Received 6 March 2024, accepted 7 April 2024, date of publication 15 April 2024, date of current version 30 April 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3389076

RESEARCH ARTICLE

Stub Signature-Based Efficient Public Data Auditing System Using Dynamic Procedures in Cloud Computing

PAROMITA GOSWAMI[1,2], NEETU FAUJDAR[2], GHANSHYAM SINGH[3], KANTA PRASAD SHARMA[2], (Member, IEEE), AJOY KUMAR KHAN[1], AND SOMEN DEBNATH[4]

[1]Department of Computer Engineering, Mizoram University, Aizawl 796004, India
[2]Department of Computer Engineering and Application, GLA University, Mathura 281406, India
[3]Centre for Smart Information and Communication Systems, Department of Electrical and Electronic Engineering Science, University of Johannesburg, Auckland Park Campus, Johannesburg 2006, South Africa
[4]Department of Computer Science and Engineering, Tripura University, Agartala 799022, India

Corresponding author: Neetu Faujdar (neetu.faujdar@gmail.com)

ABSTRACT Online cloud data storage is a rapidly growing pillar of the IT industry that offers data owners an array of attractive developments in highly sought-after online scalable storage services. Cloud users can easily access these services and have the flexibility to manage their process data effectively without worrying about the deployment or maintenance of personal storage devices. As a result, the number of cloud users has increased to purchase these convenient and cost-effective services, while Cloud Service Providers (CSP) are also rising to meet this demand for appealing cloud solutions.However, there is one major security issue related to outsourced data on shared cloud storage: its privacy and accuracy cannot be guaranteed as it may be vulnerable to unauthorized access by malicious insiders or hackers from outside sources. To address these issues, we suggest proposing a partial signature-based data auditing system so that both privacy and accuracy can be fortified while reducing the computational cost associated with auditing processes significantly. This system would involve using cryptographic techniques such as homomorphic encryption and hash functions, which would enable secure sharing between multiple parties while ensuring integrity checks on stored files at regular intervals for any potential tampering attempts made by external attackers or malicious insiders who may try to gain unauthorized access into confidential user information stored within cloud sites. Another benefit of the plan is that it supports dynamic operation on outsourced data. This research work may achieve the desired security qualities, according to the security analysis, and it is effective for real-world applications, as demonstrated by simulation outcomes of dynamic operations on various numbers of data blocks and sub-blocks.

INDEX TERMS Cloud computing, public data auditing, partial signature, data dynamics, unambiguity.

I. INTRODUCTION

Cloud computing has been visualized as the next-generation information technology (IT) for businesses due to a lengthy list of unmatched benefits in IT's historical past, including on-demand self-service, widespread network services, location-independent dynamic resources, quick resource stretchability, utilization pricing, and threat transmission [1]. Since using the services of cloud allows users to outsource their data without having to pay high hardware and software servicing fees, users benefit greatly from using it. However, if users move their data to the cloud and stop keeping it locally, they will no longer have physical control over it. It is challenging to guarantee the integrity of cloud data because hardware/software faults and human mistakes are unavoidable in the cloud [2]. To examine whether the data saved in the cloud

The associate editor coordinating the review of this manuscript and approving it for publication was Chin-Feng Lai.

is unbroken or not and to evaluate whether the information is properly stored in the cloud, a variety of data auditing approaches have been presented [3]. In global file integrity auditing methods, data blocks must first be signed by the cloud user before being delivered to the cloud. The evidence for that reason is provided by such signatures. At this stage of the integrity inspection, these data blocks are genuinely present in the cloud. After that, the data owner uploads these data blocks and their matching signatures to the cloud.

Many users of various cloud storage services, such as Google iCloud, Drive, and Dropbox, often share the information stored in the cloud. Sharing of data is one of the most numbers of individuals who can access it thanks to common cloud storage characteristics to allow others to see their information [4], [5]. Many research works discussed pairing schemes because of their time-consuming nature during computation. In research paper [6], public verification scheme has been proposed to check cloud users' data integrity and also to resist external adversaries using *Boneh_Lynn_Shacham* (BLS) signature. A unique hash function called Map-To-Point, which is also employed by the majority of traditional cryptographic systems from pairs, is required for BLS short signatures. This hash function involves more pairing operations, which makes it probabilistic and inefficient in general [7]. *Zhang_Safavi_Susilo* (ZSS) short signature is more efficient than BLS method because it performs less pairing operations. ZSS signature is utilized in the study article for batch auditing and to protect data privacy in cloud servers [8], [9]. When creating third-party cloud data auditing techniques, the bilinear paring is frequently used. However, these auditing schemes' verification processes will result in a significant cost for that calculation. As a result, a better auditing system must be created with cloud computing in mind. An algebraic signature-based cloud data integrity auditing technique that can satisfy the security features of data secrecy, privacy preservation, and free-riding attack resistance has been proposed in a research study [10]. The majority of the auditing techniques currently in use are focused on the challenging issues of discrete logarithms and large integer decomposition. These systems will confront a security risk when quantum computers get more advanced since they can handle these challenging issues with ease. Designing a data audit system that is impervious to quantum attacks is crucial. Therefore, a novel data audit system is used in this research work to ensure post-quantum security, based on the ring signature problem of learning with errors [11], [12].

Most often in research works [4], [8], [13], a cloud user divides his or her data into data blocks, creates a signature for each data block, and uploads the original data blocks together with the signatures of each data block to shared cloud storage. Signature generation can occasionally take longer for all data blocks, and this highlights problems with data secrecy. Later, cloud users pay a Third Party Auditor(TPA) to verify the authenticity of the data using signatures that are kept in a CSP's shared cloud storage. Because cloud user stores the original file block along with its identification information at cloud storage, despite the existence of all features does not guarantee data secrecy there. As a result, security risks still exist for cloud storage of outsourced data. Therefore, taking into account the current situation and in order to address the aforementioned issues, this research work proposes an efficient public data auditing scheme using stub signatures that upholds data confidentiality, data privacy, auditing correctness, unambiguity, anonymity, resists collision and forgery attacks and supports dynamic operations at the sub-block level. In this study, we present a partial signature-based outsourced data auditing scheme that can safeguard data security and privacy while conducting data integrity audits.

The proposed research offers a partial signature-based data auditing method that, while maintaining data security, is more effective and computationally time-consuming for signature creation and verification than related schemes. The following is a summary of our contributions:

- To validate the data integrity in shared cloud storage, we provide a public auditing mechanism through a TPA. The security requirements for maintaining the privacy of outsourced data stores in cloud storage are met by this auditing system, which also upholds data confidentiality, data correctness, auditing correctness, unambiguity, anonymity, etc. while the audit is being conducted.
- We present a privacy method to protect the original block elements from malicious insider and outsider assaults in cloud storage. This technique stops forgery, substitutes attacks on cloud storage, and resists block collusion issues.
- We extend an existing decentralized security model with our suggested data auditing scheme based on the partial signature.
- The linear data structure is used as the foundation for a cloud data dynamics mechanism. The suggested dynamics operation offers data addition, deletion, and update at the level of the multiple blocks and their sub-blocks.

The following is how this research project is organized: Section II explores several integrity schemes. A brief explanation of digital and partial signatures is provided in Section III. Section IV discusses the proposed system model's overview, design objectives, basic definitions, and dynamic operation. To demonstrate the superior efficacy of the suggested model, Section V briefly discusses security analysis and performance analysis using simulated results and compares the implemented result with research papers [4], [8], [10], [13]. The research study is concluded in Section VI.

II. RELATED WORKS

Many researchers have given the integrity and confidentiality of user data substantial thought. In order to improve the security and fairness of the data auditing, a TPA is taken into account when designing the auditing protocol. On the other hand, researchers are dedicated to extending their findings to support new functions. In the paper [8], the Researchers suggest a method for verifying the integrity of data based on a short signature algorithm (ZSS signature), that offers privacy and security and public auditing by adding a reliable third party (TPA). Here, by decreasing the hash function complexity in the signature process, the operational complexity is effectively decreased. The study indicates that the approach is more effective and safer. Using batch auditing, the data integrity auditing scheme this research paper [10] assures the anonymity and integrity of cloud data where the method has the benefit of only requiring one cloud server to support data dynamics and provides favored security characteristics. Based on computational Diffie-Hellman (CDH) assumptions, this study [3] provides a security framework and security verification. According to the research observations, the data integrity audit procedure now includes a limited vision component and a fog computing layer, which may significantly shorten data transmission lag times while also enhancing data audit security. Researchers develop such a plan and provide a new illustration termed data integrity audits without private key stored in the paper [14]. In this method, Researchers avoid using the hardware token by using biometric information (such as an iris scan or fingerprint) as the user's fuzzy private key, and to verify the user's identity, use a linear sketch with coding and error-correcting procedures. This scheme also provides favored security characteristics. In this research [7], a dynamic data auditing method that keeps up data protection is suggested. Here, in the initialization step, first construct the data authentication structure that is hierarchical and has several branches and secondly, create a data auditing technique, based just on Boneh-Lynn-Shacham digitally signed technique and bilinear coupling mapping technology, and thoroughly outline the data dynamically upgrading procedure. Ultimately, the system assessment process includes security assessment and evaluation metrics. For safe cloud auditing in terms of storage, connectivity, and compute costs, the research work [15], developed a new traditional authentication framework based on Ternary Hash Tree (THT) and Replica-based Ternary Hash Tree (R-THT). TPA will use this framework to carry out data audits. To ensure accessibility and assuring data integrity in the cloud, the proposed method provides auditing at the block, file, and replica levels while using storage block ordering and tree block ordering. Data uploading using proxy-oriented methods is proposed in this paper [16], [17]. Here, the bilinear pairings are used to construct a practical ID-PUIC protocol. On the basis of the computational hardness of Diffie–Hellman, the ID-PUIC protocol is proven to be secure. The researchers in the paper [18] define formally reversible attribute-based encryption with data intactness protection and present a security architecture (RABE-DI). They provide a specific RABE-DI scheme and demonstrate its integrity and confidentiality inside the specified security architecture. Finally, they demonstrate the effectiveness and applicability of their plan with an execution outcome and effectiveness review. Though numerous research publications suggested efficient techniques to swiftly reply to cloud users in order to maintain a large resource pool [19], [20], [21] but security issues are still present in cloud storage, in this research [4], the researchers suggest a remote data integrity auditing technique that enables data exchange with concealed sensitive data. A sanitizer is utilized in this method to turn the data block's signatures into correct arguments for the cleaned file while also cleaning the data blocks that correspond to the file's sensitive information. In order to fend against outside attackers, researchers offer a traditional authentication approach in that they employ a random masking mechanism rather than secure routes between cloud servers and auditors [12]. In order to fend against outside attackers, researchers offer a traditional authentication approach in that they employ a random masking mechanism rather than secure routes between cloud servers and auditors in [6]. They build a fair challenge message using Bitcoin to deter malevolent auditors from working with cloud servers. The author of this research paper [22] presented an enhanced kind of a signcryption method based upon the short signature ZSS that may fulfill the aforementioned key data verification criteria. This proposed approach can provide more features for the same computational cost as another ZSS signcryption scheme. The author proposed a blockchain-based system in his research paper [23].BCD-IV is a cloud data integrity verification scheme. TPAs are removed using chain code, which solves collusive attacks. It also created a new homomorphic verification tag (HVT) on the basis of the ZSS signature, which provides blockless verification. The suggested framework enables full dynamic operations with auxiliary data that is kept on a blockchain. The authors of [24] present a distributed machine learning-focused data integrity verification technique that can solve the key escrow problem while also lowering costs. To assure integrity, this technique is based on identity-based cryptography and two-step key generation technology. The authors use the PDP sampling auditing technique to provide data integrity verification so that the suggested scheme can withstand forgery and tampering assaults. The author provided a feasible approach for dynamic reviewing in the study paper [25] by using a dynamic list-based index table to verify the integrity of the data, which is more efficient than the state-of-the-art. Furthermore, it has been demonstrated that with a verification structure, communication and storage costs on the client side are effectively reduced.The author of this paper [26] skillfully combined the Geohash algorithm with the Symmetric-key Hidden Vector Encryption (SHVE) and Circular Shift and Coalesce Bloom Filter (CSC-BF) framework to propose an efficient Privacy-preserving Spatial Range Query (PSRQ) scheme. This method not only significantly lowers the computational cost of token generation but also increases query efficiency on large-scale datasets.The majority of current systems encrypt data using Asymmetric Scalar-Product-Preserving Encryption (ASPE), yet ASPE has been shown to be vulnerable to known plaintext attacks. Furthermore, users of the current techniques are required to supply a greater quantity of information regarding the query range, resulting in a huge number of ciphertexts and a high computational and storage overhead. In order to address these problems, the authors first proposed a new unified index structure for a basic Privacy-preserving Spatial Data Query (PSDQ) scheme that only requires users to provide a minimal amount of information about the query range. Second, the authors proposed an enhanced PSDQ scheme (PSDQ +) that uses an efficient pruning strategy and a Geohash-based R-tree structure (called GR -tree) to significantly reduce query time [27].

III. PRELIMINARIES

A. DIGITAL SIGNATURE

Three procedures make up a digital signature scheme (DS), which has the following features.

Key Generation Algorithm (SKG): The public key and corresponding secret key are both returned as a pair (vk, sk) by the key generation algorithm SKG.

Signing Algorithm (SIG): The secret key sk and the message M are input into the signing algorithm SIG, which returns a signature S.

Signature Verification Function (SVF): A candidate signature, a message, and a public key are provided to the deterministic verification algorithm SVF, which returns either 1 or 0.

B. PARTIAL SIGNATURE SCHEME

Let's say a signer distributes his message to the audience before claiming ownership of it. The signer can calculate "stub", which keeps his identity private [28]. This "stub" is regarded as a partial signature in this case, and only the owner of the public key can verify the partial signature. It is a deterministic approach. Any signature produced by the signing algorithm is essentially a pair (σ, k) in a partial signature scheme $PS = \{PKG, PSIG, PVF\}$, which is a type of digital signature scheme. Here, σ is considered a stub, while k is considered a de-anonymizer. Its three security features are unambiguity, unforgeability, and anonymity. The algorithms for the partial signature scheme $PS = (PKG, PSIG, PVF)$, which was created from the Schnorr identification protocol using the splitting structure, are listed below in Table1.

IV. PROPOSED OUTSOURCED DATA AUDITING SCHEME

The suggested system model represented in Figure 1, has been provided in this section. The auditing of the suggested scheme with an overview of the proposed system model, a basic description, an example of cloud storage security model, design objectives, and dynamic operations on data blocks is covered in more detail in this part. The four elements that make up the unabridged suggested system are as follows: Cloud User (CU), Cloud Service Provider (CSP), Public Third Party Checker (PTPC), and Remote Cloud Server(RCS) like [8], [10], [13], and [29]. In our architecture, CU divides a file into blocks, encrypts each block using a secret parameter, and then stores the encrypted data on a hybrid cloud storage server i.e. RCS. Later, in the batch auditing scheme, CU hires a PTPC to check the intactness of data blocks. Then PTPC will send a challenge message to CSP and CSP reply to PTPC as a reply message. Then PTPC will verify the reply message to inspect the accuracy of the challenged data blocks. The PTPC can assess whether the data are saved integrally after verifying the accuracy of the proof.

- Cloud User (CU): CUs are the consumers of cloud computing who offload their personal data or private files to the cloud resource's servers and take advantage of the cloud's functionalities. They employ fixed or portable tools and wired or wireless networks to access the cloud. The fact that their storage and computing capacities are constrained in comparison to the cloud, despite the fact that their terminals differ, is a shared feature.
- Cloud Service Provider (CSP): A remote cloud server can be thought of as a storage and computational pool, giving CU access to an endless number of resources. Numerous redundant and distributed servers are leased by CSP, and these servers deliver various services to CU in accordance with their needs. CSP is honest but not entirely dependable; it is interested in a user's data, particularly sensitive data. CSP will carry out all actions in line with the security assurance protocol while it is active on introducing the system.
- Public Third-Party Checker (PTPC): PTPC is an independent third party, apart from users and the cloud. The PTPC is qualified and equipped to examine cloud data. After CU submits an audit checking request to the PTPC, the PTPC will create a challenge for the data auditing process. The challenge will then be sent to CSP by the PTPC. In accordance with the challenge information, the CSP will produce proof. After examining the proof, the PTPC can produce an audit result. The PTPC can be used to audit user data in a cloud system, saving CU's computing and storage resources while guaranteeing the integrity of the verification process.
- Remote Cloud Server (RCS): An amalgam of both public and private cloud storage servers makes up a Remote Cloud Server. A brand-new remote storage server technology provides both the efficiency of a traditional dedicated server and the adaptability of both private and public cloud computing.

Here, the main objectives, as they are summarised above, are to maintain the data integrity of CU's outsourced data while

TABLE 1. Different sub-parts of partial signature algorithm.

Algorithm PKG()	**Algorithm** $PSIG(sk, M)$	**Algorithm PVF** $(vk, M, (\sigma, \kappa))$

$a \leftarrow Z_p$;
 $sk \leftarrow a$;
 $A \leftarrow g^a$;
 $vk \leftarrow A$
 $return(sk, vk)$
	$sk \leftarrow a$;
	 $B \leftarrow g^b$;
	 $\sigma = H(A \parallel B \parallel M)$;
	 $\kappa = b + \sigma * a \mod p$;
	 $return(\sigma, \kappa)$;
		$If A \notin G$ and $\sigma \neq \kappa$ and $\kappa \notin Z_p$
		 Then return 0;
		 $B = g^\kappa * A^\sigma$;
		 If $\sigma = H(A \parallel B \parallel M)$
		 Then return 1;
		 else
		 return 0;

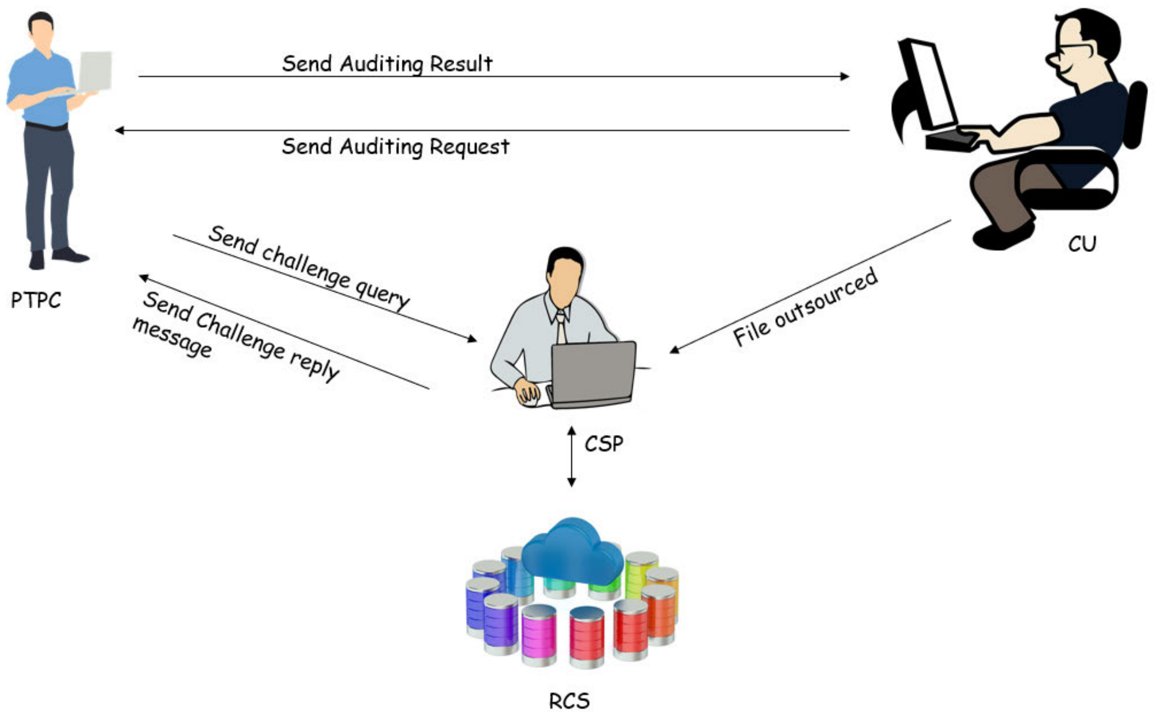

FIGURE 1. Proposed Auditing Model.

it is being stored on an insecure remote cloud server and to confirm the audit verification results produced by PTPC on the CU end. In order to make money, CU creates data before uploading it to any remote cloud storage. PTPC is an independent company that provides storage services to CUs. All outsourced data from CU is stored by CSP in RCS, an online shared data storage. By verifying the accuracy and integrity of data that is outsourced, PTPC relieves CU of the duty of managing its data. In this prototype system model, we assume that each entity has a little amount of personal storage and personal computational power to process data.

A. OVERVIEW OF THE PROPOSED AUDITING MODEL

The overview of the proposed system model is described below and Table 2 describes the definition of all notations:

- First, CU determines the maximum size of file F as l, then divides the file F into n no. of chunk size variable blocks $B = \{B_1, B_2, \ldots..B_n\}$, with the maximum size of each block being w and $1 < l < cel(\frac{l}{d})$, $2 \leq d \leq w$ i.e.each block size is w and it will vary for all blocks.
- CU selects a secret parameter x as sk from p at the start of the scheme. CU and CSP select random parameters a as sk_1 and b as sk_2 respectively where $a, b \in \mathbb{Z}$ with p prime order. CU prepares a public parameter vk_1 as g^a and CSP prepares a public parameter vk_2 as g^b. Here, p denotes $\mathbb{Z}$'s prime order. Here, g is a generator of G and G is a group of prime order p.
- Later each block elements $B_i[M_j]$, $i \in n$, $j \in w$ are encrypted by x to produce modified block elements $B_i[M'_j]$, and CU calculates hash values of each blocks like $\{H(B_1[M'_j]), H(B_2[M'_j]), .., H(B_n[M'_j])\}$, where $j \in w$.
- CU prepares a message *info_msg* which contains a file tag ϕ, generator g, encrypted block elements $B_i[M'_j]$ along with public parameter vk_1 and sends it to CSP for storing data.

TABLE 2. Definitions of notations.

Notation	Meaning
g	Multiplicative Cyclic group G with p prime order
$\mathbb{Z}_p$	Ring of integer modulo p
F	Original File
l	Size of File F
ϕ	File Tag
a, b	Random parameter of CU as sk_1 and of CSP as sk_2 where $a, b \in Z_p$
x	Secret parameter of CU where $x \in Z$
vk_1, vk_2	Public parameter of CU as A and of CSP as B accordingly
$B = \{B_1, B_2, ...B_n\}$	Original blocks
$B_i[M_j]$	Block Elements where $i \in n$,$j \in w$
$B_i[M_j']$	Encrypted Block Elements where $i \in n$,$j \in w$
H	A cryptographic hash function: $H : 0, 1* \leftarrow g$
$\{H(B_1[M_j']), H(B_2[M_j']), .., H(B_n[M_j'])\}$	Hash value of all encrypted blocks
n	Total block numbers(Block number depend on CU
w	Dynamic Block Size
σ	Stub
κ	Anonymity
τ	list of challenged blocks
t	total no. of challenged blocks
$B_i[\theta_j]$	Collude Block Elements

- CU also sends $\{H(B_i[M_j']), g, vk_1, \phi, n\}$ as a *req_gen* to PTPC for data integrity verification.
- Based of *req_gen* message PTPC generates a challenge message as *chal_pro* where it contains challenged block list τ, file tag ϕ and sends it to CSP.
- After verifying the file tag ϕ, CSP prepares stub σ and κ. Later CSP prepares a challenge-response message as a $chal\_reply = \{\phi, \sigma, \kappa, vk_2\}$ and sends it to PTPC to examine the verification test. After receiving *chal_reply* message, PTPC verifies the message to ensure the intactness of block elements in RCS, and the verification result will be sent to CU.

B. BLOCKCHAIN TECHNOLOGY: SECURITY IN CLOUD STORAGE

Our data integrity technique, which is represented in Fig.2, can be used in distributed blockchain technology to store information about user transactions involving digital currencies like bitcoin [30]. Decentralization, autonomy, openness, information modification, and anonymity are features of the underlying technology and infrastructure of this blockchain. All seven layers make up an analogue computer network, and they are how the blockchain system is divided into these layers depicted in Figure 2. This blockchain architecture offered a data integrity method that made use of recoverable proof (POR) and proven data holding (PDP) technologies in the consensus layer. It is based on challenge-response in distributed cloud storage systems and data access. The challenge-proof response-based integrity verification mechanism includes both verifiers and responders. The verifier in this case is a TPA, and the responder is a cloud service provider.

The challenge-response-based data integrity verification scheme's workflow is divided into the following three stages: The initial step in the setup procedure is for CU to preprocess the data files (by blocking them, creating different tag information, etc.) before sending the data owner's data to the cloud storage server. Stage 2 of the challenge: TPA generates the relevant challenge data in accordance with its own requirements and transmits it to the cloud server. 3. Check Proof stage: CSP produces the appropriate response data in accordance with a predetermined protocol and provides it to TPA. Later, TPA calculates using the information from the returned answer to see if the data is accurate.

C. PROPOSED OBJECTIVES

- Data Confidentiality: This encryption option assures that original data must not be disclosed in RCS during the outsourced data integrity verification procedure to either CSP or any other malicious authorized entity [10].
- Public Auditing: A third-party auditor (PTPC) is added to facilitate public auditing in the data validation process based on user authorization and privacy protection. Instead of the user, PTPC requests the cloud storage server's data intactness verification and completes the verification [8].
- Auditing Correctness: It makes sure that only when CSP correctly stores outsourced data into cloud storage can the response message from the CSP side pass the verification trial of PTPC [6].
- Verification Correctness: The validation test of CU can only be passed by the proof of CSP if both CSP and CU are truthful and CSP and CU correctly adhere to the pre-defined procedure of data storing, such as [4], [8], [10], [13], and [14].
- Collusion Resistance: If CSP did not collude with the data stored at RCS, the auditing verification test demonstrates the accuracy of the data storage [15].
- Unforgeability: If the CSP correctly maintains outsourced data at RCS, only the *chal_reply* message can pass the verification test, proving that our system model

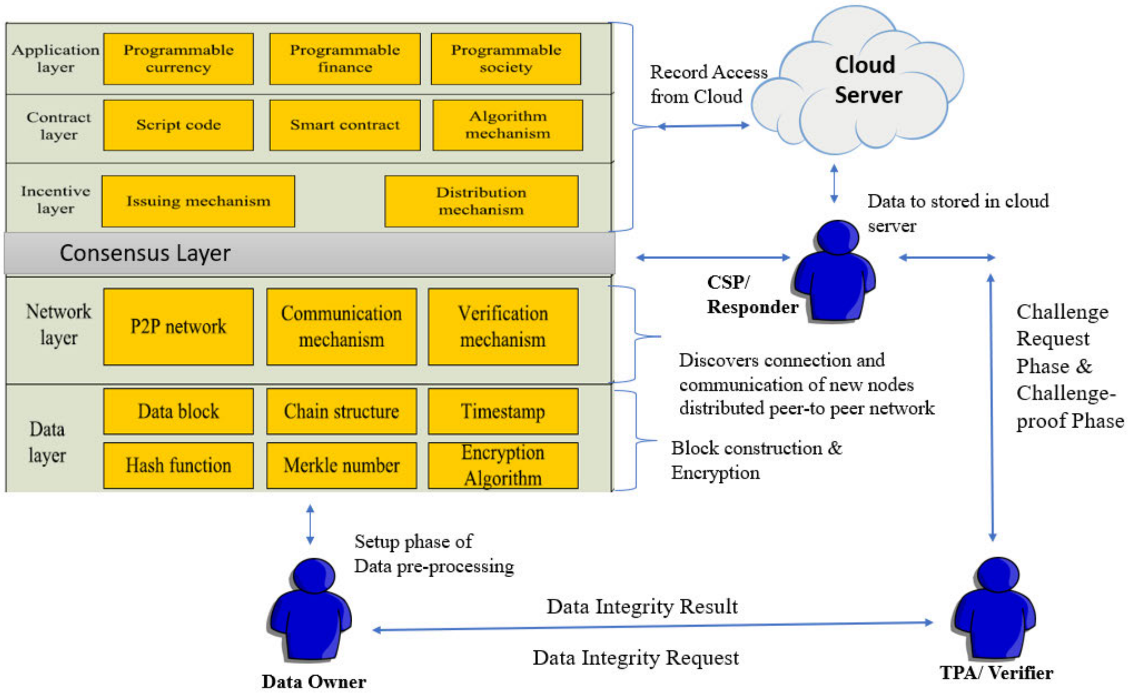

FIGURE 2. Blockchain Infrastructure using Proposed Auditing Model.

can withstand both internal and external forgery attacks [8], [10].

- Unambiguity: Given a stub σ under sk' of the challenged blocks of her choosing, an adversary is prevented from creating κ, vk such that (σ, κ, vk) validates as a signature of m under vk by unambiguity [5], [28].
- Anonymity: It is not recommended to infer the creator of the stub's identity from the stub and message [28].

D. BASIC SIGNATURE VERIFICATION SCHEME

In this research paper, we use chunk-sized variable blocks to ignore the problem of a boundary shifting problem [31]. The parameter generation phase, the sign generating phase, and the signature verification phase make up the three phases of the suggested data integrity verification technique. Note that we refer to the six steps in our basic architecture as keyGen, MsgEncrypt, RequestGen, ChalGen, SignGen, and SignVeri, respectively. Our suggested audit approach has been broken down into two parts in this section: Block Data Processing and Block Data Auditing which is depicted in Figure 4. The stages will be introduced in depth. The first three steps are under the Block Data Processing phase and the next three steps are under the Block Data Auditing phase. Here Algorithm 1 and Algorithm 2 are represented Signature Generation and Signature Verification scheme.

1) BLOCK DATA PROCESSING STAGE

$KeyGen() \rightarrow Para(sk, vk)$**:** DO generates a group generator g_1 from G_1 using a bilinear map and two random numbers r_1, r_2 to produce a pair of public and private keys (DO_{pub}, DO_{pri}) where $g_1 \in G_1$ and $r_1, r_2 \in Z_q^*$.DO also prepares a file tag ω for file identification.CU selects a random parameter x from p at the start of the scheme. CU and CSP randomly select private parameters an as sk_1 and b as sk_2 respectively where $a, b \in \mathbb{Z}_p$. CU prepares a public parameter vk_1 as g^a and CSP prepares a public parameter vk_2 as g^b. Here, p denotes $\mathbb{Z}$'s prime order. Here, g is a G's generator and G is a p's group of prime order. CU also prepares ϕ from the combination of A, B, and $\mathbb{Z}_\shortmid$ as file identifiers. $MsgEncrypt(x, B_i[M_j]) \rightarrow$

Algorithm 1 Algorithm for Stub Signature Generation Method

```
Input: $vk_1, vk_2, \tau, \phi, B', H(B[M'_{ij}]$
Output: Stub Signature $= \sigma, \kappa, t$ out
    Initialisation:
1: After verifying the chal_pro message, the CSP generates
   chal_reply message to the CU.
2: for $i = 1$ to $t$ do
3:   for $j = 1$ to $w$ do
4:     Calculates $\sigma = H[\sum_{i=1}^{t}\sum_{j=1}^{w} B_i[M'_j]||vk_1||vk_2)].a^t$
       and $\kappa = (b.H(\sum_{i=1}^{t}\sum_{j=1}^{w} B_i(M'_j) + \sigma.a) + t$ .
5:   end for
6: end for
```

$CipherMessage(B' = \{B_1[M'_j], B_2[M'_j], \ldots, B_n[M'_j]\})$**:** CU executes this phase because it generates an encrypted version for each data block. The random parameter and the data blocks are the phase's inputs, and its output is an encrypted

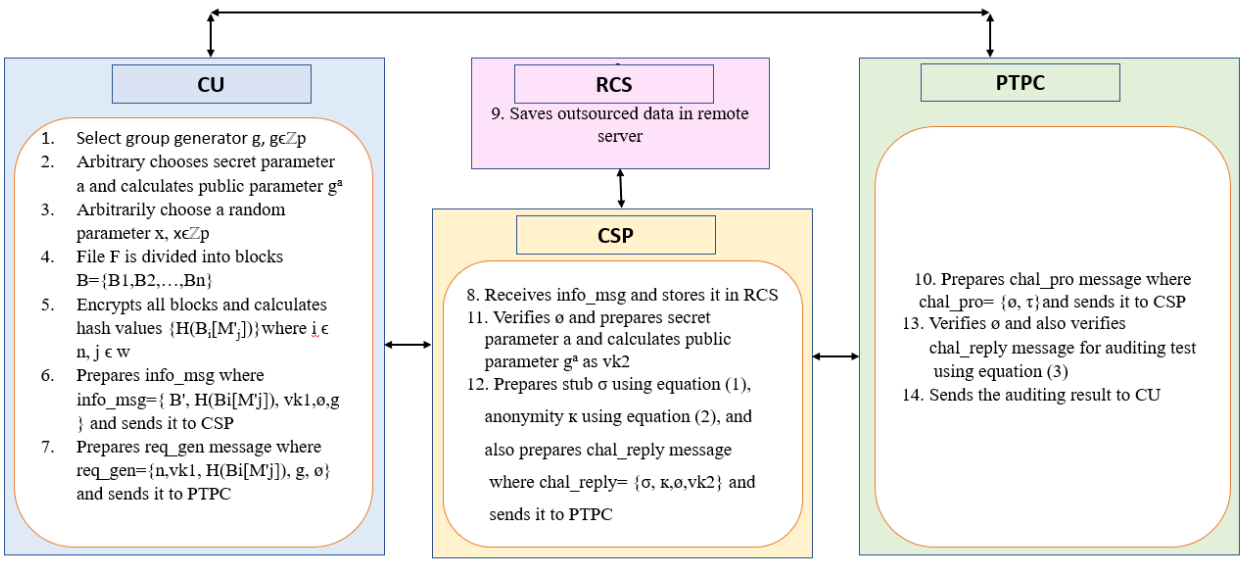
CU
1. Select group generator g, gϵℤp
2. Arbitrary chooses secret parameter a and calculates public parameter gᵃ
3. Arbitrarily choose a random parameter x, xϵℤp
4. File F is divided into blocks B={B1,B2,….,Bn}
5. Encrypts all blocks and calculates hash values {H(Bi[M'j])}where i ϵ n, j ϵ w
6. Prepares info_msg where info_msg={ B', H(Bi[M'j]), vk1,ø,g } and sends it to CSP
7. Prepares req_gen message where req_gen={n,vk1, H(Bi[M'j]), g, ø} and sends it to PTPC
RCS
9. Saves outsourced data in remote server
CSP
8. Receives info_msg and stores it in RCS
11. Verifies ø and prepares secret parameter a and calculates public parameter gᵃ as vk2
12. Prepares stub σ using equation (1), anonymity κ using equation (2), and also prepares chal_reply message where chal_reply= {σ, κ,ø,vk2} and sends it to PTPC
PTPC
10. Prepares chal_pro message where chal_pro= {ø, τ}and sends it to CSP
13. Verifies ø and also verifies chal_reply message for auditing test using equation (3)
14. Sends the auditing result to CU

FIGURE 3. **Process of Data Auditing at Block Level.**

set containing all of the data blocks' cipher versions.CU sends the information message $info\_msg = \{B', H(B_i[M_j']), vk_1, \phi\}$ to CSP where $H(B_i[M_j'])$ is a set of randomized hash values of splitting construction of Schnorr algorithm [28] based on general Md5 or SHA hash family. After receiving *info_msg*, CSP verifies ϕ and stores it in RCS. *RequestGen*() →

Algorithm 2 Algorithm for Stub Signature Verification Method

```
Input:$\sigma, \tau, H(B_i[M_j']), t, vk_1, vk_2, g$
Output:Stub Signature Verification Result (Y/N) out
    Initialisation:
1: After verifying the chal_reply message, the PTPC verifies $\sigma$ and $\kappa$.
2: if $g^{\kappa} = vk_1^{\sigma} + vk_2^{H[\sum_{i=1}^{\tau}\sum_{j=1}^{w} B_i(M_j')]}$ then
3:     Signature verification is true.
4: else
5:     Signature verification is false.
6: end if
```

req_gen$(H(B_i[M_j']), g, n, vk_1, \phi)$**:** CU executes this phase because it generates a requested message *req_gen* and sends it to PTPC for public auditing of data blocks where $req\_gen = \{H(B_i[M_j']), g, vk_1, \phi, n\}$.

2) BLOCK DATA AUDITING STAGE

- *ChalGen*(n, ϕ) → *chal_pro*(τ, ϕ, t)**:**PTPC executes this phase for preparing *chal_pro* message to inspect the intactness of CU's data, and sends it to CSP.
- *SignGen*$(vk_1, vk_2, \tau, B', H(B[M_{ij}'])$ → *stubGen*(σ, κ)**:** At first, CSP verifies ϕ and CSP should respond to the PTPC with verification of information stored at RCS after receiving the *chal_pro* message from the PTPC. CSP carries out this phase. The stub and anonymity are the outputs. CSP prepares stub σ where

$$\sigma = H[\sum_{i=1}^{t}\sum_{j=1}^{w} B_i[M_j']||vk_1||vk_2)].a^t \quad (1)$$

and also prepares the value of anonymity κ where

$$\kappa = (b.H(\sum_{i=1}^{t}\sum_{j=1}^{w} B_i(M_j') + \sigma.a) + t \quad (2)$$

Here, the concatenation operation || is used. CSP prepares challenge reply message *chal_reply* where $chal\_reply = \{\sigma, \kappa, vk_2, \phi\}$ to PTPC.
- *SignVeri*$(\sigma, \tau, H(B_i[M_j']), t, vk_1, vk_2, g)$ → *Result* (Y/N)**:** The PTPC will confirm the accuracy of the stored data blocks through *chal_reply* message once it has been received from the CSP. From the value of ϕ, PTPC can identify the CU's file. This phase's inputs are $\{\sigma, \kappa, H(B_i[M_j']), t, vk_1, vk_2, g\}$, and its output is the auditing verification result for challenged data blocks. PTPC already has hash values of all blocks received from CU, no need to worry about the size of each block, and simply performs verification proof as

$$g^{\kappa} = vk_1^{\sigma} + vk_2^{H[\sum_{i=1}^{\tau}\sum_{j=1}^{w} B_i(M_j')]} \quad (3)$$

and t should be 0.

E. SECURITY MODEL OF PROPOSED SYSTEM

In order to withstand degenerate message attacks in the shared communication channel between CSP and PTPC, our data integrity model employs the fundamental DH model concept [9] with the criterion of $G^{\bar{q}\bar{r}} \neq 1$, where q and r are security key of CSP and PTPC consequently. The idea of a two-layer security architecture is likewise used by this suggested system model from [9]. During auditing, it withstands attacks from external adversaries. This security scheme can also withstand inner forgery assaults at RCS.This security model's brilliance is that, other from CU, all other entities are unaware of the sk_1 value. RCS stores the original block components in an encrypted version, and PTPC tests the auditing of encrypted block elements—which are likewise hidden from all adversaries, including CSP and PTPC—in an encrypted manner. As a result, modifications made to block elements cannot impact the encrypted elements' initial version. By using stub signatures, PTPC can identify colluding blocks at auditing time if malicious attacks alter any block parts which is explained in Section V-A1.CU can obtain the original form of the data that was originally recorded in RCS if it uses the data recovery scheme concept as outlined in paper [9]. According to [9]'s outer security scheme, both CSP and PTPC can quickly identify any adversary attacks utilizing σ and κ values prior to message delivery which is also explained in Section V-A3.

F. DYNAMIC OPERATIONS AT SUB-BLOCK LEVEL

Even while CU which outsources its personal data to the remote cloud does not physically own the data still wants to be able to do dynamic operations on it, particularly for data that is often modified in the practical scenario according to their desires. A hash chaining structure is used to identify the data file in order to simplify the dynamic operations of the cloud data. The hash table's chaining feature enables multiple block elements to coexist in the same block position. The block elements are still inserted into the appropriate hash table slot when collisions occur. The hash function is used to create the slot in which the elements of a block should be stored when we wish to find it. We employ a search method to determine whether the block's elements are present because each slot contains a collection. The benefit is that each slot is likely to contain significantly less components from a block on average, making the search process potentially more effective.

We developed the file structure in this research work using a Python dictionary, an associative data type that allows us to store key-data pairs. When looking for the associated data values, or block elements, the key is handled as a set of block numbers. This concept is frequently described as a map. The following defines the abstract data type for maps. The structure is a group of associations between a key and a data value that is not ordered. Because each key on a map is distinct, there is a one-to-many link between a key and many values.

We build a hash table that implements the map abstract data type using two lists. A parallel list called block value will contain the block elements in such a way that every block no. contains the dynamic size of the sub-list while a list called block slot will hold the block numbers depicted in Figure 4.

When we look up a key, the related block elements will be stored in the location that matches the block slot. 2nd list contains all sub-lists collectively. Assume that the first sub-list has a total of j_1 entries and allows spaces in the list from position 1 to j_1; the second sub-list will begin from position $(j_1 + 1)$ to j_2; and so on. Consequently, total file size $n = j_1 + j_2 + \ldots. + j_m + \ldots + j_w - 1 + j_w$. The locations of requested block elements must first be found in the sub-lists before being deleted in our dynamic operations if we wish to update multiple subblock elements. Finally, we need to add the new block's elements to all of the original block's locations. We use block slots and sub-lists to help with deletion operations by utilizing the location of the block element. When deleting a subblock's elements, if one cloud user desires to remove a block's elements, the appropriate sublist automatically resizes itself and removes the block's element. Find the place in the related sublist if a cloud user wishes to insert a block element at a specific location in the sublist. Shift the element from this location to the last position in a temporary list and add a new element at the precise location, then append all the shifted items to the sublist after the specified location. The pseudo-code of our proposed method has been shown in Algorithm 3, Algorithm 4, and Algorithm 5. For an explanation of algorithms, we used *uu* as List of Block no.s, up as Block index, and *val* as the updated Sub-Block element for Block.

Here, we assume that the hash table's load factor μ is low and that the likelihood of collisions is low. We suggested chaining in this article in terms of the dynamic size of sub-lists to prevent problems with collision resolution in sub-lists. Normal hashing would demand constant time $O(1)$for searching in the best-case scenario. $(1 + \mu/2)$ is the typical number of comparisons if the search is successful, and just μ comparisons if it failed.

We have mentioned of pseudo process of all dynamic operations in Algorithms 3, 4, and 5 accordingly. In order to adjust the sub-block components as well as the sub-block position range, CU first builds a list of the block numbers we want to change, arranging them in ascending order. This list is labeled a "uu" list and an "up" list, respectively. A separate list called a "val list for list of elements" must be prepared by the central unit (CU) for update and insert operations. CSP does not require an additional list for remove operations. CU sends lists to CSP based on operations. After receiving it, CSP scans *info_msg*, locates the mentioned block numbers of the 'uu' list, and detaches all mentioned blocks from their hash value because hash values need to be altered after performing any operation on the sub-block components of mentioned blocks. Later, CSP determines the length of each block indicated in the "uu" list. For the insert operation, CSP stores the sub-block elements of the mentioned z1 blocks from the up position to the last position in a temp list and inserts the sub-block elements from its positions using a val list after concatenating the first part of z1 with the temp list as described in *Algorithm 5*. When performing a delete operation, CSP removes z1 block's sub-block elements from the locations specified in the 'up' list described in *Algorithm 4*. For update operations, CSP uses the 'val' list described in *Algorithm 3* to update the sub-block

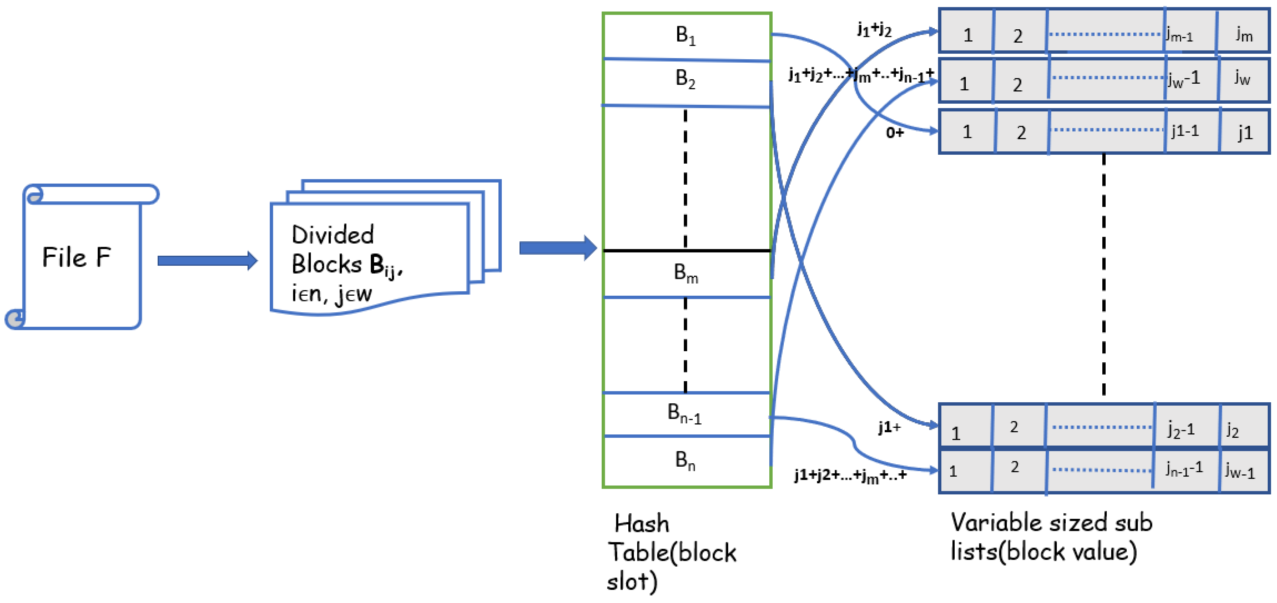

FIGURE 4. The Structure of File using Hash Table.

Algorithm 3 Process of Update Operations at Sub Block Level

```
Input: uu,up,val, [$uu \neq 0 \vee up \geq 0 \vee val$]
Output:Update Sub-Block's element   out
   Initialisation:
1:  for $i = 0$ to $n$ do
2:    if $i = uu[i]$ then
3:      $x_1 = list1[i]$
4:      $y_1 \rightarrow x_1.split(", ")$
5:      $z_1 \rightarrow y1[: -1]$
6:    else
7:      print("Block no not found")
8:    end if
9:    for $j = 0$ to $n_1$ do
10:     if $j = up[j]$ then
11:       $z_1[up] = val[0]$
12:       $val.pop(0)$
13:     else
14:       print("Sub -Block location not found")
15:     end if
16:   end for
17: end for
```

Algorithm 4 Process of Delete Operations at Sub Block Level

```
Input: uu,up, [$uu \neq 0 \vee up \geq 0$]
Output:Update Sub-Block's element   out
   Initialisation:
1:  for $i = 0$ to $n$ do
2:    if $i = uu[i]$ then
3:      $x_1 = list1[i]$
4:      $y_1 \rightarrow x_1.split(", ")$
5:      $z_1 \rightarrow y1[: -1]$
6:    else
7:      print("Block no not found")
8:    end if
9:    for $j = 0$ to $n_1$ do
10:     if $j = up[j]$ then
11:       $z_1.pop(up)$
12:     else
13:       print("Sub -Block location not found")
14:     end if
15:   end for
16: end for
```

elements of the z1 block from the places specified in 'up' list. After completing any requested operation on the CU side, CSP updates the hash value of the relevant blocks in *info_msg* along with the updated version of those blocks. Later, all new hash values are sent to CU for additional data integrity verification in accordance with the 'uu' list.

V. PERFORMANCE AND RESULTS

The suggested auditing scheme's security parameters are detailed in this section. Data confidentiality, unforgeability, collusion resistance, replica attack resistance, unambiguity, and anonymity are then introduced. Table 3 summarizes a comparison of security features which helps to understand the efficiency of the proposed model than others.

A. SECURITY ANALYSIS

We assess the proposed scheme's security by examining its fulfillment of the security guarantee given in Section IV-B: data confidentiality, auditing correctness, collision resistance, unforgeability, unambiguity, and anonymity property.

TABLE 3. Comparison table of security parameters.

Scheme	Data Confidentiality	Auditing Correctness	Collision resistance	Public Auditing	Unforgeability	Unambiguity	Anonymity
[3]	×	√	×	√	√	×	×
[8]	×	√	×	√	√	×	×
[4]	√	√	×	√	√	×	×
[10]	×	×	√	√	×	×	×
[14]	×	√	×	√	√	×	×
[15]	√	√	√	√	√	×	×
[22]	√	√	√	√	√	×	×
[23]	√	√	√	√	√	×	×
[24]	√	√	√	√	√	×	×
Our Scheme	√	√	√	√	√	√	√

Algorithm 5 Process of Insertion Operations at Sub Block Levell

```
Input: uu,up,val [$uu \neq 0 \vee up \geq 0 \vee val \neq 0$]
Output:Update Sub-Block's element   out
    Initialisation:
 1: for $i = 0$ to $n$ do
 2:   if $i = uu[i]$ then
 3:     $x_1 = list1[i]$
 4:     $y_1 \rightarrow x_1.split(", ")$
 5:     $z_1 \rightarrow y1[: -1]$
 6:   else
 7:     print("Block no not found")
 8:   end if
 9:   for $j = 0$ to $n_1$ do
10:     if $j = up[j]$ then
11:       $temp = z_1[up : n_1]$
12:       $z_1[up] = val[0]$
13:       $val.pop(0)$
14:       $z_1.append(temp)$
15:     else
16:       print("Sub -Block location not found")
17:     end if
18:   end for
19: end for
```

1) DATA CONFIDENTIALITY

Original block elements from the *chal_reply* message cannot be obtained by *PTPC* where $chal\_reply = \{\kappa, \sigma, vk_2, \phi\}$. Block data items from *info_msg* are arrived at *CSP* side and stored by *CSP*. $\sum_{i=1}^{n} \sum_{j=1}^{w} B_i[M_j'])$ are a secure version of the original blocks B that CU has encrypted through its private parameter x i.e.

$$MsgEncrypt(x, M) = \sum_{i=1}^{n} \sum_{j=1}^{w} B_i(M_j))$$
$$= (\sum_{i=1}^{n} \sum_{j=1}^{w} x \oplus B_i[M_j']))$$
$$= (\sum_{i=1}^{n} \sum_{j=1}^{w} B_i[M_j']))$$

Here, $\oplus$ is a modification operation performed by x with each and every block element. Except for CU, neither *CSP* nor *PTPC* are permitted to reveal the value of x. As a result, it is established that neither *CSP* nor any other adversaries in RCS improperly compromised data secrecy by learning the initial block parts of the CU.

2) COLLISION RESISTANCE:

The auditing verification step of the challenge response message *chal_reply* from CSP may only be passed if CSP doesn't modify the values of valid block elements $\sum_{i=1}^{t} \sum_{j=1}^{w} B_i(M_j')$ for corrupted block elements $\sum_{i=1}^{t} \sum_{j=1}^{w} B_i[\theta_j]$. Due to a lack of knowledge regarding x, CSP is unable to alter hash values. Stub σ and anonymity κ will be incorrect if CSP modifies the elements of challenged blocks and generates new hash values. Therefore, the verification's outcome will be false. This is the hash function's characteristic of collision resistance. From equation(3), we can see that

$$vk_1^{\sigma} + vk_2^{H(\sum_{i=1}^{t} \sum_{j=1}^{w} B_i[M_j'])}$$
$$\neq vk_1^{\sigma'} + vk_2^{H(\sum_{i=1}^{t} \sum_{j=1}^{w} B_i[M_j'])}$$
$$\implies g^{a*\sigma} + g^{b*H(\sum_{i=1}^{\tau} \sum_{j=1}^{w} B_i[M_j'])}$$
$$\neq g^{a*\sigma'} + g^{b*H(\sum_{i=1}^{\tau} \sum_{j=1}^{w} B_i[M_j])}$$
$$\implies g^{a*H(\sum_{i=1}^{n} \sum_{j=1}^{w} B_i[M_j']||vk_1||vk_2).a^i}$$
$$+ g^{b*H(\sum_{i=1}^{\tau} \sum_{j=1}^{w} B_i[M_j'])}$$
$$\neq g^{a*H(\sum_{i=1}^{\tau} \sum_{j=1}^{w} B_i[\theta_j]||vk_1||vk_2).a^i}$$
$$+ g^{b*H(\sum_{i=1}^{n} \sum_{j=1}^{w} B_i[M_j'])}$$

Hence, $L.H.S \neq R.H.S$ Our suggested strategy can thereby prevent collisions.

3) UNFORGEABILITY

Only the intactness of all challenged block components is proven by PTPC's verification test if CSP properly saves all block elements of CU's outsourced data(We need to demonstrate that a malicious authorized attacker cannot falsify proof (σ, κ) to deceive the PTPC, and a malicious CSP cannot falsify outsourced data for the verification test.).

A hostile authorized attacker cannot fabricate evidence (σ, κ) by tricking CSP since every time the PTPC delivers a challenge to CSP as *chal_pro* message, τ is included in the *chal_pro* message. The CSP will generate evidence (σ, κ) depending on τ from the PTPC, as shown in Equation (1&2), where τ varies by data shard.

Even if we suppose that the malicious CSP forges (σ', κ') from the prior proof, namely $(\sigma, \kappa) \neq (\sigma', \kappa')$, the PTPC verification process in Equation (3) demonstrates that the PTPC must validate τ with *req_gen* message, which also contains tau value. As a result, the (σ', κ') cannot hold Equation (3).

$$\sigma' = H[\sum_{i=1}^{t}\sum_{j=1}^{w} B_i[\theta_j]||vk_1||vk_2)].a^t$$

$$\kappa' = (b.H(\sum_{i=1}^{t}\sum_{j=1}^{w} B_i(\theta_j) + \sigma.a) + t$$

Another scenario is when an untrustworthy CSP attempts to trick the verifier by swapping out the disputed data block$\sum_{i=1}^{t}\sum_{j=1}^{w} B_i[\theta_j]$ with another data block $\sum_{i=1}^{n}\sum_{j=1}^{w} B_i[M_j']$ and preparing hash values for collude blocks when the former data blocks are broken.

So, Equation (3) can be represented as:

$$g^{\kappa'} = vk_1^{\sigma'} + vk_2^{H[\sum_{i=1}^{\tau}\sum_{j=1}^{w} B_i(M_j')]}$$

As a result, $H(\sum_{i=1}^{n}\sum_{j=1}^{w} B_i[M_j'] = H(\sum_{i=1}^{n}\sum_{j=1}^{w} B_i[\theta_j]$. However, due to the hash function's anti-collision characteristic, $H(\sum_{i=1}^{n}\sum_{j=1}^{w} B_i[\theta_j]$ cannot be equal to $H(\sum_{i=1}^{n}\sum_{j=1}^{w} B_i[M_j']$. As a result, making Equation (6) hold is impossible, and The verification process cannot accept the proof from CSP.

4) UNAMBIGUITY:

An internal adversary cannot show itself to be a genuine CSP even if it knows information about vk_2 and σ of a genuine CSP(We must show that an enemy never confirms themselves as a genuine identity due to a lack of information about CU's sk_1, CSP's sk_2 and prepares the incorrect κ').

Assume an internal adversary I_{vk_1} is there, notices CSP's stub and anonymity, and discovers his victory. He then makes the decision to assert the winning offer as his personal by sending in both his personal public verification key vk_2 and the deanonymize k' made up of the CSP's stub of challenged block elements under vk_2', which he can compute because he already knows the public key vk_2 and stub σ. However, due to a lack of information on sk_1 of CU, I_{vk_1} prepares κ' using equation(2) as follows:

$$\kappa' = (b'.H(\sum_{i=1}^{t}\sum_{j=1}^{w} B_i[M_j'] + \sigma.a') + t$$

According to equations (1),(2), and (3), PTPC verifies the auditing test as follows:

$$g^{\kappa'} = vk_1^{\sigma} + vk_2^{H(\sum_{i=1}^{t}\sum_{j=1}^{w} B_i[M_j'])}$$

From *req_gen* message, PTPC can calculate the value of a where *req_gen* $= \{H(B_i[M_j'], g, n, vk_1, \phi\}$. Now, if we elaborate equation(8) as follows:

$$\begin{aligned} g^{\kappa'} &= vk_1^{\sigma} + vk_2^{H(\sum_{i=1}^{t}\sum_{j=1}^{w} B_i[M_j'])} \\ &\implies g^{b'.H(\sum_{i=1}^{t}\sum_{j=1}^{w} B_i[M_j'] + \sigma.a') + t} \\ &= g^{a*H[\sum_{i=1}^{t}\sum_{j=1}^{w} B_i[M_j']||vk_1||vk_2).a^i} \\ &\quad + g^{b*H(\sum_{i=1}^{t}\sum_{j=1}^{w} B_i[M_j'])} \\ &\implies g^{b'.H(\sum_{i=1}^{t}\sum_{j=1}^{w} B_i[M_j']} \\ &\quad + a'.H[\sum_{i=1}^{n}\sum_{j=1}^{w} B_i[M_j']||vk_1||vk_2).a^i + t \\ &= g^{a*H[\sum_{i=1}^{t}\sum_{j=1}^{w} B_i[M_j']||vk_1||vk_2).a^i} \\ &\quad + g^{b*H(\sum_{i=1}^{t}\sum_{j=1}^{w} B_i[M_j'])} \end{aligned}$$

if $b' \neq b$ and $a' \neq a$, then t never would be 0 and adversary I_A violates the auditing criteria of PTPC.

5) ANONYMITY

The stub σ and anonymity κ cannot be used to identify the stub-creator. To further explain, we assume that the external and internal adversaries have little bit priori knowledge regarding the signer's prospective verification key, such as that it belongs to some prime set Z of keys of p order, where identities are connected to verification keys like vk_1 and vk_2 for CU and *CSP*. This set, in the worst situation, contains enormous verification keys. The adversary has little advantage over guessing when determining which one among all keys the stub was produced given knowledge of these keys, a stub created under one of them, and also the message.

B. PERFORMANCE ANALYSIS

Finding out the computational cost of signature generation with block numbers, signature affirmation with block numbers, and block size(kilobytes(KB)) is the primary goal of this analysis part. Here, comparative simulation analysis of dynamic data processes was also examined. These are the experimental settings: The processor is an Intel(R) Core(TM) i5-9300H, the RAM is 12 GB, and Windows 10 is running on it. Simulation is done in Google Colab.

1) RESEARCH SCENARIO

As per the research requirements, any research needs a research scenario to propose the research model. So, this research focuses on the given problem statement in terms of cloud storage security. The proposed research focuses on given input research parameters as per the proposed machine learning model requirements such as *Block_no*,*Sub_Block_no*, Time, etc. It analyzes the performance of the proposed model, research requires some data sets. The sample of data sets is described in Table 4 and Table 5.

TABLE 4. Sample values for signature generation and signature verification.

$Block\_no$	Time for Signature Generation	Time for Signature Verification
100	0.1305	.66164)
150	0.1705)	.9623
200	0.2128	1.2539
250	0.2567	1.567
300	0.2947	1.8548
350	0.3487	2.154
400	0.4026	2.4633)

TABLE 5. Sample values for all Dynamic operation.

$Block\_no$	Sub Block no	Time for Insert Operation	Time for Update Operation	Time for Delete Operation
50	50	0.012	0.008	0.0129
	100	0.0209	0.0349	0.0199
	200	0.0277	0.0447	0.0369

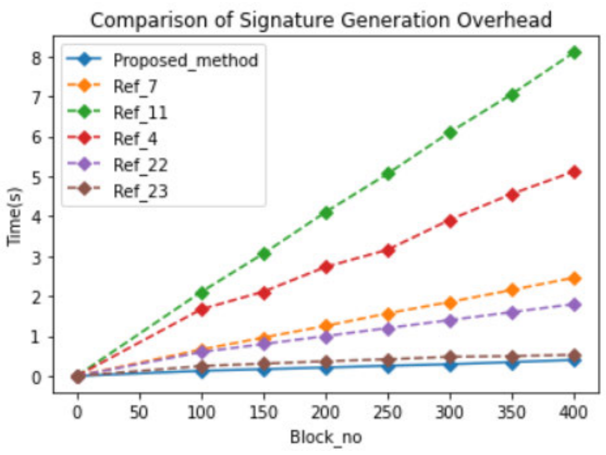

FIGURE 5. Signature Generation Overhead with Block no.

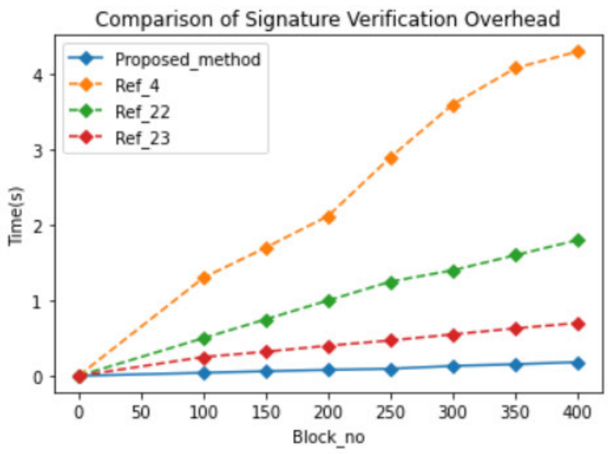

FIGURE 6. Signature Verification Overhead with Block no.

2) COMPUTATIONAL OVERHEAD

This analysis part mainly focuses on determining the computational cost of signature creation time with multiple block numbers, signature verification time with multiple block numbers. A comparison of three aspects of the suggested method with earlier research [4], [8], [13], [14], [22], [23], [24] was later briefly described, as well as evidence of the suggested work's effectiveness. Figure 5 displays the computational cost of signature generation time with various data blocks 100,150,200,250,300,350,400 for the proposed scheme and the research papers [4], [8], [13], [23], [24].
In terms of computation complexity, we compare our partial signature-based auditing scheme to the BLS signature-based scheme [13], the ZSS signature-based scheme [4], [22], [23], and the algebraic signature-based scheme [4], [24] in terms of computational overhead for signature generation and verification, signature's type, and the size of the signature, among other things. Table 4 is defined to show the meaning of all crypto operations in order to establish unambiguous comparisons.
Our technique is more efficient since it does not use pairing operations like the other signatures stated in [4], [8], [22], and [23]. Furthermore, unlike the BLS signature, a partial signature does not require a particular hash function. It can make use of a widely used SHA family hash function, like the ZSS signature.

Pairing process and exponential process are typically regarded as high-cost procedures. It is already proven in the article [32]that the cost of an exponential operation is smaller than the cost of a pairing operation (for both curved and non-curved). Regrettably, the attribute-based signature demanded more exponential operations than alternative approaches.

In papers [4], [8], [10], [13], [22], [23], cloud users have the responsibility to prepare signature for all blocks. For all the above-mentioned research papers, a file F is broken up into n no. of blocks of varying sizes where $B = \{B1, B2, \ldots, Bn\}$ and the ZSS signature, BLS signature, and identity-based signature are used at each block level. In the sign generation phase, scheme [4] spends n(1Hash + 1Mul + 2Exp)for the signature generation, scheme [8] spends nHash+ nMul+ nInv for the signature generation, scheme [22] spends n(1Exp + 1Inv + 1Mul+1Add + 3Hash) for the

TABLE 6. Notations in computational overhead.

Notation of computational operation	Meaning
$Pair$	Bilinear pairing operation
Mul	Multiplication Operation
Inv	Inverse operation
$Hash$	Hash operation
Exp	Exponential operation
$Encrpt$	Symmetric encryption operation
Add	Addition operation

signature generation, scheme [23] spends nHash+ nMul+ 2 nAdd + n Inv for the signature generation, scheme [24] spends n(2Exp+1Mul+1Hash) for the signature generation respectively. In Table 7, a comparison of the computational overhead of existing research works with proposed research work is depicted. Unlike other mentioned schemes, CSP is responsible for preparing stub signatures of challenged blocks in our proposed scheme, because our proposed signature scheme prepares signatures for challenged blocks for which a request is made from the PTPC side, whereas all other mentioned schemes prepare signatures for all blocks. As a result of this approach, our suggested signature method has less computational overhead than previous signature-based schemes in general. We further assume that CSP necessitates (4+t) addition operations, 3 multiplication operations,1 hash operation, and 1 exponential operation. Because there is no original version, it enhances the level of data privacy and security in RCS for outsourced *data*. As a result, the computational cost of the CSP to prepare a stub signature as a challenge answer for a public auditing test is $(4 + t)Add + 3Mul + 1Exp + 1Hash$. Bilinear pairing is required for BLS, ZSS, and algebraic signature-based data auditing. A research paper [4] requires four pairing operations, a paper [8], [13] requires two pairing operations, a paper [22] requires n pairing operations, and a data auditing verification technique [23]requires one pairing operation. Because of the partial signature notion, our suggested data auditing approach does not involve any pairing operations. Reference [24] requires (t+1) Exp operation, but our proposed approach requires only one Exp operation.

Figure 5 depicts the computational cost of signature creation time for the proposed approach and the research publications [4], [8], [13], [23], [24] with various data blocks 100,150,200,250,300,350,400. The average computing time of the suggested technique is 0.2595 seconds, whereas the average sign generation time in [4], [8], [13], [23], and [24] research works was 1.5595 5.0842 3.3228,1.00,.4085 seconds respectively. The average computation times for sign production with three system models differ by 1.30 seconds, 4.824 seconds, 3.063 seconds, 0.7405 seconds, and 0.1490 seconds, respectively.

3) COMMUNICATIONAL OVERHEAD

The research paper [4] shows BLS signature-based data auditing and [8] shows data auditing verification based on ZSS signature. Both ZSS and BLS signature performs bilinear pairing operations. The BLS signature process, which has an additional communication cost of around 960 bits, provides the basis for the scheme [4] and the ZSS signature mechanism which is about 720 bits [8]. Our proposed signature is a Schnorr signature and it has no overhead when compared to the base entropy scheme. Based on the Schnorr protocol [28], we create a modified partial signature method for a 240-bit full signature using an 80-bit stub and a 160-bit de-anonymizer.

In the article [32], it is already proved that the cost of exponential operation is less than the cost of pairing operation (for both curved and without curved). It has already been demonstrated in a research study [8] that as a signature technique based on the ZSS requires less time to sign documents than a BLS's signature scheme as the number of data blocks increases.

The research paper [13], [24] shows data integrity verification based on BLS signature [8], [22], [23] shows data auditing verification based on ZSS signature and [4] shows data auditing verification based on algebraic signature. Both ZSS and BLS signature performs bilinear pairing operations. The extra communication overhead of the scheme is based on the BLS signature mechanism which is about 960 bits [4] and the ZSS signature mechanism which is about 720 bits [8]. Our proposed signature is a Schnorr signature and it has no overhead when compared to the base entropy scheme. Based on the Schnorr protocol [28], we prepare a modified partial signature scheme with an 80-bit stub and a 160-bit de-anonymizer for a 240-bit full signature. It has already been demonstrated in a research study [8] that as the number of data blocks rises, the signature time required for a signature scheme based on the ZSS is shorter than that required for a signature scheme based on the BLS. The extra communication overhead caused by the CU in the proposed system based on the partial signature technique is mostly the *info_message* value uploaded to the CSP, which is approximately 80 bits. The extra communication cost for the proof of *chal_pro* message is approximately 240 bits, which is forwarded by the CSP to the PTPC as *chal_reply* message.As a result, the additional communication overhead generated by the proposed scheme is around 320 bits, which is less than the communication overhead required by the ZSS and BLS-based signature mechanisms. Here, we consider three scenarios like 1st is CU to CSP communication, 2nd is CU to PTPC communication, and 3rd is PTPC to CSP and CSP to PTPC communication which depict in Table 8. Like the other methods [22], [23], [24], [25], CU sends a file to CSP to store their data in cloud storage, and simultaneously CU hires PTPC as an auditor to check their outsourced's data integrity, where PTPC prepares a challenge message and sends it to the CSP and PTPC then validates the proof that the CSP sent after it produces evidence of verification and transmits it to PTPC. First communication overhead is O(n) where n is the total blocks requested to store in CSP like [22], [23], and [24]. 2nd communication overhead is O(1) like

TABLE 7. Comparison of computational overhead.

Reference	Cloud UserKind of Signature	Size of Signature(bits	Overhead of Signature preparation	Overhead of Signature Verification
[4]	Algebric	Na	$n(Hash + Mul + 2Exp)$	$4Pair + 2Mul + 2(t-1)Add + 2Exp + (t+1)Exp + (t+1)Mul + tHash$
[8]	ZSS	720	$nHash + nMul + +nInv$	$Mul + 2Pair + Add$
[13]	BLS	960	$n(Hash + Mul_2Exp)$	$tMulExp + t2Hash + 3Exp + 2Pair + 2Mul$
[22]	ZSS	260	$n(1Exp+1Inv+1Mul+1Add+3Hash)$	$n(1Add + 2Hash + Pair)$
[23]	ZSS	480	$nHash+nMul+2nAdd+nInv$	$1Pair + 2Mul(2tAdd + 1) + 2Add$
[24]	Algebric	NA	$n(2Exp + 1Mul + 1Hash)$	$(t+1)Exp + 1Mul$
Proposed Method	Partial	240	$(t+4)Add + 3Mul + 1Hash + 1Exp$	$3Exp + (t+1)Add$

TABLE 8. Comparison of communication overhead.

Reference	CU to CSP	CU to PTPC	CSP to PTPC
[22]	O(n)	O(1)	O(n)
[23]	O(n)	O(1)	O(t)
[24]	O(n)	O(1)	O(t)
[25]	zero	O(n)	O(t)
Our's	O(n)	O(1)	O(t)

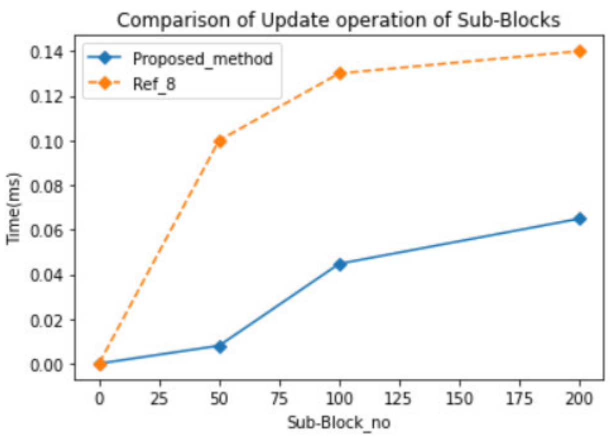

FIGURE 7. Comparison of Update operation with Block no.

[22], [23], and [24] and 3rd communication overhead claims $O(t)$ for stub signature of t no. of challenged blocks like [23], [24], and [25]. In [25], there is not a single communication happening between the user and CSP.

4) SIMULATION OVERHEAD OF DYNAMIC DATA OPERATIONS

In a manner similar to [10], we carry out each operation three times in accordance with the various data blocks.50 data blocks are available. We simulate the three separate data block operations with varying numbers of sub-blocks 50, 100, and 200, respectively, in order to show the regular pattern of the time cost in Figure 7, Figure 8, and Figure 9 respectively. which displays the computational time required for the data sub-blocks update, delete and, insertion simulation.

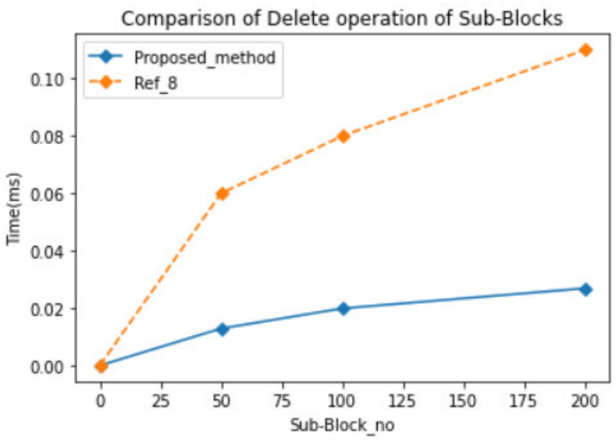

FIGURE 8. Comparison of Delete operation with Block no.

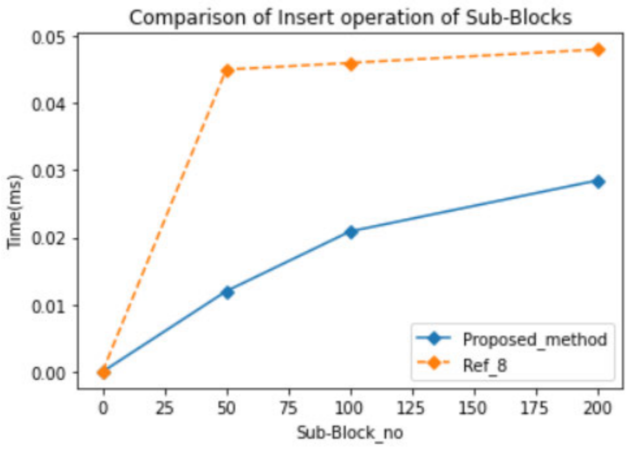

FIGURE 9. Comparison of Insert operation with Block no.

We have already mentioned of pseudo process of all dynamic operations in Algorithm 1, 2 and 3 accordingly.

We have considered here 50 blocks and for this we have taken the simulation result of 50 sub-blocks, 100 sub-blocks and 200 sub-blocks respectively for all dynamic operations. From this figure, we can see that every time a sub-block update, delete and, insertion time will increase for the growing no. of sub-blocks. Our proposed model takes 0.039ms for update operation,0.019ms for delete operation, and 0.020 ms for insertion operation while [10] takes 0.092 ms for update operation,0.073 ms for delete operation, and 0.046 ms for insertion operation which are less than [10] averagely.

VI. CONCLUSION AND FUTURE WORKS

Dynamic operations are accomplished to protect the integrity of outsourced data on shared cloud storage by auditing correctness at the block level. Consistency of outsourced data for CU at RCS is ensured through the use of a partial signature and original data encryption technology using random number generation. It aids in thwarting replace and forgeability attacks. Additionally, PTPC is capable of detecting block corruption during auditing and notifying CU. Additionally, this user-friendly auditing architecture protects the privacy of CU's data from all entities, including CSP and PTPC, while conducting the audit. Additionally, the dynamic data update operation is carried out while maintaining public auditing on the same with reduced computational cost and has already been shown through the aforementioned experimental results to be a better solution than the current options. This research article does not consider the security of message communication between CU and PTPC. CU also does not have its algorithm to generate an original version of source data if data has been colluded by some malicious attackers. Hence, our future work in this research article is to expand this auditing model for CU-side audit message verification and error localization.

REFERENCES

[1] P. Mell and T. Grance. (2009). *Draft Nist Working Definition of Cloud Computing*. [Online]. Available: http://csrc.nist.gov/groups

[2] K. Ren, C. Wang, and Q. Wang, "Security challenges for the public cloud," *IEEE Internet Comput.*, vol. 16, no. 1, pp. 69–73, Jan. 2012.

[3] B. Shao, G. Bian, Y. Wang, S. Su, and C. Guo, "Dynamic data integrity auditing method supporting privacy protection in vehicular cloud environment," *IEEE Access*, vol. 6, pp. 43785–43797, 2018.

[4] W. Shen, J. Qin, J. Yu, R. Hao, and J. Hu, "Enabling identity-based integrity auditing and data sharing with sensitive information hiding for secure cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 2, pp. 331–346, Feb. 2019.

[5] S. Debnath, B. Bhuyan, and A. K. Saha, "Privacy preserved secured outsourced cloud data access control scheme with efficient multi-authority attribute based signcryption," *Multiagent Grid Syst.*, vol. 16, no. 4, pp. 409–432, Dec. 2020.

[6] Y. Zhang, C. Xu, H. Li, and X. Liang, "Cryptographic public verification of data integrity for cloud storage systems," *IEEE Cloud Comput.*, vol. 3, no. 5, pp. 44–52, Sep. 2016.

[7] F. Zhang, R. Safavi-Naini, and W. Susilo, "An efficient signature scheme from bilinear pairings and its applications," in *Public Key Cryptography—PKC 2004: 7th International Workshop on Theory and Practice in Public Key Cryptography, Singapore, March 1–4, 2004. Proceedings 7*. Berlin, Germany: Springer, 2004.

[8] H. Zhu, Y. Yuan, Y. Chen, Y. Zha, W. Xi, B. Jia, and Y. Xin, "A secure and efficient data integrity verification scheme for cloud-IoT based on short signature," *IEEE Access*, vol. 7, pp. 90036–90044, 2019.

[9] P. Goswami, N. Faujdar, S. Debnath, A. K. Khan, and G. Singh, "ZSS signature-based audit message verification process for cloud data integrity," *IEEE Access*, vol. 11, pp. 145485–145502, 2023.

[10] J. Shen, D. Liu, D. He, X. Huang, and Y. Xiang, "Algebraic signatures-based data integrity auditing for efficient data dynamics in cloud computing," *IEEE Trans. Sustain. Comput.*, vol. 5, no. 2, pp. 161–173, Apr. 2020.

[11] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," *J. Cryptol.*, vol. 17, no. 4, pp. 297–319, Sep. 2004.

[12] M. Tian, Y. Zhang, Y. Zhu, L. Wang, and Y. Xiang, "DIVRS: Data integrity verification based on ring signature in cloud storage," *Comput. Secur.*, vol. 124, Jan. 2023, Art. no. 103002.

[13] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for secure cloud storage," *IEEE Trans. Comput.*, vol. 62, no. 2, pp. 362–375, Feb. 2013.

[14] W. Shen, J. Qin, J. Yu, R. Hao, J. Hu, and J. Ma, "Data integrity auditing without private key storage for secure cloud storage," *IEEE Trans. Cloud Comput.*, vol. 9, no. 4, pp. 1408–1421, Oct. 2021.

[15] M. Thangavel and P. Varalakshmi, "Enabling ternary hash tree based integrity verification for secure cloud data storage," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 12, pp. 2351–2362, Dec. 2020.

[16] A. Bender, J. Katz, and R. Morselli, "Ring signatures: Stronger definitions, and constructions without random oracles," in *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4–7, 2006. Proceedings 3*. Berlin, Germany: Springer, 2006.

[17] H. Wang, D. He, and S. Tang, "Identity-based proxy-oriented data uploading and remote data integrity checking in public cloud," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 6, pp. 1165–1176, Jun. 2016.

[18] C. Ge, W. Susilo, J. Baek, Z. Liu, J. Xia, and L. Fang, "Revocable attribute-based encryption with data integrity in clouds," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 5, pp. 2864–2872, Sep. 2022.

[19] S. Singhal and A. Sharma, "A job scheduling algorithm based on rock hyrax optimization in cloud computing," *Computing*, vol. 103, no. 9, pp. 2115–2142, Sep. 2021.

[20] S. Singhal and D. Mangal, "Mutative ABC based load balancing in cloud environment," in *Futuristic Trends in Network and Communication Technologies: Third International Conference, FTNCT 2020, Taganrog, Russia, October 14–16, 2020, Revised Selected Papers, Part I 3*. Singapore: Springer, 2021.

[21] S. Singhal and A. Sharma, "Mutative BFO-based scheduling algorithm for cloud environment," in *Proc. Int. Conf. Commun. Artif. Intell. (ICCAI)*. Singapore: Springer, 2021, pp. 589–599.

[22] E. N. Witanto and S.-G. Lee, "Cloud storage data verification using signcryption scheme," *Appl. Sci.*, vol. 12, no. 17, p. 8602, Aug. 2022.

[23] Y. Bai, Z. Zhou, X. Luo, X. Wang, F. Liu, and Y. Xu, "A cloud data integrity verification scheme based on blockchain," in *Proc. IEEE SmartWorld, Ubiquitous Intell. Comput., Adv. Trusted Comput., Scalable Comput. Commun., Internet People Smart City Innov. (SmartWorld/SCALCOM/UIC/ATC/IOP/SCI)*, 2021, pp. 357–363.

[24] X.-P. Zhao and R. Jiang, "Distributed machine learning oriented data integrity verification scheme in cloud computing environment," *IEEE Access*, vol. 8, pp. 26372–26384, 2020.

[25] A. Kc and B. Muniyal, "Dynamic list based data integrity verification in cloud environment," *J. Cyber Secur. Mobility*, vol. 11, pp. 433–460, Jul. 2022.

[26] Y. Miao, Y. Yang, X. Li, Z. Liu, H. Li, K. R. Choo, and R. H. Deng, "Efficient privacy-preserving spatial range query over outsourced encrypted data," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 3921–3933, 2023.

[27] Y. Miao, Y. Yang, X. Li, L. Wei, Z. Liu, and R. H. Deng, "Efficient privacy-preserving spatial data query in cloud computing," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 1, pp. 122–136, Jan. 2024.

[28] M. Bellare and S. Duan, "Partial signatures and their applications," *Cryptol. ePrint Arch.*, 2009.

[29] S. K. Pasupuleti, "Privacy-preserving public auditing and data dynamics for secure cloud storage based on exact regenerated code," in *Research Anthology on Privatizing and Securing Data*. Hershey, PA, USA: IGI Global, 2021, pp. 1003–1022.

[30] P. Wei, D. Wang, Y. Zhao, S. K. S. Tyagi, and N. Kumar, "Blockchain data-based cloud data integrity protection mechanism," *Future Gener. Comput. Syst.*, vol. 102, pp. 902–911, Jan. 2020.

[31] A. Bhalerao and A. Pawar, "A survey: On data deduplication for efficiently utilizing cloud storage for big data backups," in *Proc. Int. Conf. Trends Electron. Informat. (ICEI)*, May 2017, pp. 933–938.

[32] F. Benhamouda, G. Couteau, D. Pointcheval, and H. Wee, "Implicit zero-knowledge arguments and applications to the malicious setting," in *Proc. Annu. Cryptol. Conf.* Berlin, Germany: Springer, 2015, pp. 107–129.

PAROMITA GOSWAMI received the MCA degree from the University of North Bengal, West Bengal, in 2014, and the M.Tech. degree in multimedia and software systems from NITTTR, Kolkata, in 2016. She is currently an Assistant Professor with the Department of Computer Engineering and Application, GLA University, Mathura, India. She is a Research Scholar with Mizoram University, Aizwal, India. Her research interests include cloud computing, big data, and information security.

NEETU FAUJDAR received the B.E. degree in information technology from VTU University, Karnataka, in 2011, the M.Tech. degree in computer science engineering from Invertis University, Bareilly, in 2013, and the Ph.D. degree in computer science engineering from the Jaypee University of Information Technology, Solan, India, in 2017. She is currently an Assistant Professor with the Department of Computer Engineering and Application, GLA University, Mathura, India. She has published more than 50 research articles. Her research interests include HPCA, GPU computing, the IoT, networking, and cloud computing. She is having memberships in professional societies, such as the International Association of Engineers (IAENG) and the Institute of Research Engineers and Doctors (IRED).

GHANSHYAM SINGH received the Ph.D. degree in electronics engineering from Indian Institute of Technology, Banaras Hindu University, Varanasi, India, in 2000. He is currently a Full Professor and the Director of the Centre for Smart Information and Communication Systems, Department of Electrical and Electronic Engineering Sciences, University of Johannesburg, Auckland Park Kingsway Campus, South Africa. Earlier, he was associated with the Central Electronics Engineering Research Institute, Pilani, and the Institute for Plasma Research, Gandhinagar, India, where he was a Research Scientist. He was a Visiting Researcher with Seoul National University, Seoul, South Korea. He was a Professor with the Department of Electronics and Communication Engineering, Jaypee University of Information Technology, Waknaghat, Solan, India. He has more than 22 years of teaching and research experience in academia and research and development institutions. He is the author/coauthor of more than 290 scientific research papers of the refereed journal and international conferences. He is the author of several books and book chapters published by Springer, Elsevier, IET, Wiley, and CRC. He has executed several sponsored research projects of ISRO and DRDO and currently involved on 5G resource allocation sponsored by SENTECH. His research and teaching interests include millimeter/THz wave technologies and their applications in communication and imaging and next-generation communication systems (5G/6G)/cognitive radio/NOMA. He has supervised 15 Ph.D. and 42 M.Tech. theses (awarded). He is an editor/the guest editor of several peer-reviewed journals and delivered keynote and plenary talks in international conferences.

KANTA PRASAD SHARMA (Member, IEEE) has published research manuscripts with SCI and SCOPUS indexing international journals and conference proceedings. His research interests include navigation and communication, wireless sensor networks, and finger control technology and deep learning with image processing. He is a member of ACM and the Computer Society of India. He is a member of the reviewer and editorial board of more than 22 international and national journals and a TPC member of various international conferences. He received the best research paper award at an international platform. He has successfully edited many books as an Editor of the *Smart IoT for Research and Industry*, *EAI/Springer Innovations in Communication and Computing* Series (Springer Nature), and *Evolving Networking Technologies: Developments and Future Directions* (Wiley), and has published various special issues on SCI and Scopus indexing journal as the guest editor in multimedia under Springer.

AJOY KUMAR KHAN received the B.Tech. and M.Tech. degrees in computer science and engineering from Calcutta University, and the Ph.D. degree in information technology from Assam University (a Central University). He is currently a Professor and the Head of the Computer Engineering Department, Mizoram University. He has more than 15 years of teaching experience in the field of computer science and engineering. He has published more than 70 research articles. He has supervised several M.Tech. theses and three Ph.D. scholars and currently guiding ten Ph.D. scholars in the field of security and privacy. He organized several FDP/workshops in national level and also delivered invited lectures more than 30 in national workshop/FDP/seminar. His research interests include applied cryptography and information security, such as smart cards security, cyber security, digital forensics, the IoT security, cloud data security, big data security, neural cryptography, quantum cryptography, and blockchain technology.

SOMEN DEBNATH received the B.E. degree in computer science and engineering from NIT Agartala, the M.Tech. degree in computer science and engineering from Tripura University respectively, and Ph.D. degree from North Eastern Hill University, Shillong, with a focus on cloud data security. He has been an Associate Professor with the Department of Computer Science and Engineering, Tripura University (a Central University under MHRD), since 2023. He has more than 13 years of teaching experience in the field of computer science and engineering, including NIT Manipur, Mizoram University (a Central University), and Tripura University (a Central University). He has published more the 30 papers in journals and conference proceedings and book chapters indexed by SCI/Scopus. He filled two patents and PI/CoPI of four ongoing research projects funded by DST and UGC. He has supervised several M.Tech. theses and currently guiding four Ph.D. scholars in the field of security and privacy. He organized several FDP/ workshops at national level and also delivered invited lectures of more than 20 in national workshop/FDP/seminar. His research interests include cloud data outsourcing, access control, cryptography, and information security. He is an Associate Member of the Institute of Engineers and a Life Member of CSI and ISTE.