating an auditing scheme. While considering the privacy of the data owners in cloud environment, the identity of the data owners has to be protected from the internal and external adversarial attacks. In the proposed work, before uploading the data blocks to the cloud storage, data owner generates a signature (y^e) to prove the ownership of the data block. Later, the cipher text and the signature of the data blocks are sent to the Third Party Auditor along with the metadata. The signature of the data owner is generated using,

$$y^e = \left(x \times h_1^{\beta} \times h_2^{\beta \oplus H_1(R)} \times h_3^{\beta \oplus H_2(R)}\right) mod\ n. \quad (5)$$

While creating the signature of the data owner, the data block is hashed using anti-collision one-way hash function 'HF', makes impossible for an adversary to acquire the information about the data owners from the signature. Similarly, to secure the data blocks in the cloud storage, LDuAP uses an asymmetric Cramer Shoup cryptosystem, which is proven to be secure against IND-CCA2 (indistinguishable adaptive chosen cipher text attacks) under the assumption of Decisional Diffie-Hellman (DDC) algorithm. The proposed LDuAP auditing scheme effectively protects the data owner's privacy from internal and external adversarial attacks. In addition, the proposed LDuAP scheme is tested using a Coverity–SAST (Static Application Security Testing) tool in the Polaris software integrity platform. The common weakness enumeration (CWE) of the proposed LDuAP scheme is derived and shown in the Fig. 5.

To assess the vulnerability of the proposed cryptographic model, the defect density of weak encoding for password (CWE-261), missing cryptographic step (CWE-325), reversible one-way Hash (CWE-328), small space of random values (CWE-334), use of cryptographically weak pseudo-random number generator (CWE-338), use of password hash with insufficient computational effort (CWE-916) are derived using the Coverity- SAST security testing tool.

Likewise, to assess the proposed dual auditing scheme, the defect density of key exchange without entity

Table 7 Computation overhead to perform dual integrity verification

Schemes	Wang et al. (2011)	Yu et al. (2016)	Proposed scheme
Client (signature or tag generation)	$nHash_{G1} + nMul_{G1} + nExp_{zq} + nExp_{G1}$	$3mul + nInv + 2Hash + 4Exp +$	$3Mul_{QR} + Hash_R + 3Exp_{QR} + Mod_{sign}$
TPA (challenge and Proof verification)	$nMulExp_{G1} + nHash_{G1} + nHash_{zq} + 3Exp_{G1} + 2Pair_{G1,G2} + Mul_{G1} + Mul_{zq}$	$nMul + nInv + nhash + 2exp + chas h + (c+2)Exp$	*nInsert + nCompare (each with O(k) time complexity) and O(1) for creation of challenge query*
CSP (proof generation)	$nMulExp_{G1} + nExp_{G2} + nHash_{zq} + nAdd_{zq} + (n+1)Mul_{zq}$	$c{\cdot}Add + 2cMul + 2Hash + cExp$	$3Mul_{QR} + Hash_R + 3Exp_{QR} + Mod_{sign} + nHash_{ct}$

authentication (CWE-322), improper verification of cryptographic signature (CWE-347), use of less trusted source (CWE-348), missing support for integrity check (CWE-353), download of code without integrity check (CWE-494), improper enforcement of message integrity during transmission in a communication channel (CWE-924) are derived.

7.2 Cross verification

As per the discussion made in Sect. 6.3, the proposed public auditing scheme does not assure the correctness of the integrity verification at all time. Hence, a direct verification method is proposed to ensure the correctness of the integrity method. For some random data blocks, data owner generates a private challenge query $Chal_{private}$ and sends to the Cloud Service Provider. After receiving the $Chal_{private,}$ CSP verifies the ownership of the data block and allows the data owner to perform direct verification as shown in Algorithm 5. Since, the integrity is directly verified, the correctness of the proposed auditing scheme is assured and the trust issues associated with the TPA is reduced. However, it slightly increases the computation and communication overhead on the data owner's side.

7.3 Lightweight auditing

To measure the computation overhead of the proposed LDuAP auditing schemes, four important factors are considered such as: (i) size of the signature. (ii) Computation time to generate signature and metadata. (iii) Computation time to challenge CSP and verify the integrity proof. (iv) Computation time to generate integrity proof.

(i) *Size of the signature* In accordance with the signature generation process explained in Algorithm 2, each data block 'R' is hashed using a collision resistance hash function 'HF' and produces 'L' bit hash value. These 'L' bit hash value is split into two equal halves, $H_1(R)$ and H_2 (R) each with L/2 bits. Later, the value of 'e' is chosen from the prime number of $\left(\frac{(L+1)}{2}\right)$ bit and a random string 'β' is chosen from $\left(\frac{L}{2}\right)$ bit hash values to compute the signature. The splitting of hash values into two halves reduces the value of 'e' and subsequently reduces the size of the signature. Choosing a prime number either from $\left(\frac{(L+1)}{2}\right)$ or $\left(\frac{L}{2}\right)$ bits, does not make any major changes in aspects of security of the signature.

Let us consider, if the hash values are not split into two halves, then the value of 'e' should be chosen from a prime number of (L + 1) bits values (i.e.) The value of 'e' should be chosen from any of these 161bit prime numbers 159, 399, 493, 685, 709, 765, 973, 1011, 1099, 1263. Choosing a larger prime number for 'e' will put an extra load on the Data owner. To reduce it, we have split the hash values into two halves and chose a prime number from $\left(\frac{(L+1)}{2}\right)$ bit. (i.e.) 81 bits. So that, the value of 'e' can be chosen from any of the 81 bit prime numbers, such as, 51, 63, 163, 205, 333, 349, 429, 433, 481, 553. So using a lesser prime value for 'e' in Eq. (2) produces a short signature. To be precise, the size of the signature is reduced by 50% compared to the existing schemes. It subsequently reduces the computation overhead on the data owner's side. In addition, the size of the signature required to achieve 128 bit security is very less in the proposed LDuAP scheme compared to the existing scheme. Table 6 compares the required size of the signature in various scheme to achieve 128bit security. Likewise, the signature size and time taken to generate signature is compared with existing RSA and BLS methods in Fig. 6.

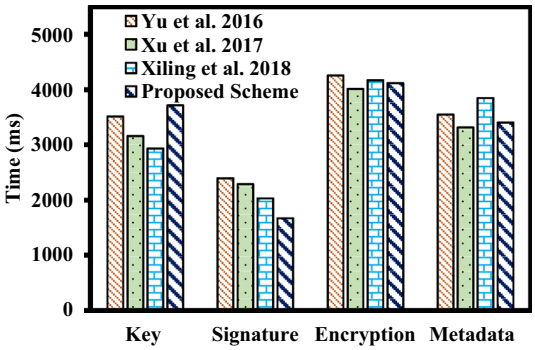

Fig. 7 Time taken on data owner to execute 750 data blocks

(ii) *Computation time to generate signature and metadata* The time taken to create the signature for a single data block is, $3Mul_{QR} + Hash_{R} + 3Exp_{QR} + Mod_{sign.}$ Here, $3Mul_{QR}$ represents the number of multiplicative operations used to create the signature. The quadratic residues such as, x, $h_{1,}$ $h_{2,}$ h_3 are multiplied with each other. Likewise, one collision resistance hash function (HF), three exponential operation and one modulo 'n' operation is used by the data owner to create the signature.

On the other hand, to generate metadata, the proposed LDuAP scheme uses, 'n' number of hash function to hash the cipher texts (i.e.) U_1, U_2 and Enc. Here, the value of 'n' is decided based on the total number of data blocks and the size of the BMHT. So, the total computation time required to create metadata remains as, $nHash_{CT}$.

(iii) *Computation time to challenge CSP and proof verification* To perform public auditing, Third Party Auditor challenges the Cloud Service Provider to prove the integrity of the data block with a timestamp (T_s). Later, to perform integrity proof verification, the TPA uses a space efficient probabilistic BMHT. The proof verification is performed by comparing hash bits received from the CSP and the hash bits received from the data owner. Challenging the CSP does

Table 8 Time taken by the data owner to execute a single data block

Schemes	Key generation (ms)	Signature (ms)	Encryption (ms)	Metadata creation (ms)	Private auditing (ms)
Yu et al. (2016)	3.516	2.311	4.212	3.501	Not supported
Xu et al. (2017)	3.125	2.279	4.453	3.385	Not supported
Xiling et al. (2018)	2.474	2.063	4.176	3.868	Not supported
Proposed scheme	3.89	1.666	4.112	3.446	5.604

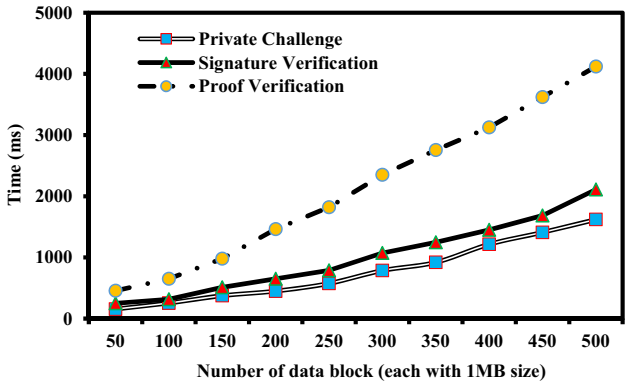

Fig. 8 Computation overhead to perform private auditing for 500 data blocks

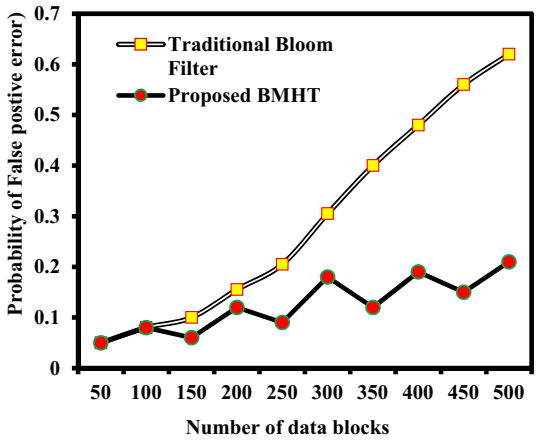

Fig. 10 Comparison of False positive errors in different schemes

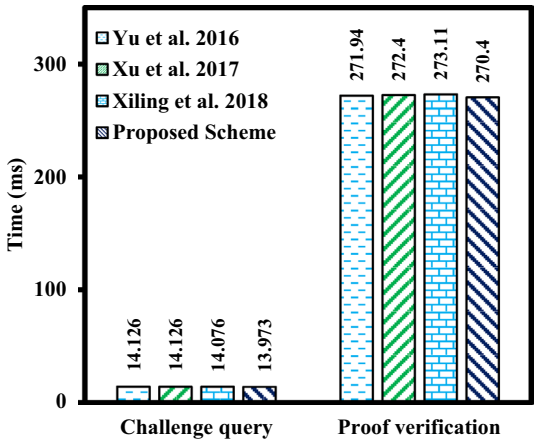

Fig. 9 Computation overhead on TPA to perform public auditing

not use any complex operations. However, the computation time to insert and verify the hash bits in the BMHT is O (n).

(iv) *Computation time to signature verification and proof generation* To verify the signature of the data owner, the cloud service provider calculates the value of 'y^e' as shown in Algorithm 4. In the proposed scheme, the time to generate the signature and verifying it are almost same (i.e.) $3Mul_{QR} + Hash_R + 3Exp_{QR} + Mod_{sign}$. Likewise, to generate the proof for challenged data blocks, CSP hashes the cipher text values using 'n' function. So the required computation time is, $nHash_{CT.}$ The computational overhead of the proposed system is compared with existing schemes system in Table 7.

(v) *Computation time to perform private auditing* Existing public auditing schemes does not support direct or private verification. Because, it puts an extra workload (both in computation and communication perspective) on the data owners. However, to overcome the trust issues associated with the TPA, private auditing is introduced in the proposed LDuAP scheme.

In existing private auditing scheme, to perform integrity verification, the cipher text stored in the cloud has to be transferred to the premises of the data owner and later it has to be decrypted. This tedious process increases the computation and computational time of the private auditing. In addition, transferring the cipher text between the data owner and CSP to verify the integrity increases the data vulnerability and possibility of network attacks.

To overcome these issues, the proposed LDuAP scheme allows the data owners to perform private integrity verification on the premises of the CSP without decrypting the cipher text. It reduces the computation and communicational overhead of the private auditing. To verify the integrity of the cipher text, the proposed private auditing scheme uses three addition operations and two exponential operations as

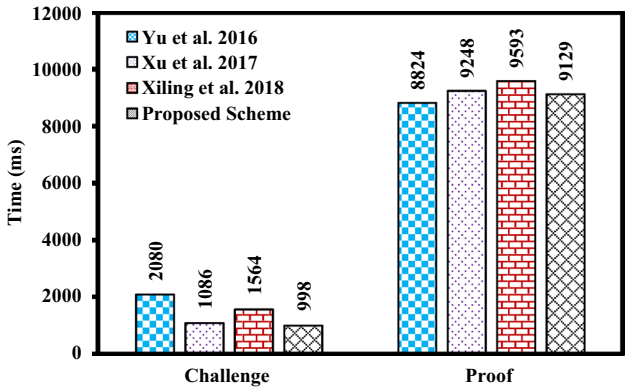

Fig. 11 Communication cost of the public auditing

shown in Eq. (1). The computation time required to perform direct verification is, $3Add_{CT} + 2Exp_{key}$.

In addition, the proposed private auditing scheme allows the data owners to verify the integrity only for a limited number of random data blocks. If TPA performs 'm' number of auditing in time 't', then the data owner is allowed to perform only 'm/10'number of direct auditing. It further helps in maintaining the workload on the data owner.

8 Implementation and result analysis of the LDuAP scheme

The experimental analysis of the proposed LDuAP scheme is carried out on a private cloud. Eucalyptus v4.2.0 is installed on an Intel Xeon E5 2620 v4 processor with a processing speed of 2.1 GHz, 64 GB of RAM and 4 TB of storage space to create the private cloud. An open source mhealth dataset, which consists of 1,72,824 vital signs value of the patients is used to evaluate the performance of the proposed LDuAP scheme. The values in the dataset are sensed using various medical sensors.

The main objective of the proposed LDuAP scheme is to reduce the computation and communication overhead of the integrity verification. In this section, to evaluate the total overhead of the proposed auditing scheme, computational overhead on data owner and TPA are measured separately. Likewise, the communication overhead to perform public and private auditing is measured and compared with the existing auditing schemes.

8.1 Computation overhead on the data owner

Data owner performs five major tasks in the proposed auditing scheme, such as, (1) signature generation, (2) key generation, (3) metadata generation, (4) encrypting the data and (5) private auditing for random data blocks.

As explained in the Sect. 7.3, the proposed scheme uses a lightweight signature to verify the ownership of the data. It subsequently reduces the computational overhead on the data owner and TPA. The time taken to generate the proposed lightweight signature for a total of 500 data blocks is measured and compared with existing RSA and BLS signatures in Fig. 6. Likewise, the computation time taken by the data owner to generate asymmetric keys, metadata information and encrypting a total of 750 data blocks each with 1 MB size is measured and shown in Fig. 7. In addition, the time taken to execute a single data block on the data owner is measured and compared with existing schemes in Table 8.

The time taken to generate asymmetric keys in the proposed LDuAP scheme is slightly higher than the existing schemes. Because, the proposed algorithm uses three exponent operations to generate keys. However, the time taken to generate the signature, encryption and metadata creation is low compared to the existing schemes (Yu et al. 2016; Xu et al. 2017; Xiling et al. 2018).

On the other hand, to perform private auditing for random number of data blocks 'R', the data owner perform two major operations such as, (1) challenges ($Chal_{private}$) the CSP directly using the block identification number (B_{id}) and data owner's signature, (2) verifies the integrity of the cipher text without decrypting it in the premises of the CSP. The time taken to perform private auditing for 500 data blocks is measured and shown in Fig. 8.

8.2 Computation overhead on TPA

Third Party Auditor perform two major tasks, such as, (1) challenging the CSP to prove the integrity of the data block and (2) proof verification. TPA uses metadata information to challenge the CSP and it uses the BMHT to verify the proof generated by CSP. The time taken to perform these operations for a single data block in the premises of the TPA is measured and shown in Fig. 9.

Likewise, the probability of false positive errors in the proposed BMHT is compared with existing traditional Bloom filter. To measure the probability of false positive errors in BMHT, a total of 500 data block 'R' each with 1 MB is used. The number of hash function (HF) used to hash the input data block is set to 3. The total size of the array used to measure the probability of false positive error in traditional bloom filter is fixed to 500. (i.e.) number of data blocks to be inserted. On the other hand, to calculate the initial matrix size (m*n) of the proposed BMHT, Eq. (2) is used.

Later, when the filter reaches a certain fill ratio threshold value, a new matrix is created with a tightened error probability. The creation of new and large BMHT causes a

sudden reduction in the probability of false positive errors in BMHT. The comparison of the probability of false positive errors in the proposed BMHT is compared with traditional Bloom filter in Fig. 10.

8.3 Communication overhead of the proposed LDuAP auditing scheme

While considering the communication overhead of the proposed public auditing scheme, uploading the cipher text to the cloud storage server and uploading the metadata to TPA is performed only once. However, the communication between the CSP and TPA happens in a loop to ensure the integrity of the data stored in the cloud. So, the communication between the TPA and CSP is measured and represented in Fig. 11.

On summarizing, the performance of the proposed public auditing scheme is evaluated based on the computation and communication overhead of the proposed LDuAP scheme. It effectively reduces the size of the signature by 50% and subsequently reduces the computation overhead on the data owner. In addition, the proposed BMHT in the TPA effectively reduces the probability of false positive errors and increases the accuracy of public auditing.

9 Conclusion and future works

To verify the integrity of the outsourced data, auditing is performed at frequent interval in the cloud storage servers. However, existing auditing scheme faces three major problems such as, (1) trust issues associated with the TPA, (2) false positive errors in the index table produces erroneous integrity results and (3) size of the signature used to verify the ownership is large and subsequently increases the computation overhead of the auditing. To overcome the issues, an LDuAP based on the Cramer-Shoup cryptosystem is proposed. It combines both public and private auditing schemes to overcome the trust issues associated with the TPA. It also introduces a low-error index table called, BMHT. It reduces the false positive errors in the hash table and increases the accuracy of the public auditing. In addition, to reduce the computation overhead of the auditing, a lightweight signature is used for ownership verification. The computational overhead of the proposed LDuAP scheme is convincing enough to work with real-time applications. However, the size of the cipher text encrypted using the Cramer Shoup cryptosystem is slightly high compared to the existing encryption algorithms. In future, the efficiency of the cloud storage can be improved by reducing the size of the cipher text encrypted by the Cramer Shoup cryptosystem.

Acknowledgements This research work was financially support by Science and Engineering Research Board (SERB), Department of Science and Technology, Government of India (Grant number: ECR/2016/000546).

Funding This research work was financially supported by Science and Engineering Research Board (SERB), Department of Science and Technology, Government of India under the research Grant number ECR/2016/000546.

Declarations

Conflict of interest The author(s) has no conflict of Interest.

References

Aditya T, Baruah PK, Mukkamla R (2011) Space efficient bloom filters for enforcing integrity of outsourced data in cloud environments. In: Proc. IEEE 4th Int. Conf. Cloud Comp., pp 292–299. https://doi.org/10.1109/cloud.2011.40

Ateniese G, Burns R, Curtmola R, Herring J, Kissner L, Peter Z, Song D (2007) Provable data possession at untrusted stores. In: Proc. 14th ACM Conf. Comp. Comm. Sec., pp 598–610. https://doi.org/10.1145/1047915.1047917

Ateniese G, Kamara S, Katz J (2009) Proofs of storage from homomorphic identification protocols. In: Proc. Int. Conf. Theory App. Cryp. Inf. Sec. Adv. Cryp., pp319–333. https://doi.org/10.1007/978-3-642-10366-7_19

Ateniese G, Burns R, Reza C, Joseph H, Khan O, Kissner L, Peterson Z, Song D (2011) Remote data checking using provable data possession. ACM Trans Inf Syst Secur 14(1):1–34. https://doi.org/10.1145/1952982.1952994

Bowers KD, Juels A, Oprea A (2009) HAIL: a high-availability and integrity layer for cloud storage. In: Proc. 16th ACM Conf. Comp. Comm. Sec, pp 188–198. https://doi.org/10.1145/1653662.1653686

He D, Kumar N, Wang H, Wang L, Choo KK (2017) Privacy-preserving certificateless provable data possession scheme for big data storage on cloud. Appl Math Comput 314:31–43. https://doi.org/10.1016/j.amc.2017.07.008

Jiang M, Zhao C, Xiang G (2013) A modified algorithm based on the bloom filter. In: Proc. Int. Con. Image Sig. Proc., pp 1087–1091. https://doi.org/10.1109/CISP.2013.6745220

Jiang M, Zhao C, Mo Z, Jing W (2018) An improved algorithm based on Bloom filter and its application in bar code recognition and processing. J Image Video Proc 139:1–12. https://doi.org/10.1186/s13640-018-0375-6

Kang B, Jiaqiang W, Dongyang S (2017) Attack on privacy-preserving public auditing schemes for cloud storage. Math Probl Eng. https://doi.org/10.1155/2017/8062182

Khan Z, Anwar B, Bordbar E, Ritter RH (2018) A protocol for preventing insider attacks in untrusted infrastructure-as-a-service clouds. IEEE Trans Cloud Comput 6(4):942–954. https://doi.org/10.1109/TCC.2016.2560161

Li L, Yang Y, Wu Z (2017) FMR-PDP: Flexible multiple-replica provable data possession in cloud storage. In: IEEE Symposium on Computers and Communications (ISCC), pp 1115–1121. https://doi.org/10.1109/ISCC.2017.8024675

Liu C, Ranjan R, Yang C, Zhang X, Wang L, Chen J (2015) MuR-DPA: Top-down levelled multi-replica Merkle hash tree based secure public auditing for dynamic big data storage on cloud.

IEEE Trans Comput 64(9):2609–2622. https://doi.org/10.1109/TC.2014.2375190

Liu L, Vel OD, Han Q, Zhang J, Xiang J (2018) Detecting and preventing cyber insider threats: a survey. IEEE Commun Surv Tutor 20(2):1397–1417. https://doi.org/10.1109/COMST.2018.2800740

Mukundan R, Madria S, Linderman M (2014) Efficient integrity verification of replicated data in cloud using homomorphic encryption. Distrib Parallel Databases 32(4):507–534. https://doi.org/10.1007/s10619-014-7151-0

Nianmin Y, Haifeng M, Yong H (2014) A method for memory integrity authentication based on bloom filter. J Algorithms Comput Technol. https://doi.org/10.1260/1748-3018.8.3.267

Shacham H, Waters B (2008) (2008) Compact proofs of retrievability. Proc Asia Crypt 5350:90–107. https://doi.org/10.1007/978-3-540-89255-7_7

Tabrizchi H, Kuchaki RM (2020) A survey on security challenges in cloud computing: issues, threats, and solutions. J Supercomput 76:9493–9532. https://doi.org/10.1007/s11227-020-03213-1

Tian H, Chen Y, Chang CC, Hong J, Huang Y, Chen Y, Liu J (2017) Dynamic-hash-table based public auditing for secure cloud storage. IEEE Trans Serv Comput 10(5):701–714. https://doi.org/10.1109/TSC.2015.2512589

Venkatesh M, Sumalatha MR, SelvaKumar C (2012) Improving public auditability, data possession in data storage security for cloud computing. In: Proc. Int. Conf. Recent Trends Inf. Tech.., pp 463–467. https://doi.org/10.1109/ICRTIT.2012.6206835

Walid KI, Khater HM, Mohamed ER (2019) Cryptographic accumulator based scheme for critical data integrity verification in cloud storage. IEEE Access 7:65635–65651. https://doi.org/10.1109/access.2019.2917628

Wang H (2013) Proxy provable data possession in public clouds. IEEE Trans Serv Comput 6(4):551–559. https://doi.org/10.1109/TSC.2012.35

Wang H (2015) Identity-based distributed provable data possession in multi-cloud storage. IEEE Trans Serv Comput 8(2):328–340. https://doi.org/10.1109/TSC.2014.1

Wang Q, Wang C, Ren K, Lou W, Li J (2011) Enabling public auditability and data dynamics for storage security in cloud computing. IEEE Trans Parallel Distrib Syst 22(5):847–859. https://doi.org/10.1109/TPDS.2010.183

Wang XA, Liu Y, Zhang J, Yang X, Zhang M (2018) Improved group-oriented proofs of cloud storage in IoT setting. Concurr Comput Pract Exp. https://doi.org/10.1002/cpe.4781

Xiang S, Wang F, Cao Q (2016) A Bloom Filter based scalable data integrity check tool for large-scale dataset. In: Int. Para. Stor. data Inten. Scal. Comp. Syst., pp 55–60. https://doi.org/10.1109/pdsw-discs.2016.014

Xiling L, Zequan Z, Lin Z, Jian M, Chaoyong C (2018) An effective integrity verification scheme of cloud data based on BLS signature. Secur Commun Netw. https://doi.org/10.1155/2018/2615249

Xu Z, Wu L, Khan MK, Choo KR, He D (2017) A secure and efficient public auditing scheme using RSA algorithm for cloud storage. J Super Comput 73(12):5285–5309. https://doi.org/10.1007/s11227-017-2085-8

Yan Y, Lei W, Gao G, Wang H, Wenyu X (2018) A dynamic integrity verification scheme of cloud storage data based on lattice and Bloom filter. J Inf Secur Appl 39:10–18. https://doi.org/10.1016/j.jisa.2018.01.004

Yu Y, Xue L, Au MH, Susilo W, Ni J, Zhang Y, Vasilakos AV, Shen J (2016) Cloud data integrity checking with an identity-based auditing mechanism from RSA. Future Gener Comput Syst 62:85–91. https://doi.org/10.1016/j.future.2016.02.003

Yuhan L, Fu A, Yu Y, Zhang G (2017) IPOR: An efficient IDA-based proof of retrievability scheme for cloud storage systems. IEEE Int. Conf. Comm., pp. 1–6. https://doi.org/10.1109/ICC.2017.7997106

Zhang S, Hang Z, Yahui YW (2017) A joint Bloom filter and cross-encoding for data verification and recovery in cloud. IEEE Sym. Comp. Comm., pp 614–619. https://doi.org/10.1109/iscc.2017.8024596

Zhang J, Wang B, He D, Wang XA (2019) Improved secure fuzzy auditing protocol for cloud data storage. Soft Comput 23:3411–3422. https://doi.org/10.1007/s00500-017-3000-1

Zhang J, Wang B, Wang XA, Wang H, Xiao S (2020) New group user based privacy preserving cloud auditing protocol. Future Gener Comput Syst 106:585–594. https://doi.org/10.1016/j.future.2020.01.029

Zhu Y, Ahn GJ, Hu H, Yau SS, An HG, Hu CJ (2013) Dynamic audit services for outsourced storages in clouds. IEEE Trans Serv Comput 6(2):227–238. https://doi.org/10.1109/TSC.2011.51