IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 17, 2022, accepted March 1, 2022, date of publication March 14, 2022, date of current version March 25, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3159699

Towards Lightweight Provable Data Possession for Cloud Storage Using Indistinguishability Obfuscation

SMITA CHAUDHARI[1,2] **AND GANDHARBA SWAIN**[1]

[1]Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur, Andhra Pradesh 522502, India
[2]Dr. D.Y. Patil Institute of Technology, Pimpri, Pune 411018, India

Corresponding author: Smita Chaudhari (smita.m.c@gmail.com)

ABSTRACT Cloud Computing has proved to be a boon for many individuals and organizations who cannot afford infrastructure and maintenance cost of resources. But the untrusted nature of Cloud Server (CS) brings many challenges related to security and trust. Public auditing is one process that enables users to delegate the integrity verification of outsourced data to external party such as Third-Party Auditor (TPA). Provable Data Possession (PDP) is one approach of auditing that can verify the integrity using cryptographic algorithms. Many PDP schemes are based on bilinear pairing and homomorphic authenticators that involves complex computations that leads to increased verification time. The lightweight auditing processes is a need today using modern cryptographic techniques. Indistinguishability Obfuscation (IO) is one of the modern but weaker primitive that, if used with one-way functions, provide multiple cryptographic constructs. Sahai and Waters proposed construction of cryptographic constructs using IO. Zhang *et al.* proposed lightweight public auditing scheme using these IO. But still there are many goals to achieve such as group support, collusion handling, privacy-preserving etc. using IO. In this paper, we are trying to explore these issues and lists future research directions in this field.

INDEX TERMS Public auditing, provable data possession, homomorphic verifiable tags, indistinguishability obfuscation.

I. INTRODUCTION

Evolution in the field of computer technology and Internet-based services open up the era of Cloud Computing. High-speed processors with improved service computing architecture led to the development of large data centers. Flexible and increased network bandwidth enables a client to access or subscribe to data and services from remote data centers. This improved computing architecture helps many organizations or individuals to expand their businesses without any investment in infrastructure and maintenance costs. This paradigm mainly provides services in computing and storage. Multiple computing resources are easily assigned and released to cloud users based on their needs. Cloud datastore is one of the widely used services among cloud users. Cloud users can easily store and share information on cloud storage. Although it provides a promising platform, this paradigm has brought many security and performance issues with it. The main concern is that of integrity verification of outsourced data at remote centers since user is not having any physical control over it. Cloud Server (CS) may delete infrequent data to create space for other information. CS may also hide accidental data loss from users due to disk crashes, natural disasters or any other reason to maintain reputation. Because of mass outsourced data and resource capacity constraints on user side, it is not efficient for cloud user to conduct periodic integrity checks [1].

Multiple approaches are suggested to address this issue. Evidence generated through different logs is used to solve disputes between cloud user and CS [2]. But Log-based auditing faces many challenges such as: increased growth in log size, security, integrity, and privacy of logs [3]. Traditional cryptographic methods cannot be used as it is in this paradigm. Alternatively, CS needs to assure cloud users that their data is intact and can be accessible at any time. Auditing is a process that offers clear and recognizable footprints of

The associate editor coordinating the review of this manuscript and approving it for publication was Mingjun Dai.

resource access for various activities. Traditional IT audits are mostly conducted in two ways: internal and external. The internal audit generally concentrates only on certain organizational processes, their optimization, and risk management. The external audit focuses on an organization's ability to satisfy the constraints of various laws and regulations. But when an organization's data is outsourced on the cloud, it discloses cloud-specific security concerns. Ryoo *et al.* [4] highlight the challenges of cloud security auditing from traditional IT security audit practices. According to Hsien *et al.* [5], the role of verifiers in cloud security audit is mainly categorized as private auditability and public auditability. In private auditing, cloud users or organizations directly verify data in cloud storage whereas, in public auditing, the verification task is delegated to an external verifier such as a Third-party Auditor (TPA) who can check outsourced data on behalf of cloud user. TPA uses cryptographic algorithms to check the integrity of outsourced data. Wang *et al.* [1] presented a framework for describing this problem and suggested a set of cryptographically desirable properties for public auditing services. Hsien *et al.* [5] surveyed basic requirements and evaluation metrics to analyze security and efficiency of the public auditing system. According to Hsien et al, evaluation parameters for public auditing systems are classified into two categories: Security evaluation and Performance Evaluation. Parameters for security evaluation include block-less verification, batch auditing, dynamic data support, and privacy-preserving. Performance evaluation parameters are computing cost and Storage cost. Kollar *et al.* [6] summarized a state of different methodologies for cloud security audit and focused on research issues. Methodologies include message authentication code, homomorphic authenticators, BLS signatures, etc. Most of the auditing schemes based on these cryptographic primitives involve complex computations cost that increases verification time. There is a need to propose lightweight auditing schemes using modern cryptographic techniques. This paper concentrate on existing challenges to achieve lightweight auditing schemes using a modern cryptographic technique: Indistinguishability Obfuscation (IO).

The organization of the remaining sections of this paper is as described below: Section 2 describes the general strategies or approaches used by researchers for integrity verification of cloud data. Section 3 discusses the architecture and design goals of public auditing system. Section 4 elaborates on pros and cons of different cryptographic techniques used for public auditing system. Section 5 discusses the use of modern cryptographic technique such as Indistinguishability Obfuscation (IO) for public auditing. Section 6 addresses the future research challenges to achieve lightweight PDP using IO for public auditing.

II. GENERAL STRATEGIES OF INTEGRITY VERIFICATION FOR CLOUD STORAGE

Liu and Chen [7] listed three approaches for information auditing: strategic-oriented, Process-oriented, and resource-oriented. The process-oriented approach focuses on activities the system takes to achieve expected outcomes. This approach provides sufficient evidence for both digital forensics and reputation estimation for CS. More practical strategies for cloud data integrity are as follows:

A. SENTINEL EMBEDDING

In this strategy, cloud user produces sentinels to secure its own data at CS. Generally, sentinels are One-Way Functions (OWF). Predefined number of sentinels are appended to the file. Cloud user can check the integrity of outsourced data by giving challenges to CS by checking the fixed number of sentinels. Since CS is not aware about the location of sentinels, it can't modify the data. Juels and Kaliski [8] proposed a new cryptographic building block based on sentinels known as Proof of Retrievability (PoR). But it faces the problem of a restricted number of challenges since the number of sentinels is predefined. To address this problem, Schaum and Waters [9] proposed a new Compact PoR construct which suggests private verifiability based on Pseudo-Random Functions (PRF) and public verifiability using short signature based on Boneh, Lynn, and Shacham (BLS). Most of the verification schemes proposed using PoR are based on public-key cryptosystem which increases the computational burden on users. Guan *et al.* [10] proposed symmetric-key-based PoR using modern cryptographic construct Indistinguishability Obfuscation. Li *et al.* [11] also proposed OPoR verification scheme for resource-constrained devices such as mobiles where heavy tag generation task is outsourced to cloud audit server to make the process lightweight.

B. RANDOM SAMPLING AUTHENTICATORS

Another way to check the integrity of remote data is based on authenticators. Authenticators are nothing but metadata of an outsourced file. Cloud user produces authenticators or tags before uploading a file on CS. User keeps some part of metadata related to the file at its end. With the help of this metadata, users can verify the integrity of outsourced data. Authenticators are generated using cryptographic algorithms such as RSA, ECC, etc. The algorithms used to calculate authenticators are playing major role in this strategy. Ateniese *et al.* [12] proposed Provable Data Possession (PDP) scheme based on random sampling authenticators.

This research paper concentrates on the PDP model. Ateneiese presented a framework for PDP as shown in Fig. 1. Before outsourcing the file F at an untrusted store, client C preprocesses the file and generates metadata m. Client stores m locally at its store and forward updated file F' to Server S. The server stores the file and replies to challenges generated by client. PDP scheme uses Homomorphic Verifiable Tags (HVT) to generate authenticators. The homomorphic property allows to calculate a single value for the tags generated on multiple file blocks. The client precomputes tags for each block of file and stores the file, the tags at the server. For verification, client generates a challenge for random blocks and server computes the proof of possession

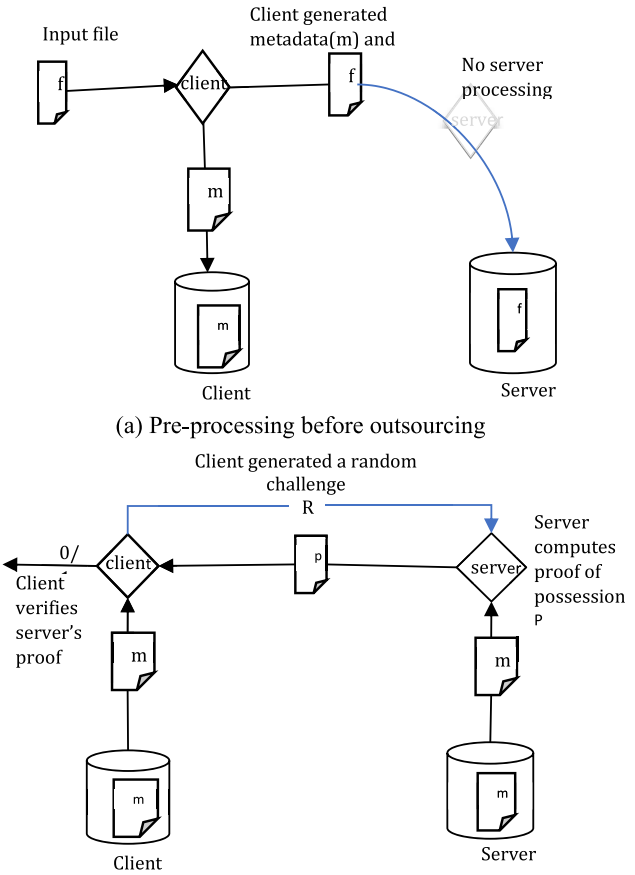


FIGURE 1. Framework for provable data possession.

using challenged blocks and tags. The PDP scheme comprises four polynomial-time algorithms:

KeyGen, *TagBlock*, *GenProof* and *CheckProof*. *KeyGen*, *TagBlock,* and *CheckProof* are executed at client side whereas *GenProof* is executed at server end.

KeyGen$(1^k)\rightarrow$**(*pk*, *sk*):** Key generation algorithm that takes security parameter k as input and produces Secret key pair (pk, sk).

TagBlock **(*pk*, *sk*, *m*)**$\rightarrow T_m$**:** Generates verification metadata using public key pk, secret key sk, and message m. Generates verification metadata T_m.

GenProof **(*pk*, *F*, *chal*,** $\sum$**)**$\rightarrow$***prf*** **:** Server generates proof prf based on challenge blocks. It takes as input public key pk, ordered collection of blocks F, a challenge $chal$, and verification metadata $\sum$ corresponding to ordered file blocks F.

CheckProof**(*pk*, *sk*, *chal*, *prf*)**$\rightarrow${0,1}**:** Validates proof of possession using public key pk, secret key sk, challenge $chal$, and proof prf. Generates 1 if successful otherwise 0.

III. FRAMEWORK OF PUBLIC AUDITING FOR CLOUD STORAGE

Cloud computing provides two basic service models, computing and storage. To meet user demands, all cloud computing services need high-performance cloud storage. So, cloud storage is one of the important components of the cloud computing system. The data in cloud storage is shared and accessed by many users which may create security and privacy issues. Cloud Server (CS) is one of the semi-trusted entities which can hide some unintentional failures and data errors to maintain their reputation. So, the user must have some way to check the integrity of his data stored on cloud storage. User can't download and check integrity due to resource constraints. So public auditing process is widely used where external entities can verify the integrity of out-sourced data on behalf of cloud user without downloading the entire file. The cloud user delegates the task of integrity verification to external entity called Third Party Auditor (TPA). TPA with the help of cryptographic techniques can verify the integrity.

A. SYSTEM MODEL OF CLOUD STORAGE AUDITING

As mentioned in Wang *et al.* [13] work, the auditing process mainly consists of three components: Cloud User, Cloud Server (CS), and TPA as shown in Fig. 2.

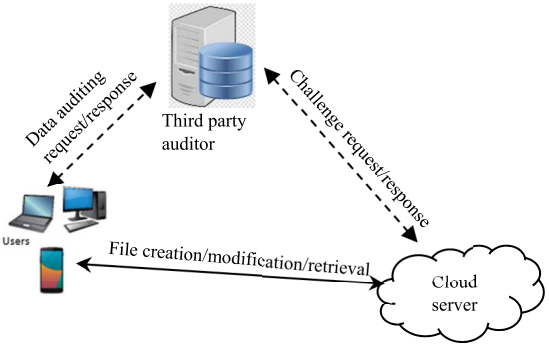


FIGURE 2. Architecture of cloud storage auditing.

1) CLOUD USERS

A Cloud user is an individual user or a group member who outsources large data files on CS. These users completely rely on cloud server storage for data maintenance and computation by Cloud Server Provider (CSP). It has vast storage space and computation infrastructure to maintain and satisfy the cloud user's needs.

2) THIRD-PARTY AUDITOR (TPA)

TPA is an external expert and having capability to check the correctness of cloud storage on behalf of cloud user either periodically or upon user's requests. TPA is a trusted but curious entity in auditing framework that may lead to privacy issues.

B. DESIGN GOALS

Wang *et al.* [14] stated design goals for public auditing system. Our extended design goals can be summarized as follows:

1. Public Verification: TPA can check the integrity of outsourced data without downloading the entire file.
2. Data Dynamic Support: Cloud users regularly update outsourced files. Auditing schemes should support static as well as dynamic updates generated by cloud user.
3. Collusion Resistant: Untrusted CS or revoked user may generate collusion attack in collaboration with TPA to pass the verification of some fabricated event.
4. Privacy-Preserving: User's data contents and identity must not be revealed to TPA during auditing.
5. Lightweight: TPA must be able to complete verification tasks with minimum communication and computation overhead.

IV. STATE-OF-THE-ART CRYPTOGRAPHIC TECHNIQUES FOR AUDITING

Most of the auditing schemes are based on cryptographic primitives to generate authenticators for outsourced data. This section depicts a literature survey of different cryptographic techniques used to achieve above goals during auditing such as MAC, Commitment Scheme, Homomorphic Authenticators, Merkle Hash Tree, Ring signature, etc. Fig. 3 depicts different cryptographic techniques used in PDP.

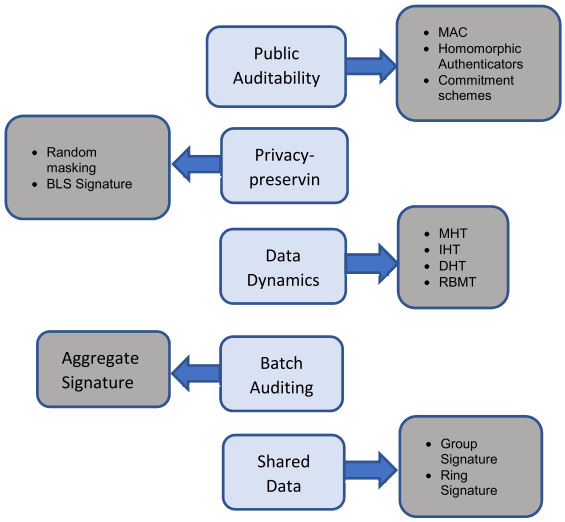


FIGURE 3. Cryptographic techniques used for PDP.

A. PUBLIC VERIFICATION

As surveyed by Zhang *et al.* [15], a widely used cryptographic method to verify the integrity of data is Message Authentication Code (MAC). Generally, before outsourcing, Cloud user precomputes MAC of file F with secret keys and store locally. During auditing, a cloud user reveals a secret key to CS and asked for a fresh MAC for outsourced file F. Then it compares a fresh MAC with locally stored MAC to verify the integrity. This verification covers all the data blocks during auditing. The problem with this technique is the restricted number of verifications because of a limited number of secret keys. Another problem is if user updates the data, cloud user has to download the entire file, calculate a new MAC and outsource the file again which is usually impractical due to large communication overhead. Moreover, the major problem with MAC is that not possible to support public auditability as the private keys are involved during verification.

Another solution is to use a signature instead of MAC for public auditing. Cloud user generates signature on file F before outsourcing. During auditing, requesting a signature for random number of blocks. Even though both the schemes assure integrity, these solutions only support static data. Communication overhead is increased during dynamic data auditing because of frequently modified MAC tags and signatures.

1) HOMOMORPHIC AUTHENTICATORS

To address the problem with MAC and signature for public verification, Atenese *et al.* [12] proposed two PDP schemes: S-PDP and E-PDP using Homomorphic Verifiable Tags (HVT) based on RSA. Homomorphic Authenticators are one type of metadata associated with each file block. These are unforgeable authenticators that can be aggregated securely. By verifying only aggregated authenticators, a verifier gets assurance that a sequential grouping of blocks is properly computed. By using this technique, before outsourcing the data cloud user needs additional information i.e. metadata encoded along with the data. To compute the metadata, a data file is partitioned into blocks $m_i (i = 1,\ldots, n)$. Homomorphic authenticator σ_i is computed on each block. These authenticators are metadata to ensure the integrity of the file. To ensure that data is intact at CS, the authenticators should be verified by the data owner or TPA. A challenge $ch = \{(i, v_i)\}$ for a set of arbitrarily chosen blocks is submitted to CS. v_i can be arbitrary weights. Because of the feature of the homomorphic authenticators, the server only needs to calculate a reply based on a linear arrangement of the sampled data blocks $\mu = \sum_i v_i m_i$ and an aggregated authenticator $\sigma \prod_i \sigma_i^{v_i}$, which is calculated from $\{m_i, \sigma_i, v_i\}_{i \in ch}$. The reply of μ and σ are checked by TPA. It gives assurance of data correctness on a huge portion of cloud data. The problem with Homogeneous Authenticators is that the linear combination of blocks may disclose some information to TPA. RSA-based Homomorphic Authenticators are used by many researchers to ensure the correctness of cloud storage Wang *et al.* [16] proposed a privacy-preserving public auditing scheme using Homomorphic Linear Authenticators (HLA) combined with random masking. Wang *et al.* [17] suggested Oruta public auditing scheme for shared data using HLA. Panda, a public auditing scheme with user revocation is proposed by Wang *et al.* [18]. Public auditing system using RSA based Homomorphic Authenticators consists of mainly 4 functions *Key_Gen(), Sig_Gen(), Gen_Proof(), Verify_Proof()*.

Key_Gen ():

The cloud user C partitions the file F into blocks $m_1 \ldots\ldots m_n$.

- Key Generation Centre (KGC) compute C's private key S_{IDc}& public key P_{IDc}

- A signing key pair (ssk, spk) is generated by C.
- A private key sk: = (S_{IDc}, ssk) and public key pk: = (P_{IDc}, spk) is formed.

Sig_Gen(sk,F):

- An arbitrary element name for the file F is chosen by data owner C.
- A file label t = name$||sig_{ssk}$(name) is computed with a signature on it.
- By taking an arbitrary element for each block of F, cloud user C generates the signature and transfers it to CS.

Gen_Proof (F,spk): In the auditing protocol, TPA gets backs the label t and verifies it by using spk. The procedure is finished if the test fails.

- TPA put a random challenge $chal = \{(i, v_i)\}_{i\in I}$.
- After receiving random challenge $chal$, CS computes $\mu = \sum_i v_i . m_i$
- CS randomly picks $r \leftarrow \mathbb{Z}_p$, computes $R = e(u, v)^r$ and $\gamma = h(R)$ and send it to TPA, where h is any secure hash function.

Verify_Proof (μ, R):

- $\gamma = h(R)$ is computed and then check m_i.

Gennaro and Wichs [19] proposed a new primitive called fully homomorphic authenticators, that generates short tags or authentication metadata. The generated tag is independent of the size of authenticated input to computation. Demirel *et al.* [20] summarized homomorphic message authentication codes (MACs) and homomorphic signatures used to construct privately and publicly verifiable computing schemes.

2) COMMITMENT SCHEMES

To generate proof in auditing systems, commitment schemes are widely used. It allows one to commit to a chosen value (or chosen statement) while keeping it hidden to others, with the ability to reveal the committed value later. Commitment schemes [21] are designed so that a party cannot change the value or statement after they have committed to it. These schemes hide a value but ensure that it cannot be changed later e. g. sealed bid at an auction. There are two stages in these commitment schemes: Commit and Reveal. During commit stage, the sender electronically ''locks'' a message in a box and sends the box to the receiver. During reveal stage, sender proves to the receiver that a certain message is contained in the box. Every commitment scheme must possess the following properties:

a: COMMITMENT MUST BE HIDING

It states that at the end of the commit stage, no adversarial receiver learns information about the committed value. Commitment schemes are computationally hiding if a receiver has limited computational power and cannot learn anything about the committed message. Commitment schemes are unconditional hiding if receiver has unlimited computational power and the commitment phase does not leak any information about the committed message.

b: COMMITMENT MUST BE BIDING

It states that at the end of the reveal stage, there is only one value that an adversarial sender can successfully ''Reveal''. Commitment schemes are computationally binding if a receiver has limited computational power and cannot reveal two different values. Commitment schemes are unconditional binding if receiver has unlimited computational power and cannot reveal two different values.

Two widely used commitment schemes for cloud auditing are Pedersen Commitment and Vector Commitment.

Pedersen Commitment: Pedersen commitment [22] is a perfectly hiding and computationally binding scheme that uses three stages as follows:

***Setup*:** The Receiver selects

- Large primes p and q in such a way that q divides p-1
- G is a generator of the order-q subgroup of Z_p^*
- Random secret a from Z_q
- Calculate $h = g^a \ mod \ p$
  - values p, q, g, and h are published publicly, a kept private.

***Commit*:**

To commit to some $x \in Z_q$, sender calculates $C = g^x h^r mod \ p$ where r is a random value such that r$\in Z_q$. Sender sends this Commitment value C to the receiver. This is nothing but the same as $g^x(g^a)^r \ mod \ p = g^{x+ar} mod \ p$

***Reveal*:**

Sender exposes x and r to open the commitment, receiving party then validates that $C = g^x h^r \ mod \ p$

Cuvelier *et al.* [23] used Pedersen commitment to generate the proof which is perfectly Zero-knowledge proof for verification of election ballot system. Cuvelier and Periera [24] proposed verifiable MPC with PPAT which uses Pedersen commitment with NIZK. This work is illustrated by three different test applications: Solving a system of linear equations, electronic auctions, and finding the shortest path in a graph.

Vector Commitment: Catalano and Fiore [25] proposed a vector commitment scheme, in which user commit to an ordered sequence of q values (i. e. vector) instead of a single message. This commitment is done in such a way that later it is possible to open that commitment based on the positions of the vectors. More specifically, the vector commitment scheme has to preserve binding in terms of position. It states that an adversary can't open a commitment to two different values at the same position. The size of the commitment string and the size of each opening is independent of the vector length. Vector commitment constructs satisfy the computational binding but is not crucial regarding hiding property. It contains six algorithms such as: *VC.KeyGen*, *VC.Com*, *VC.Open*, *VC.Ver*, *VC.Update*, *VC.ProofUpdate*. Thokcham and Saikia [26] used a vector commitment scheme along with ring signature to generate proof during public auditing of cloud storage. Jiang *et al.* [27] proposed public auditing scheme for shared dynamic data with group user revocation using vector commitment and verifier-local revocation group signature.

B. DYNAMIC DATA SUPPORT

Cloud users store as well as share documents with multiple users. Dropbox and Google Docs are the best examples to provide such services to cloud users. Once a file is uploaded and shared, multiple users can insert, delete or modify the contents of the shared file. To handle these updates during public auditing, different cryptographic constructs that can support dynamic operations are discussed next.

1) MERKLE HASH TREE (MHT)

In cloud storage system, data owners may update data dynamically at any time. Block-level functions such as block alteration, block addition, and block removal are performed to maintain dynamic updates. But for auditing purposes, this is the most critical task in cloud data storage system. The homomorphic authenticator scheme uses the index information to calculate the authenticators for each block. Because of this index information, CS is prevented from using the same authenticator for another block to achieve the verification evidence. So, any update such as insertion or deletion operation will modify the indices of all the blocks. This modification leads to the recalculation of all equivalent authenticators.

MHT is a well-defined authentication structure in cryptography. The purpose of MHT is to efficiently prove that a set of elements are unaltered. MHT is a binary tree where the leaves are the hash values of authentic data values.

The Merkle Hash Tree (MHT) [28] as shown in Fig. 4 is used to achieve data dynamics during auditing. The leaves of the MHT are considered as the file blocks m_i. A publicly certified root value R and the Auxiliary Authentication Information (AAI) of every leaf is used to verify the block m. For the path joining from the leaf to the root, AAI includes all the siblings of the nodes. Suppose verifier received verification request for $\{x_2, x_7\}$ with authentic h_r. The prover specifies the verifier with AAI $< h(x_1), h_d >$ and $< h(x_8), h_e >$ respectively for x_2 and x_7. The verifier verifies by computing

$$h(x_2) \rightarrow h_c = h(h(x_1)||h(x_2)) \rightarrow h_a = h(h(h_c)||h(h_d))$$
$$h(x_7) \rightarrow h_f = h(h(x_7)||h(x_8)) \rightarrow h_b = h(h(h_e)||h(h_f))$$

and finally calculates $h_r = h(h(h_a)||h(h_b))$. Here $||$ is a concatenation operation. Check if the calculated h_r is the same as authentic h_r. Auditing schemes proposed in [14], [16], [29] utilized MHT during auditing to support dynamic updates on outsourced data.

Index Hash Table (IHT) is used by Oruta [17] and Panda [18] to support dynamic updates on shared data. In MHT, when a user modifies a single block in shared data, the indices of all blocks after the modified block are completely changed. Cloud user has to re-calculate the block tags after every modification. To avoid this recalculation, Zhu *et al.* [30] proposed IHT for dynamic updates with auditing. IHT stores the modifications on file blocks as well as generate the hash value for each block. The structure of IHT is similar to File Allocation Table (FAT) in operating system. It stores a serial number, block number, version number, random number, etc. A random number is a unique identifier for each block. Even-though, IHT increases complexity but it gives assurance regarding the behavior of untrusted CS. The problem with IHT is that because of single-dimensional sequence structure, insertion and deletion operations are inefficient since it involves adjustment of average N/2 number of blocks. Again, during addition and deletion operation, the block numbers of some blocks are implicitly modified which creates unnecessary computational overhead. To overcome these issues, Tian *et al.* [31] proposed a modified two-dimensional data structure called Dynamic Hash Table (DHT) which consists of basically two elements: file element and block element. Each file is organized using a linked list with the corresponding file element as the header node. Because of the linked list, DHT minimizes the computation overhead during insertion and deletion operations.

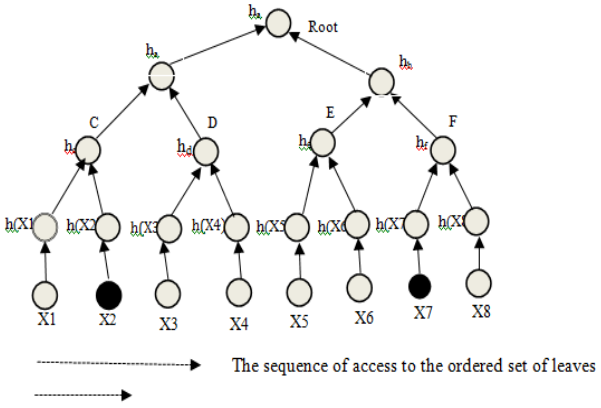


FIGURE 4. Merkle hash tree authentication.

2) RANK BASED MERKLE TREE (RBMT)

MHT supports dynamic updates but introduces some additional information for each tree node, such as status value and height value that leads to increased storage cost. The maintenance cost of MHT is also increased during updates since it has to update two values: status and height for each affected node. RBMT as shown in Fig. 5 is formed by utilizing MHT storage data structure and some parameters in node information [32]. Using cryptographic hash function H and BLS hash function h: $\{0,1\} \rightarrow G$, RBMT for file F comprising $(m_1, m_2 \ldots . m_n)$ blocks are formally defined as:

1. It is a binary tree
2. The leaf node is a 3-tuple $(h, sign, r)$, where $h = h(m_i)$. The value of $sign$ =0 if a node is left child else $sign$ =1 for the right child. The third value r indicates rank and having a default value of 1 for the leaf node.
3. The non-leaf node is a 3-tuple $(h, sign, r)$, where $h = H(h_{lchild}||h_{rchild})$. h_{lchild} and h_{rchild} represents the value of the left and right child of a node. "*Sign*" is the same as above. "r" is the number of nodes reachable from the node. It is calculated as

$$r = r_{lchild} + r_{rchild}$$

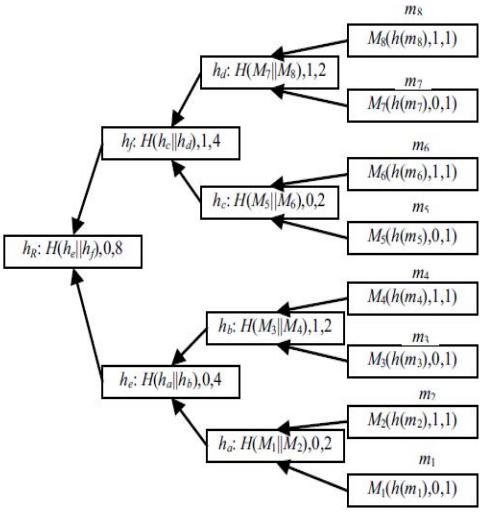


FIGURE 5. Rank based Merkle tree.

where r_{lchild} and r_{rchild} are the rank values of the left and right nodes respectively.

C. PRIVACY-PRESERVING USER GROUP SUPPORT

As mentioned in [17], group users are normally of two types: original user and group user. Most of the time, organizations consist of multiple departments. Generally, employees from individual departments work in a group on documents shared between them. Cloud storage provides group services where the original user shares a document to multiple group members based on access policies. Original user delegates the verification task to TPA. TPA is one of the trusted but curious entities in auditing systems that can infer some information as well as the identity of user during auditing and can harm privacy. There are two types of user groups: static and dynamic. Static groups are predefined before sharing the data on cloud and the membership of users is not changed during data sharing. Whereas, in dynamic groups, a new user can be added and an existing user can be removed from the group at any time.

1) GROUP SIGNATURE

In group signature, a central entity called group manager is involved [33]. Group manager is responsible for the formation of group, addition of members, and revocation of members in a group. Initially, group manager selects its own private key and public parameters for group consisting of the public key of group. Using his own private key, group manager provides certificates to each member. These certificates consist of secret keys of each member. Using this secret key, the user can sign any arbitrary messages and any verifier can verify the authenticity of user using public group key. Thus, the group signature proves that the signer belongs to the group. Prearranged groups, separate procedures for user insertion and revocation, and a common symmetric key among group users are major problems with group signature. To create a confidential network among group members, Group Key Agreement (GKA) protocol is used. Rather than a common symmetric key among group members, Wu et. al [34] proposed Asymmetric Group Key Agreement (ASGKA) protocol in which the public key can be used to validate signature as well as encrypt messages. Any signature can be used to decrypt ciphertext under this public key.

2) RING SIGNATURE

Rivest *et al.* [35] formalized a notion of a ring signature scheme for a group. This scheme comprises only users. There is no centralized entity such as a manager or data owner. This scheme is mainly suitable when the group members do not wish to participate or collaborate in generating the signature. With this scheme, there are no prearranged groups, no procedures for altering, deleting groups, no means to issue specific keys among members, and no way to revoke the anonymity of genuine signers until the signer's wish. These features make this scheme very useful for generating proofs in auditing system where groups are involved. Boneh *et al.* [36] proposed a security model for aggregate signature that aggregates the signatures of n distinct users into a single short signature. This single short signature proves to the verifier that the signature on the original message is indeed calculated by n users. These aggregated signatures greatly help the public auditing system to verify the integrity of outsourced data where user groups are involved. All such signature scheme mostly needs certificates to be stored and process during verification. It led to increased time and computation costs. To address this complex certificate management and the key-escrow problem of ID-PKC, Choi *et al.* [37] proposed a secure certificate-less short signature scheme in a random oracle. Zheng *et al.* [38] proposed a general construction of linkable group signature to achieve anonymity, auditing, and tracing. Kumawat and Paul [39] proposed an accountable ring signature with constant size and based on Indistinguishability Obfuscation or OWF. Schlage and Schwenk [40] presented a CDH-based ring signature scheme that is anonymous and unforgeable under Computational Diffie-Hellman (CDH) assumption in bilinear group. This scheme is also space-efficient. It comprises two algorithms: Ring_sign and Ring_verify.

a: RING-SIGN

This algorithm takes as input given message M. For a size of ring n, each group member chooses a secret key $Sk = x_i$ which belongs to Z_p and public key $Pk = g^{x_i}$.

- Signer t uses global parameter d, $u_0, u_1 \ldots . u_l \in G_1$ of l random elements.
- Signer t will choose random $r_i \in Z_p$ for all other members of the group and generates $s_i = g^{r_i}$. Signer t again computes signature on behalf of the group

$$s_t = (d.\prod_{i=1,i\neq t}^{n} \boldsymbol{Pk}_i^{r_i}(u_0.\prod_{j=1}^{l} \boldsymbol{u}_j^{m_j})^{-r_{n+1}})^{\frac{1}{X^t}} \quad (1)$$

TABLE 1. Comparison of auditing functionalities.

	Public Auditing	Privacy Preserving	Data Dynamics	Batch Auditing	Shared Data
Q. Wang et. al. [14]	RSA based Homomorphic Authenticators	X	Merkle Hash Tree	Bilinear Aggregate Signature	X
C. Wang et. al. [16]	Homomorphic Linear Authenticators	HLA + Random Masking	Merkle Hash Tree	Modified Bilinear Aggregate Signature	X
B. Wang et. al. [17]	Homomorphic Authenticators	HARS	Index Hash Table	Bilinear Maps	Ring Signature
B. Wang et. al. [18]	Homomorphic Authenticators	X	Index Hash Table	X	Proxy Re-Signature with User Revocation
H. Tian et. al. [31]	Homomorphic Verifiable Authenticators	BLS based Signature	Dynamic Hash Table	Aggregate BLS Signature	X
T. Chakraborty et. al. [29]	ECC based Homomorphic authenticators	X	Merkle Hash Tree	X	X
Sumathi D. et. al. [41]	Elliptical Curve Digital Signature Algorithm (ECDSA)	X	Modified ECDSA	X	X
Ni et al. [42]	RSA Accumulator	Identity-based cryptography	MHT	X	Proxy Re-Signature with User Revocation

In (1), signer t computes a signature s_t using his private key X_t with different parameters such as public keys of n group members PK, the global parameter d, $u_0, u_1 \ldots . u_l$, message M divided into l elements $(M_1 \ldots . M_l)$, and random number r. The final signature is $\sigma = (s_1, s_2 \ldots . s_{n+1})$

b: RING-VERIFY

As per (2), the verifier verifies the signature using received ring signature σ, message M, and public keys PK of all members. The verifier checks the following equality

$$\prod_{i=1}^{n} \boldsymbol{e(s_i, Pk_i)}.\mathrm{e}(\boldsymbol{s_{n+1}}, \boldsymbol{u_0} \prod_{j=1}^{l} \boldsymbol{u_j^{m_j}}) = \mathrm{e}(\mathrm{g}, d) \qquad (2)$$

In (2), using public keys PK, signature $(s_1, s_2 \ldots . s_{n+1})$, and received message M, recomputes $\prod_{i=1}^{n} \boldsymbol{e(s_i, Pk_i)}$. $e(s_{n+1}, \boldsymbol{u_0} \prod_{j=1}^{l} \boldsymbol{u_j^{m_j}})$. This recomputed part is verified using global parameters d and g.

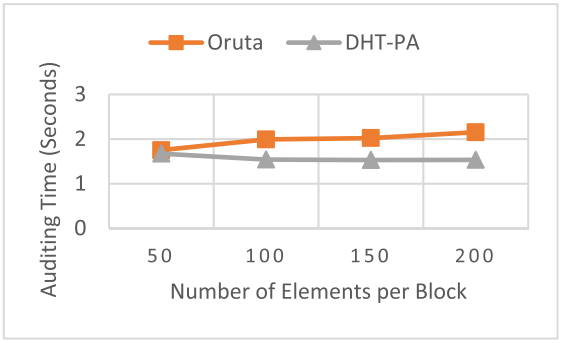
D. COMPARATIVE ANALYSIS OF AUDITING FUNCTIONALITIES

Based on the above cryptographic techniques, multiple researchers have proposed public auditing schemes for cloud storage to verify the integrity of outsourced data. Table 1 shows the survey of different auditing functionalities proposed by different researchers.

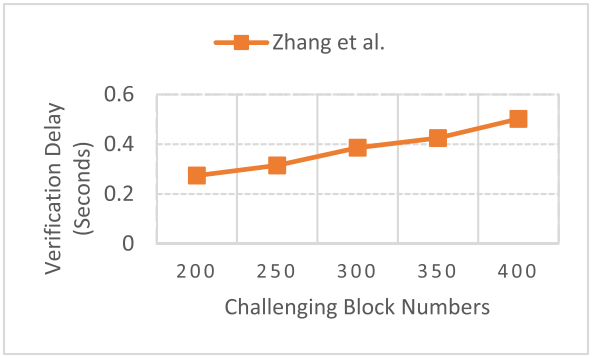
Many researchers have proposed public auditing schemes using homomorphic authenticators, Elliptical curve cryptography for integrity verification. To support dynamic data auditing, MHT is one of the dominating authentications structures in auditing system. To support batch auditing, it is necessary to aggregate the signatures so that TPA can perform multiple auditing tasks simultaneously. Bilinear aggregate signature and its variants are utilized in auditing system by multiple researchers. Shared data means group user support is achieved by ring signature and its variants because of its anonymity. Most of the auditing schemes proposed a combination of HLA and ring signature to maintain user and data privacy from TPA during the auditing process. Chakraborty *et al.* [29] proposed public auditing using elliptical curve cryptography. Another cryptography technique known as ID-based cryptography is recently used by [43]–[45]. Key Generation Centre (KGC) is involved in an ID-based auditing system that generates key parameters for cloud user based on their attributes. As we observed, many researchers have proposed public auditing schemes using homomorphic authenticators, dynamic data support using MHT, etc. Homomorphic authenticators are costly in terms of computation since it involves multiple exponentiations, pairing, and multiplication operations.

These techniques increase the auditing time and computation cost.

As per the survey, there is a need to propose lightweight auditing schemes with modern cryptographic techniques. Many new cryptographic primitives such as signcryption, program obfuscation, and structure-preserving cryptography has powerful functionalities which previously not used by researchers. These can provide cloud users good and rich cloud services. Indistinguishability Obfuscation (IO) is a modern cryptographic technique currently used by many researchers. Using IO, even we obfuscate two programs with unique functionality, they are still computationally not distinct from each other. IO can be used with OWF to construct multiple essential building blocks such as public-key Encryption, ID-based encryption, as well as Chosen-Ciphertext Secure Public Key Encryption,



a) Auditing time in Wang et al. and Tian et al. scheme



b) Verification delay in Zhang et al. scheme

FIGURE 6. **Auditing time comparison between HVT and IO.**

and NIZKs. Zhang *et al.* [46] proposed public auditing using IO. Fig. 6 shows the auditing time comparison between schemes that uses HVT and IO. Fig. 6 a) shows the auditing time using HVT whereas Fig. 6 b) shows the verification time using IO technique. For the block size of 200 KB, the auditing time in Oruta and DHT-PA using HVT is nearer to 2.1 seconds and 1.5 seconds respectively. Whereas it is 0.3 seconds using IO. Fig. 6 shows that the auditing time using HVT is more as compared to IO Technique.

V. LIGHTWEIGHT PDP USING IO

This section describes the latest state of PDP using IO. We are trying to elaborate on the basic functionality of IO and its applicability to achieve different goals of public auditing.

A. . WHAT IS INDISTINGUISHABILITY OBFUSCATION (IO)?

The concept of obfuscation was first proposed by Barak *et. al.* [47]. According to this work, program obfuscation is a technique that creates computer programs ''unintelligible'' but still maintains their functionality. They proposed two methods for IO: Virtual Black-Box Obfuscation (VBO) and Indistinguishability Obfuscation (IO). VBO is similar to a black box instantiating the program. Although it has several applications in cryptography such as converting normal private-key encryption to public-key encryption, authors showed that VBO is not possible to achieve. So, they have proposed another concept Indistinguishability Obfuscation (IO) which obfuscates any two different (same size) functions that implement unique functionalities, that are still computationally differentiable with respect to each other. Garg *et al.* [48] suggested the first candidate construction of IO for general circuits. IO can be defined as below:

Assume $\{C_l\}$ is a circuit class with security parameters l. A uniform PPT algorithm iO having input l, circuit $C \in \{C_l\}$ and outputs a circuit C' is called indistinguishable obfuscator if the following conditions are fulfilled:

1. For all security parameters l, Circuit C, and input x, we have probability as:

$$\Pr[C'(\mathrm{x}) = C(\mathrm{x})] = 1, \quad \text{where } C' = iO(l, C) \tag{3}$$

Equation (3) satisfies the completeness property of IO. It states that circuit C' must behave exactly the same as circuit C if C' is generated by an independent invocation of iO on C.

2. For any (not essentially uniform) PPT adversaries D, for all security parameter $l \in N$, for all pairs of circuits C_0, $C_1 \in C_l$, there exists a negligible function $Negl$ such that if $C_0(\mathrm{x}) = C_1(\mathrm{x})$ for all inputs x, then

$$|\Pr[D(iO(l, C_0)) = 1] - Pr[D(iO(l, C_1)) = 1]| \leq Negl(l) \tag{4}$$

Equation (4) satisfies the indistinguishability property of IO. It states that the secrets embedded in the obfuscated program cannot be extracted by D.

How to obfuscate the program is a major issue. The idea is to puncture the program (which is to be obfuscated). The concept states that we have to separate a key part of the program in such a manner that the functionality of the program still has to remains the same. And since we remove a key element, attacker can't win the security game. Suppose we want to transform a natural private key encryption scheme into a public-key encryption scheme by process of obfuscation. Consider a simple private-key encryption scheme: Assume that the key K is a secret key of a pseudo-random function (PRF), r is a random string. We can represent PRF as $PR = PRF\ (K,\ r)$. For private-key encryption scheme, using a random string r, encrypt the message m and generate the encoded text $C = (r,\ PR \oplus m)$.

How to convert this private-key encryption scheme into a public-key encryption scheme using IO? Concerning to security point of view, the attacker should not be capable to regain message m*, if given the challenge ciphertext $C^* = (r^*, e^*)$, encoding certain message m*. To achieve this, consider a flawed solution as follow: Let's assume that obfuscated version of encryption function as a public key. So obfuscated encryption function becomes: $f_k(r,\ m) = (r,\ PR \oplus m)$. So, we have to develop the function f_k using a ''Punctured'' PRF key such that it properly defines the PRF at all other random strings than r, but not going to reveal any information about $PRF\ (K,\ r^*)$. Then attacker can't crack the security of challenge cipher-text even though he is aware of this punctured PRF confidential key. It means we have to substitute the original function $f_k(r,\ m) = (r,\ PR \oplus m)$ using obfuscation of punctured function in such a way that it is not differentiable

to the attacker. Still, there is a trivial attack possible on this solution. The attacker simply feed input (r^*, *0)* to the obfuscated program and accordingly determine *PRF (K,r*)* and regain m^* from the challenge cipher text. So, the idea is to search for an alternative to ''shift'' the puncturing to a place that is not at all retrieved by the program functionally.

Using a Pseudo-Random Generator (PRG) which associates λ bits to λ bits, where 2λ is a security parameter, we can tackle this issue. We can rewrite the new PRF using PRG as *N_PRF = (k, PRG(r))*. So, revised private-key encryption function is, $f_k(r, m) = (PRG(r), N\_PRF \oplus m)$. Because of the keyless nature of PRG, the attacker can't differentiate the original security game in which the challenge cipher text was generated.

According to Moran and Rosen [49], IO is one of the weaker primitives than VBO. Therefore, it is not possible to construct different cryptographic constructs using only IO. Sahai and Waters [50] have shown how to create basic cryptographic primitives such as: Public-key cryptosystem, short signature, NIZK, etc. from IO and One-Way Function (OWF). Here we elaborate implementation of public-key cryptography using IO. IO implies witness encryption. Witness encryption and OWF imply public-key encryption. This chain of implication results in a public-key encryption scheme where ciphertexts are themselves obfuscated programs. A pseudo-Random generator (PRG) is an OWF used for implementation. This scheme mainly contains three procedures: Setup, Encrypt, and Decrypt. Assume PRG be a pseudo-random generator that maps $\{0, 1\}^{\lambda}$ to $\{0, 1\}^{2\lambda}$. Let F be a puncturable PRF that contains an input of λ bits and outputs a single bit.

- Setup: This algorithm chooses puncturable PRF key k for F and generates an obfuscated program for PKE Encrypt function as below.

PKE Encrypt

Input: Punctured PRF key k, message m, random value r
Calculate t = PRG(r)
Output c= (=t, =F (k, t) $\bigoplus$ m).

The obfuscated program PKE Encrypt* is as below. The public key PK is the obfuscated program and secret key SK is k.

PKE Encrypt*

Input: Punctured PRF key k (), message m, random value r
Calculate t = PRG(r)
Output c = (=t, =F (k, t) ⊕ m).

- Encrypt (PK, m): This algorithm selects an arbitrary value r and executes the obfuscated program of PK on input m, r
- Decrypt (SK, c = (c_1, c_2)): The output of decryption algorithm m'= F(K, c_1) ⊕ c_2

Mahmoody *et al.* [51] prove some lower bounds on assumptions that imply IO in a black-box way based on the computational assumption of P = NP. Holmgren and Lombardi [52] formulated a exponentially secured OWF and constructed collision-resistant hash families.

B. PUBLIC AUDITING OF CLOUD STORAGE USING IO

Most of the auditing schemes are based on homomorphic authenticators which involve multiple computations. Guan *et al.* [10] explored IO to construct a lightweight PoR scheme for public verification and symmetric-key primitives for encryption. This scheme has proved better performance for low-power devices users. It also supports dynamic updates. Some researchers used IO for PDP also. Boneh *et al.* [53] proposed a secure system for a service provider who has created a service and desires to outsource the execution on an untrusted cloud. Malicious cloud may try to infer information from the input-output messages that create harm to privacy of the users who use that service. The scheme allows service users to simultaneously authenticate and securely communicate with an obfuscated program while hiding this authentication and communication from the untrusted cloud.

Many schemes use semi-trusted proxy to convert encrypted messages of client into a ciphertext of the receiver. But in a real scenario, these proxies are not trustworthy and reliable. Consider the situation where client outsourced his encrypted large data on the cloud. To share this data with the receiver, he has to share his transform key with the proxy. Then proxy converts encrypted data into a re-encrypted ciphertext that can be decrypted using the receiver's secret key. However, proxy may use faulty implementation of the re-encryption algorithm or returning a random result to save computation time. To address this issue, Liu *et al.* [54] proposed a novel generic scheme for verifiable proxy re-encryption using IO. Sun *et al.* [55], [56] proposed public data integrity verification scheme based on IO that relieves the user from computation cost to calculate authenticators without splitting the data files into blocks. The privacy of user data will be maintained because of data breaches from TPA. Here we elaborate Zhang *et al.* [46] work in detail which proposed a lightweight public auditing scheme using IO with one-way function MAC for cloud storage. This scheme reduces the computation overhead at the auditor side just by calculating the MAC tag computations. TPA only needs to verify the validity of the MAC tag to check the integrity of outsourced data. The scheme achieves data privacy against the external auditor and resistance against external adversaries. Although IO construction requires a huge overhead for obfuscation, using this scheme it's only one-time cost/computation on user side. The scheme also extended to support dynamic updates using MHT. TPA can perform multiple verification tasks efficiently using batch auditing. The scheme consists of five algorithms: *Setup*, *Store*, *Audit*, *Prove* and *Verify*.

Setup: Let G and G_T are two multiplicative groups produced by g with order p comprises bilinear

map e: G × G → G_T. Data owner D selects a signing key pair (ssk, spk), α, v where $\alpha \rightarrow Z_p$ and v = $g^{\alpha} \in$ G. D chooses s random elements $u_1, u_2 \ldots .u_s$ and determines pseudorandom permutation and function key $\pi_{key}(\ )$ and $f_{key}(\ )$ respectively. The secret and public parameters are sk = (α, ssk) and pk = (v, spk, $u_1, u_2 \ldots .u_s$).

Store: Data owner transforms the data file F into blocks n and each block is again split into s sectors F = $\{f_{i,j}\}_{1\leq i\leq n, 1\leq j\leq s}$. D computes file tag as τ = name|| n||u_1,$u_2 \ldots .u_s$||sig_{ssk}(name||n||u_1,$u_2 \ldots .u_s$) based on randomly selected names. Also computes tag for each data block as:

$$\sigma_i = (\mathrm{H(i||name)} \prod_{j=1}^{s} u_{\mathrm{j}}^{f_{ij}})^{\alpha}, \quad \mathrm{i} \in [1, \mathrm{n}]$$

where H() is any secure hash function.

D has to outsource $\tilde{F}$ = {F = $\{f_{i,j}\}_{\mathrm{i}\in[1,\ \mathrm{n}],\mathrm{j}\in[1,\ \mathrm{s}]}$, ϕ = $\{\sigma_i\}_{\mathrm{i}\in\ [1,\mathrm{n}]}$, τ} on cloud.

Audit: During this phase, D selects a MAC key k and shares it to TPA using a secure channel. D also generates a circuit $Audit_K$ as described below.

Uniform PPT algorithm iO takes security parameters, audit circuit $Audit_k$ and computes public parameter P as P = iO ($Audit_K$). TPA produces a challenge message using data blocks to be audited. Generates {k_1, k_2} which are keys for pseudorandom permutation and function respectively. TPA send these keys to CS.

$Audit_k$

Input: t, {(i,), } i ∈ [1,n], j ∈ [1,s], σ, {v, g, spk}
Constant: MAC Key K
Validate τ
  If no Valid
    Output $\perp$
  Else
    Deconstruct τ to retrieve name,
    If Ver σ, name) = 1
      Generate $\mathrm{MAC_k}$ (name||})
    Else
      Output $\perp$

Prove: Using {k_1, k_2}, CS computes i = $\pi k_1(\xi)$ and v_i = f$k_2(\xi)$ where $\xi \in$ [1, c] and c is the size of I (Input blocks to be audited). Based on public parameters and corresponding $\tau, f_{i,j}, \sigma_i$,c CS computes $\sigma = \prod_{i\in \mathrm{I}} \sigma_i^{v_i}$, $\mu_j = \sum_{i\in \mathrm{I}} v_i f_{ij}$ and Prf = P τ, {(i, v_i), μ_j}$_{\mathrm{i}\in\mathrm{I}}$, σ, {v, spk}). CS send this Prf to TPA for verification phase.

Verify: To verify the correctness of data, TPA computes $\pi k_1(\xi)$ and v_i = f$k_2(\xi)$ and verify whether

$$\mathrm{Prf} = MAC_k(\mathrm{name||\{(i,}\ v_i)_{\mathrm{i}\in\mathrm{I}}\})$$

If equal, verification is successful otherwise verification fails.

Zhang *et al.* [46] scheme focused only on some of the design goals of PDP such as public verification, dynamic data updates and batch verification. Many existing PDP scheme using IO not addresses many security goals such as user group support, revocation of users, collusion resistance, privacy-preserving etc. There are many MPC applications in real life such as e-voting, e-auction etc. where computations are performed by third party or worker. In such systems, many times clients are interested to verify the integrity of his submitted input. PDP is generally used to develop such verifiable MPC applications. IO can be used to minimize the computation as well as verification time in such systems. Next part surveys existing work and challenges to use IO in such multi-party applications as well as different goals to achieve in PDP.

1) VERIFIABLE MULTI-PARTY COMPUTATION USING IO

MPC is a technique in which several parties share their secret input without revealing anything to each other. The required function is calculated on shared input and the result of computation is shared to all the parties. MPC is having multiple applications in real life. Suppose two millionaires want to know which of them has more money without revealing exactly how much money they have. SMC [57], [58] is a computationally efficient technique that enables a client to outsource computation to external parties or cloud providers, convincing that the client's sensitive or shared information is not misused even if some parties are corrupted or cannot be completely trusted. In most practical cases of secure multi-party computation such as auctions, e-voting, benchmarking services, or cloud services in general, computations are performed by assigning all the sensitive inputs to a trusted third party or worker. A worker is responsible for computation and distribution of the result to all users. The basic overview of secret sharing MPC is as follows:

- Initially, all parties share their secret inputs i.e. input x is shared so that $x^i = \sum_{j=1}^{n} x_j^i$ and party $\mathrm{P_j}$ holds x_j^i where n is the number of parties.
- The parties carry out addition and multiplication on these shared values. Parties may communicate certain values by doing computation at local sites. The result of an operation is revealed to all the parties.
- At the end, the parties 'Open' the result of computation. This stage comprises each party sharing their 'final' share with each other (verifies that no errors were hosted by the attacker during this communication).

Gennaro *et al.* [59] introduced the formalization of ''Verifiable Computation'' that enable weak clients to outsource computation of function F on various dynamically chosen inputs $x_1, x_2 \ldots .x_n$ to external party such as worker. The worker then returns the result of computation as well as proof that was carried out correctly on given value x_i. Here users and worker are doing the interaction among themselves in many rounds to complete the process of MPC protocol. The main focus of researchers is to minimize these rounds of interactions and communication complexity to complete the MPC task. The trust in SMC is one of the major problems which encompass two main issues: the integrity of the computation, and the secrecy of the inputs. To verify the correctness of computation, most of the researchers have given approaches

using cryptographic primitives. One such approach proposed by Baum *et al.* [60] is an efficient MPC scheme with public verifiability. They have come up with a publicly verifiable, secure computation scheme. The scheme offers an improved version of the SPDZ (pronounced ''Speedz'') protocol. In this protocol, even though every party involved is dishonest, someone can verify the correctness of the result. Garg *et al.* [61] focus to minimize the communication complexity in terms of the number of rounds minimized to two. They also obtain Universally Composable (UC) security with abort against static malicious adversaries and fairness if there is an honest majority. They come up with a technical tool two-round MPC from IO. He and Wang [57] discussed applications of secure MPC in computational geometric problems. Cuvelier and Pereira [24] present verifiable Multiparty computation with Perfectly Private Audit Trail [PPAT] for MPC using Pedersen commitment scheme. This scheme is single-phase: the users share their inputs asynchronously, and afterward, all parties can collect the result. They presented three distinct applications:

resolving a system of linear equations, an auction system, and the exploration of the shortest path in a shared graph. This protocol uses non-interactive zero-knowledge (NIZK) proof to check the correctness of the result. In this scheme, the time required for verification is increasing as the number of users are increased. Table 2 shows the computation and verification cost of PPAT. Because of these operations, verification time for auction application in PPAT scheme at the client is increased as shown in Table 3.

Online auction is a popular example to be explored in terms of computationally efficient and secure multiparty computations. In 1999, Cachin [62] proposed a novel and efficient private bidding and auction scheme using a combination of homomorphic encryption with ϕ-hiding assumption. Cuvelier and Pereira [24] also presented PPAT scheme for electronic auction. Galal and Youssef [63] proposed solution for a verifiable online auction with blockchain using Ethereum. The comparative study of auditing features for verifiable MPC is shown in Table 4.

As we can observe from Table 4, Cuvelier and Pereira [24] and Baum *et al.* [61] has proposed public verification for SMC. But both the schemes are based on homomorphic encryption and NIZK. Cuvelier and Pereira [24] suggested verifiable MPC using any commitment scheme and NIZK. NIZK verifies the proof by comparing multiple proof relations which involve scalar multiplication and Group operations.

2) COLLUSION RESISTANT PUBLIC AUDITING USING IO

Security threats can be classified as *internal* and *external* threats. Many organizations concentrate only on external threats because of confidence that internal threats can be monitored by organization policies and access rights. Hence, they concentrate on an unfamiliar outsider who can get unauthorized access to their information. Although one entity can't get unauthorized access, dishonest internal and external participants may collaborate and launch a collusion attack to get sensitive data. Public auditing for cloud storage system comprises Cloud users, CS, and TPA. For the efficient processing of the auditing system, many auditing schemes assume all these entities to be honest and fully trusted. But in practice, some of these entities may be dishonest and can collude with each other to generate a collusion attack.

In most auditing techniques, TPA is assumed to be expert, reliable, and having capability to validate the outsourced data on behalf of cloud users. But in real life, certain TPAs are honest but curious. They may collude with CS to pass

TABLE 2. Complexity cost for NIZK.

	Computation	Verification
π_{ope}	2 Sm_p + 1 A	Sm + 2 Sm_p + 2 A
π_{or}	4 Sm_p + 2 A	2 Sm + 3 Sm_p + 3 A
π_{DL}	4 Sm_p + 2 A	2 Sm + 4 Sm_p + 4 A
π_{mul}	6 Sm_p + 3 A	4 Sm + 5 Sm_p + 6 A

Where,
Sm - Scalar Multiplication of an EC Point without precomputation.
Sm_p - Scalar Multiplication of an EC Point with precomputation.
A - EC Point Addition in Group G.
π_{ope} - proof of opening commitments
π_{or} - or-proof of knowledge
π_{DL} - Proof of equality of Discrete Logarithms between two commitments
π_{mul} - Proof of multiplication

TABLE 3. Verification time for auction system.

Number of clients	Verification Time at Client (in seconds)	Verification Time at Client (in seconds)
10	$3.94e10^{-1}$	$3.87e10^{-1}$
100	4	4.17
1000	40.08	42.08

TABLE 4. Comparative study for verifiable MPC.

Sr. No.	Reference Paper	Uses IO	Public Verification	Secure Multiparty Computation
1.	[46]	Yes	Yes	No
2.	[60]	No	Yes	Yes
3.	[61]	Yes	No	Yes
4.	[24]	No	Yes	Yes

the verification of some corrupted events. Huang *et al.* [64] studied the problem or situations where certain TPAs are semi-trusted or malicious. They proposed a feedback-based audit scheme where no need for users to interact with CS and can check the integrity of outsourced data by themselves instead of TPA. TPA generates the feedback loop through processing the proof from CSP and returns it to user which is yet unforgeable to TPA and checked exclusively by user.

In certain situations, during partial and total file loss, CS is one entity that is not trusted. CS may try to deceive users by manipulating verification tags and proving to possess correct files. CS may delete less frequently accessed user data to create space for new data. To manipulate this event, CS may collude with TPA to pass the verification and deceive the user. Chow *et al.* [65] proposed Server-Aided Verification (SAV) scheme based on pairing-based signatures. In business processes, a substantial part of verification computations is outsourced to untrusted servers which allow resource-constrained devices to enjoy security guarantees. To gain an unfair advantage, an adversary may collude with an untrusted server to complete his task. The authors proposed a generic pairing-based SAV protocol to address this issue. Su *et al.* [66] proposed map-based dynamic data integrity verification and recovery scheme that can prevent multiple cloud servers from colluding to fabricate consistent signatures. Huang *et al.* [67] addressed the trust problem between data owners and CS by collaborative auditing blockchain framework for cloud storage. Researchers in [68], [69] also addressed collusion resistance for cloud storage.

Cloud users many times create groups and share the contents with each other. Revoked users from a group may collude with CS or TPA to get unauthorized access of sensitive information. To avoid collusion due to revoked users, it is necessary to re-sign the blocks signed by the revoked user previously or regularly update the valid user list to CS or TPA so that they can differentiate between valid and revoked user.

Cloud users many times create groups and share the contents with each other. Revoked users from group may collude with CS or TPA to get unauthorized access to sensitive information. To avoid collusion due to revoked user, it is necessary to re-sign the blocks signed by the revoked user previously or regularly update the valid user list to CS or TPA so that they can differentiate between valid and revoked user. Wang *et al.* [18] proposed a user revocation scheme where CS has to re-sign the blocks of a revoked user using proxy re-signature. This method avoids the unnecessary computational burden of an existing user to re-sign the blocks of revoked user. Group signature is a cryptographic technique in which any group member can sign the data but the identity of the signer is anonymous in generated signature. To create a confidential network among group members, Group Key Agreement (GKA) protocol is used. Rather than a common symmetric key among group members, Wu *et al.* [34] proposed Asymmetric Group Key Agreement (ASGKA) protocol in which the public key can be used to validate signature as well as encrypt messages while any signature can be used to decrypt ciphertext under this public key. AS in Wang *et al.* scheme, if CS received the secret key of revoked user, it may convert data m to m' by colluding with revoked user. During revocation, this m' will become valid data. To address this situation, Jiang *et al.* [27] proposed a group user revocation scheme in which the data owner is involved during revocation. Revocation uses ASGKA that negotiates a shared secret key instead of a common secret key. Kumar and Parthiban [70] presented a technique to overthrow collusion attack in auditing mechanism utilizing vector commitment and verifier local abrogation group signature. Luo *et al.* [71] presented collusion-resistant auditing using Shamir secret sharing. Scheme allows the group together with proxies and cloud to convert the signatures from revoked users into ones from the existing users after their revocation. To overcome the overhead of computation, Hequn *et al.* [85] utilized backup files for resigning after user has revoked. In this scheme, they store original as well as backup files on cloud during upload. When user is revoked, the existing user will resign on the backup file instead of the original. So revoked users do not have to share their security credentials with cloud. The lazy revocation model is used by Tian *et al.* [72] for revocation to avoid collusion. The idea is that during user revocation, other than block tag generations, all cryptographic information to protect the block needs to be done. Tag regeneration of related blocks is postponed after the signature conversion of revoked user. Rabaninejad *et al.* [73] proposed CORPA, collusion resistant auditing service using a proxy re-signature scheme which is based on Homomorphic Authenticable Signature HAS_{uni}. In some schemes, instead of CS, existing user re-sign the blocks of revoked user. In this scenario, CS may collude with revoked user and get the secret key of the existing user. To address such collusion attack, Mara *et al.* [74] introduce CRUPA: Collusion Resistant User Revocable Public Auditing by the concept of regression. To secure the secret key of an existing user from CS, information proprietor is used to re-compute the Re-sign key using regression and send it to CS. CS further re-signs the blocks of revoked user using Re-sign key. Most of the time, there is large number of blocks are signed and outsourced on CS by revoked user. To resign all this vast information, create additional computational generation and a new private key update technique. By this strategy, revocation is realized by overhead on resigning entities (either CS or Existing user). Zhang *et al.* [75] explored a strategy for key updating the private keys of non-revoked members rather than authenticators of revoked users. Auditing is based on identity-based cryptography. Thokcham and Saikia [26] proposed collusion resistant auditing using CDH based Ring Signature scheme. Collusion between revoked user and CS is avoided through data owner by regularly updating valid members list to CS and TPA after each user revocation. Comparison between different auditing schemes with functionalities such as dynamic data support, user revocation, collusion attack resistance, etc. is shown in Table 5.

TABLE 5. Comparative study of auditing functionalities with collusion resistant.

Scheme	Third-Party Auditor	Dynamic Data Operation	User Revocation	Immune to Collusion Attack	Membership to several groups using same key set	Lightweight
Wang et al. [18]	Yes	Full	Yes	No	No	No
Thokcham and Saikia [26]	Yes	Full	Yes	Yes	Yes	No
Zhang et al. [46]	Yes	Full	No	No	No	Yes
Jiang et al. [27]	Yes	Partial	Yes	Yes	No	No

TABLE 6. Comparison of communication overhead in auditing schemes.

Scheme	Proof Generation	Verification	User Revocation
Zhang et. al.[20]	$\|k_1\| + \|k_2\| +$|HMAC|	2 $Hash_{z_p}$	NA
Wang et.al.[11]	cM+cE	(c+n)E+(c+3n)M+(n+1)P+cH	2E+M+2P+H
Thokcham and Saikia [26]	(q-c)(M+E)	(n+4)P+6E+7M+(c+1)M	(2n+2)E+nM

From Table 5, Zhang *et al.* [46] is one lightweight auditing scheme using IO. This scheme identified collusion attack between malicious auditor and cloud server but has not given any solution for this. Also, it doesn't support user groups. Thokcham and Saikia [26] proposed collusion handling between revoked user and cloud server with integrity verification. This scheme uses vector commitment for integrity verification which increases computation cost as well as verification time because of bilinear pairing. Table 6 shows the communication overhead for different schemes. As we can observe from Table 6, Wang *et al.* and Thokcham scheme need multiple computations such as multiplication, exponentiation, pairing, and hashing that lead to an increase in the cost of communication overhead on TPA as well as cloud user. Compare to these two schemes, Zhang *et al.* scheme need to perform only hash operations for auditing that naturally decrease the communication overhead. Fig. 7 shows the performance of verification time for Wang and Thokcham Scheme. As we observed from the graph, if number of users are increased, the verification time is also increasing by these schemes. So, there is a need to propose collusion resistant public auditing scheme that reduces computational overhead during revocation and is also independent of the number of group users.

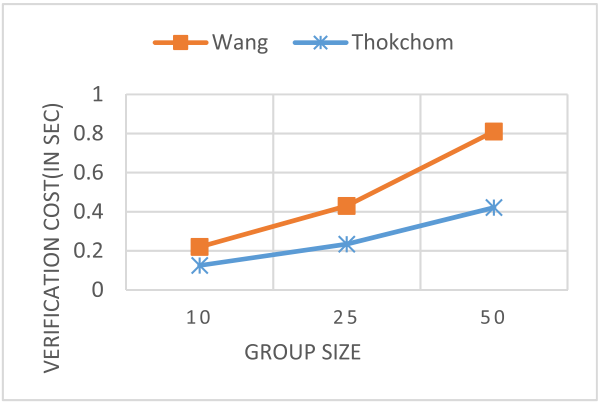


FIGURE 7. Verification cost in Wang and Thokcham scheme.

3) LIGHTWEIGHT AND PRIVACY_PRESERVING PUBLIC AUDITING USING IO

In auditing model, TPA is one of the trusted entities but still curious and may compromise data and user privacy. During auditing, TPA may infer user information who has signed the blocks. Wang *et al.* [76] proposed Knox: privacy-preserving auditing mechanism using group signature to construct homomorphic authenticators. The scheme uses MAC to reduce storage space of verification information. Auditing time is independent of group users. But the time required is high in this scheme. Wang *et al.* [16] achieved privacy from TPA by integrating homomorphic authenticators with a random masking technique. The linear combination of sampled blocks is masked with randomness in the server's response so that TPA is not able to build up a correct group of linear equations and cannot infer user's data contents. Yang *et al.* [77] proposed privacy-preserving cloud auditing scheme for multiple users with authorization and Traceability. Yang *et al.* [78] proposed another framework to achieve identity privacy as well as traceability and formalized public auditing using group signature. This scheme uses a group manager to construct authenticators and generate two lists that record the member list who perform latest modification on each block. Thokcham and Saikia [26] used

TABLE 7. Comparative study of auditing functionalities with privacy-preserving.

Scheme	Third-Party Auditor	Dynamic Data Operation	User Group Support	Privacy Preserving
Wang et al. [17]	Homomorphic Authenticators (HA)	Index Hash Table	Ring Signature	Homomorphic Authenticable Ring Signature (HARS)
Zhang et al. [46]	Indistinguishability Obfuscation (IO)	MHT	NA	NA
Thokcham and Saikia[26]	Vector Commitment	MHT	CDH based Ring Signature	CDH based Ring Signature
Tian et al. [72]	Homomorphic Verifiable Authenticators (HVA)	Dynamic Hash Table	Group Signature	Random Masking
ODPDP [84]	Efficient HVA	Rank Based Merkle Tree (RBMT)	NA	NA

CDH based ring signature scheme to attain group user privacy from TPA during auditing. The communication cost in certificate-based signatures is increasing due to handling of certificates. Some researchers [77], [79], [80] proposed privacy-preserving certificate-less auditing for cloud storage. Results show a great reduction in auditing time. Blockchain-based Privacy-preserving public auditing is proposed by some researchers [81], [82] that attain increased security levels.

In most of these schemes, cloud user and TPA are actively involved during verification phase. This may create an additional burden either on cloud users or on TPA. Tu *et al.* [83] proposed user-focus auditing which tries to reduce the overhead of user by pre-generating challenges for TPA before auditing. Guo *et al.* [84] proposed the ODPDP scheme which relieves user's verification overhead by migrating frequent auditing tasks to TPA. In this scheme, a contract takes place between user and TPA regarding the frequency of verification tasks. TPA generates challenges based on this contract. After each verification, a log file is generated at TPA which contains an audit data log. There is no need for a user to be available during verification, as per his convenience he can check the audit log.Jayaraman and Mohammed [86] also proposed Secure Privacy Conserving PDP (SPC-PDP) model for healthcare system. Based on literature survey, it is observed that ODPDP scheme reduces user overhead but increases the burden on TPA in terms of computation. It uses HVT for integrity verification and RBMT to support dynamic data. The computation cost of TPA in ODPDP during auditing is $(l{+}s{+}1)Exp_G{+}$ 2Pair. Where l is the number of challenged blocks, s denotes number of sectors per block, Exp_G is exponentiation operation on group G and Pair is pairing operation. Compare to ODPDP scheme, Zhang *et al.* [20] scheme create less burden on TPA during auditing i.e. 2 $Hash_{Zp}$ where Hash is hashing operation on Z_p. It means TPA has to compute only 2 hash functions for verification. ODPDP scheme not proposing any solution for identity privacy. The comparison of different auditing schemes concerning privacy-preserving is shown in Table 7.

Zhang *et al.* [89] shows that Privacy-Preserving public auditing protocol proposed by Liu *et al.* [90] for regenerating-code-based cloud storage is not secure since the proxy delegated by data owner can forge authenticators for any data block. Li *et al.* [91] commented on data auditing system called P-ATHAT proposed by Sun *et al.* [92]. P-ATHAT consists of two schemes:

Tree-based private auditing: User need to download the entire data to audit the data integrity.

Tree-based public auditing: No need to download entire data and learning the data contents.

This letter claimed that P-ATHAT scheme is vulnerable to an adversarial CS i.e. if a data block m_i is arbitrarily modified to m'_i, the CS is able to forge a valid tag tag'_i, for m'_i to pass the TPA's auditing. To avoid this, author also suggested that proof information should be blinded to avoid above vulnerability. Prakash *et al.* [95] proposed a multisector public auditing that uses the data blocks masking to preserve privacy from TPA. This scheme is based on homomorphic authenticators. To handle user groups, most of the auditing schemes uses group and ring signature techniques that suffers from large tag size resulting in increased communication and verification cost as well as storage space. Yan *et al.* [96] proposed certificateless PDP in a workgroup that preserves privacy with respect to data uploader as well as group user. This scheme experience reduction in verification time but cost of tag generation is linearly increasing with number of tags. For 300 tags, the cost of tag generation is 3000ms while for 300 challenged blocks, verification time if 1500 ms. Researchers in [97] and [98] proposed privacy-preserving certificateless cloud auditing with group user support. The tag of each message in these schemes is only one element i.e. $|G_1|$. Yan and Gui [99] also proposed an identity-based public auditing scheme with user privacy-preserving. This scheme ensures the relationship of data and data uploader in the proof generation phase instead of integrity auditing phase. It will reduce the overhead on TPA and also maintain privacy from TPA. The tag size in this scheme is again one

group element $|G_1|$. The tag generation time in this scheme is reduced because of compact tag size. For 300 blocks, tag generation time is 4.2 ms.

4) OTHER AREAS OF PUBLIC AUDITING USING IO

Cloud storage services are utilized in many fields to outsource data. CS and cloud users face multiple challenges because of this paradigm. Handling duplicate data on cloud storage is a major concern for CS. Multiple researchers have given solutions for cloud de-duplication. Liu at al. [102] proposed ''one-tag checker'' integrity auditing scheme using Convergent Encryption (CE) to avoid redundancy. Tag deduplication is avoided using message derived private key. Integrity auditing is achieved using HA and proxy-resignature scheme. Gao *et al.* [103] proposed low-entropy cloud data auditing scheme for file and authenticator deduplication that also utilizes a random-message locked key to compute authenticators. If we compare these schemes the proof generation time is linearly increasing as the number of blocks are increased. Gao *et al.* [103] also proposed cloud data auditing scheme with file and authenticator deduplication that achieve low-entropy.IO can be used with this scheme to share re-key in audit circuit so that malicious CS not deduce it.

To achieve a lightweight PDP, Gao *et al.* [104] proposed ''checking only when it is necessary'' auditing scheme with encrypted keyword for sensitive information. Based on label Relation Authentication Label (RAL), scheme generates the auditing proof as well as authenticate the relation that file contains the queries keyword.

Key-exposure is one of the serious problem reported in cloud paradigm. The secret key might be exposed due to weak security settings at client side. Once malicious CS received such keys for auditing, it can hide data lost incidents by forging fake authenticators. Yu *et al.* [105] proposed key-exposure resilient auditing for cloud storage. Using IO during auditing, normally keys we can share through audit circuit that data owner obfuscate and executed by CS to generate the proof. IO based auditing can give solution for such Key-exposure problem.

VI. NEW CHALLENGES AND RECOMMENDATIONS

PDP is one of the matured and widely used model for integrity verification of cloud storage. But most of the techniques used with PDP are based on cryptographic techniques. Homomorphic authenticator is one of the dominating techniques used by many researchers. But this scheme increases the verification time because of expensive computations on large numbers. Many PDP schemes are based on IO which is one of the modern cryptographic techniques. Many PDP scheme realizes efficient verification using IO but authenticator generation make these schemes inefficient. This is one of the limitations of IO that avoid to use it in PDP. Even though it creates one time overhead of obfuscating the verification circuit on user, it greatly reduces the verification time. There is a scope to improve the auditing processes utilizing IO by considering multiple perspectives of auditing. Verification of MPC applications such as E-voting, E-auctions that uses lightweight devices such as mobiles is still an open issue to resolve because of increased verification time. It is necessary to propose Verifiable MPC such that there are reduced number of rounds and verification round must be independent from another round so that verification time will be reduced. Communication overhead in such systems totally depends upon message sizes transferred between different entities during auditing. Tag generation, proof generation phases must employ lightweight cryptographic techniques so that the computational overhead on lightweight devices should be reduced. IOT applications also uses lightweight devices. How can we utilize the IO technique for achieving verifiability in IOT environment is one of the future research direction in this field.

Group and ring signature techniques are widely used for group support in cloud paradigm. These techniques are based on certificates that increases the burden of certificate handling during verification. So, this is one of the avenues for future research to propose group support auditing using certificateless signature. Collusion attack is mostly generated by revoked users or TPA to get access to group information. Revocation policies should be strengthened to avoid such events. Zhang *et al.* [87] proposed revocable and certificateless public auditing model for cloud storage that supports the key update of the group user and the tag update of the revoked user. In most of the existing revocation procedures, existing user has to resign the blocks of revoked user that may create unnecessary burden on such users. Using symmetric key cryptography for integrity verification, cloud user has to share the secret key with TPA so that TPA can verify the file blocks during auditing. Even though TPA is trusted entity in this paradigm, he is curious about the data and identity of users. In such situations TPA may also collude with CS and fabricate the collusion attack to pass some forged audit.

To support dynamic data updates during auditing, MHT is widely used. During verification of MHT, auxiliary authentication information has to be generated and verified using certified root. This process may create many duplicate and unnecessary information during auditing for each verification. It may affect the total verification time. So, it is necessary to use IO with other constructs such as DHT, RBMT with multiple dynamic updates in batch mode to reduce verification time.

Continuous involvement of user during verification may create unnecessary burden on user. TPA also has to perform large computations during auditing. So, from the computation perspectives, the technical challenge of auditing service can be addressed by employing lightweight auditing process where user can audit the integrity of data as per his convenience. At the same time the computational burden of TPA should be reduced to generate lightweight auditing process. Wang *et al.* [88] presented lightweight certificate-based public/private auditing scheme based on asymmetric bilinear pairing. This scheme focuses on minimizing the

computation cost of tag generation at user side since large data is outsourced by cloud users. Uses public/private audit model. Private auditing is efficient but during disputes, user can't become judge. So public auditing is invoked during any disputes in the absence of user. Such type of public/private auditing model can be used to generate lightweight processes.

Another major challenge observed by Ding *et al.* [93] is that if the user's auditing key is exposed to the malicious CS, the user's data may be deleted by CS without being detected. Ding proposed intrusion-resilience public auditing scheme to relieve damage caused by the key-exposure problem. As observed in this scheme, proof verification time depends on number of challenged blocks. As challenged blocks are increased, the verification time is also increasing. As a future work, modified auditing scheme can be proposed using IO to maintain constant and reduced verification time during auditing.

George and Nargunam [94] presented a comprehensive analytical and comparative survey for different auditing techniques and inferred that IO is one of the efficient methods to reduce overhead at auditor side as compared to attribute-based methods. George also suggested to evolve efficient auditing scheme using IO that can reduce overhead on CS as well as TPA. Chen *et al.* [100] proposed a threshold hybrid encryption for integrity verification using AES and Shamir Secret Sharing without trusted center. Scheme also proposed re-signature technique to avoid collusion due to revoked user. As future research, this scheme can be modified using IO to reduce verification cost. Shen *et al.* [101] explored how to employ fuzzy private key for integrity auditing without storing private key. We can improve Zhang *et al.* [46] scheme by this fuzzy approach where cloud user has to share the MAC key with TPA through secure channel.

There are still multiple issues during auditing that are not addresses properly or not able to generate efficient auditing schemes. Auditing using file and authenticator deduplication, key-exposure resilient are some of the issues where IO technique can be utilized.

VII. CONCLUSION

In this paper, we reviewed different cryptographic constructs to implement lightweight auditing process for integrity verification of outsourced data. There are two basic approaches of auditing such as PDP and PoR. Our work concentrates on lightweight PDP processes. In this review, initially we presented an overview of multiple cryptographic techniques to achieve goals of public auditing system. We compared multiple auditing schemes proposed by many researchers with respect to auditing goals and cryptographic techniques. From this comparison, we came to know that with the use of IO, auditing time is greatly reduced. We explored the Zhang *et al.* work that uses IO to achieve lightweight public auditing process. Further, we identified the challenges to develop lightweight PDP processes with respect to verifiable MPC, Group support, Collusion resistant and privacy-preserving using IO. We analyzed the performance based on communication cost, proof generation time and proof verification time. The study of this research work show that generation of lightweight PDP processes still has some open issues. Based on the survey, challenges are listed and recommendations are suggested.

REFERENCES

[1] C. Wang, K. Ren, W. Lou, and J. Li, "Toward publicly auditable secure cloud data storage services," *IEEE Netw.*, vol. 24, no. 4, pp. 19–24, Jul./Aug. 2010.

[2] H. Chen, S. Tu, C. Zhao, and Y. Huang, "Provenance cloud security auditing system based on log analysis," in *Proc. IEEE Int. Conf. Online Anal. Comput. Sci. (ICOACS)*, May 2016, pp. 155–159.

[3] R. K. L. Ko, B. S. Lee, and S. Pearson, "Towards achieving accountability, auditability and trust in cloud computing," in *Proc. Int. Conf. Adv. Comput. Commun.*, vol. 193. Berlin, Germany: Springer, 2011, pp. 432–444.

[4] J. Ryoo, S. Rizvi, W. Aiken, and J. Kissell, "Cloud security auditing: Challenges and emerging approaches," *IEEE Secur. Privacy*, vol. 12, no. 6, pp. 68–74, Nov. 2014.

[5] W. Hsien, C. Yang, and M. Hwang, "A survey of public auditing for secure data storage in cloud computing," *Int. J. Netw. Secur.*, vol. 18, no. 1, pp. 133–142, 2016.

[6] M. Kolhar, M. M. Abu-Alhaj, and S. M. A. El-atty, "Cloud data auditing techniques with a focus on privacy and security," *IEEE Secur. Privacy*, vol. 15, no. 1, pp. 42–51, Jan. 2017.

[7] B. Liu and Y. Chen, "Auditing for data integrity and reliability in cloud storage," in *Handbook on Data Centers*, 2015, pp. 535–559.

[8] A. Juels and B. S. Kaliski, "PORs: Proofs of retrievability for large files," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, 2007, pp. 584–597.

[9] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2008, pp. 90–107.

[10] C. Guan, K. Ren, F. Zhang, F. Kerschbaum, and J. Yu, "Symmetric-key based proofs of retrievability supporting public verification," in *Proc. Eur. Symp. Res. Comput. Secur.*, 2015, pp. 203–223.

[11] J. Li, X. Tan, X. Chen, D. S. Wong, and F. Xhafa, "OPoR: Enabling proof of retrievability in cloud computing with resource-constrained devices," *IEEE Trans. Cloud Comput.*, vol. 3, no. 2, pp. 195–205, Apr. 2015.

[12] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, 2007, pp. 598–609.

[13] C. Wang, Q. Wang, K. Ren, and W. Lou, "Ensuring data storage security in cloud computing," in *Proc. 17th Int. Workshop Quality Service*, 2009, pp. 1–9.

[14] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 5, pp. 847–859, May 2011.

[15] 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./Oct. 2016.

[16] 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.

[17] B. Wang, B. Li, and H. Li, "Oruta: Privacy-preserving public auditing for shared data in the cloud," *IEEE Trans. Cloud Comput.*, vol. 2, no. 1, pp. 43–56, Jan. 2014.

[18] B. Wang, B. Li, and H. Li, "Panda: Public auditing for shared data with efficient user revocation in the cloud," *IEEE Trans. Services Comput.*, vol. 8, no. 1, pp. 92–106, Jan. 2015.

[19] R. Gennaro and D. Wichs, "Fully homomorphic message authenticators," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2013, pp. 301–320.

[20] D. Demirel, L. Schabhüser, and J. Buchmann, "Homomorphic authenticators," in *Proc. Privately Publicly Verifiable Comput. Techn.*, 2017, pp. 27–35.

[21] I. Damgård, "Commitment schemes and zero-knowledge protocols," in *Proc. Eur. Educ. Forum*, 1998, pp. 63–86.

[22] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Proc. Annu. Int. Cryptol. Conf.*, 1991, pp. 129–140.

[23] E. O. Cuvelier Pereira and T. Peters, "Election verifiability or ballot privacy: Do we need to choose?" in *Proc. Eur. Symp. Res. Comput. Secur.* Berlin, Germany: Springer, 2013, pp. 481–498.

[24] E. Cuvelier and O. Pereira, "Verifiable multi-party computation with perfectly private audit trail," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.*, Guildfold, U.K., 2016, pp. 367–385.
[25] D. Catalano and D. Fiore, "Vector commitments and their applications," in *Proc. Int. Workshop Public Key Cryptogr.*, 2013, pp. 55–72.
[26] S. Thokcham and D. Saikia, "Privacy preserving integrity checking of shared dynamic cloud data with user revocotion," *J. Inf. Secur. Appl.*, vol. 50, pp. 2126–2214, Feb. 2020.
[27] T. Jiang, X. Chen, and J. Ma, "Public integrity auditing for shared dynamic cloud data with group user revocation," *IEEE Trans. Comput.*, vol. 65, no. 8, pp. 2363–2373, Aug. 2016.
[28] R. C. Merkle, "Protocols for public key cryptosystems," in *Proc. IEEE Symp. Secur. Privacy*, Apr. 1980, pp. 122–132.
[29] T. K. Chakraborty, A. Dhami, P. Bansal, and T. Singh, "Enhanced public auditability & secure data storage in cloud computing," in *Proc. 3rd IEEE Int. Adv. Comput. Conf. (IACC)*, Feb. 2013, pp. 101–105.
[30] Y. Zhu, H. Wang, Z. Hu, G.-J. Ahn, H. Hu, and S. S. Yau, "Dynamic audit services for integrity verification of outsourced storages in clouds," in *Proc. ACM Symp. Appl. Comput.*, 2011, pp. 1550–1557.
[31] H. Tian, Y. Chen, C.-C. Chang, H. Jiang, Y. Huang, Y. Chen, and J. Liu, "Dynamic-hash-table based public auditing for secure cloud storage," *IEEE Trans. Services Comput.*, vol. 10, no. 5, pp. 701–714, Sep. 2017.
[32] J. Zou, Y. Sun, and S. Li, "Dynamic provable data possession based on ranked Merkle hash tree," in *Proc. Int. Conf. Identificat., Inf. Knowl. Internet Things (IIKI)*, Oct. 2016, pp. 4–9.
[33] M. Manulis, N. Fleischhacker, F. Gunther, K. Franziskus, and B. Poettering, "Group signatures: Authentication with privacy," Surv. Federal Office Inf. Secur., Bonn, Germany, Tech. Rep., 2012.
[34] Q. Wu, Y. Mu, W. Susilo, B. Qin, and J. Domingo-Ferrer, "Asymmetric group key agreement," in *Proc. Annu. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2009, pp. 153–170.
[35] R. Rivest, A. Shamir, and Y. Tauman, "How to leak a secret," in *Proc. Theory Appl. Cryptol. Inf. Secur.*, 2001, pp. 552–565.
[36] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Proc. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2003, pp. 416–432.
[37] K. Y. Choi, J. H. Park, and D. H. Lee, "A new provably secure certificateless short signature scheme," *Comput. Math. Appl.*, vol. 61, no. 7, pp. 1760–1768, 2011.
[38] H. Zheng, Q. Wu, B. Qin, L. Zhong, S. He, and J. Liu, "Linkable group signature for auditing anonymous communication," in *Proc. Australas. Conf. Inf. Secur. Privacy*, 2018, pp. 304–321.
[39] S. Kumawat and S. Paul, "A new constant-size accountable ring signature scheme without random oracles," in *Proc. Int. Conf. Inf. Secur. Cryptol.*, 2017, pp. 157–179.
[40] S. Schäge and J. Schwenk, "A CDH-based ring signature scheme with short signatures and public keys," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.*, 2010, pp. 129–142.
[41] D. Sumathi and R. V. Pujeri, "A modified elliptic curve digital signature algorithm for public verifiability with data dynamics in cloud computing," *J. Comput. Sci.*, vol. 10, no. 10, pp. 2077–2087, Oct. 2014.
[42] J. Ni, K. Zhang, Y. Yu, and T. Yang, "Identity-based provable data possession from RSA assumption for secure cloud storage," *IEEE Trans. Dependable Secure Comput.*, early access, Nov. 9, 2020, doi: 10.1109/TDSC.2020.3036641.
[43] 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.
[44] J. Xue, C. Xu, J. Zhao, and J. Ma, "Identity-based public auditing for cloud storage systems against malicious auditors via blockchain," *Sci. China Inf. Sci.*, vol. 62, no. 3, p. 32104, Mar. 2019.
[45] J. R. Gudeme, S. K. Pasupuleti, and R. Kandukuri, "Attribute-based public integrity auditing for shared data with efficient user revocation in cloud storage," *J. Ambient Intell. Humanized Comput.*, vol. 12, no. 2, pp. 2019–2032, Feb. 2021.
[46] Y. Zhang, C. Xu, X. Liang, H. Li, Y. Mu, and X. Zhang, "Efficient public verification of data integrity for cloud data store systems from indistinguishability obfuscation," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 3, pp. 676–688, Mar. 2017.
[47] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang, "On the (Im)possibility of obfuscating programs," in *Proc. Annu. Int. Cryptol. Conf.*, 2001, pp. 1–18.
[48] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters, "Candidate indistinguishability obfuscation and functional encryption for all circuits," *SIAM J. Comput.*, vol. 45, no. 3, pp. 882–929, 2016.
[49] T. Moran and A. Rosen, "There is no indistinguishability obfuscation in pessiland," IACR Cryptol. ePrint Arch., Tech. Rep. 643, 2013, p. 643.
[50] A. Sahai and B. Waters, "How to use indistinguishability obfuscation: Deniable encryption, and more," in *Proc. 46th Annu. ACM Symp. Theory Comput.*, May 2014, pp. 475–484.
[51] M. Mahmoody, A. Mohammed, S. Nematihaji, R. Pass, and A. Shelat, "Lower bounds on assumptions behind indistinguishability obfuscation," in *Proc. Theory Cryptogr. Conf.*, 2016, pp. 49–66.
[52] J. Holmgren and A. Lombardi, "Cryptographic hashing from strong one-way functions (or: One-way product functions and their applications)," in *Proc. IEEE 59th Annu. Symp. Found. Comput. Sci. (FOCS)*, Oct. 2018, pp. 850–858.
[53] D. Boneh, D. Gupta, I. Mironov, and A. Sahai, "Hosting services on an untrusted cloud," in *Proc. Annu. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2015, pp. 404–436.
[54] M. Liu, Y. Wu, J. Chang, R. Xue, and W. Guo, "Verifiable proxy re-encryption from indistinguishability obfuscation," in *Proc. Int. Conf. Inf. Commun. Secur.*, 2015, pp. 363–378.
[55] L. Sun, C. Xu, Y. Zhang, and K. Chen, "An efficient $\rangle\mathcal{O}$-based data integrity verification scheme for cloud storage," *Sci. China Inf. Sci.*, vol. 62, no. 5, May 2019, Art. no. 059101.
[56] L. Sun, C. Xu, Y. Zhang, and K. Chen, "Public data integrity auditing without homomorphic authenticators from indistinguishability obfuscation," *Int. J. Inf. Secur.*, vol. 19, pp. 711–720, Jan. 2020.
[57] F. He and T. Wang, "Research and application of secure multi-party computation in several computational geometry problems," in *Proc. Int. Conf. Ind. Control Electron. Eng.*, Aug. 2012, pp. 1434–1437.
[58] P. Pullonen and S. Siim, "Combining secret sharing and garbled circuits for efficient private IEEE 754 floating-point computations," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.*, 2015, pp. 172–183.
[59] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: Outsourcing computation to untrusted workers," in *Proc. Annu. Cryptol. Conf.*, 2010, pp. 465–482.
[60] C. Baum, I. Damgård, and C. Orlandi, "Publicly auditable secure multi-party computation," in *Proc. Int. Conf. Secur. Cryptogr. Netw.*, 2014, pp. 175–196.
[61] S. Garg, C. Gentry, S. Halevi, and M. Raykova, "Two-round secure MPC from indistinguishability obfuscation," in *Proc. Theory Cryptogr. Conf.*, Berlin, Germany, 2014, pp. 74–94.
[62] C. Cachin, "Efficient private bidding and auctions with an oblivious third party," in *Proc. 6th ACM Conf. Comput. Commun. Secur.*, 1999, pp. 120–127.
[63] H. S. Galal and A. M. Youssef, "Verifiable sealed-bid auction on the Ethereum blockchain," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.*, 2018, pp. 265–278.
[64] K. Huang, M. Xian, S. Fu, and J. Liu, "Securing the cloud storage audit service: Defending against frame and collude attacks of third party auditor," *IET Commun.*, vol. 8, no. 12, pp. 2106–2113, Aug. 2014.
[65] S. S. M. Chow, M. H. Au, and W. Susilo, "Server-aided signatures verification secure against collusion attack," *Inf. Secur. Tech. Rep.*, vol. 17, no. 3, pp. 46–57, Feb. 2013.
[66] Z. Su, Y. Yang, Q. Shen, Z. Wu, and X. Li, "MB-DDIVR: A map-based dynamic data integrity verification and recovery scheme in cloud storage," in *Proc. ICICS*, in Lecture Notes in Computer Science, 2016, pp. 335–345.
[67] P. Huang, K. Fan, H. Yang, K. Zhang, H. Li, and Y. Yang, "A collaborative auditing blockchain for trustworthy data integrity in cloud storage system," *IEEE Access*, vol. 8, pp. 94780–94794, 2020.
[68] J. Li, W. Yao, J. Han, Y. Zhang, and J. Shen, "User collusion avoidance CP-ABE with efficient attribute revocation for cloud storage," *IEEE Syst. J.*, vol. 12, no. 2, pp. 1767–1777, Jun. 2018.
[69] D. S. Kasunde and A. A. Manjrekar, "Verification of multi-owner shared data with collusion resistant user revocation in cloud," in *Proc. Int. Conf. Comput. Techn. Inf. Commun. Technol. (ICCTICT)*, Mar. 2016, pp. 182–185.
[70] S. Kumar and L. Parthiban, "Cloud data integrity auditing over dynamic data for multiple users," *Int. J. Intell. Eng. Syst.*, vol. 10, no. 5, pp. 239–246, Oct. 2017.
[71] Y. Luo, M. Xu, K. Huang, D. Wang, and S. Fu, "Efficient auditing for shared data in the cloud with secure user revocation and computations outsourcing," *Comput. Secur.*, vol. 73, pp. 492–506, Mar. 2018.

[72] H. Tian, F. Nan, H. Jiang, C.-C. Chang, J. Ning, and Y. Huang, "Public auditing for shared cloud data with efficient and secure group management," *Inf. Sci.*, vol. 472, pp. 107–125, Jan. 2019.
[73] R. Rabaninejad, M. A. Attari, M. R. Asaar, and M. R. Aref, "A lightweight auditing service for shared data with secure user revocation in cloud storage," *IEEE Trans. Services Comput.*, vol. 15, no. 1, pp. 1–15, Jan. 2022.
[74] G. C. Mara, U. Rathod, S. R. RG, R. Buyya, K. R. Venugopal, S. S. Iyengar, and L. M. Patnaik, "CRUPA: Collusion resistant user revocable public auditing of shared data in cloud," *J. Cloud Comput.*, vol. 9, no. 1, pp. 1–18, Dec. 2020.
[75] Y. Zhang, J. Yu, R. Hao, C. Wang, and K. Ren, "Enabling efficient user revocation in identity-based cloud storage auditing for shared big data," *IEEE Trans. Dependable Secure Comput.*, vol. 17, no. 3, pp. 608–619, Jun. 2020.
[76] B. Wang, B. Li, and H. Li, "Knox: Privacy-preserving auditing for shared data with large groups in the cloud," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.*, 2012, pp. 507–525.
[77] X. Yang, M. Wang, T. Li, R. Liu, and C. Wang, "Privacy-preserving cloud auditing for multiple users scheme with authorization and traceability," *IEEE Access*, vol. 8, pp. 130866–130877, 2020.
[78] G. Yang, J. Yu, W. Shen, Q. Su, Z. Fu, and R. Hao, "Enabling public auditing for shared data in cloud storage supporting identity privacy and traceability," *J. Syst. Softw.*, vol. 113, pp. 130–139, Mar. 2016.
[79] Y. Ming and W. Shi, "Efficient privacy-preserving certificateless provable data possession scheme for cloud storage," *IEEE Access*, vol. 7, pp. 122091–122105, 2019.
[80] K. Zhao, D. Sun, G. Ren, and Y. Zhang, "Public auditing scheme with identity privacy preserving based on certificateless ring signature for wireless body area networks," *IEEE Access*, vol. 8, pp. 41975–41984, 2020.
[81] L. Huang, G. Zhang, and A. Fu, "Privacy-preserving public auditing for non-manager group shared data," *Wireless Pers. Commun.*, vol. 100, no. 4, pp. 1277–1294, Jun. 2018.
[82] P. Banerjee, N. Nikam, and S. Ruj, "Blockchain enabled privacy preserving data audit," 2019, *arXiv:1904.12362*.
[83] T. Tu, L. Rao, H. Zhang, Q. Wen, and J. Xiao, "Privacy-preserving outsourced auditing scheme for dynamic data storage in cloud," *Secur. Commun. Netw.*, vol. 2017, pp. 1–17, Dec. 2017.
[84] W. Guo, H. Zhang, S. Qin, F. Gao, Z. Jin, W. Li, and Q. Wen, "Outsourced dynamic provable data possession with batch update for secure cloud storage," *Future Gener. Comput. Syst.*, vol. 95, pp. 309–322, Jun. 2019.
[85] H. Liu, B. Wang, K. Lu, Z. Gao, and Y. Zhan, "Public auditing for shared data utilizing backups with user revocation in the cloud," *Wuhan Univ. J. Natural Sci.*, vol. 23, no. 2, pp. 129–138, Apr. 2018.
[86] I. Jayaraman and M. Mohammed, "Secure privacy conserving provable data possession (SPC-PDP) framework," *Inf. Syst. e-Bus. Manage.*, vol. 18, no. 3, pp. 351–377, Sep. 2020.
[87] Y. Zhang, T. Zhang, S. Xu, G. Xu, and D. Zheng, "Revocable and certificateless public auditing for cloud storage," *Sci. China Inf. Sci.*, vol. 63, no. 10, Oct. 2020, Art. no. 209302.
[88] F. Wang, L. Xu, K.-K.-R. Choo, Y. Zhang, H. Wang, and J. Li, "Lightweight certificate-based public/private auditing scheme based on bilinear pairing for cloud storage," *IEEE Access*, vol. 8, pp. 2258–2271, 2020.
[89] J. Zhang, R. Lu, B. Wang, and X. An Wang, "Comments on 'privacy-preserving public auditing protocol for regenerating-code-based cloud storage'," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 1288–1289, 2021.
[90] J. Liu, K. Huang, H. Rong, and H. Wang, "Privacy-preserving public auditing for regenerating-code-based cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 7, pp. 1513–1528, Jul. 2015.
[91] S. Li, Y. Zhang, C. Xu, and K. Chen, "Cryptoanalysis of an authenticated data structure scheme with public privacy-preserving auditing," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 2564–2565, 2021.
[92] Y. Sun, Q. Liu, X. Chen, and X. Du, "An adaptive authenticated data structure with privacy-preserving for big data stream in cloud," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3295–3310, 2020.
[93] R. Ding, Y. Xu, J. Cui, and H. Zhong, "A public auditing protocol for cloud storage system with intrusion-resilience," *IEEE Syst. J.*, vol. 14, no. 1, pp. 633–644, Mar. 2020.
[94] A. S. George and A. S. Nargunam, "Remote cloud data auditing protocols: A comprehensive analysis and comparative study," in *Proc. 5th Int. Conf. Intell. Comput. Control Syst. (ICICCS)*, May 2021, pp. 64–70.
[95] G. Prakash, M. Prateek, and I. Singh, "Secure public auditing using batch processing for cloud data storage," in *Proc. 1st Int. Conf. Smart Syst., Innov. Comput., Smart Innov., Syst. Technol.*, 2018, pp. 137–148.
[96] H. Yan, Y. Liu, Z. Zhang, and Q. Wang, "Efficient privacy-preserving certificateless public auditing of data in cloud storage," *Secur. Commun. Netw.*, vol. 2021, pp. 1–11, May 2021.
[97] G. Wu, Y. Mu, W. Susilo, F. Guo, and F. Zhang, "Privacy-preserving certificateless cloud auditing with multiple users," *Wireless Pers. Commun.*, vol. 106, no. 3, pp. 1161–1182, 2019.
[98] G. Wu, Y. Mu, W. Susilo, and F. Guo, "Privacy-preserving cloud auditing with multiple uploaders," in *Proc. Inf. Secur. Pract. Exp.* in Lecture Notes in Computer Science, vol. 10060, 2016, pp. 224–237.
[99] H. Yan and W. Gui, "Efficient identity-based public integrity auditing of shared data in cloud storage with user privacy preserving," *IEEE Access*, vol. 9, pp. 45822–45831, 2021.
[100] Y. Chen, H. Liu, B. Wang, B. Sonompil, Y. Ping, and Z. Zhang, "A threshold hybrid encryption method for integrity audit without trusted center," *J. Cloud Comput.*, vol. 10, no. 1, p. 3, Dec. 2021.
[101] 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.
[102] X. Liu, W. Sun, W. Lou, Q. Pei, and Y. Zhang, "One-tag checker: Message-locked integrity auditing on encrypted cloud deduplication storage," in *Proc. IEEE Conf. Comput. Commun.*, May 2017, pp. 1–9, doi: 10.1109/INFOCOM.2017.8056999.
[103] X. Gao, J. Yu, W.-T. Shen, Y. Chang, S.-B. Zhang, M. Yang, and B. Wu, "Achieving low-entropy secure cloud data auditing with file and authenticator deduplication," *Inf. Sci.*, vol. 546, pp. 177–191, Feb. 2021.
[104] X. Gao, J. Yu, Y. Chang, H. Wang, and J. Fan, "Checking only when it is necessary: Enabling integrity auditing based on the keyword with sensitive information privacy for encrypted cloud data," *IEEE Trans. Dependable Secure Comput.*, early access, Aug. 24, 2021, doi: 10.1109/TDSC.2021.3106780.
[105] J. Yu and H. Wang, "Strong key-exposure resilient auditing for secure cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 8, pp. 1931–1940, Aug. 2017.

SMITA CHAUDHARI received the B.E. degree in computer engineering from North Maharashtra University, Maharashtra, India, in 1999, the M.E. degree in computer engineering from Mumbai University, India, in 2011. She is working as an Assistant Professor with Savitribai Phule Pune University. Currently, she is a Research Scholar of computer science and engineering with Koneru Lakshmaiah Education Foundation, Vaddeswaram, Andhra Pradesh, India. Her research interests include information security and cloud computing.

GANDHARBA SWAIN received the M.C.A. degree from the University College of Engineering, Burla, in 1999, the M.Tech. degree in CSE from the National Institute of Technology, Rourkela, India, in 2004, and the Ph.D. degree in computer science and engineering from Siksha 'O' Anusandhan University, Bhubaneswar, India, in 2014. Presently, he is working as a Professor with the Department of CSE, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Andhra Pradesh, India. He has more than 20 years of teaching experience, and authored two books and more than 90 research articles. Many of his articles are published in journals of reputed publishers like Elsevier, Springer, Hindawi, Wiley, and Inderscience. He has done extensive research work on digital image steganography, particularly addressed the various problems like fall off boundary problem, range mismatch problem, fall in error problem, detection by RS analysis, detection by PDH analysis, and tradeoff between PSNR and capacity. His other research interests include security, image tamper detection, and block chain technology.