Hindawi
Security and Communication Networks
Volume 2022, Article ID 6705948, 32 pages
https://doi.org/10.1155/2022/6705948

*Research Article*

# Certificateless Reliable and Privacy-Preserving Auditing of Group Shared Data for FOG-CPSs

**Manohar Sai Burra and Soumyadev Maity**

*Department of Information Technology, Indian Institute of Information Technology, Allahabad, Uttar Pradesh 211015, India*

Correspondence should be addressed to Manohar Sai Burra; rsi2018504@iiita.ac.in

Received 7 October 2021; Accepted 27 December 2021; Published 4 February 2022

Academic Editor: Ruhul Amin



FOG-enabled cyber-physical systems (FOG-CPSs) open new security challenges as the local edge devices are easier to compromise than a traditional cloud server. Remote data integrity checking (RDIC) plays an important role in safeguarding against data corruption from a storage server. Certificateless cryptography (CLPKC)-based RDIC schemes do not suffer from the drawbacks of the public key infrastructure (PKI)-based RDIC protocols. Most of the CLPKC-based RDIC schemes proposed in the literature deal with personal data. However, in a FOG-CPS, it is also important to audit a data file shared by a group of edge devices. Most of the existing group shared data auditing schemes lack mechanisms to defend against a semi-trusted data auditor applicable for a FOG-CPS scenario. In order to address these issues, in this paper, we propose a novel CLPKC-based group shared data auditing protocol tailored to the specific security requirements of a FOG-CPS. Besides, we perform a detailed cryptanalysis of two existing CLPKC-based privacy-preserving group shared data auditing schemes. The formal security analysis of our proposed protocol establishes metadata and data integrity proof unforgeability and claimed zero-knowledge privacy and reliability properties through rigorous proofs in the random oracle model setting. Performance evaluations establish the efficiency of our proposed protocol.

## 1. Introduction

After the introduction of FOG computing by Cisco [1], considering the capability of FOG to provide computing at the edge of the network while preserving the advantages of cloud computing such as ubiquity, decentralized management, and so on, FOG-assisted CPS (FOG-CPS) [2–8] is emerging as a new research domain to address the data management issues of CPS. The ability of FOG-CPS to provide services at the physical proximity of the network is very much useful in ensuring the low-latency and reliability requirements of the real-time jobs in a CPS. However, it opens new security challenges since the local edge devices are much easier to compromise by an attacker than a traditional cloud server, and hence it cannot be fully trusted. Resource limitations of the edge devices offer additional challenge in designing security protocols for a FOG-CPS.

Ensuring the *availability* of correct data at the right time is of utmost importance for a safety-critical CPS. Data auditing plays an important role in safeguarding against intentional or unintentional deletion or modification of data from a cloud data storage. Since the stored data are enormous, retrieving the complete file to check data integrity is impractical in communication and computation costs [9]. A large number of remote data integrity checking (RDIC) protocols have been proposed in the literature [10–40] which enable a cloud user to verify the integrity of data files stored at the cloud server remotely. An RDIC protocol involves two phases, namely, (a) preprocessing phase: in which, a cloud user, using its security credentials, generates some metadata from the data blocks of its data file and then outsources the metadata along with the data file to the cloud service provider (CSP), and (b) verification phase: in which, a trusted data auditor sends a challenge message to the CSP, who computes a proof of possession on the challenged data file using the stored metadata and data. The auditor verifies the proof generated by the CSP to determine the integrity of the stored data file [41]. Majority of the RDIC protocols

proposed in the literature [11, 12, 26–31, 33, 34, 38, 40] deal with personal data where a data file is solely owned by a single cloud user. However, in a FOG-CPS scenario, group of edge devices, either located in a specific geographical area or are dedicated for some specific task, forms a cluster, and data collected or generated by all edge devices in a cluster are kept in a single data file so that it can be efficiently accessed by all the others in the cluster. Auditing of such shared data file imposes additional challenges since the metadata for different blocks of the same data file may now be generated by different users (edge devices) using their own security credentials, and hence, while auditing, the CSP needs to aggregate all such heterogeneous pieces of information.

A good number of RDIC protocols have also been proposed by the researchers for auditing group shared data [10, 13–25, 32, 35–37, 39]. Some of these schemes, such as [10, 42], are dependent on the traditional public key infrastructure (PKI)-based system where a trusted certification authority (CA) is responsible for managing and distributing public key certificates of all the users and the certificate revocation lists (CRLs). Although PKI-based security protocols serve excellent in traditional networks such as the Internet, they are inefficient in a ubiquitous network like FOG-CPS. For a resource-constrained edge device, it is impractical to store and verify public key certificates associated with the large number of edge devices in a FOG-CPS. Moreover, obtaining up-to-date certificates and CRLs from the CA might be problematic for the edge devices since the devices might be deployed in an area having intermittent Internet connectivity. To address the problems associated with PKI, a number of shared data auditing protocols [13–15] have been proposed based on identity-based cryptography (IBC). In IBC [43], the public key of an user is derived from the publicly known identity of the user, and hence it does not suffer from the complexities of certificate management problems of a PKI-based system. However, in all IBC-based shared data auditing protocols [13–15] and in the protocols proposed in [44], where the private key of an user is generated by a trusted private key generator (PKG), the security of the whole system gets collapsed when the PKG gets compromised. This problem of IBC is known as the key escrow problem, in literature [45]. The certificateless cryptography (CLPKC) [45] was introduced to eliminate both the key escrow problem of IBC and the certificate management problem of PKI at the same time. In CLPKC, like IBC, the authenticity of the public key of an user is implicit; however, the full private key of an user is known only to that user. Recently, a number of CLPKC-based shared data auditing protocols have been proposed in the literature [16–24, 39] for traditional cloud storage systems. However, these protocols cannot be directly adopted for a FOG-CPS because of its discriminating characteristics as discussed below.

In FOG-CPS, the data storage and computing facilities are brought close to the CPS network for efficient bandwidth utilization and faster communication [46, 47]. Hence, implementing a data auditing protocol for a FOG-CPS allows the best utilization of FOG computing benefits when the data auditor is part of the edge network. This motivates us to design a protocol where we can opt an edge device from the CPS network to perform the data auditing task to improve FOG-CPS network latency and performance. However, since an edge device is susceptible to compromise either by physical tampering or through security attacks, we cannot fully trust the data auditor in this scenario. Here, we identify the two main security vulnerabilities that may arise from such a semi-trusted data auditor. First, ensuring *confidentiality* of the data is as important as ensuring its availability for a safety-critical CPS. Hence, a data auditing protocol designed for such scenario must ensure that no information regarding the content of the data file is leaked to the semi-trusted data auditor during its interactions with the storage server. The second concern is regarding the *reliability* of the auditing service provided by a semi-trusted data auditor. The reliability of the auditing service may be hampered due to two possible reasons, namely, (a) selfish nature of the auditor device and (b) collusion between CSP and the auditor. The data auditor is supposed to perform auditing duties on the request of the users (other edge devices) and periodically, as per the frequency specified in a service-level agreement (SLA). But, the edge device acting as auditor may behave selfishly to save its computational resources by either skipping or delaying the auditing tasks. Moreover, if the auditor is compromised by the storage server, that is, if there exists a collusion between the data auditor and the CSP, the auditor may disclose all the challenge vectors in advance to the CSP, using which, the storage server can precalculate data possession proofs and then delete/modify the data blocks of the data file. The threat of auditor-CSP collusion was first identified by Armknecht et al. for non-shared data auditing protocol [48], without data privacy against auditor, proposed for a traditional cloud storage system. In a traditional RDIC protocol, it might not be considered a serious threat as the possibility of collusion between the CSP and data auditor may be negligible. However, in a FOG-CPS scenario, where an edge device is supposed to work as the auditor and the storage server is also part of the edge network, the chances of an auditor being compromised by the storage server cannot be neglected.

The CLPKC-based group shared data auditing protocols proposed in [16–22] trust the data auditor on the data *confidentiality* and hence do not provide any data privacy mechanisms against the auditor. In these protocols, the auditor can extract the data blocks from the information received from the cloud as part of the data auditing process. Recently, a few privacy-preserving CLPKC-based shared data auditing protocols have been proposed by the researchers [23, 24, 39] for traditional cloud storage systems. But, none of these schemes can ensure the *reliability* criteria of the auditing service against a semi-trusted data auditor in a FOG-CPS, as defined above.

Moreover, the schemes in [23, 24] employ a weaker notion of data privacy which only ensures that a compromised auditor will not be able to extract any information regarding the challenged data blocks from the data possession proofs received from the CSP. However, the data possession proofs sent by the CSP leaks information regarding the metadata corresponding to the challenged

blocks, since the CSP does not employ any masking technique to hide the part of a data possession proof computed from the metadata. A compromised data auditor can use this leaked information to perform an offline guessing attack to find a valid message block corresponding to the leaked metadata.

Furthermore, our cryptanalysis shows that the metadata generation mechanisms used in [23, 39] cannot fully protect the integrity and authenticity of metadata, corresponding to a data block, generated by an user. In these schemes, an attacker having access to a valid data block along with its corresponding metadata can easily replace the data block with a forged block. Uploading the data and the metadata blocks using a confidential channel between the user and the CSP can prevent access to the metadata blocks against unauthorized entities. However, we cannot prevent the CSP itself to get access to the data and metadata blocks as those are essential for the calculation of correct data possession proofs by the CSP. A compromised CSP can always launch the above-mentioned attack by replacing the public key of the user, which is also known as type-I attack in CLPKC literature [45]. To mitigate this problem, an user may explicitly authenticate its public key using some mechanisms such as digital signature. However, this would contradict with the main purpose of CLKPC, which is to authenticate public keys implicitly without use of any explicit signature or certificates.

In this paper, our contribution is twofold. First, we have performed a detailed cryptanalysis of the CLPKC-based privacy-preserving shared data auditing schemes proposed in [23, 39] to pinpoint the exact vulnerabilities in the metadata generation mechanisms used in these protocols. Second, to address the limitations of the existing research as discussed above, we have proposed a novel group shared data auditing protocol tailored to the specific security requirements of a FOG-CPS. The proposed data auditing protocol takes advantage of the localized storage and computing facilities available at the edge of the FOG-CPS network by delegating an edge device geographically close to the storage resources to perform the data auditing task. Our protocol ensures *zero-knowledge data privacy* [49] against a semi-trusted data auditor where the auditor can learn no information regarding either a data block or a metadata block corresponding to a data file while auditing for the integrity of the file. The proposed protocol is also reliable against a semi-trusted data auditor. It enables a user to verify the data auditing reports generated by the auditor and detect any procrastination of auditing tasks done by the auditor. In order to defend against possible collusion of auditor and the CSP in advance on the challenge vectors, we ensure that the challenge vectors are generated from a time-varying and verifiable universal source of randomness. We have designed our protocol using the CLPKC mechanism, and hence it is free from both the certificate management problem associated with traditional PKI as well as the key escrow problem associated with IBC. We have performed a detailed security analysis of our proposed protocol. We have proved that the metadata generated by our protocol are unforgeable even by a CSP. To be specific, the metadata generation mechanism used in our protocol is secure against both type-I and type-II attacks in the random oracle model setting, as per the security notions defined in certificateless cryptography [45]. We have also established proofs in support of the claimed zero-knowledge privacy (ZKP) and the reliability properties of our proposed protocol. The comparative performance evaluations establish the efficiency of our proposed protocol.

The rest of the paper is organized as follows. We review the literature in Section 2, which covers generic remote data integrity checking protocols, and the research related to FOG-CPSs. In Section 3, we provide the necessary preliminary concepts and security model required for the proposed work. In Section 4, we show the detailed cryptanalysis of the schemes in [23, 39]. In Section 5, we provide detailed information about the entities involved, data structures and files used, and the assumptions underlying our proposed scheme. In Section 6, we describe in detail the proposed basic CLS-RDIC scheme. In Section 7, we describe in detail the proposed CRPPA scheme. Section 8 presents a detailed security analysis of both the proposed basic CLS-RDIC and CRPPA schemes. In Section 9, we provide qualitative and experimental computation cost comparisons of our proposed CRPPA scheme with some other existing schemes. Section 10 concludes the paper.

Table 1 shows the list of acronyms and their abbreviations used throughout the entire paper.

## 2. Related Work

In this section, we first provide a brief summary of the research done in the field of FOG-CPSs, their importance, and the need for security in FOG-CPSs. After this, we survey the existing remote data integrity checking (RDIC) protocols proposed for cloud storage services and discuss their applicability for FOG-CPSs.

*2.1. FOG-CPSs.* There is a massive increase in the number of cyber-physical system (CPS) devices connecting to the Internet to access cloud computing services. With the current growth rate, the available bandwidth of the cloud computing system will not be able to cater to the growing demand, and this situation increases the latency of the cloud computing network [47]. To overcome the bandwidth and latency issues in a traditional cloud computing environment, Cisco proposed a new cloud computing paradigm called FOG computing [1]. In FOG computing, FOG nodes are deployed in less centralized geographically close locations to end-user devices (CPS devices) to provide storage and computing facilities, improving bandwidth and latency issues. Since the FOG-CPS devices are less centralized, they are prone to physical tampering and security attacks. Therefore, there is a strong security requirement regarding the confidentiality, integrity, and availability of data for the FOG-CPS [2, 50, 51].

Several works have been proposed to provide security for the critical data of CPS devices in FOG-CPSs. The authors in [3] employed a modified version of the attributed-based encryption technique to provide secure communication between the FOG-CPS devices even when the certificate authority is comprised. The scheme in [4] employs a

Table 1: List of acronyms.

| Acronym | Abbreviation |
|---|---|
| CPS | Cyber-physical system |
| RDIC | Remote data integrity checking |
| CSP | Cloud service provider |
| PKI | Public key infrastructure |
| PKC | Public key cryptography |
| CA | Certification authority |
| CRL | Certificate revocation list |
| IBC | Identity-based cryptography |
| PKG | Private key generator |
| CLPKC | Certificateless public key cryptography |
| SLA | Service-level agreement |
| PDP | Provable data possession |
| KGC | Key generation center |
| TPA | Third-party auditor |
| CLS | Certificateless signature |
| CDH | Computational Diffie–Hellman |
| DDH | Decisional Diffie–Hellman |
| DL | Discrete log |
| PPT | Probabilistic polynomial time |
| DPT | Deterministic polynomial time |
| ROM | Random oracle model |
| CLS-RDIC | Certificateless signature-based remote data integrity checking |
| CRPPA | Certificateless reliable privacy-preserving auditing |
| Adv1 | Adversary 1 |
| Adv2 | Adversary 2 |
| Adv3 | Adversary 3 |
| Adv4 | Adversary 4 |
| Adv5 | Adversary 5 |
| Adv6 | Adversary 6 |
| PBC | Pairing-based cryptography |

lightweight signature mechanism to generate group signatures, which helps to handle batch authentication of edge nodes efficiently. The authors in [5] provided a framework for the user and data privacy in the FOG-enabled smart city network. The schemes in [6, 7] provide mechanisms for privacy-preserving secure data aggregation mechanisms for FOG-CPSs. In [8], FOG nodes store cache data to improve the edge network latency; the scheme employs a modified version of a Merkle hash tree data structure to enable end users to audit the integrity of their cached data. Chen et al. [52] proposed a lightweight pairing-free CLPKC-based matchmaking encryption mechanism for secure communication in the Internet of things (IoT) to prevent leakage of information and identity forgery. Further, the data sharing mechanisms in scheme [52] are efficient, and the authors have provided formal security proofs of the protocols based on the standard hard assumptions. The scheme in [53] provides mechanisms to establish a group shared key for secure data sharing and storage in CSP and can be adopted to FOG-CPS networks. The authors in [54] designed a CLPKC-based searchable encryption technique for an honest-but-curious platform to provide secure data sharing in an industrial IoT environment against keyword guessing attacks. The scheme in [54] is secure against a malicious key generation center, provides user data security and access control on search, and supports on-demand user revocation.

Further, many FOG-CPS schemes have been proposed to take advantage of blockchain technology, but the devices in the FOG-CPS are resource-limited, and they cannot directly adopt the traditional blockchain. Therefore, the works in [55–58] developed scalable public blockchain techniques for FOG-CPSs. The authors in [57] proposed a group chain concept which utilizes a two-chain structure to build the blockchain for regulating the behaviour of groups and preventing malicious nodes from performing joint attacks. The authors in [58] constructed a Ethereum blockchain for the resource-constrained device to provide security of the data generated from the CPS devices. They even designed a secure networked clock for blockchain data synchronization for the non-real-time CPS devices.

*2.2. Remote Data Integrity Checking (RDIC) Protocols.* Blum et al. [59] introduced the first RDIC scheme in which the data owner maintains a small amount of metadata corresponding to the actual outsourced data, and using these metadata, the data owner checks if the actual data stored remotely are intact or corrupted. Ateniese et al.'s RDIC scheme [9] generates probabilistic provable data possession (PDP) proofs where the data verifier checks the data integrity of a file by only verifying the validity of a few random data blocks of the file. In the probabilistic PDP schemes, the cloud user splits the data file into several smaller blocks, and each block is denoted with an index. For each of the data blocks, the cloud user computes metadata using its security credentials. The cloud user then uploads both the data file and metadata to the cloud server and deletes the original data file from its storage. Suppose the cloud user wants to verify the integrity of the remote data file. The user initiates a challenge-response protocol to the CSP, in which the user sends a challenge set that contains some random indices corresponding to the data blocks of the data file. The CSP computes a PDP proof corresponding to the challenge set and sends the PDP proof as a response to the cloud user. The cloud user verifies the PDP proof and determines whether the integrity of the remote data file is intact or corrupted. Juels et al. [60] proposed a symmetric-key-cryptography-based RDIC scheme called the proof of retrievability (POR) that utilizes efficient error-coding techniques. The main drawback of the scheme is that it supports only private verification. The authors in [32] provided a POR mechanism based on BLS signature [61] that supports public verification but implicitly suffers from data privacy issues against the public verifier.

The majority of public key cryptography-based RDIC protocols [11, 12, 26–30, 33, 34, 37, 40] developed until now employ PKI or IBC [43]. The PKI-based RDIC schemes [11, 30, 37] require the user to verify the certificate of the other user for authentication each time before initiating a communication. The PKI has a very high computational and communication overhead, and introducing the PKI into an RDIC scheme certainly overloads the users in terms of required computation power. On the other hand, the ID-based RDIC schemes [12, 26–29, 33, 34, 40] are efficient in terms of computation and communication overhead than PKI.

Zhao et al. [33] introduced the first ID-based RDIC scheme that ensures public verification of remote data integrity and provides privacy against a third-party auditor (TPA). Wang et al. [34] developed the first IBC-based RDIC for the multi-cloud storage model. Yu et al. [12] proposed a secure symmetric key agreement between the three parties of a public RDIC protocol, namely, the cloud user, the public verifier (auditor), and the CSP, to provide data privacy against the public verifier efficiently. Still, their tag-generation mechanism is susceptible to the attacks shown in Section 4 and hence is not secure. Zhang et al. [25] improved the scheme in [48], but it lacks data privacy. The IBC-RDIC schemes have an inherent key escrow problem where the private key of a user is generated using the master secret key of the PKG. The compromise of the master secret key of the PKG will result in a compromise of the security credentials of all the users. Thus, the encrypted communication under a user's public key gets exposed. Certificateless public key cryptography (CLPKC) [45] eliminates the burden of certificate management of PKI and the key escrow problem of IBC. Many RDIC schemes [16, 18, 19, 22, 23, 35, 36, 39, 62–66] have been proposed based on CLPKC. He et al.'s [64] certificateless public auditing scheme for cloud-assisted WBANs is not a privacy-preserving scheme. Kang et al. [65] improved He et al.'s [64] scheme to provide privacy-preserving property, but it lacks public verifiability. Sasikala et al. [35] proposed a certificateless RDIC scheme for the cloud using lattices, but it was proved to be insecure in [36].

A number of existing RDIC schemes in the literature [11, 12, 26–30, 33, 34, 40] deal with personal data where a data file is solely owned by a single cloud user. However, in a FOG-CPS scenario, group of edge devices, either located in a specific geographical area or are dedicated for some specific task, forms a cluster, and data collected or generated by all edge devices in a cluster are kept in a single data file so that it can be efficiently accessed by all the others in the cluster. Auditing of such shared data file imposes additional challenges since the metadata for different blocks of the same data file may now be generated by different users (edge devices) using their own security credentials, and hence, while auditing, the CSP needs to aggregate all such heterogeneous pieces of information. A number of schemes that support group shared data include [10, 13–25, 32, 35–37, 39].

We now focus on the CLPKC-based group shared data schemes considering the advantages of CLPKC over IBC and PKI. The CLPKC-based group shared data auditing protocols proposed in [16–22, 62, 63] are not data privacy-preserving. Further, Li et al.'s [62] metadata generation mechanism suffers from metadata forgeability. Jaya et al.'s [63] scheme on certificateless multi-replica RDIC considers the fully trusted KGC and TPA. In reality, the TPA or KGC should not gain any sensitive or confidential information about the cloud user. A few recently proposed privacy-preserving CLPKC-based group shared data auditing protocols [23, 24, 39] employ traditional cloud storage systems, and these schemes cannot ensure the *reliability* criteria of the auditing service against a semi-trusted data auditor. The schemes in [23, 24] employ a weaker notion of data privacy where the data possession proofs sent by the CSP leak information regarding the metadata corresponding to the challenged blocks. Further, the schemes [23, 39] are susceptible to metadata forgery attacks discussed in Section 4.

In summary, the adoption of FOG computing across various CPS domain areas to address the data management issues is exponentially growing, and ensuring the availability of correct data at the right time is of utmost importance for a safety-critical CPS. Thus, the need arises for remote data integrity checking (RDIC) mechanism for FOG-CPSs. However, many RDIC schemes proposed in the literature employ ID-based cryptography (IBC) with an inherent key escrow problem [12, 26–29, 33, 34, 40] or depend on public key infrastructure (PKI) for key management, which is expensive [11, 30, 37]. Certificateless cryptography (CLPKC)-based RDIC schemes overcome the drawbacks of IBC and PKI mechanisms. However, several CLPKC-based RDIC schemes have been proposed in the literature to deal only with personal data. But, in a FOG-CPS scenario, a group of edge devices performs some specific task and forms a cluster. Data generated by all edge devices in a cluster are kept in a single data file so that all others can efficiently access it in the cluster. Hence, RDIC schemes [11, 12, 26–30, 33, 34, 40] designed for personal data auditing cannot be adopted for FOG-CPS scenarios. Although there are few existing CLPKC-based group shared data auditing schemes, none of these schemes [16–22, 62, 63] provide mechanisms to ensure confidentiality of user data and reliability of data auditing task against a semi-trusted data auditor at the same time, which are essential for a FOG-CPS. Therefore, we have proposed a CLPKC-based group shared data auditing protocol that ensures zero-knowledge data privacy against a semi-trusted data auditor where the auditor can learn no information regarding either a data block or a metadata block corresponding to a data file while auditing for the integrity of the file. At the same time, the proposed protocol also ensures the reliability of the auditing task by a semi-trusted data auditor where the auditor cannot collude with the CSP in advance on the random challenge vectors used as a part of RDIC and cannot delay or skip the auditing tasks. To the best of our knowledge, integrating both the confidentiality and the reliability objectives, as defined above, have not been addressed by any of the existing CLPKC-based group shared data auditing protocols.

## 3. Preliminaries

In this section, we discuss the preliminary concepts which have been used as the building blocks to design the proposed protocol. At first, we introduce the negligible function, the general forking lemma [67], and the concepts of bilinear map and computational Diffie–Hellman (CDH) problem. We then provide the outline of a certificateless signature (CLS) scheme which is a part of certificateless cryptography (CPLKC) [45] used in our protocol for metadata generation purpose. In the last two subsections, we first provide an overview of a CLS-based RDIC scheme and then describe the standard adversary model used by the researchers to analyze the security of these schemes.

*3.1. Negligible Function ($\xi$).* A function $\xi(x)$: $\mathbb{N} \longrightarrow \mathbb{R}$ is called negligible, if $\forall c \in \mathbb{N}$, $\exists Q_c \in \mathbb{N}$ such that $\forall x > Q_c$, $|\xi(x)| < 1/x^c$.

In cryptography, a scheme is said to be "provably secure," if the probability of security failure is a negligible function of the input $x$ which is the security parameter (cryptographic key length) of the security scheme.

*3.2. General Forking Lemma [67].* Let $x$ be a set of global parameters and $H$ be the co-domain of a random oracle function RO, with $|H| \geq 2$. Let $A$ be a randomized algorithm which takes $x$ and a set $\{h_1, \ldots, h_v\}$ as inputs where each $h_i$ value $(\forall i = 1, \ldots, v)$ is randomly drawn from the set $H$ for some $v \geq 1$. The algorithm outputs a pair $(I, \theta)$ where $I \in \{0, \ldots, v\}$ and $\theta$ is called *side output*.

*Acceptance*. We say that the algorithm $A$ has accepted the inputs if it uses one of the values in $\{h_1, \ldots, h_v\}$ to generate its side output $\theta$. If it accepts, then $I$ is the index of the inputs $\{1, 2, \ldots, v\}$. If it does not use any of the values in $\{h_1, \ldots, h_v\}$ to generate $\theta$, then $I = 0$.

*Acceptance Probability*. The acceptance probability for the above-mentioned randomized algorithm $A$ is defined as $\Pr[\text{acc}] = \Pr[I \neq ]$.

*Forking*. The forking protocol on algorithm $A$, denoted as $\text{Fork}_A(x)$, is as shown below:

(1) \$ $\leftarrow$ Random Tape;
(2) $\{h_1, \ldots, h_v\} \overset{R}{\leftarrow} H$;
(3) $(I, \theta) \leftarrow A(x, h_1, \ldots, h_v; \$)$;
(4) If $(I = 0)$ then return 0;
(5) $\{h'_I, \ldots, h'_v\} \overset{R}{\leftarrow} H$;
(6) $(I', \theta') \leftarrow A(x, h_1, \ldots, h_{I-1}, h'_I, \ldots, h'_v; \$)$;
(7) If $((I \neq I')$ or $(h_I = h'_I))$ then return 0;
(8) else return 1;

The above protocol $\text{Fork}_A(x)$ actually consists of two rounds. In the first round, we input the value of $x$, a randomly chosen set of $\{h_1, \ldots, h_v\}$ values, and a random tape \$. The protocol aborts and returns 0 if algorithm $A$ does not accept $(I = 0)$ the inputs in the first round. In the second round, we provide the same $x$ and \$ values to algorithm $A$; however, the input set $\{h_1, \ldots, h_v\}$ is updated keeping $h_1$ to $h_{I-1}$ the same. The second round aborts and returns 0 if either the acceptance indices $I$ and $I'$ of the two rounds are different or if the randomly generated elements $h_I$ and $h'_I$ are same. Otherwise, the protocol $\text{Fork}_A(x)$ returns 1. It can be noted that the main objective of the forking protocol is to generate two different side output values $\theta$ and $\theta'$ using two different random oracle values $h_I$ and $h'_I$.

*Forking Probability*. The forking probability for the above protocol $\text{Fork}_A(x)$ is defined as $\Pr[frk] = \Pr[\text{Fork}_A(x) = 1]$ where $x$ is a randomly generated global parameter $x$. Bellare et al. [67] proved that $\Pr[frk] \geq \Pr[\text{acc}] \cdot ((\Pr[\text{acc}]/q) - (1/|H|))$.

*3.3. Bilinear Map.* Consider two cyclic multiplicative groups $G_1$, $G_2$, for $|G_1| = |G_2| = q$, where $q$ is a large prime integer. A bilinear map [61] is a function of the form $\hat{e}$: $G_1 \times G_1 \longrightarrow G_2$, that satisfies the following properties.

*Bilinearity.* $\forall m, n \in {F_q}^*, \forall X, Y \in G_1$, $e(X^m, Y^n) = e(X, Y)^{mn}$.

*Non-Degeneracy.* $e(X, Y) \neq 1$ for all values of $X$ and $Y$, where 1 is the identity element of $G_2$.

*Computability*. There exists a polynomial-time algorithm to compute $\hat{e}$.

The bilinear map can be implemented using a Weil pairing [43] or a Tate pairing [68].

*3.4. Computational Diffie–Hellman (CDH) Problem.* Let $G_1$ be a cyclic multiplicative group of order $q$, and let $g$ be the generator of the group. The CDH problem [69] in $G_1$ is as follows: given $\langle q, G_1, g, h_1, h_2 \rangle$ for some $h_1, h_2 \in G_1$, where $h_1 = g^\alpha$ and $\alpha \in {Z_q}^*$, compute ${h_2}^\alpha$.

The decision version of the above problem, also known as the decisional Diffie–Hellman (DDH) problem, is defined as follows: given $\langle q, G_1, g, h_1, h_2, h_3 \rangle$ for some $h_1, h_2$, and $h_3 \in G_1$, where $h_1 = g^\alpha, h_2 = g^\beta$, and $h_3 = g^\gamma$, for $\alpha, \beta, \gamma \in {Z_q}^*$, decide whether $g^{\alpha\beta} = g^\gamma$. The DDH problem is easy to decide in $G_1$ if there exists a bilinear map in group $G_1$. However, till date, there exists no efficient algorithm to compute ${h_2}^\alpha$ in polynomial time, and thus CDH is believed to be a hard problem in group $G_1$.

*3.5. Certificateless Signature Scheme.* A certificateless signature (CLS) scheme is part of the CLPKC mechanism [45], which is used to generate and verify digital signatures without the use of any public key certificates. In order to defend against the key escrow problem, CLPKC uses the concept of partial keys where part of the private key of an user is generated by a trusted key generation center (KGC) and the rest of the part is generated and known to the user only. A CLS mechanism consists of the following seven algorithms:

(i) Setup: it is a probabilistic polynomial-time (PPT) algorithm that takes a security parameter $k$ as input and outputs public parameter-list params and a secret master key $(\alpha)$ for the key generation center (KGC). The KGC executes the algorithm and announces the public parameters params to all the participating entities.

(ii) Partial-Private-Key-Extract: the KGC runs this algorithm at the request of the user to generate a partial-private key for it. The user sends its identity $ID_u$ to KGC as part of the request. The input to this deterministic polynomial-time (DPT) algorithm is the identity $ID_u$, public

parameters params, and master secret key $\alpha$, and it outputs the partial-private-key $D_u$. The KGC transfers the partial-private-key $D_u$ to the corresponding user through a secure side channel.

(iii) Set-Secret-Value: this algorithm is executed by a user $u$. The input to this PPT algorithm is params and $ID_u$ and it outputs a secret value $\beta_u$ which is kept secret to the user $u$ only.

(iv) Private-Key-Gen: a user $u$ runs this PPT algorithm to generate its private key as $\langle D_u, \beta_u \rangle$.

(v) Set-Public-Key: user $u$ runs this DPT algorithm with its identity $ID_u$, full private key $\langle D_u, \beta_u \rangle$, and the public parameters params as the inputs to generate its public key $P_u$. User $u$ announces its public key publicly without the need for any certificate.

(vi) Sign: this PPT algorithm is executed by a user $u$ to generate a signature on a given message $m \in \{0,1\}^*$. The algorithm takes $m$, $ID_u$, the full private key $\langle D_u, \beta_u \rangle$, and the public parameters params as the inputs and produces $\text{Sig}_m$ as the signature.

(vii) Verify: the input to this DPT algorithm is $m$, $\text{Sig}_m$, params, $ID_u$, and $P_u$ and outputs accept if the signature is valid, or reject otherwise. The algorithm can be executed by any entity having the required inputs in its hand.

A CLS scheme can be considered as secure only if there is no existential forgery possible for signatures, either by (a) any compromised user who can try to achieve this by replacing the public key of another user or (b) even by a compromised but passive attacker KGC. Type-I and type-II attackers are defined in Section 3.7 which describes these two attack scenarios in detail.

*3.6. Overview of a CLS-RDIC Scheme.* A CLS signature-based remote data integrity checking (CLS-RDIC) scheme consists of four types of entities (roles), namely, (i) cloud user (data owner), (ii) the cloud service provider (CSP), (iii) a trusted third-party auditor (TPA), and (iv) the KGC. It consists of eight algorithms: the CLS mechanism, as described in Section 3.5, contributes the first four algorithms, namely, setup, partial-private-key-extract, private-key-gen, and set-public-key. The rest of the algorithms of CLS-RDIC are briefed below:

(i) Tag-Gen: a cloud user runs this algorithm with a data block $m$, corresponding authenticating information $w = (\text{file-id} \parallel \text{block} - \text{index} \parallel \text{public info. of user})$, params , and $ID_u$ as inputs to generate a tag $(\sigma)$, where $m \in \mathcal{M}$, for $\mathcal{M}$ is the message space. For every data block uploaded in the data file $ID_F$, the value $w$ is updated in a public log file. Set of all tags corresponding to all the blocks of the data file constitutes the metadata, stored by the CSP along with the original data file.

(ii) Challenge: the TPA runs this algorithm with $c$ (number of data blocks challenged), $ID_F$ (id of the data file), params, and inputs from a pseudorandom source to generate a challenge vector called chal. The chal contains information regarding the specific blocks challenged, along with some random values, for which the CSP needs to generate a data integrity proof.

(iii) Proof-Gen: the CSP runs this DPT algorithm when it receives a challenge request chal from the TPA. The inputs to this algorithm are chal, params, $\{\sigma_i\}$, and $\{m_i\}$ (all tags and data blocks corresponding to the challenge), and it outputs data integrity proof $P$ for the challenged blocks.

(iv) Proof-Ver: the TPA runs this DPT algorithm with the inputs $P$, chal, $ID_F$, and params and outputs accept if the proof $P$ is valid, or reject otherwise.

The security requirements for a CLS-RDIC scheme are discussed in detail in the following subsection.

*3.7. Security Model for a CLS-RDIC Scheme.* A CLS-based RDIC scheme is considered to be secure only when (a) the underlying CLS mechanism is secure, *i.e.*, when the generated tags are unforgeable by any internal or external attacker, and (b) the CSP is unable to cheat the TPA, *i.e.*, when CSP can pass the auditing test only if it stores all the data blocks of a file correctly. According to the CLPKC literature [45], the tag unforgeability is again considered against two different types of attackers, namely, (i) type-I attacker: any internal/external attacker except the KGC, and (ii) type-II attacker: a compromised KGC. A compromised CSP that aims to cheat the TPA is termed as type-III attacker. We use the random oracle model (ROM)-based analysis [70] to validate the security of a CLS-RDIC scheme against the three types of attackers defined above. In the random oracle model-based analysis, the hypothetical attacker's power and the proposed protocol's ultimate security objectives are modeled as a game between the attacker and a challenger. The games corresponding to the above three types of adversaries are defined in the following three subsections.

*3.7.1. Type-I Adversary (Adv1).* In type-I attack, the adversary may be an external attacker or an internal attacker such as a compromised user/users or even the CSP. The objective of the attacker is to generate a valid tag for a forged data block for a data file corresponding to a victim user. It is quite obvious to assume that the adversary has access to all the public parameters used in the protocol including the valid public keys of all genuine users. Since the adversary may be a compromised user/users, it may have valid partial-private-keys and personal secret-values corresponding to a set of legitimate users. Additionally, an adversary may have access to valid tags corresponding to a set of data blocks, generated by the victim user using its valid security credentials. The adversary may obtain this information either by eavesdropping to the communication between the victim user and CSP during metadata upload phase or when the adversary is the CSP itself. However, we assume that, in any case, the personal secret value of the victim user is not

known to the adversary. Since there is no use of public key certificates in a CLS-RDIC scheme, attacker may try to replace the public key of the victim user by advertising a false public key claiming it as the public key of the victim user.

Game-1, defined below, between a challenger ($C$) and an adversary Adv1, models the hypothetical type-I attacker discussed above. The game consists of the following phases.

*Setup*. The challenger $C$ runs the setup algorithm of the CLS-RDIC scheme to generate the master secret key msk and public parameters params. $C$ forwards only the params to Adv1 and keeps with itself msk.

*Queries*. In this phase, Adv1 can adaptively make the following queries to the challenger $C$.

(i) Hash Query: Adv1 can query for hash values of any binary string, related to any hash function used for tag generation in the CLS-RDIC scheme, to the challenger $C$. $C$ is required to send back the correct output.

(ii) Partial-Private-Key Query: Adv1 can submit any identity $ID_u$ to $C$. $C$ runs the Partial-Private-Key-Extract algorithm of the CLS-RDIC scheme to provide the corresponding partial private key to Adv1.

(iii) Secret-Value Query: Adv1 can submit any identity $ID_u$ to $C$. Challenger $C$ runs the Set-Secret-Value algorithm of the CLS-RDIC scheme to provide the corresponding secret value to Adv1.

(iv) Public-Key Query: Adv1 can submit any identity $ID_u$ to $C$. Challenger $C$ runs the Set-Public-Key algorithm of the CLS-RDIC scheme to provide the corresponding public key to Adv1.

(v) Public-Key-Replacement Query: Adv1 can submit any identity $ID_u$ and a public key $P_u'$ of its choice to $C$. Challenger $C$ updates the public key corresponding to $ID_u$ as $P_u'$.

(vi) Tag-Gen Query: Adv1 can submit any tuple $(m, w, ID_u)$ of its choice to $C$. Challenger $C$ runs the Tag-Gen algorithm of the CLS-RDIC scheme to generate the corresponding tag $\sigma$ and sends the tag to Adv1.

*Forge-Tag*. Finally, Adv1 outputs a tuple $(\sigma', m', w', ID_u')$, where $\sigma'$ is a forged tag on the forged data block $m'$ with authenticating information $w'$ for user $ID_u'$.

Adv1 wins game-1 only if (i) $\sigma'$ is valid tag corresponding to the forged data block $m'$ and the authenticating information $w'$ for the targeted user $ID_u'$, (ii) Adv1 has not queried for the secret value corresponding to the identity $ID_{u'}$, (iii) Adv1 does not query for the partial-private-key and replace public key simultaneously for the identity $ID_u'$, and (iv) Adv1 has not queried the tag for the $(m', w', ID_u')$ tuple at any stage.

*3.7.2. Type-II Adversary (Adv2).* In type-II attack, we consider a compromised KGC as a passive attacker in the sense that it may try to forge a data block for a victim user but without any attempt to replace the public key of the victim user. Like type-I attacker, we assume that the adversary has access to all the public parameters used in the protocol including the valid public keys of all genuine users. Additionally, since the attacker is the KGC itself, it has access to the master secret key msk of the KGC. Like type-I attack scenario, adversary may have access to valid tags corresponding to a set of data blocks, generated by the victim user using its valid security credentials. Similar to type-I attacker, we assume that the personal secret value of the victim user is not known to the adversary. However, unlike a type-I attacker, the adversary does not try to replace the public key of the victim user.

Game-2, defined below, between a challenger ($C$) and an adversary Adv2, models the hypothetical type-II attacker discussed above. The game consists of the following phases.

*Setup*. Challenger $C$ runs the setup algorithm of the CLS-RDIC scheme to obtain the master secret key msk and public parameters params. $C$ forwards both msk and params to Adv2.

*Queries*. In this phase, Adv2 can make the following queries adaptively to challenger $C$.

(i) Hash Query: this query is the same as the hash query in game-1.

(ii) Secret-Value Query: same as the Secret-Value query defined in game-1.

(iii) Public-Key Query: same as the Public-Key query defined in game-1.

(iv) Tag-Gen Query: same as the Tag-Gen query defined in game-1.

It can be noted here that we do not include the Partial-Private-Key Query for Adv2 since it can calculate the partial-private-key of any user using msk it has in its possession.

*Forge-Tag*. Finally, Adv2 outputs a tuple $(\sigma', m', w', ID_u')$, where $\sigma'$ is the forged tag on the forged data block $m'$ with authentication information $w'$ for user $ID_u'$.

Adv2 wins game-2 only if (i) $\sigma'$ is valid tag corresponding to the forged data block $m'$ and the authenticating information $w'$ for the targeted user $ID_u'$, (ii) Adv2 has not queried for the secret value corresponding to the $ID_u'$, and (iii) Adv2 has not queried for the tag for the $(m', w', ID_u')$ tuple at any stage.

It can be noted that we consider only passive attack for a compromised KGC as providing security against active attacks launched by a compromised certificate authority (CA) in a traditional public key infrastructure (PKI)-based protocol is also equally difficult. So, a malicious-but-passive KGC, in this attacker model, may try to forge a tag for a user but without trying to replace the user's public key. Thus,

security against type-II attacker establishes the same level of confidence on a KGC-based CLPKC system as that of a traditional CA-based PKI system.

*3.7.3. Type-III Adversary (Adv3).* In type-III attack, we consider a compromised CSP as the possible attacker. However, unlike type-I attack, where the objective of the attacker is to generate tag for a forged data block, in type-III attack, the attacker's objective is to generate a valid data possession proof for a given random challenge without storing all the data blocks of the corresponding data file properly. Like type-I and type-II attacks, we assume that the adversary in this case also has access to all the public parameters used in the protocol including the valid public keys of all genuine users. Since type-III attacker is a compromised CSP, we need to assume that the adversary has got access to all the valid tags corresponding to all the data blocks of the challenged data file. It has also access to all the data blocks of the file only except one of the challenged blocks which, we assume, has been deleted or modified by the attacker.

Game-3, defined below, between a challenger ($C$) and an adversary Adv3, models the hypothetical type-III attacker discussed above. The game consists of the following phases.

*Setup*. The challenger $C$ runs the setup algorithm of the CLS-RDIC scheme to obtain the master secret key msk and public parameters params. In addition, $C$ generates a random data file with $n$ number of data blocks and file-id $ID_F$. The challenger forwards only the params to Adv3 and keeps with itself msk and the data file.

*Queries*. In this phase, Adv3 can make the following queries adaptively to the challenger $C$.

(i) Hash Query: this query is same as the hash query in game-1.

(ii) Data-Block Query: Adv3 can query for any block index $i$ of the data file $ID_F$. Challenger $C$ retrieves corresponding data block from the data file and forwards it to Adv3.

(iii) Public-Key Query: same as the Public-Key query defined in game-1.

(iv) Tag-Gen Query: Adv3 can submit any block index $i$ of the data file $ID_F$ to $C$. Challenger $C$ runs the Tag-Gen algorithm of the CLS-RDIC scheme to generate the corresponding tag $\sigma_i$ and sends it back to Adv3.

*Challenge*. $C$ runs the Challenge algorithm of the CLS-RDIC scheme to generate a challenge vector chal for data file $ID_F$ and forwards chal to Adv3.

*Forge-Proof*. Adv3 outputs a data possession proof $P$ corresponding to the given challenge chal.

Adv3 wins the game-3 only if (i) $P$ is valid data possession proof corresponding to the given challenge chal and (ii) at least one of the challenged blocks does not appear in any Data-Block query from Adv3.

## 4. Cryptanalysis of Existing Schemes

In this section, we perform a detailed cryptanalysis of the CLPKC-based privacy-preserving shared data auditing protocols proposed in [23, 39]. Specifically, we analyze the tag unforgeability of these schemes against type-I and type-II attackers as defined in Sections 3.7.1 and 3.7.2. Since the tag-generation mechanisms used in these two protocols are very similar, we mainly discuss the vulnerabilities for one of these schemes [23] and highlight the similarities of the other scheme [39] with [23]. In the following, we first provide a brief description of the tag-generation mechanism used in the protocol [23] and then perform its cryptanalysis.

*4.1. Brief Description of Jaya et al.'s Scheme [23].* The algorithms involved in the tag generation of the protocol proposed by Jaya et al. [23] are described as follows. The notations used in describing these algorithms are little bit different from the notations used in [23]. We use the notations same as the notations that we use in our proposed protocol description, so that they can be compared easily.

*Setup*. The KGC runs the setup algorithm and performs the following steps:

(1) Selects $q, g, G_1, G_2, e$ as described in our Preliminaries section.

(2) Chooses a master secret key $\alpha \in Z_q^*$ at random and sets the KGC's public key $P_{\text{pub}} = g^{\alpha}$. The KGC also chooses another random value $\gamma_o \in Z_q^*$ and sends it confidentially to the group manager (GM).

(3) Selects three secure hash functions $H_1: \{0,1\}^* \longrightarrow G_1$, $H_2: \{0,1\}^* \longrightarrow G_1$, and $h(.): G_1 \longrightarrow Z_q^*$.

The KGC publishes the public parameters $\text{params} = \langle G_1, G_2, q, \hat{e}, g, P_{\text{pub}}, H_1, H_2, h(.)\rangle$.

*Join*. Any user entity $U_j$, for $1 \le j \le d$, which is part of network and wants to join a group, requests the GM. The GM after reviewing the request generates a group key $\delta_{U_j}$ for the user where $\delta_{U_j} = g^{\gamma_o} \cdot H_1(ID_{U_j})^{\gamma_o}$ and sends $\delta_{U_j}$ to the corresponding user through a secure channel.

*PartialPvtKeyGen*. Once the user receives the group key from the GM, the user requests the KGC for the partial private key and sends $(ID_{U_j}, \delta_{U_j})$ as part of the request. The KGC validates both $(ID_{U_j}, \delta_{U_j})$ as per the verification equation below:

$$e\left(\delta_{U_j}, g\right) = e\left(g \cdot H_1\left(ID_{U_j}\right), g^{\gamma_o}\right). \tag{1}$$

If invalid, KGC aborts, else generates $Q_{U_j} = H_1(ID_{U_j})$ and $D_{U_j} = Q_{U_j}^{\alpha}$ where $ID_{U_j}$ is the unique public identity and $D_{U_j}$ is the partial private key of the user $U_j$. KGC forwards $D_{U_j}$ to the corresponding user $U_j$ via a secure channel.

*SetSecretValue*. User $U_j$ chooses the values $\beta_{U_j} \in Z_q^*$ and $g_1 \in G_1$ at random and stores with itself $\beta_{U_j}$ as a secret. The user computes $\zeta = g_1^{\beta_{U_j}}$ and makes it public.

*PvtKeyGen*. User $U_j$ sets its private key $S_{U_j} = \left\{D_{U_j}, \beta_{U_j}\right\}$.

*SetPublicKey*. User $U_j$ computes its public key as $PK_{U_j} = g^{\beta_{U_j}}$.

*SignGen*. User $U_j$ generates tag for a block $m_i$ of the file $M = \langle m_1, m_2, \ldots, m_n \rangle$ with file identity $F_{id}$, where $n$ is the number of blocks, $m_i \in Z_q{}^*$.

(1) The group user $U_j$ computes the tag $\sigma_i$ as $\sigma_i = (D_{U_j} \cdot g_1)^{m_i} H_2(w_i)^{\beta_{U_j}}$, where $w_i = F_{id} \| n \| i$.

(2) The group user $U_j$ uploads its message block along with authenticating information and the generated tag $\{(m_i, w_i, \sigma_i)\}$ to the cloud.

A correctly generated tag $\sigma_i$ satisfies the following equation, which can be verified by the cloud after receiving the tag.

$$e(\sigma_i, g) = e\left(H_1\left(ID_{U_j}\right)^{m_i}, P_{\text{pub}}\right) \cdot e(g_1^{m_i}, g) \cdot e\left(H_2(w_i), PK_{U_j}\right). \tag{2}$$

### 4.2. Analysis of Jaya et al.'s Scheme

*4.2.1. Type-I Attack Analysis.* Suppose a type-I attacker targets user $U_j$ as the victim user. According to the definition provided in Section 3.7.1, we can assume that type-I attacker has access to (1) all the public parameters used in the protocol, (2) the public key $PK_{U_j}$ of the victim user, and (3) a valid tuple $(m_i, w_i, \sigma_i)$ for the $i^{\text{th}}$ block of the data file of user $U_j$, for some $i$. Having these kinds of information, the attacker can execute the following steps to generate a forged tag $\sigma_i'$ for a forged data block $m_i'$, keeping the authenticating information same as $w_i$ and replacing the original public key $PK_{U_j}$ of the victim user by a forged public key $PK_{U_j}{}'$.

(1) Generates a forged data block $m_i'$ of its choice.

(2) Calculates $k = (m_i'/m_i)$, where $m_i$ is the data block to be replaced. Note that $m_i' = km_i$.

(3) Generates a forged public key $PK_{U_j}{}'$ for user $U_j$ as $PK_{U_j}{}' = (PK_{U_j})^k$. It replaces the original public key $PK_{U_j}$ of the victim user by the forged public key $PK_{U_j}{}'$.

(4) Computes the forged tag as $\sigma_i' = \sigma_i^k$.

(5) The attacker outputs the tuple $(m_i', w_i, \sigma_i')$.

Note that the attacker does not need access to the partial-private key $D_{U_j}$ or the personal secret key $\beta_{U_j}$ of the target user $U_j$.

**Lemma 1.** *The forged tag $\sigma_i'$ generated using the above mechanism is a valid tag on the forged data block $m_i'$ with authenticating information $w_i$ under the public values $(ID_{U_j}, PK_{U_j}{}')$ of the victim user $U_j$.*

*Proof.* In order to prove the lemma, it is sufficient to show that the tuple $(m_i', w_i, \sigma_i')$ satisfies the correctness equation as shown in equation (2) under the replaced public key $PK_{U_j}{}'$ of user $ID_{U_j}$. The correctness proof is given below:

$$\begin{aligned}
e(\sigma_i', g) &= e\left(\sigma_i^k, g\right) \\
&= e\left(\left(\left(D_{U_j} \cdot g_1\right)^{m_i} \cdot H_2(w_i)^{k\beta_{U_j}}\right), g\right) \\
&= e\left(\left(D_{U_j}^{km_i} \cdot g_1^{km_i}\right) H_2(w_i)^{k\beta_{U_j}}\right), g) \\
&= e\left(D_{U_j}^{km_i}, g\right) \cdot e\left(g_1^{km_i}, g\right) \cdot e\left(\left(H_2(w_i)^{k\beta_{U_j}}\right), g\right) \\
&= e\left(H_1\left(ID_{U_j}\right)^{\alpha km_i}, g\right) \\
&\quad \cdot e\left(g_1^{m_i'}, g\right).e\left((H_2(w_i)), g^{k\beta_{U_j}}\right) \\
&= e\left(H_1\left(ID_{U_j}\right)^{km_i}, g^{\alpha}\right) \cdot e\left(g_1^{m_i'}, g\right) \\
&\quad \cdot e\left(H_2(w_i), \left(PK_{U_j}\right)^k\right) \\
&= e\left(H_1\left(ID_{U_j}\right)^{m_i'}, P_{\text{pub}}\right) \cdot e\left(g_1^{m_i'}, g\right) \\
&\quad \cdot e\left(H_2(w_i), PK_{U_j}{}'\right).
\end{aligned} \tag{3}$$

□

*4.2.2. Type-II Attack Analysis.* Like type-I attack, a type-II attacker also has the access to all public parameters, including the public key $PK_{U_j}$ of the victim user $U_j$, and a valid tuple $(m_i, w_i, \sigma_i)$ for some $i$. In addition, it has the master secret key $\alpha$ in its possession. The attacker can execute the following steps to generate a forged tag $\sigma_i'$ for a forged data block $m_i'$, keeping the authenticating information same as $w_i$.

(1) Generates a forged data block $m_i'$ of its choice.

(2) Calculates $D_{U_j} = H_1(ID_{U_j})^{\alpha}$.

(3) Calculates $X = (D_{U_j} \cdot g_1)^{m_i}$

(4) Calculates $Y = (\sigma_i / X)$.

(5) Computes the forged tag as $\sigma_i' = (D_{U_j} \cdot g_1)^{m_i'}.Y$.

(6) The attacker outputs the tuple $(m_i', w_i, \sigma_i')$.

Note that in the above attack, the attacker does not need access to the personal secret key $\beta_{U_j}$ of the victim user $U_j$. Also, it does not replace the original public key $PK_{U_j}$ of the user.

**Lemma 2.** *The forged tag $\sigma_i'$ generated using the above mechanism is a valid tag on the forged data block $m_i'$ with authenticating information $w_i$ under the public values $(ID_{U_j}, PK_{U_j})$ of the victim user $U_j$.*

*Proof.* Let us prove the lemma by showing the validity of the forged tuple $(m_i', w_i, \sigma_i')$ under the public key $PK_{U_j}$ of user $U_j$, as per equation (2). The correctness proof is given below:

$$\begin{aligned} Y &= \left(\frac{\sigma_i}{X}\right) \\ &= \frac{\left(\left(D_{U_j} \cdot g_1\right)^{m_i} H_2(w_i)^{\beta_{U_j}}\right)}{\left(D_{U_j} \cdot g_1\right)^{m_i}} \\ &= H_2(w_i)^{\beta_{U_j}}, \end{aligned} \tag{4}$$

and hence

$$\begin{aligned} e(\sigma_i', g) &= e\left(\left(\left(D_{U_j} \cdot g_1\right)^{m_i'} \cdot Y\right), g\right) \\ &= e\left(D_{U_j}^{m_i'}, g\right) \cdot e\left(g_1^{m_i'}, g\right) \cdot e(Y, g) \\ &= e\left(H_1\left(ID_{U_j}\right)^{\alpha m_i'}, g\right) \cdot e\left(g_1^{m_i'}, g\right) \cdot e\left(H_2(w_i)^{\beta_{U_j}}, g\right) \\ &= e\left(H_1\left(ID_{U_j}\right)^{m_i'}, g^{\alpha}\right) \cdot e\left(g_1^{m_i'}, g\right) \cdot e\left(H_2(w_i), g^{\beta_{U_j}}\right) \\ &= e\left(H_1\left(ID_{U_j}\right)^{m_i'}, P_{\text{pub}}\right) \cdot e\left(g_1^{m_i'}, g\right) \cdot e\left(H_2(w_i), PK_{U_j}\right). \end{aligned} \tag{5}$$

□

*4.2.3. Remarks on the Analysis.* The above analysis shows that if an attacker (type I or type II) obtains a valid data block and tag pair $(m_i, \sigma_i)$ corresponding to some block $i$ of a data file $F_{id}$ belonging to a victim user $U_j$, then it can replace the block by generating a valid tag $\sigma_i'$ for a forged block $m_i'$ while keeping the authenticating information same as $w_i = F_{id} \| n \| i$. More importantly, we can also observe that the forged block $m_i'$ can be generated as per the choice of the attacker. Hence, the tag-generation mechanism used in [23] is susceptible to *selective forgery attack* under both type-I and type-II attackers. It can be noted that a selective forgery attack is more powerful than an existential forgery attack and can become detrimental in a cyber-physical system (CPS) environment where the truthfulness of data is directly associated with safety and security of property and life.

An obvious solution to prevent the above attack could be to prevent an unauthorized entity in obtaining access to a data block and its tag pair. It can be noted that in an RDIC protocol, the metadata (tags) generated by an user are required only by the CSP. Hence, we can keep both data and tag blocks of a file confidential between the user and the CSP only. This simple solution can certainly strengthen the security by completely eliminating type-II attack (compromised KGC) and eliminating type-I attack by any internal/external entity except the CSP. A compromised CSP can still launch type-I attack since it has access to the valid tag blocks generated by an user and we cannot hide those from the CSP since those are mandatory in calculating data-possession proof during data auditing. Our observation reveals the fact that type-I tag forgery attack is possible against Jaya et al.'s scheme [23] since the tag generation algorithm (SignGen) does not link the authenticating information $w_i$ of a block with the public key of the user. This enables a type-I attacker to easily replace the public key and forge a tag for a chosen message as we have shown in our analysis. In our protocol, we utilize the above conclusions to design a novel tag-generation mechanism which is provably secure against both type-I and type-II attacks.

*4.3. Brief Overview on the Tag-Generation Mechanism of the Scheme in [39].* In the following, we briefly describe the tag generation algorithm used in [39]. Again, we keep the notations consistent to the notations used in our proposed protocol.

*Tag-Gen.* Using this algorithm, a group user $U_j$ for $1 \le j \le d$ generates tag for a block $m_i$ of the file $M = \langle m_1, m_2, \ldots, m_n \rangle$ with file identity $F_{id}$ where $n$ is the number of blocks, $m_i \in {Z_q}^*$. User $U_j$ computes the tag $\sigma_i$ as

$$\sigma_i = \left(D_{U_j}\right)^{m_i} H_2(w_i)^{\beta_{U_j}}, \tag{6}$$

where $w_i = i \| F_{id}$. The partial private key is calculated as $D_{U_j} = H_1(ID_{U_j} + \eta)^{\alpha}$ where $\langle \alpha, \eta \in Z_q^* \rangle$ are secretly kept with the group manager, which acts as the KGC in this protocol. User $U_j$ sends the message block and tag pair $\{(m_i, \sigma_i)\}$ to the cloud and updates a public log file with the values index $i$, the user public identity $H_1(ID_{U_j} + \eta)$, and the user public key $P_{U_j} = g^{\beta_{U_j}}$ where $\beta_{U_j}$ is secret to the user $U_j$.

It can be noted that the tag generation algorithm discussed above is quite similar to that used in [23] in the sense that in this algorithm, the authenticating information for a block does not contain the information regarding the public key of the user. In addition, the protocol in [39] also does not include any mechanism to hide the valid tags generated by an user from the group manager, acting as the KGC in this protocol. Hence, type-I and type-II attacks can be launched in the same ways as we have shown in Sections 4.2.1 and 4.2.2.

## 5. Proposed System Model

In this section, the entities involved, data structures and files used, and the assumptions underlying our proposed protocols are stated in detail.

*5.1. Entities Involved.* The proposed protocols consist of the following entities.

*The CSP.* In cloud computing, a CSP is responsible for providing the computing and storage facilities to its users. For a FOG-CPS, we assume that the storage is implemented inside the FOG network at the physical proximity of the CPS system [71]. In our protocol, we view the storage facility implemented inside the FOG systems as a local storage server, and without loss of generality, we refer to it as the CSP.

*The Cloud User Group*. The cloud user is an edge device in our scheme. A group of edge devices forming a cluster constitutes a user group. There exists a group manager who manages the usual activities of a user group [72]. These users contribute data blocks to commonly shared files kept inside the local CSP. During registration of each cloud user, the user establishes a shared secret key with the CSP for confidential communication with it.

*The Data Auditor*. Unlike the existing RDIC approaches, which use an external fully trusted third-party auditor (TPA), in our protocol, one of the edge devices in a user group would act as the data auditor. Selection of the auditor can be done by the group manager or it can be elected by the group members using some leader election mechanism such as that in [73]. While selecting/electing the auditor, load balancing should be a criterion to be considered. Without loss of generality, throughout the rest of the paper, we assume that there exists a dedicated edge device in a user group acting as the data auditor. Since an edge device is easier to compromise, we assume that the auditor is a semi-trusted entity.

*Audit-Verifier*. Any edge device belonging to the user group sharing the data file $F$ can act as an audit-verifier to check whether the data auditor is performing its assigned duties correctly or not. We recommend that each edge device invoke this algorithm occasionally so that the load of verification is distributed among all edge devices in the group. Alternatively, the group manager can also prepare a schedule to distribute the verification duty among the group members.

*The KGC*. It acts as the security server that publishes the public parameter list and grants security credentials to all of the participating entities in the protocol.

*5.2. Data Structures and Files.* The data structures and files used and maintained by different entities in our protocol are listed below.

*Data File*. A shared data file $F$ with identity $ID_F$, stored with the CSP and shared among $d$ users (edge devices), can have $n$ number of data blocks $\langle m_1, m_2, \ldots, m_n \rangle$ at any instant in time. Each of these data blocks may be contributed by any one of the $d$ users of that group.

*Metadata File*. For each data file $F$, the CSP maintains a metadata file/tag file which contains the tags $\langle S_1, S_2, \ldots, S_n \rangle$ associated with each data blocks of the data file generated by the corresponding users.

*Indicator Matrix*. For each such data file $F$, the CSP maintains an indicator matrix $I$. The indicator matrix is a $(n \times d)$ 2D matrix whose each row corresponds to a block index $i\,(1 \le i \le n)$ and each column corresponds to an user $u\,(1 \le u \le d)$. We use the notation $I_{i,u}$ to denote $I[i][u]$ which is defined as follows:

$$I_{i,u} = \begin{cases} 1, & \text{if user } u \text{ is the generator of block } i, \\ 1, & \text{otherwise.} \end{cases} \tag{7}$$

*Log-File-1*. The data auditor maintains the Log-File-1. Corresponding to every data audit instance, there is an entry in this file. The fields contained in each entry are as shown below:

$$\langle t, t', ID_F, \text{Result, Signed Proof} \rangle, \tag{8}$$

where $t$ and $t'$ indicate the time instances when the audit was invoked and when the corresponding record was inserted into the public blockchain (explained later), respectively, for the file identified by $ID_F$. Result is an 1 bit field indicating whether the CSP passed the specific audit instance or not. A copy of the proof of data possession digitally signed and sent by the CSP, in response to the audit instance, is kept in the last field of the entry. All the cloud users have read access to Log-File-1.

*Log-File-2*. The CSP maintains Log-File-2 which contains the indicator matrix $I$ for every data file. The auditor and all the cloud users have read access to Log-File-2.

*5.3. Assumptions.* We assume the existence of a scalable public blockchain, suited for FOG-CPS environments, inside the network. There are several works that have proposed scalable public blockchain techniques for FOG-CPSs [55–58] as discussed in Section 2.1. FOG nodes with moderate storage and computing power store the complete blockchain information, and some edge devices with low storage and computing power only maintain a copy of the header information of the blocks in the blockchain.

A service-level agreement (SLA) is executed between the group manager, on behalf of a cloud user-group, and the data auditor which determines the frequency of audits or their specific time instants. The auditor is supposed to invoke data auditing protocol as per the requirements specified in the SLA.

## 6. Proposed Basic CLS-RDIC Protocol

In this section, we propose a basic CLS-RDIC group shared data auditing protocol applicable for the system model we described in the previous section. Usually, in the existing CLS-RDIC schemes, the data auditor generates the random challenge values used during a data auditing instance by itself. In our protocol, we restrict the auditor to use only time-varying verifiable source of randomness. For this purpose, the data auditor is required to retrieve the latest hash-block from the public blockchain available inside the network. It can be noted that given a time instant $t$, the latest hash-block is identified uniquely in the blockchain. This mechanism ensures that the auditor cannot send the challenge values in advance to the CSP even when they collude. We have proposed a novel tag generation and tag uploading mechanism which is provably secure against both type-I and type-II attacks. Besides, the data auditing mechanism of the

proposed basic CLS-RDIC scheme is also provably secure against type-III attack. However, we do not consider the data privacy requirement and the reliability of auditing service requirement against a compromised data auditor. In other words, the proposed basic CLS-RDIC protocol assumes the data auditor as fully trusted like many of the existing schemes. In the complete version of our protocol, which is proposed in the next section, we relax this assumption and provide additional mechanisms to ensure the security requirements against a semi-trusted auditor.

The proposed basic CLS-RDIC shared data auditing protocol consists of a total of nine algorithms, namely, Setup, Partial-Private-Key-Extract, Set-Secret-Value, Private-Key-Gen, Set-Public-Key, Tag-Gen, Challenge, Proof-Gen, and Proof-Ver. Detailed descriptions of each of these algorithms are provided below.

*Setup*. This algorithm is executed by the KGC during the setup phase of the protocol. Steps of the algorithm are as follows.

(1) On input $k$, the algorithm generates two cyclic multiplicative groups $G_1, G_2$ of prime order $q$ and an admissible bilinear map $\hat{e}$: $G_1 \times G_1 \longrightarrow G_2$. KGC selects a random generator $g \in G_1$.

(2) KGC chooses a random $\alpha \in Z_q^*$ as the master-secret-key (msk) and sets $P_{\text{pub}} = g^{\alpha}$.

(3) KGC selects three secure hash functions $H_1$: $\{0,1\}^* \longrightarrow G_1$, $H_2$: $\{0,1\}^* \longrightarrow G_1$, for partial-private-key and tag generation purposes, respectively, and $H_3$: $G_2 \longrightarrow Z_q^*$ for masking of data possession proof. The security analysis for the proposed scheme views $H_1$ and $H_2$ as random oracles.

(4) KGC selects another global hash function defined as $\Omega$: $\{0,1\}^* \longrightarrow \{0,1\}^{2l}$. The function produces a $2l$-bit binary string from a given hash-block of a blockchain. The bit string is further segregated into two $l$-bit strings, used as the seeds in two pseudorandom functions defined below.

(5) Two pseudorandom generation functions are chosen by the KGC as per the following definitions:
$f$: $\{0,1\}^l \times \{Z_{n+1} - \{0\}\} \longrightarrow Z_q^*$ $\pi$: $\{0,1\}^l \times \{Z_{n+1} - \{0\}\} \longrightarrow \{Z_{n+1} - \{0\}\}$.
The first input in both of the above two functions is an $l$-bit string which is used as a seed to initialize the random number generators. Both of these functions are used for generating the random challenge vectors.

(6) The KGC publishes the public parameters $\text{params} = \langle G_1, G_2, q, \hat{e}, g, P_{\text{pub}}, H_1, H_2, \Omega, f, \pi \rangle$.

*Partial-Private-Key-Extract*. On receiving a request from user with identity $ID_u$, KGC calculates $Q_u = H_1(ID_u)$ and $D_u = Q_u^{\alpha}$ and forwards the partial-private-key $D_u$ to the corresponding user through a secure channel.

*Set-Secret-Value*. User with identity $ID_u$ randomly chooses $\beta_u \in Z_q^*$ and keeps it as a personal secret value.

*Private-Key-Gen*. User with identity $ID_u$ sets the private key for itself as $\langle D_u, \beta_u \rangle$.

*Set-Public-Key*. User with identity $ID_u$ sets its public key as $P_u = g^{\beta_u}$. The user announces its public key to all the participating entities in the network. It can be noted that because of the use of CLPKC, there is no need for any certificate to be attached with the public key.

*Tag-Gen*. The steps followed by an user $u$ to generate the encrypted tag-signature $T_i$ on data block $m_i$ of a data file $F$ are as shown below:

(1) Calculates the authenticating tag information $w_i = (ID_F \| i \| P_u)$ for the block $m_i$.

(2) Computes the tag-signature $S_i$ for block $m_i$ as $S_i = D_u^{m_i} H_2(w_i)^{\beta_u}$.

(3) Encrypts the tag using its shared secret key $K_u$ with the CSP as $T_i = \text{Encr}_{K_u}[S_i]$.

(4) User $u$ sends the data block $m_i$ along with the encrypted tag $T_i$ to the CSP.

$$u \longrightarrow \text{CSP: } \{ID_u, P_u, ID_F, i, m_i, T_i\}. \tag{9}$$

The CSP extracts the tag-signature $S_i$ by decrypting $T_i$ using the shared secret key $K_u$. It calculates the authenticating tag information $w_i = (ID_F \| i \| P_u)$ and $Q_u = H_1(ID_u)$ and verifies the tag-signature by checking the following equality:

$$e(S_i, g) \stackrel{?}{=} e\left(Q_u^{m_i}, P_{\text{pub}}\right).e(H_2(w_i), P_u). \tag{10}$$

Upon successful verification, the CSP updates the indicator matrix $I$ corresponding to the data file $F$ as $I_{i,u} = 1$. It also stores the data block $m_i$ as the $i'$th block in the data file $F$ and the tag $S_i$ as the $i'$th tag value in the corresponding tag file.

*Challenge*. In order to audit the data file $F$, the auditor chooses a number $c\,(1 \le c \le n)$ as the size of the challenge vector and sends the challenge request as $\text{chal} = \langle t, c, ID_F \rangle$ to the CSP, where $t$ is the instant in time when the audit gets performed.

$$\text{Auditor} \longrightarrow \text{CSP: } t, c, ID_F. \tag{11}$$

*Proof-Gen*. Upon receiving the challenge from the auditor, the CSP first retrieves the hash-block corresponding to the given instant $t$ from the public blockchain and extracts the two random values $k_1, k_2 \in \{0,1\}^l$ from the retrieved hash-block using the global function $\Omega$. The CSP computes the set $C = \{(i, v_i)\}$, where $i = \pi(k_1, x)$ and $v_i = f(k_2, x)$ for $1 \le x \le c$. Next, it computes proof of possession for the challenged blocks as follows.

For each user $u \in \{1, \ldots, d\}$, the CSP computes $\mu_u$ and $\sigma_u$ as follows:

$$\mu_u = \sum_{i \in C} v_i m_i I_{i,u},$$
$$\sigma_u = \prod_{i \in C} S_i^{(v_i I_{i,u})}. \tag{12}$$

CSP sets $\mu = \{\mu_1, \mu_2, \ldots \mu_d\}$ and $\sigma = \{\sigma_1, \sigma_2, \ldots \sigma_d\}$ and generates the data possession proof as $P = (\sigma, \mu)$. Finally, the CSP signs the generated proof $P$ with its private key using the certificateless signature mechanism and sends the signed proof to the auditor.

$$\text{CSP} \longrightarrow \text{Auditor: } P = (\sigma, \mu), \text{Sig}_{\text{CSP}}(P). \tag{13}$$

*Proof-Ver*. The auditor verifies the received proof in the following way:

(1) The auditor first verifies the signature $\text{Sig}_{\text{CSP}}(P)$ against the public key of the CSP using the certificateless signature verification mechanism. It proceeds only if the verification is successful.

(2) Next, the auditor retrieves the hash-block corresponding to time $t$ from the public blockchain. The retrieved hash-block is used to extract the random values $k_1, k_2 \in \{0,1\}^l$ using the global function $\Omega$.

(3) The auditor calculates the public value $Q_u = H_1(ID_u); \forall u \in \{1, \ldots, d\}$ of the users. Then, it aggregates the values in $(\sigma, \mu)$, respectively, as $\sigma_\pi = \prod_{u=1}^{d} \sigma_u$ and $\mu_\pi = \prod_{u=1}^{d} Q_u^{\mu_u}$.

(4) The auditor computes the challenge vectors $C = \{(i, v_i)\}$, where $i = \pi(k_1, x)$ and $v_i = f(k_2, x)$ for all $1 \le x \le c$.

(5) For each $i \in C$, the auditor retrieves the public key information for user $u$ such that $I_{i,u} = 1$ and calculates the authenticating tag information $w_i = ID_F \| i \| P_u$.

(6) Auditor accepts the proof as valid only if the following equality holds:

$$e(\sigma_\pi, g) \stackrel{?}{=} e\left(\mu_\pi, P_{\text{pub}}\right). \prod_{u=1}^{d} e\left(\prod_{i \in C} H_2(w_i)^{v_i I_{i,u}}, P_u\right). \tag{14}$$

Once the verification of the received proof $P$ is done, the data auditor creates a report-message of the form Report = $\langle t, ID_F, \text{Result}, \text{Signed Proof} \rangle$ where $t$ is the instant of time used in the challenge message, $ID_F$ is the identity of the file audited, Result = 1 if the verification resulted in a success, 0 otherwise, and the Signed Proof field contains $P, \text{Sig}_{\text{CSP}}(P)$. The auditor submits the hash digest of the Report message along with its identity to the public blockchain. The blockchain verifies the authenticity of the sender and includes the sent digest value along with its sender's identity in the next generated hash-block in the blockchain. Finally, the auditor inserts an entry $\langle t, t', ID_F, \text{Result}, \text{Signed Proof} \rangle$ in Log-File-1 where the values of $t, ID_F$, Result, and Signed Proof are exactly the same as those in the Report message. Value of $t'$ is the time instant when the hash-block containing the digest of the Report message is inserted in the public blockchain.

**Lemma 3.** *An honest CSP will always pass the data auditing test as described above.*

*Proof.* In order to prove this lemma, it is sufficient to show that equation (14) always holds true if all values are entered correctly. The correctness proof is given below.

$$\begin{aligned} e(\sigma_u, g) &= e\left(\prod_{i \in C} S_i^{v_i I_{i,u}}, g\right) = \prod_{i \in C} e(S_i, g)^{v_i I_{i,u}} \\ &= \prod_{i \in C} e\left(Q_u^{m_i}, P_{\text{pub}}\right)^{v_i I_{i,u}} \prod_{i \in C} e(H_2(w_i), P_u)^{v_i I_{i,u}}. \end{aligned} \tag{15}$$

(from equation (10))

$$\begin{aligned} &= e\left(Q_u^{\sum_{i \in C} m_i v_i I_{i,u}}, P_{\text{pub}}\right). e\left(\prod_{i \in C} H_2(w_i)^{v_i I_{i,u}}, P_u\right) \\ &= e\left(Q_u^{\mu_u}, P_{\text{pub}}\right). e\left(\prod_{i \in C} H_2(w_i)^{v_i I_{i,u}}, P_u\right). \end{aligned} \tag{16}$$

Hence,

$$\begin{aligned} e(\sigma_\pi, g) &= e\left(\prod_{u=1}^{d} \sigma_u, g\right) = \prod_{u=1}^{d} e(\sigma_u, g) \\ &= \prod_{u=1}^{d} e\left(Q_u^{\mu_u}, P_{\text{pub}}\right) \prod_{u=1}^{d} e\left(\prod_{i \in C} H_2(w_i)^{v_i I_{i,u}}, P_u\right) \\ &= e\left(\mu_\pi, P_{\text{pub}}\right). \prod_{u=1}^{d} e\left(\prod_{i \in C} H_2(w_i)^{v_i I_{i,u}}, P_u\right). \end{aligned} \tag{17}$$

Soundness of the proposed basic CLS-RDIC protocol against type-I, type-II, and type-III attackers has been proved later in the security analysis section. □

## 7. The Proposed Complete Protocol

In this section, we propose the complete version of our protocol which is an extension over the basic CLS-RDIC protocol we proposed in the previous section. The security credential generation and the tag-generation mechanisms of the complete version are exactly the same as those in the basic version. In addition, like the basic version, in the complete version, we restrict the data auditor to use public blockchain as a time-varying verifiable source of randomness for challenge generation purpose. However, unlike the basic version, in the complete version of our protocol, we consider the data auditor as a semi-trusted entity, applicable for a FOG-CPS environment we described earlier. Hence, we need to protect *data confidentiality* and ensure the *correctness* and the *timeliness* of the auditing service provided by such semi-trusted data auditor. In the following, we first describe the additional attack types that we introduce for

such a semi-trusted data auditor and then provide the construction of our proposed certificateless reliable privacy-preserving auditing (CRPPA) protocol for shared data auditing in FOG-CPSs.

*7.1. Security Model for the Complete Scheme.* Type-I, type-II, and type-III attackers are also applicable for the complete version of the proposed protocol. In addition, we define the following types of adversaries to capture a semi-trusted data auditor scenario.

*7.1.1. Type-IV Adversary (Adv4).* In type-IV attack scenario, we consider a compromised data auditor as a passive attacker that can try to extract the content of a data file from a number of challenge-response interactions with the CSP while auditing the file. A privacy-preserving RDIC scheme must be able to provide data privacy against a curious auditor. The RDIC protocol needs to incorporate mechanisms to mask the values computed from the data blocks and corresponding tag blocks of the files by the CSP, so that the auditor learns zero information regarding any data block or tag block of the file from the responses received from the CSP. This should be ensured even when the number of such interactions is very large. This security requirement is analogous to the requirement in a *zero-knowledge authentication* protocol, such as that in [49, 74], where a claimant needs to prove the possession of a secret value without disclosing any information regarding the secret to the verifier.

*7.1.2. Type-V Adversary (Adv5).* Security of an RDIC protocol against a type-III attacker (Section 3.7.3) provides the assurance that a compromised CSP cannot generate a valid data possession proof for a given random challenge without storing all the data blocks of the corresponding data file properly. This assurance is valid under the assumption that the data auditor is honest and it performs its assigned tasks correctly. In type-V attack, we consider the possible collusion between a compromised data auditor and the CSP. The objective of the attack is same as that in type-III attack, i.e., to hide intentional/unintentional deletion or modifications of data blocks of a data file from its users. The compromised data auditor can achieve this objective in two possible ways. Simplest way is to generate false audit reports whenever the CSP fails to provide valid data possession proofs. Alternatively, the compromised auditor can disclose the random challenge vectors in advance to the CSP, using which, the CSP can precalculate data possession proofs and then delete or modify data blocks of a data file. Security of an RDIC scheme against type-V attacker requires mechanisms to detect a compromised auditor when it generates incorrect audit report and to prevent it from disclosing the random challenges in advance to anybody.

*7.1.3. Type-VI Adversary (Adv6).* In type-VI attack, the adversary is a compromised selfish data auditor, where the objective of the attacker is to save its computational resources. In this attack, the compromised selfish auditor may deviate from the SLA by skipping or delaying the auditing duties. An RDIC protocol, secure against type-VI attacker, must be able to verify *timeliness* of the data auditing tasks performed by a data auditor.

Our proposed scheme has considered six attack scenarios where an edge device (user) is a victim. Type-I and type-II adversaries arise from the security vulnerabilities of the key distribution mechanism of the CLPKC. In type-I attack, the compromised user tries to perform a tag forgery to impersonate a legitimate user, and the attack may be launched by replacing the public key of the legitimate user. In type-II attack, the compromised but passive KGC performs a tag forgery to impersonate a legitimate user. However, the KGC is trusted not to replace the legitimate user's public key. In type-III attack, a compromised CSP generates a forgery on the data possession proof to suppress a data loss or data corruption event. Security against type-III is a mandatory requirement of an RDIC scheme. In type-IV attack, a compromised data auditor performs cryptanalysis on the challenge-response messages to extract the users' confidential data or metadata. A public RDIC scheme must protect user data against type-IV attacks. In a type-V attack, the data auditor reveals the random challenge vectors in advance to the CSP to help CSP precompute the integrity proofs so that CSP can suppress a data loss or data corruption event in the future. In type-VI attack, the compromised data auditor skips or delays an auditing task to save computational resources. Table 2 summarizes the six types of attack scenarios, the entities involved in the attack, the objective of the attack, and the method of attack.

*7.2. Construction of the Proposed CRPPA Protocol.* The proposed certificateless reliable privacy-preserving auditing (CRPPA) protocol for shared data auditing in FOG-CPSs consists of a total of ten algorithms. The Setup, Partial-Private-Key-Extract, Set-Secret-Value, Private-Key-Gen, Public-Key-Gen, Tag-Gen, and Challenge algorithms of the CRPPA protocol are exactly the same as those in our proposed basic CLS-RDIC scheme. However, in the Proof-Gen algorithm, we use proper masking techniques to ensure that no information regarding either the data blocks or the tags is leaked to the auditor even if it executes millions of challenge verification instances with the CSP. The Proof-Ver algorithm has been modified accordingly. Additionally, we introduce the Verify-Proof algorithm which can be used by an audit-verifier to occasionally verify the activities of the auditor. The Proof-Gen, Proof-Ver, and the Verify-Proof algorithms of the proposed CRPPA protocol are explained below.

*Proof-Gen*. Upon receiving the challenge from the auditor, the CSP retrieves the random values $k_1, k_2$ using the global function $\Omega$ from the public blockchain as described in the proposed basic CLS-RDIC scheme. The CSP computes proof of possession for the challenged blocks as follows.

The CSP sets $C = \{(i, v_i)\}$, where $i = \pi(k_1, x)$ and $v_i = f(k_2, x)$ for $1 \le x \le c$.

Table 2: Types of attack scenarios.

| | Entities involved in the attack | Objective of the attack | Method of the attack | Attack type first introduced |
|---|---|---|---|---|
| Type-I | Compromised users | Impersonating a legitimate user | Generate forged tag by replacing public key | Al-Riyami et al. [45] |
| Type-II | Compromised but passive KGC | Impersonating a legitimate user | Generate forged tag without replacing public key | Al-Riyami et al. [45] |
| Type-III | Compromised CSP | Suppress data loss/corruption event | Generate forged data possession proof (without storing the data file properly) | Ateniese et al. [9] |
| Type-IV | Compromised data auditor | Extract confidential data and/or metadata of the users | Performing cryptanalysis on the challenge-response messages | Wang et al. [75] |
| Type-V | Collusion of data auditor and CSP | Suppress data loss/corruption event | Disclosing the random challenge vectors in advance to the CSP | Armknecht et al. [48] |
| Type-VI | Compromised data auditor | Save computational resources (selfish behaviour) | Skipping or delaying the auditing tasks | Zhang et al. [25] |

For each user $u \in \{1, \ldots, d\}$,

(1) The CSP computes the public value $Q_u = H_1(ID_u)$ and selects three random values $r_{m_u} \xleftarrow{R} Z_q^*$, $r_{\sigma_u} \xleftarrow{R} Z_q^*$, $\rho_u \xleftarrow{R} Z_q^*$, which are used for masking the $\sigma$ and $\mu$ values.

(2) CSP calculates $R_u = e(Q_u, P_{\text{pub}})^{r_{m_u}} \cdot e(g_1, g)^{r_{\sigma_u}}$ and $\gamma_u = H_3(R_u)$.

(3) CSP generates $(\mu_u, \sigma_u)$ as follows:

$$\mu_u = \sum_{i \in C} v_i m_i I_{i,u}, \qquad \sigma_u = \prod_{i \in C} S_i^{(v_i I_{i,u})}. \tag{18}$$

(4) Next, CSP calculates $\mu_u' = (r_{m_u} + \gamma_u \mu_u)$, $\Sigma_u = \sigma_u \cdot g_1^{\rho_u}$, and $\varepsilon_u = (r_{\sigma_u} + \gamma_u \rho_u)$. The masking of $(\sigma, \mu)$ helps to provide zero-knowledge data privacy against the auditor.

(5) The CSP calculates the integrity proof as $P = \langle \mu', \Sigma, \varepsilon, R \rangle$ where $\mu' = \{\mu_1', \mu_2', \ldots, \mu_d'\}$, $\Sigma = \{\Sigma_1, \Sigma_2, \ldots, \Sigma_d\}$, $\varepsilon = \{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_d\}$, $R = \{R_1, R_2, \ldots, R_d\}$.

Finally, the CSP signs the generated proof $P$ with its private key using the certificateless signature mechanism and sends the signed proof to the auditor.

$$\text{CSP} \longrightarrow \text{Auditor: } P = \langle \mu', \Sigma, \varepsilon, R \rangle, \text{Sig}_{\text{CSP}}(P). \tag{19}$$

*Proof-Ver*. The auditor first checks if $\text{Sig}_{\text{CSP}}(P)$ is a valid signature on proof $P = \langle \mu', \Sigma, \varepsilon, R \rangle$ using the public key of the CSP and then executes the following steps to verify the proof $P$:

(1) The auditor retrieves the hash-block corresponding to time $t$ from the public blockchain. The retrieved hash-block is used to extract the random values $k_1, k_2 \in \{0,1\}^l$ using the global function $\Omega$.

(2) For each user $u \in \{1, 2, \ldots, d\}$, the auditor calculates $\gamma_u = H_3(R_u)$.

(3) The auditor aggregates the values in the received proof $P = \langle \mu', \Sigma, \varepsilon, R \rangle$ as $R_\pi = \prod_{u=1}^d R_u$, $\Sigma_\pi = \prod_{u=1}^d \Sigma_u^{\gamma_u}$, $\mu_\pi' = \prod_{u=1}^d Q_u^{\mu_u'}$, and $\varepsilon_\pi = \sum_{u=1}^d \varepsilon_u$.

(4) The auditor generates the challenge vectors $C = \{(i, v_i)\}$, where $i = \pi(k_1, x)$ and $v_i = f(k_2, x)$ for all $1 \le x \le c$.

(5) For each $i \in C$, the auditor gets public key information of user $u$ such that $I_{i,u} = 1$ and calculates the authenticating tag information $w_i = ID_F \| i \| P_u$.

(6) The auditor accepts the proof only if the following equation holds correct.

$$R_\pi \cdot e(\Sigma_\pi, g) \stackrel{?}{=} \left( \prod_{u=1}^{d} e\left( \prod_{i \in C} H_2(w_i)^{v_i I_{i,u} \gamma_u}, P_u \right) \right) \cdot e\left(\mu_\pi', P_{pub}\right) \cdot e(g_1, g)^{\varepsilon_\pi}. \tag{20}$$

Once the verification of the received proof $P$ is done, the data auditor creates a report-message of the form Report = $\langle t, ID_F, \text{Result}, \text{Signed Proof} \rangle$ where $t$ is the instant of time used in the challenge message, $ID_F$ is the identity of the file audited, Result = 1 if the verification resulted in a success, 0 otherwise, and the Signed Proof field contains $P, \text{Sig}_{\text{CSP}}(P)$. The auditor submits the hash digest of the Report message along with its identity to the public blockchain. The blockchain verifies the authenticity of the sender and includes the sent digest value along with its sender's identity in the next generated hash-block in the blockchain. Finally, the auditor inserts an entry $\langle t, t', ID_F, \text{Result}, \text{Signed Proof} \rangle$ in Log-File-1 where the values of $t, ID_F, \text{Result}$, and Signed Proof are exactly the same as those in the Report message. Value of $t'$ is the time instant when the hash-block containing the digest of the Report message is inserted in the public blockchain.

**Lemma 4.** *An honest CSP will always pass the data auditing test as described in the proposed CRPPA protocol.*

*Proof.* In order to prove this lemma, it is sufficient to show that equation (20) always holds true if all values are entered correctly. The correctness proof is given below.

$$\begin{aligned} R_u \cdot e(\Sigma_u^{\gamma_u}, g) &= e\left(Q_u, P_{\text{pub}}\right)^{r_{m_u}} \\ &\quad \cdot e(g_1, g)^{r_{\sigma_u}} \cdot e\left(\left(\sigma_u \cdot g_1^{\rho_u}\right)^{\gamma_u}, g\right) \\ &= e\left(Q_u, P_{\text{pub}}\right)^{r_{m_u}} \cdot e(g_1, g)^{r_{\sigma_u}} \\ &\quad \cdot e\left(\left(S_i^{\nu_i I_{i,u}} \cdot g_1^{\rho_u}\right)^{\gamma_u}, g\right) \\ &= e\left(Q_u, P_{\text{pub}}\right)^{r_{m_u}} \cdot e(g_1, g)^{r_{\sigma_u}} \cdot e\left(Q_u, P_{\text{pub}}\right)^{\gamma_u \mu_u} \cdot \\ &\quad e\left(\prod_{i \in C} H_2(w_i)^{\nu_i I_{i,u} \beta_u \gamma_u}, g\right) e\left(g_1^{\gamma_u \rho_u}, g\right) \\ &= e\left(Q_u^{\mu_u'}, P_{\text{pub}}\right) \cdot e(g_1, g)^{\varepsilon_u} \\ &\quad \cdot e\left(\prod_{i \in C} H_2(w_i)^{\nu_i I_{i,u} \gamma_u}, P_u\right). \end{aligned} \tag{21}$$

Hence,

$$\begin{aligned} R_\pi \cdot e(\Sigma_\pi, g) &= \prod_{u=1}^{d} R_u \cdot e\left(\prod_{u=1}^{d} \Sigma_u^{\gamma_u}, g\right) \\ &= \prod_{u=1}^{d} \left(R_u \cdot e(\Sigma_u^{\gamma_u}, g)\right) \\ &= \prod_{u=1}^{d} e\left(Q_u^{\mu_u'}, P_{pub}\right) \cdot e\left(g_1^{\varepsilon_u}, g\right) \\ &\quad \cdot e\left(\prod_{i \in C} H_2(w_i)^{\nu_i I_{i,u} \gamma_u}, P_u\right)) \\ &= e\left(\mu_\pi', P_{\text{pub}}\right) \cdot e(g_1, g)^{\varepsilon_\pi} \\ &\quad \cdot \left(\prod_{u=1}^{d} e\left(\prod_{i \in C} H_2(w_i)^{\nu_i I_{i,u} \gamma_u}, P_u\right)\right). \end{aligned} \tag{22}$$

*Verify-Proof*. In order to audit data file $F$, an audit-verifier first chooses an audit-interval $T = [T_1, T_2]$ from the past and retrieves all the entries from Log-File-1 where the audit instance $t$ lies in the interval $T$ and the $ID_F$ field matches the id of file $F$. Next, it verifies whether the audit instances are as per the SLA or not. If any one of the extracted entries contains a zero value in its Result field, the audit-verifier retrieves the Signed Proof corresponding to that entry and invokes the *Proof-Ver* algorithm to check whether the data possession proof sent by CSP was valid or not. If it is a valid proof, then it reports to the group manager against the data auditor as misbehaving. If the proof is invalid, it reports to the group manager against the CSP as misbehaving. When all entries contain 1 in the Result fields, the audit-verifier retrieves the signed proofs from all the entries and performs an aggregate proof verification in the following way.

We use the notation $\text{Signed Proof}(t)$ to denote the signed proof at audit instance $t \in T$. Hence, $\text{Signed Proof}(t) \equiv \langle P(t), \text{Sig}_{\text{CSP}}(P_t) \rangle$ where $P(t) = \langle \mu'(t), \Sigma(t), \varepsilon(t), R(t) \rangle$, and

$$\begin{aligned} \mu'(t) &= \{\mu_1'(t), \mu_2'(t), \ldots, \mu_d'(t)\}, \\ \Sigma(t) &= \{\Sigma_1(t), \Sigma_2(t), \ldots, \Sigma_d(t)\}, \\ \varepsilon(t) &= \{\varepsilon_1(t), \varepsilon_2(t), \ldots, \varepsilon_d(t)\}, \\ R(t) &= \{R_1(t), R_2(t), \ldots, R_d(t)\}. \end{aligned} \tag{23}$$

(1) For each audit instance $t \in T$, the audit-verifier performs the following:

(i) Checks the validity of the signature $\text{Sig}_{\text{CSP}}(P(t))$ using the public key of the CSP. It proceeds only if the signature is valid.

(ii) Calculates $\gamma_u(t) = H_3(R_u(t))$, for all $u \in \{1, 2, \ldots, d\}$.

(iii) Calculates $C(t) = \{(i, v_i)\}$, where $i = \pi(k_1(t), x)$ and $v_i = f(k_2(t), x)$, for all $1 \le x \le c$, where $k_1(t)$ and $k_2(t)$ are the random strings extracted using the global function $\Omega$ from the hash-blocks at time instant $t$ of the public blockchain.

(iv) The audit-verifier computes the following aggregates values for the audit instance $t$:

$$\begin{aligned} \mu_\pi'(t) &= \prod_{u=1}^{d} Q_u^{\mu_u'(t)}, \\ \Sigma_\pi(t) &= \prod_{u=1}^{d} \Sigma_u(t)^{\gamma_u(t)}, \\ \varepsilon_\pi(t) &= \sum_{u=1}^{d} \varepsilon_u(t), \\ R_\pi(t) &= \prod_{u=1}^{d} R_u(t). \end{aligned} \tag{24}$$

(2) Finally, the audit-verifier computes the following aggregate values by aggregating the above aggregate values for all $t \in T$:

$$\begin{aligned} \mu_{\text{agg}}' &= \prod_{\forall t \in T} \mu_\pi'(t), \\ \Sigma_{\text{agg}} &= \prod_{\forall t \in T} \Sigma_\pi(t), \\ \varepsilon_{\text{agg}} &= \sum_{\forall t \in T} \varepsilon_\pi(t), \\ R_{\text{agg}} &= \prod_{\forall t \in T} R_\pi(t), \\ C_{\text{agg}} &= \cup_{\forall t \in T} \{C(t)\}. \end{aligned} \tag{25}$$

(3) For all $i \in C_{\text{agg}}$, the verifier retrieves the public key $P_u$ of user $u$ such that $I_{i,u} = 1$ and calculates the authenticating tag information as $w_i = ID_F \| i \| P_u$.

(4) The audit verification results in a success only if the following equality holds:

$$R_{\text{agg}} \cdot e\left(\Sigma_{\text{agg}}, g\right) \stackrel{?}{=} \prod_{\forall t \in T}\left(\left(\prod_{u=1}^{d} e\left(\prod_{i \in C(t)} H_2\left(w_i\right)^{\nu_i I_{i,u} \gamma_u(t)}, P_u\right)\right)\right) \cdot e\left(\mu_{\text{agg}}^{A'}, P_{\text{pub}}\right) \cdot e\left(g_1, g\right)^{\varepsilon_{\text{agg}}}. \quad (26)$$

□

**Lemma 5.** *If the data auditor performs all the proof verifications honestly for each audit instance $t \in T$, then the above-mentioned aggregate audit verification must result in success.*

*Proof.* The above-mentioned aggregate audit verification is initiated only when the Result field in each entry corresponding to each audit instance $t \in T$ shows 1, in the Log-File-1. This can happen only when the Proof-Ver algorithm resulted in success for each of the instance $t \in T$, since we assume that the data auditor executed the Proof-Ver algorithm honestly for each of these instances. Hence, each of the instance $t \in T$ must satisfy equation (20), i.e.,

$$R_\pi(t) \cdot e\left(\Sigma_\pi(t), g\right) \stackrel{?}{=} \left(\prod_{u=1}^{d} e\left(\prod_{i \in C(t)} H_2\left(w_i\right)^{\nu_i I_{i,u} \gamma_u(t)}, P_u\right)\right) \cdot e\left(\mu_\pi'(t), P_{\text{pub}}\right) \cdot e\left(g_1, g\right)^{\varepsilon_\pi(t)}. \quad (27)$$

Let us take the product of the left-hand side (LHS) of the above equation for all $t \in T$:

$$\begin{aligned} \prod_{\forall t \in T} (\text{LHS}) &= \prod_{\forall t \in T} \left(R_\pi(t) \cdot e\left(\Sigma_\pi(t), g\right)\right) \\ &= \prod_{\forall t \in T} \left(R_\pi(t)\right) \cdot \prod_{\forall t \in T} e\left(\Sigma_\pi(t), g\right) \\ &= \prod_{\forall t \in T} \left(R_\pi(t)\right) \cdot e\left(\prod_{\forall t \in T} \Sigma_\pi(t), g\right) \\ &= R_{\text{agg}} \cdot e\left(\Sigma_{\text{agg}}, g\right). \end{aligned} \quad (28)$$

Similarly, we take the product of the right-hand side (RHS) of equation (27) for all $t \in T$:-

$$\begin{aligned} &\prod_{\forall t \in T} (\text{RHS}) \\ &= \prod_{\forall t \in T}\left(\prod_{u=1}^{d} e\left(\prod_{i \in C(t)} H_2\left(w_i\right)^{\nu_i I_{i,u} \gamma_u(t)}, P_u\right)\right) \cdot \prod_{\forall t \in T} e\left(\mu_\pi'(t), P_{\text{pub}}\right) \cdot \prod_{\forall t \in T} e\left(g_1, g\right)^{\varepsilon_\pi(t)} \\ &= \prod_{\forall t \in T}\left(\left(\prod_{u=1}^{d} e\left(\prod_{i \in C(t)} H_2\left(w_i\right)^{\nu_i I_{i,u} \gamma_u(t)}, P_u\right)\right)\right) \cdot e\left(\prod_{\forall t \in T} \mu_\pi'(t), P_{\text{pub}}\right) \cdot e\left(g_1, g\right)^{\sum_{\forall t \in T} \varepsilon_\pi(t)} \\ &= \prod_{\forall t \in T}\left(\left(\prod_{u=1}^{d} e\left(\prod_{i \in C(t)} H_2\left(w_i\right)^{\nu_i I_{i,u} \gamma_u(t)}, P_u\right)\right)\right) \cdot e\left(\mu_{\text{agg}}', P_{\text{pub}}\right) \cdot e\left(g_1, g\right)^{\varepsilon_{\text{agg}}}. \end{aligned} \quad (29)$$

Equations (28) and (29) directly establish that equation (26) must hold true. Hence, the aggregate verification must result in success.

So, if the above-mentioned aggregate audit verification does not result in success, the audit-verifier reports against the data auditor as misbehaving to the group manager. The above procedure verifies whether the data auditor performs its assigned auditing task correctly or not. In order to check whether all the auditing tasks were performed timely or not, the audit-verifier proceeds as follows.

For each of the audit instance $t \in T$, the audit-verifier constructs a Report message the same way as it is mentioned in the Proof-Ver algorithm, i.e., Report = $\langle t, ID_F, \text{Result}, \text{SignedProof} \rangle$. The verifier then retrieves the $t'$ field corresponding to the entry from Log-File-1 and checks whether the hash digest of the Report message exists in the hash-block corresponding to time $t'$ in the public blockchain or not. If the verification fails for any audit instance $t \in T$, the audit-verifier reports against the timeliness of auditing performed by the data auditor to the group manager.

Soundness of the aggregate verification used in the *Verify-Proof* algorithm has been analyzed in the next section. □

## 8. Security Analysis

*8.1. Soundness Proof for the Proposed Basic CLS-RDIC Scheme.* In this subsection, we prove the security of our proposed basic CLS-RDIC scheme against type-I, type-II, and type-III attackers. In Section 8.2, we establish the security of our proposed CRPPA scheme based on the fact that the proposed basic scheme is secure against type-I, type-II, and type-III attackers. In Theorems 1 and 2, we analyze the security of the tag-generation mechanism of the basic CLS-RDIC scheme against type-I and type-II attackers as defined in Sections 3.7.1 and 3.7.2, respectively. In Theorem 3, we analyze the security of the data auditing challenge-response protocol of the basic CLS-RDIC scheme against type-III attacker as defined in Section 3.7.3.

The following theorem proves that, in the random oracle model, the tags generated by the proposed basic CLS-RDIC scheme are unforgeable under an adaptive chosen-message type-I attacker assuming that the CDH problem in $G_1$ is hard.

**Theorem 1.** *If an adversary Adv1, as defined in game-1, has a success probability of $\xi(k)$ in generating a forged tag against our proposed basic CLS-RDIC scheme, then there exists a challenger C that solves an instance of a CDH problem in group $G_1$. Suppose Adv1 makes at most $q_1$ $H_1$ hash queries, $q_2$ partial-private-key queries, $q_3$ secret-value queries, $q_4$ public-key queries, $q_5$ public-key-replace queries, $q_6$ $H_2$ hash queries, and $q_7$ tag-gen queries, where $q_1, q_2, q_3, q_4, q_5, q_6, q_7$ all are in polynomial$(k)$; then, the success probability, $\xi'(k)$, of challenger C in solving the above problem is lower bounded by*

$$\xi'(k) \geq \frac{\xi(k)}{e \cdot (1 + q_2 + q_7)}, \tag{30}$$

*where $e \approx 2.71828$ is the base of the natural logarithm. In addition, the time complexity classes are $O(C) \approx O(Adv1) + O(q_1 + q_2 + q_3 + q_4 + q_5 + q_6 + q_7)$.*

*Proof.* Let Adv1 be the adversary as in game-1 who is capable of forging a tag. Then, we construct a simulator $C$ using Adv1 as subroutine to solve the CDH problem in $G_1$. $C$ simulates game-1 in the following way.

*Setup*. Given a CDH problem instance $\langle q, G_1, g, h_1, h_2 \rangle$, $h_1 = g^{\alpha}$, for some $\alpha \in Z_q^*$. $C$ chooses another cyclic multiplicative group $G_2$, $(|G_1| = |G_2| = q)$, and an admissible bilinear map $\hat{e} = G_1 \times G_1 \longrightarrow G_2$. $C$ sets the public parameters params $= \langle G_1, G_2, q, \hat{e}, g, P_{\text{pub}} = h_1, H_1, H_2 \rangle$ and sends params to Adv1. Note that the master secret key($\alpha$) is unknown to the challenger $C$.

*$H_1$ Hash Query*. Adv1 adaptively queries for $H_1$ for any identity $ID_u$. For every $H_1$ query, $C$ maintains a list $H_1$-List $= \{(ID_u, Q_u, r_u, S_u, P_u, b_u)\}$. If the $ID_u$ is already queried before, $C$ simply retrieves the tuple from the $H_1$-List and returns $Q_u$ to Adv1. If $ID_u$ is new, $C$ randomly picks $r_u \in Z_q^*$ and tosses a coin $b_u \in \{0, 1\}$, where the probability of $b_u = 0$ is $\varphi$ and that of $b_u = 1$ is $1 - \varphi$. If $(b_u = 0)$, then $C$ sets $Q_u = g^{r_u}$, chooses a random value $\beta_u \in Z_q^*$, and sets $S_u = \beta_u$ and $P_u = g^{\beta_u}$. Else if $(b_u = 1)$, then $C$ sets $Q_u = h_2 \cdot g^{r_u}$, chooses a random value $\beta_u \in Z_q*$, and sets $S_u = \beta_u$ and $P_u = g^{\beta_u}$. Finally, $C$ inserts the values $(ID_u, Q_u, r_u, S_u, P_u, b_u)$ into $H_1$-List. Return $Q_u$ to Adv1.

*Partial-Private-Key Query*. Adv1 adaptively queries for partial-private-key for any identity $ID_u$. $C$ scans for $ID_u$ in $H_1$-List. If $ID_u$ does not exist in $H_1$-*List*, $C$ queries itself for $H_1(ID_u)$. If $b_u = 0$, then retrieve $r_u$ from $H_1$-List set $D_u = h_1^{r_u}$ and return $D_u$ to Adv1. Else if $b_u = 1$, then abort.

*Secret-Value Query*. Adv1 adaptively queries for set-secret-value for any identity $ID_u$. $C$ scans for $ID_u$ in $H_1$-*List*. If $ID_u$ does not exist in $H_1$-List, $C$ queries itself for $H_1(ID_u)$. If $(S_u = \perp)$, $C$ chooses a random value $\beta_u \in Z_q*$ and sets $S_u = \beta_u$ and $P_u = g^{\beta_u}$. $C$ updates the values $(S_u, P_u)$ into $H_1$-List and returns $S_u$ to Adv1.

*Public-Key Query*. Adv1 adaptively queries for public-key for any identity $ID_u$. $C$ scans for $ID_u$ in $H_1$-List. If $ID_u$ does not exist in $H_1$-List, $C$ queries itself for $H_1(ID_u)$. $C$ retrieves $P_u$ and returns it to Adv1.

*Public-Key-Replace Query*. Adv1 adaptively queries for a public-key replacement for any identity $ID_u$ with $(ID_u, P_u')$. $C$ scans for $ID_u$ in $H_1$-List. If $ID_u$ does not exist in $H_1$-List, $C$ queries itself for $H_1(ID_u)$. $C$ sets $S_u = \perp$ and $P_u = P_u'$ and updates $H_1$-List.

*$H_2$ Hash Query*. Adv1 adaptively queries for $(w_k, P_u)$. For every $H_2$ query, $C$ maintains a list $H_2$-List $= \{(w_k, H_k^2, r_k, P_u)\}$. $C$ scans $H_2$-List for $(w_k, P_u)$. If $(w_k, P_u)$ does not exist in $H_2$-List, $C$ randomly chooses $r_k \in Z_q*$, sets $H_k^2 = g^{r_k}$, and updates $H_2$-List. $C$ returns $H_k^2$ to Adv1.

*Tag-Gen Query*. Adv1 adaptively queries with the values $(m, w_k, ID_u)$. If $ID_u$ does not exist in $H_1$-List, $C$ queries itself for $H_1(ID_u)$. $C$ retrieves $P_u$ from $H_1$-List. If $(b_u = 0)$, then, $C$ queries for $D_u$. $C$ scans for $(w_k, P_u)$ in $H_2$-List, and if it exists, then retrieve $H_k^2$; else, $C$ self-queries for $H_k^2$ with the values $(w_k, P_u)$. Now $C$ retrieves $S_u$ from $H_1$-List for $ID_u$. It computes tag $\sigma = D_u^m \cdot (H_k^2)^{S_u}$. Else if $b_u = 1$ and $(w_k, P_u)$exists in $H_2$-List, then abort. Else select randomly $r_k \in Z_q*$, compute $H_k^2 = g^{r_k} \cdot h_1^{-m}$, set $P_u = h_2$, compute $\sigma = h_1^{r_u m} \cdot h_2^{r_k}$, and finally insert the tuple $(w_k, H_k^2, \perp, P_u)$ to $H_2$-List. $C$ returns $\sigma$ to Adv1

*Forge-Tag*. Adv1 outputs a tuple $(\sigma', m', w_k', ID_u', P_u')$, where $\sigma'$ is the forged tag on the message $m'$ with authenticating information $w_k'$ for identity $ID_u'$ with public key $P_u'$.

*Analysis*. If Adv1 successfully forges the tag, then the tuple should be valid. If $(b_u = 0)$, then $C$ aborts. Else if $(b_u = 1)$, $C$ uses the following verification equation: $e(\sigma', g) = e(H_1(ID_u')^{m'}, P_{\text{pub}}) \cdot e(H_2(w_k', P_u'), P_u')$. $C$ retrieves the tuple from $H_1$-List for the forged $ID_u'$, and $C$ retrieves the value of $Q_u = h_2 \cdot g^{r_u}$ from $H_1$-List and $H_2(w_k', P_u') = g^{r_{k'}}$ from $H_2$-List. As per the verification equation above, $C$ is able to calculate solution to CDH problem, i.e., given an instance of the CDH problem $(g, \{h_1 = g^{\alpha}\}, h_2)$, $C$ calculates $h_2^{\alpha}$ as follows.

As per the verification equation:

$$\begin{aligned} e(\sigma', g) &= e\left((h_2 \cdot g^{r_u})^m, P_{\text{pub}}\right) \cdot e\left(g^{r_{k'}}, P_u'\right), \\ e(\sigma', g) &= e\left((h_2)^{\alpha \cdot m'} \cdot (g^{\alpha})^{r_u \cdot m'}, g\right) \cdot e\left((P_u')^{r_{k'}}, g\right), \\ h_2^{\alpha} &= \left(\frac{\sigma'}{(P_u')^{r_{k'}} \cdot (h_1)^{r_u \cdot m'}}\right)^{1/m'}. \end{aligned} \tag{31}$$

We now calculate the minimum success probability $\xi'(k)$ with which the challenger $C$ can solve an instance of the given CDH problem in $G_1$. For this to happen, the challenger $C$ must not abort during the above simulation. There are three phases in the simulation where the challenger $C$ can abort, i.e., in *partial-private-key query* phase, *tag-gen query* phase, and *analysis* phase. The *partial-private-*

*key query* phase and the *tag-gen query* phase do not abort when $b_u = 0$, whereas the *analysis* phase does not abort when $b_u = 1$. Hence, the probability that the challenger $C$ does not abort the above simulation is $(1-\varphi)\cdot\varphi^{q_2+q_7}$. The quantity $(1-\varphi)\cdot\varphi^{q_2+q_7}$ is maximized when $\varphi = \varphi_{\max} = q_2 + q_7/1 + q_2 + q_7$. By substituting the above value of $\varphi_{\max}$, we get the lower bound on the probability that the challenger $C$ does not abort as $1/e\cdot(1+q_2+q_7)$. Hence, the minimum success probability with which the challenger $C$ can solve an instance of the given CDH problem is $\xi'(k) \geq \xi(k)/e\cdot(1+q_2+q_7)$. It can be noted that since $q_2$ and $q_7$ are polynomial$(k)$, the quantity $1/(1+q_2+q_7)$ is non-negligible. Thus, if $\xi(k)$ is non-negligible, then the success probability $\xi'(k)$ is also non-negligible.

The running time of $C$ is the sum of upper bounded running time of adversary algorithm Adv1 ($O$ (Adv1)) and upper bounded running time of all the algorithms used in simulating the challenger $C$, i.e., $H_1$ hash query ($t_1$), partial-private-key query ($t_2$), secret-value query ($t_3$), public-key query ($t_4$), public-key-replace query ($t_5$), $H_2$ hash query ($t_6$), and tag-gen query ($t_7$) executed $q_1, q_2, q_3, q_4, q_5, q_6$, and $q_7$ times, respectively. Therefore, $O(C) = O(\text{Adv1}) + O(q_1\cdot t_1 + q_2\cdot t_2 + q_3\cdot t_3 + q_4\cdot t_4 + q_5\cdot t_5 + q_6\cdot t_6 + q_7\cdot t_7)$. Now since all the algorithms used in the simulation of challenger $C$ are polynomial-time computable, $O(C) = O(\text{Adv1}) + O(q_1 + q_2 + q_3 + q_4 + q_5 + q_6 + q_7)$. It can be noted that $q_1, q_2, q_3, q_4, q_5, q_6$, and $q_7$ all are in polynomial$(k)$. Therefore, $O(C) \approx O(\text{Adv1})$. It can be observed that if Adv1 is a polynomial-time algorithm, then the simulated CDH problem solver $C$ is also a polynomial-time algorithm.

The following theorem proves that, in the random oracle model, the tags generated by the proposed basic CLS-RDIC scheme are unforgeable under an adaptive chosen-message type-II attacker assuming that the CDH problem in $G_1$ is hard. □

**Theorem 2.** *If an adversary Adv2, as defined in game-2, has a success probability of $\xi(k)$ in generating a forged tag against our proposed basic CLS-RDIC scheme, then there exists a challenger $C$ that solves an instance of a CDH problem in group $G_1$. Suppose Adv2 makes at most $q_1$ $H_1$ hash queries, $q_2$ secret-value queries, $q_3$ public-key queries, and $q_4$ $H_2$ hash queries, where $q_1, q_2, q_3$, and $q_4$ all are in polynomial$(k)$; then, the success probability, $\xi'(k)$, of challenger $C$ in solving the above problem is lower bounded by*

$$\xi'(k) \geq \frac{\xi(k)}{e\cdot(1+q_2)}, \tag{32}$$

*where $e \approx 2.71828$ is the base of the natural logarithm. In addition, the time complexity classes are $O(C) \approx O(Adv2) + O(q_1 + q_2 + q_3 + q_4)$.*

*Proof.* Let Adv2 be the adversary as in game-2 who is capable of forging a tag. Then, we construct an adversary $C$ using Adv2 as subroutine to solve the CDH problem in $G_1$. $C$ simulates game-2 in the following way.

*Setup*. Given an instance of CDH problem $(g, G_1, \{h_1 = g^{\alpha}\}, h_2)$. $C$ randomly picks $s \in Z_q^*$, chooses another cyclic multiplicative group $G_2$, $(|G_1| = |G_2| = q)$, and an admissible bilinear map $\hat{e}$: $G_1 \times G_1 \longrightarrow G_2$, and sets the public parameters params $= \langle G_1, G_2, q, \hat{e}, g, (P_{\text{pub}} = g^s), H_1, H_2\rangle$. $C$ sends params and $s$ to Adv2, and $s$ is the master secret key.

*$H_1$ Hash Query*. Adv2 adaptively queries for $H_1$ for any identity $ID_u$. For every $H_1$ query, $C$ maintains a list $H_1$-List $= \{(ID_u, Q_u, r_u, S_u, P_u)\}$. If the $ID_u$ is already queried before, $C$ simply retrieves the tuple from the $H_1$-List and returns $Q_u$ to Adv2. If $ID_u$ is new, $C$ randomly picks $r_u, \beta_u \in Z_q^*$, sets $Q_u = g^{r_u}$, and tosses a coin $b_u \in \{0,1\}$, where the probability of $b_u = 0$ is $\varphi$ and that of $b_u = 1$ is $1-\varphi$ . If $(b_u = 0)$, then $C$ sets $P_u = g^{\beta_u}$ and $S_u = \beta_u$. Else if $(b_u = 1)$, then $C$ sets $P_u = (h_1)^{\beta_u}$ and $S_u = \perp$. Finally, $C$ inserts the values $(ID_u, Q_u, r_u, S_u, P_u)$ into $H_1$-List. $C$ returns $Q_u$ to Adv2.

*Secret-Value Query*. Adv2 adaptively queries for set-secret-value for any identity $ID_u$. $C$ scans for $ID_u$ in $H_1$-List. If $ID_u$ does not exist, $C$ queries itself for $H_1(ID_u)$. If $(b_u = 1)$, abort; else, $C$ retrieves $S_u$ and returns it to Adv2.

*Public-Key Query*. Adv2 adaptively queries for public-key for any identity $ID_u$. $C$ scans for $ID_u$ in $H_1$-List. If $ID_u$ does not exist, $C$ queries itself for $H_1(ID_u)$. $C$ retrieves $P_u$ and returns it to Adv2.

*$H_2$ Hash Query*. Adv2 adaptively queries for $(w_k, P_u)$. For every $H_2$ query, $C$ maintains a list $H_2$-List $= \{(w_k, H_k^2, r_k, P_u)\}$. $C$ scans $H_2$-List for $(w_k, P_u)$. If $(w_k, P_u)$ does not exist, $C$ randomly chooses $r_k \in Z_q*$, sets $H_k^2 = (h_2)^{r_k}$, and updates $H_2$-List. $C$ retrieves $H_k^2$ and returns it to Adv2.

*Forge-Tag*. Adv2 outputs a tuple $(\sigma', m', w_k', ID_u', P_u')$, where $\sigma'$ is the forged tag on the message $m'$ with authenticating information $w_k'$ for identity $ID_u'$ with public key $P_u'$.

*Analysis*. If Adv2 successfully forges the tag, then the tuple should be valid. If $(b_u = 0)$, then $C$ aborts. Else if $(b_u = 1)$, $C$ uses the following verification equation: $e(\sigma_{m'}, g) = e(H_1(ID_u')^{m'}, P_{pub})\cdot e(H_2(w_k', P_u'), P_u')$. $C$ retrieves the tuple from $H_1$-List for the forged $ID_u'$, and $C$ retrieves the value of $Q_u = g^{r_u}$ from $H_1$-List and $H_2(w_k', P_u') = (h_2)^{r_k'}$ from $H_2$-List.

As per the verification equation:

$$\begin{aligned} e(\sigma', g) &= e\left(g^{r_u\prime\cdot m'}, P_{\text{pub}}\right)\cdot e\left(h_2^{r_k'}, g^{\alpha\cdot\beta_u}\right), \\ e(\sigma', g) &= e\left(g^{r_u\prime\cdot m'\cdot s}, g\right)\cdot e\left(h_2^{r_k'\alpha\cdot\beta_u}, g\right), \\ h_2^{\alpha} &= \left(\frac{\sigma'}{g^{r_u\prime\cdot m'\cdot s}}\right)^{1/r_k'\beta_u}. \end{aligned} \tag{33}$$

We now calculate the minimum success probability $\xi'(k)$ with which the challenger $C$ can solve an instance of

the given CDH problem in $G_1$. For this to happen, the challenger $C$ must not abort during the above simulation. There are two phases in the simulation where the challenger $C$ can abort, i.e., in *secret-value query* phase and *analysis* phase. The *secret-value query* phase does not abort when $b_u = 0$, whereas the *analysis* phase does not abort when $b_u = 1$. Hence, the probability that the challenger $C$ does not abort the above simulation is $(1-\varphi)\cdot\varphi^{q_2}$. The quantity $(1-\varphi)\cdot\varphi^{q_2}$ is maximized when $\varphi = \varphi_{\max} = q_2/1+q_2$. By substituting the above value of $\varphi_{\max}$, we get the lower bound on the probability that the challenger $C$ does not abort as $1/e\cdot(1+q_2)$. Hence, the minimum success probability with which the challenger $C$ can solve an instance of the given CDH problem is $\xi'(k) \geq \xi(k)/e\cdot(1+q_2)$. It can be noted that since $q_2$ is polynomial$(k)$, the quantity $1/(1+q_2)$ is non-negligible. Thus, if $\xi(k)$ is non-negligible, then the success probability $\xi'(k)$ is also non-negligible.

The running time of $C$ is the sum of upper bounded running time of adversary algorithm Adv2 ($O$ (Adv2)) and upper bounded running time of all the algorithms used in simulating the challenger $C$, i.e., $H_1$ hash query $(t_1)$, secret-value query $(t_2)$, public-key query $(t_3)$, and $H_2$ hash query $(t_4)$ executed $q_1, q_2, q_3$, and $q_4$, times respectively. Therefore, $O(C) = O(\text{Adv2}) + O(q_1\cdot t_1 + q_2\cdot t_2 + q_3\cdot t_3 + q_4\cdot t_4)$. Now since all the algorithms used in the simulation of challenger $C$ are polynomial-time computable, $O(C) = O(\text{Adv2}) + O(q_1+q_2+q_3+q_4)$. It can noted that $q_1, q_2, q_3$, and $q_4$ all are in polynomial$(k)$. Therefore, $O(C) \approx O(\text{Adv2})$. It can be observed that if Adv2 is a polynomial-time algorithm, then the simulated CDH problem solver $C$ is also a polynomial-time algorithm.

The following theorem proves that, in the random oracle model, the data audit proofs generated by the proposed basic CLS-RDIC scheme are unforgeable by a type-III attacker assuming that the CDH and DL problem in $G_1$ are hard. □

**Theorem 3.** *If an adversary Adv3, as defined in game-3, has a success probability of $\xi(k)$ in generating a forged data audit proof against our proposed basic CLS-RDIC scheme, then there exists a challenger C that solves an instance of a CDH or DL problem in group $G_1$. Suppose Adv3 makes at most $q_1$ $H_1$ hash queries, $q_2$ data-block queries, $q_3$ public-key queries, and $q_4$ tag-gen and $H_2$ queries, where $q_1, q_2, q_3, q_4$ all are in polynomial$(k)$; then, the success probability, $\xi'(k)$, of challenger C in solving the above problem is lower bounded by*

$$\xi'(k) \geq \xi(k)\cdot\left(1-\frac{1}{q}\right), \tag{34}$$

*where $q$ is the order of the group $G_1$. In addition, the time complexity classes are $O(C) \approx O(Adv3) + O(q_1+q_2+q_3+q_4)$.*

*Proof.* Let Adv3 be the adversary as in game-3 who is capable of generating a valid data possession proof for the challenged blocks. Then, we construct a PPT simulator $C$ using Adv3 as subroutine to solve either the DH or the CDH problem in $G_1$. $C$ simulates game-3 in the following way.

*Setup*. Given an instance of CDH problem $(g, G_1, (h_1 = g^a), (h_2 = g^b))$ and an instance of the DL problem $(g, G_1, (h_2 = g^b))$. $C$ chooses another cyclic multiplicative group $G_2$, where $(|G_1| = |G_2| = q)$, and an admissible bilinear map $\hat{e}$: $G_1 \times G_1 \longrightarrow G_2$, sets the public parameters params $= \langle G_1, G_2, q, \hat{e}, g, P_{\text{pub}} = h_1\rangle$, and sends params to Adv3, and $a$ and $b$ are unknown to $C$.

*$H_1$ Hash Query*. Adv3 adaptively queries for $H_1$ for any identity $ID_u$. For every $H_1$ query, $C$ maintains a list $H_1$-List $= \{(ID_u, Q_u, r_u, \beta_u, P_u)\}$. If $ID_u$ is already queried before, $C$ simply retrieves the tuple from the $H_1$-List and returns $Q_u$ to Adv3. If $ID_u$ is new, $C$ randomly picks $r_u, \beta_u \in Z_q^*$ and sets $Q_u = g^{r_u}h_2^{\beta_u}$ and $P_u = h_2^{\beta_u}$. Finally, it inserts the values $(ID_u, Q_u, r_u, \beta_u, P_u)$ into $H_1$-List. $C$ returns $Q_u$ to Adv3.

*Data-Block Query*. The file $F = \langle m_1, \ldots, m_n\rangle$ with file identifier $ID_F$ is available with $C$, for $n$ is the number of blocks in the file. $C$ maintains an indicator matrix $I$. Adv3 can adaptively query for block index $i$. Then, $C$ retrieves corresponding data block from the date file $ID_F$ and forwards to Adv3.

*Public-Key Query*. Adv3 adaptively queries for the public key of any identity $ID_u$. $C$ scans for $ID_u$ in $H_1$-List, and if $ID_u$ does not exist, $C$ queries itself for $H_1(ID_u)$. $C$ retrieves $P_u$ and returns it to Adv3.

*Tag-Gen Query and $H_2$ Query*. $C$ maintains a list $H_2'$-List $= \langle i, b_i, H_2(ID_F\|i\|P_u), \sigma_i\rangle$ and computes tag for each index in $ID_F$ corresponding to user $u$ shown in indicator matrix $I$ as follows.

$C$ randomly picks $b_i \in Z_q^*$, for $i \in 1, \ldots, n$. For each index $i$, $C$ computes $H_2(ID_F\|i\|P_u) = g^{b_i}h_1^{-m_i}$ and $\sigma_i = h_1^{r_u m_i}\cdot h_2^{b_i\beta_u}$, where $\beta_u$ and $r_u$ are retrieved from $H_1$-List corresponding to $P_u$.

The validity of the tag simulation can be verified as follows.

As per the tag-gen algorithm, $\sigma_i = D_u^{m_i}H_2(ID_F\|i\|P_u)^{b\beta_u}$.

As per simulation, $\sigma_i = Q_u^{am_i}(g^{b_i}h_1^{-m_i})^{b\beta_u}$.

By replacing value of $Q_u$, $\sigma_i = (g^{r_u}\cdot h_2^{\beta_u})^{am_i}\cdot g^{bb_i\beta_u}\cdot h_1^{-bm_i\beta_u}$.

By replacing values of $h_1, h_2$, $\sigma_i = g^{ar_um_i}\cdot g^{abm_i\beta_u}\cdot g^{bb_i\beta_u}\cdot g^{-abm_i\beta_u}$.

$\sigma_i = (g^a)^{r_um_i}\cdot(g^b)^{b_i\beta_u} = h_1^{r_um_i}\cdot h_2^{b_i\beta_u}$.

$C$ adds the following tuple to its list $H_2'$-List $= \langle i, b_i, H_2(ID_F\|i\|P_u), \sigma_i\rangle$. $C$ sends $\langle\sigma_1, \ldots, \sigma_n\rangle$, corresponding file $F = \langle m_1, \ldots, m_n\rangle$ and indicator matrix $I_{i,u}$ to Adv3.

*Challenge*. $C$ chooses a number $c$ $(1 \leq c \leq n)$ as the size of the challenge vector and two random values $k_1, k_2 \in \{0,1\}^l$ using the global function $\Omega$, which act as the seeds for the two pseudo-random functions $\pi$ and $f$, respectively. $C$ sends the challenge request as chal $= \langle c, C = \{(i, v_i)\}, ID_F\rangle$ to the Adv3, where $i = \pi(k_1, x)$, $v_i = f(k_2, x)$ for $1 \leq x \leq c$.

*Forge-Proof.* Eventually Adv3 outputs proof of possession $P' = (\mu', \sigma')$ where $\mu' = \{\mu_1', \mu_2', \ldots, \mu_d'\}$ and $\sigma' = \{\sigma_1', \sigma_2', \ldots, \sigma_d'\}$.

*Analysis.* $C$ calculates actual proof $P = (\mu, \sigma)$ where $\mu = \{\mu_1, \mu_2, \ldots, \mu_d\}$ and $\sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_d\}$.

Further $C$ computes the following values:

(1) $\sigma_\pi = \prod_{u=1}^{d} \sigma_u$ and $\sigma_\pi' = \prod_{u=1}^{d} \sigma_u'$.

(2) $\mu_\pi = \prod_{u=1}^{d} Q_u^{\mu_u}$ and $\mu_\pi' = \prod_{u=1}^{d} Q_u^{\mu_u'}$.

(3) For each, $i \in C$ do the following:

(a) Get $u$ such that $I_{i,u} = 1$.

(b) Calculate $w_i = ID_F \| i \| P_u$.

Both the proofs $P$ and $P'$ should be valid as per the following equality:

$$e(\sigma_\pi, g) = e\left(\mu_\pi, P_{\text{pub}}\right). \prod_{u=1}^{d} e\left(\prod_{i \in C} H_2(w_i)^{\nu_i I_{i,u}}, P_u\right), \tag{35}$$

$$e(\sigma_\pi', g) = e\left(\mu_\pi', P_{\text{pub}}\right). \prod_{u=1}^{d} e\left(\prod_{i \in C} H_2(w_i)^{\nu_i I_{i,u}}, P_u\right). \tag{36}$$

If $\mu_\pi = \mu_\pi'$, then Adv3 stores the challenged message blocks correctly which contradicts our assumption. So, there are two cases. □

*Case 1.* $\sigma_\pi = \sigma_\pi'$.

Equating (35) and (36),

$$\begin{aligned}
e\left(\prod_{u=1}^{d} (Q_u)^{\mu_u}, P_{\text{pub}}\right) &= e\left(\prod_{u=1}^{d} (Q_u)^{\mu_u'}, P_{\text{pub}}\right) e\left(\prod_{u=1}^{d} (Q_u)^{\mu_u}, g^a\right) \\
&= e\left(\prod_{u=1}^{d} (Q_u)^{\mu_u'}, g^a\right) e\left(\prod_{u=1}^{d} (Q_u)^{\mu_u \cdot a}, g\right) \\
&= e\left(\prod_{u=1}^{d} (Q_u)^{\mu_u' \cdot a}, g\right),
\end{aligned}$$

$$\prod_{u=1}^{d} Q_u^{\mu_u \cdot a} = \prod_{u=1}^{d} Q_u^{\mu_u' a} \prod_{u=1}^{d} (Q_u^a)^{\Delta\mu_u} = 1, \text{ for } \quad \Delta\mu_u = \mu_u - \mu_u',$$

$$\begin{aligned}
(g^a)^{\sum_{u=1}^{d} r_u \Delta\mu_u} \cdot (h_2^a)^{\sum_{u=1}^{d} \beta_u \Delta\mu_u} &= 1 (h_2^a)^{\sum_{u=1}^{d} \beta_u \Delta\mu_u} \\
&= \left( (h_1)^{\sum_{u=1}^{d} r_u \Delta\mu_u} \right) h_2^a \\
&= (h_1)^{-\frac{\sum_{u=1}^{d} r_u \Delta\mu_u}{\sum_{u=1}^{d} \beta_u \Delta\mu_u}} \\
h_1^b &= (h_1)^{-\frac{\sum_{u=1}^{d} r_u \Delta\mu_u}{\sum_{u=1}^{d} \beta_u \Delta\mu_u}}.
\end{aligned} \tag{37}$$

Then, $C$ can compute solution to discrete log of $h_2$ as

$$b = -\frac{\sum_{u=1}^{d} r_u \Delta\mu_u}{\sum_{u=1}^{d} \beta_u \Delta\mu_u}. \tag{38}$$

It can be observed that the solution to an instance of DL problem is found, unless the denominator $\sum_{u=1}^{d} \beta_u \Delta\mu_u$ is zero. However, at least one of $\Delta\mu_u$ is non-zero, and the values $\{\beta_u\}$ are randomly chosen by the challenger $C$ from $Z_q*$. Hence, the probability that denominator is zero is $1/q$. We now calculate the minimum success probability $\xi'(k)$ with which the challenger $C$ can solve an instance of the given DL problem in $G_1$. For this to happen, the denominator in the exponent must be non-zero and its probability is $(1 - 1/q)$.

Hence, the minimum success probability $\xi'(k)$ with which the challenger $C$ can solve an instance of the given DL problem in $G_1$ is $\xi'(k) \geq \xi(k) \cdot (1 - 1/q)$. It can be noted that since $q$ is exponential$(k)$, the quantity $(1 - 1/q)$ is non-negligible. Thus, if $\xi(k)$ is non-negligible, then the success probability $\xi'(k)$ is also non-negligible.

*Case 2.* $\sigma_\pi \neq \sigma_\pi'$.

Dividing (26) by (20):

$$\begin{aligned}
\frac{\sigma_\pi'}{\sigma\pi} &= \prod_{u=1}^{d} (Q_u)^{\Delta\mu_u a} \sigma_\pi' \sigma_\pi^{-1} \\
&= \prod_{u=1}^{d} (Q_u)^{\Delta\mu_u a} \\
(g^a)^{\sum_{u=1}^{d} r_u \Delta\mu_u} \cdot (h_2^a)^{\sum_{u=1}^{d} \beta_u \Delta\mu_u} &= \sigma_\pi' \sigma_\pi^{-1} \\
(h_2^a)^{\sum_{u=1}^{d} \beta_u \Delta\mu_u} &= \left( \sigma_\pi' \sigma_\pi^{-1} \left( h_1^{-\sum_{u=1}^{d} r_u \Delta\mu_u} \right) \right) \\
h_2^a &= \left( \sigma_\pi' \sigma_\pi^{-1} \left( h_1^{-\sum_{u=1}^{d} r_u \Delta\mu_u} \right) \right)^{1/\sum_{u=1}^{d} \beta_u \Delta\mu_u}.
\end{aligned} \tag{39}$$

$C$ knows $\sigma_\pi, \sigma_\pi', \Delta\mu_u, h_1, \{r_u, \beta_u\}$, and hence $C$ can compute $h_2^a$, the solution to the CDH problem.

It can be observed that the solution to an instance of CDH problem is found, unless the denominator in the exponent $\sum_{u=1}^{d} \beta_u \Delta\mu_u$ is zero. However, we have already noted that not all of $\Delta\mu_u$ can be zero, and the values $\{\beta_u\}$ are randomly chosen by the challenger $C$ from $Z_q *$. Hence, the probability that denominator is zero is $1/q$. We now calculate the minimum success probability $\xi'(k)$ with which the challenger $C$ can solve an instance of the given CDH problem. For this to happen, the denominator in the exponent must be non-zero and its probability is $(1/q)$. Hence, the minimum success probability $\xi'(k)$ with which the challenger $C$ can solve an instance of the given CDH problem in $G_1$ is $\xi'(k) \geq \xi(k) \cdot (1 - 1/q)$. It can be noted that since $q$ is exponential$(k)$, the quantity $(1 - 1/q)$ is non-negligible. Thus, if $\xi(k)$ is non-negligible, then the success probability $\xi'(k)$ is also non-negligible.

The running time of $C$ is the sum of upper bounded running time of adversary algorithm Adv3 ($O$ (Adv3)) and upper bounded running time of all the algorithms used in simulating the challenger $C$, i.e., $H_1$ hash query $(t_1)$, data-block query $(t_2)$, public-key query $(t_3)$, and tag-gen and $H_2$ query $(t_4)$ executed $q_1, q_2, q_3$, and $q_4$ times, respectively. Therefore,

$O(C) = O(\text{Adv3}) + O(q_1 \cdot t_1 + q_2 \cdot t_2 + q_3 \cdot t_3 + q_4 \cdot t_4)$.

Now since all the algorithms used in the simulation of challenger $C$ are polynomial-time computable, $O(C) = O(\text{Adv3}) + O(q_1 + q_2 + q_3 + q_4)$. It can noted that $q_1, q_2, q_3$, and $q_4$ all are in polynomial$(k)$. Therefore, $O(C) \approx O(\text{Adv3})$. It can be observed that if Adv3 is a polynomial-time algorithm, then the simulated DL or CDH problem solver $C$ is also a polynomial-time algorithm.

*8.2. Soundness Proof for the Proposed CRPPA Scheme.* The proposed CRPPA scheme uses the same tag-generation mechanism of that of our proposed basic CLS-RDIC scheme. Hence, the security proofs for tag-unforgeability will be same as Theorems 1 and 2 against type-I and type-II attackers, respectively. Since the proposed CRPPA scheme employs zero-knowledge mechanisms to establish privacy-preserving data possession proof, we provide both the soundness and the zero-knowledge privacy proofs to establish the security of the privacy-preserving data auditing mechanism against type-III and type-IV adversaries, respectively. Theorem 2 bases on Theorem 1 to establish the soundness of the privacy-preserving mechanism used in the CRPPA scheme against type-III attacker. Theorem 5 establishes the zero-knowledge data privacy proof against type-IV attacker. Finally, the soundness property of the *Verify-Proof* algorithm of the proposed CRPPA scheme is established in Theorem 4.

The following theorem proves that, in the random oracle model, the tags generated by the proposed CRPPA scheme are unforgeable by a type-III attacker assuming the unforgeability of the data audit proof of the proposed basic CLS-RDIC scheme.

**Theorem 4.** *If an adversary Adv3 has a success probability of $\xi(k)$ in generating a forged data audit proof against our proposed CRPPA scheme, then there exists a challenger C that*

*generates a forged data audit proof against the proposed basic CLS-RDIC scheme. Suppose Adv3 makes at most $q_1$ $H_3$ hash queries and $q_2$ data-block and tag-gen queries, where $q_1, q_2$ both are in polynomial$(k)$, and $Pr[frk]$ denotes the forking probability as per forking lemma [67]. Then, the success probability, $\xi'(k)$, of challenger $C$ in solving the above problem is lower bounded by*

$$\xi'(k) \geq \xi(k) \cdot \left(\frac{1}{q_1} - \frac{1}{q}\right), \tag{40}$$

*where $q$ is the size of the co-domain $(Z_q*)$ of $H_3$ hash function. In addition, the time complexity classes are $O(C) \approx O(Adv3) + O(q_1 + q_2)$.*

*Proof.* Let Adv3 be the adversary as in game-3 who is capable of generating a valid data possession proof against the proposed CRPPA scheme for a given random challenge chal without having at least one of the challenged blocks in its possession. Then, we show that we can construct a PPT simulator $C$ using Adv3 as subroutine to forge the data possession proof for the proposed basic CLS-RDIC scheme for the same challenge chal. It can be noted here that the simulator $C$ acts as a challenger for type-III adversary against the CRPPA scheme and acts as type-III adversary against the proposed basic CLS-RDIC scheme at the same time. We use the forking theorem introduced by Bellare and Neven in [67] to establish our proof.

Given the public parameters params $= \langle G_1, G_2, q, \hat{e}, g, P_{\text{pub}} \rangle$ and instance of the challenge set chal $= \langle c, k_1, k_2, ID_F \rangle$, $C$ segregates the challenge chal into $d$ subsets, for each subset belongs to a unique user. $C$ simulates game-3 in the following way.

*Setup.* $C$ forwards params$' = \langle G_1, G_2, q, \hat{e}, g, g_1, P_{\text{pub}} \rangle$ and a random tape to Adv3.

*$H_3$ Hash Query.* Adv3 queries for $H_3$ for any value $R \in G_2$. For each $H_3$ query, $C$ maintains a $H_3$-List $= \{(R, \gamma)\}$. If $R$ is already queried, $C$ retrieves the tuple from the $H_3$-List and returns $\gamma$ to Adv3. If $R$ is new, $C$ randomly chooses a $\gamma \in Z_q*$ and sends $\gamma$ to Adv3 as response.

*Data-Block and Tag-Gen Queries.* Since the simulator $C$ is acting as a type-III adversary against the proposed basic CLS-RDIC scheme, it is also entertained for Data-Block and Tag-Gen queries, as defined in game-3. The Data-Block and Tag-Gen queries received by $C$ from Adv3 are entertained by it by utilizing its own privileged in this regard.

*Challenge.* $C$ forwards the challenge set corresponding to a user as $\text{chal}_u = \langle c_u, k_1, k_2, ID_F \rangle$ to Adv3.

*Forge-Proof.* Adv3 outputs privacy-preserving integrity proof $P_{1u} = \langle \mu_{1u}', \Sigma_u, \varepsilon_{1u}, R_u \rangle$. The value $R_u$ must have been queried to the hash oracle where $H_3(R_u) = \gamma_{1u}$. Since the output of $H_3$ hash oracle is uniformly random in $Z_q*$, the adversary Adv3 cannot obtain $H_3(R_u)$ without querying $H_3$. However, $H_3(R_u)$ is essential in generating a valid proof $P_{1u}$. Hence, the value of $\gamma_{1u}$ must have been queried.

$C$ uses the forking technique to rewind the simulation to the instance where the value $R_u$ is queried to the hash oracle. $C$ replies $\gamma_{2u}$ as the hash value of $R_u$. Finally, Adv3 outputs privacy-preserving integrity proof $P_{2u} = \langle \mu_{2u}', \Sigma_u, \varepsilon_{2u}, R_u \rangle$.

*Analysis.* Calculate: $\mu_u = \mu_{2u}' - \mu_{1u}'/\gamma_{2u} - \gamma_{1u}$, $\rho_u = \varepsilon_{2u} - \varepsilon_{1u}/\gamma_{2u} - \gamma_{1u}$, $\Sigma_u = g_1^{\rho_u}\sigma_u$, and $\sigma_u = \Sigma_u/g^{\rho_u}$.

Since both proofs $P_{1u}$ and $P_{2u}$ are valid proofs, they must satisfy auditor verification equation (20):

$$\begin{aligned} R_u \cdot e\left(\Sigma_u^{\gamma_{1u}}, g\right) &= e\left(Q_u^{\mu_{1u}'}, P_{\text{pub}}\right) \cdot e\left(g_1, g\right)^{\varepsilon_{1u}}, \\ &\quad e\left(\prod_{i \in C_u} H_2\left(w_i\right)^{v_i I_{i,u} \gamma_{1u}}, P_u\right), \\ R_u \cdot e\left(\Sigma_u^{\gamma_{2u}}, g\right) &= e\left(Q_u^{\mu_{2u}'}, P_{\text{pub}}\right) \cdot e\left(g_1, g\right)^{\varepsilon_{1u}}, \\ &\quad e\left(\prod_{i \in C_u} H_2\left(w_i\right)^{v_i I_{i,u} \gamma_{2u}}, P_u\right). \end{aligned} \tag{41}$$

Dividing (14) by (10), we get

$$\begin{aligned} e\left(\Sigma_u^{\gamma_{2u} - \gamma_{1u}}, g\right) &= e\left(Q_u^{\mu_{2u}' - \mu_{1u}'}, P_{\text{pub}}\right) \cdot e\left(g_1, g\right)^{\varepsilon_{2u} - \varepsilon_{1u}}, \\ &\quad e\left(\prod_{i \in C_u} H_2\left(w_i\right)^{v_i I_{i,u} \left(\gamma_{2u} - \gamma_{1u}\right)}, P_u\right). \end{aligned} \tag{42}$$

Taking root of $(\gamma_{2u} - \gamma_{1u})$ on both sides:

$$\begin{aligned} e\left(\Sigma_u, g\right) &= e\left(Q_u^{\mu_{2u}' - \mu_{1u}'/\gamma_{2u} - \gamma_{1u}}, P_{\text{pub}}\right) \cdot e\left(g_1, g\right)^{\varepsilon_{2u} - \varepsilon_{1u}/\gamma_{2u} - \gamma_{1u}}, \\ &\quad e\left(\prod_{i \in C_u} H_2\left(w_i\right)^{v_i I_{i,u}}, P_u\right). \end{aligned} \tag{43}$$

Substituting the value of $\Sigma_u, \mu_u, \rho_u$, we get

$$e\left(\sigma_u, g\right) = e\left(Q_u^{\mu_u}, P_{\text{pub}}\right).e\left(\prod_{i \in C} H_2\left(w_i\right)^{v_i I_{i,u}}, P_u\right), \tag{44}$$

which is nothing but the auditor verification equation for the user $u$, and the values $(\sigma_u, \mu_u)$ correspond to the integrity proof of our proposed basic CLS-RDIC scheme.

Repeating the above simulation for all the users, we finally output the integrity proof corresponding to the challenge chal for the proposed basic CLS-RDIC scheme, which is a set of $\{\sigma_u, \mu_u\}$ for all $u$ in chal.

We now calculate the minimum success probability $\xi'(k)$ with which the challenger $C$ can generate a forged data audit proof against the basic CLS-RDIC scheme. For this to happen, the challenger $C$ must not abort during the above simulation. The challenger aborts only when forking is unsuccessful. So, let us calculate the success probability of forking as discussed in Section 3.2. Here in this simulation, the random oracle RO is the hash function $H_3$. We respond to a total of $q_1$ number of RO queries received from the adversary Adv3. For each of these queries, we return a random value from the co-domain $Z_q*$. The output

generated by the adversary algorithm Adv3 is the proof $P_{1u}$ which contains a component $R_u$. As stated earlier, $R_u$ must have been queried to the hash oracle $H_3$. Hence, the acceptance probability of the adversary algorithm Adv3 is $\Pr[\text{acc}] = 1$. Thus, applying the forking lemma as stated in Section 3.2, the forking success probability $\Pr[frk] \geq (1/q_1 - 1/|Z_q*|) = (1/q_1 - 1/q)$. Hence, the minimum success probability with which the challenger $C$ can generate a forged data audit proof against the basic CLS-RDIC scheme is $\xi'(k) \geq \xi(k) \cdot \Pr[frk] = \xi(k) \cdot (1/q_1 - 1/q)$. It can be noted that $\Pr[frk]$ is non-negligible, since $q_1$ is in polynomial$(k)$ and $q$ is in exponential$(k)$. Thus, if $\xi(k)$ is non-negligible, then the success probability $\xi'(k)$ is also non-negligible.

The running time of $C$ is the sum of upper bounded running time of adversary algorithm Adv3 ($O$ (Adv3)) and upper bounded running time of all the algorithms used in simulating the challenger $C$, i.e., $H_3$ hash query $(t_1)$ and data-block and tag-gen queries $(t_2)$ executed $q_1$ and $q_2$ times, respectively. Therefore, $O(C) = O(\text{Adv3}) + O(q_1 \cdot t_1 + q_2 \cdot t_2)$. Now since all the algorithms used in the simulation of challenger $C$ are polynomial-time computable, $O(C) = O(\text{Adv1}) + O(q_1 + q_2)$. It can be noted that $q_1$ and $q_2$ both are in polynomial$(k)$. Therefore, $O(C) \approx O(\text{Adv3})$. It can be observed that if Adv3 is a polynomial-time algorithm, then the simulated data audit proof forger $C$ against basic CLS-RDIC scheme is also a polynomial-time algorithm. □

**Theorem 5.** *The proposed CRPPA protocol is secure against type-IV attacker, in the random oracle model. Suppose Adv4 makes at most $q_1 H_1$ hash queries, where $q_1$ is in polynomial$(k)$. Then, the success probability that the simulator $S$ can generate a valid data possession proof corresponding to a given challenge is*

$$\xi'(k) \approx 1 - \frac{1}{q}, \tag{45}$$

*where $q$ is the order of the group $Z_q*$. In addition, the time complexity classes are $O(C) \approx O(q_1)$.*

*Proof.* We prove the above theorem by showing that it is possible to design a simulator $S$ that can generate a valid data possession proof corresponding to any given challenge chal without any knowledge about the data and tag blocks, if it has control over the random oracle hash function $H_3$ used in the masking technique of CRPPA. We show the success probability of such simulation to be non-negligible.

*Setup*. The simulator $S$ receives a random tape and public parameters params.

*$H_3$ Hash Query*. In order to simulate $H_3$ hash function as a random oracle, $S$ maintains a $H_3$-List = $\{(R, \gamma)\}$. If $R$ is already queried, $S$ retrieves the tuple from the $H_3$-List and returns $\gamma$ as the response. If $R$ is new, $S$ randomly chooses a $\gamma \in Z_q*$ and returns $\gamma$ as the response.

*Challenge*. $S$ receives the challenge set corresponding to a user $ID_u$ as chal = $\langle c, k_1, k_2, ID_F, ID_u \rangle$.

*Forge-Proof*. After receiving the challenge chal = $\langle c, k_1, k_2, ID_F, ID_u \rangle$, $S$ computes the forged proof $P$ using the following steps:

(1) $C = \{(i, v_i)\}$, where $i = \pi(k_1, x)$ and $v_i = f(k_2, x)$ for all $1 \leq x \leq c$ where $c$ is the number of challenged indices.
(2) $Q_u = H_1(ID_u)$.
(3) Choose $\gamma, \mu', \varepsilon$ randomly from $Z_q^*$.
(4) Choose $\Sigma$ randomly from $G_1$.
(5) Compute $R = e(Q_u^{\mu'}, P_{\text{pub}}) \cdot e(g_1, g)^{\varepsilon} \cdot e(\prod_{i \in C} H_2(w_i)^{v_i \gamma}, P_u) / e(\Sigma, g)^{\gamma}$.
(6) If $R$ already exists in the $H_3$-List, then abort the simulation.
(7) Else, set $H_3(R) \leftarrow \gamma$.

The simulator $S$ returns proof $P = \langle \mu', \Sigma, \varepsilon, R \rangle$. It is easy to check that $P$ is a valid data integrity proof corresponding to the given challenge chal.

*Success Probability*. The success probability of this simulation $\xi'(k) = 1 - (\text{failure probability of simulation}) =$

$$\xi'(k) = 1 - P(F), \tag{46}$$

where $P(F) =$ the probability with which simulator $S$ aborts. The simulation aborts only in step 6, when the value of $R$ computed in step 5 already exists in $H_3$-List.

Let polynomial number $(q_l)$ of $H_3$ queries be received by the simulator $S$ during the simulation. The probability that the value $R$ computed in step 5 matches with any one of the previously queried $H_3$ queries is proportional to $q_l/q$, where $q$ is the order of the group $Z_q*$. The value $q_l/q$ is an inverse exponential function of $k$, and thus $\xi'(k) \approx 1 - 1/q$ is non-negligible.

The running time of $C$ is the upper bounded running time of the algorithm $H_3$ hash query $(t_1)$ executed $q_1$ times, which is $(O(q_1 \cdot t_1))$. Now the algorithm used in the simulation $S$ is polynomial-time computable, $O(C) = O(q_1)$. It can be noted that $q_1$ is in polynomial$(k)$. Therefore, $O(C) \approx O(\text{polynomial}(k))$. Hence, simulator $S$ is polynomial-time algorithm. □

**Theorem 6.** *The aggregate verification done in the Verify-Proof algorithm of the proposed CRPPA protocol is sound. In other words, if the aggregate audit verification as per the Verify-Proof algorithm in Section 7.2 results in success, then each of the $t \in T$ instances of the data possession proofs is correct.*

*Proof.* We prove the above given theorem through proof by contradiction. We know that $\forall t \in T$, $P(t) = \langle \mu'(t), \Sigma(t)(t), \varepsilon(t), R(t) \rangle$, where $\mu'(t) = \{\mu_1'(t), \mu_2'(t), \ldots, \mu_d'(t)\}$, $\Sigma(t) = \{\Sigma_1(t), \Sigma_2((t)), \ldots, \Sigma_d(t)\}$, $\varepsilon(t) = \{\varepsilon_1(t), \varepsilon_2(t), \ldots, \varepsilon_d(t)\}$, $R(t) = \{R_1(t), R_2(t), \ldots, R_d(t)\}$. We assume that there is at least one proof, say, the $x^{\text{th}}$ instance (for some $x \in T$) where the proof $P(x)$ is not valid as per the *Proof-Ver* algorithm of the CRPPA protocol,

but the aggregate verification for all the $t \in T$ instances still passes the verification test as per the *Verify-Proof* algorithm.

As per the assumption above, the $x^{\text{th}}$ instance does not pass the verification test of *Proof-Verify* algorithm. Hence,

$$\prod_u R_u(x) \cdot e\left(\prod_u \Sigma_u(x)^{\gamma_u(x)}, g\right) \neq \prod_u \left(e\left(Q_u^{\mu_u'(x)}, P_{\text{pub}}\right) \cdot e\left(g_1^{\varepsilon_u(x)}, g\right)\right) \cdot e\left(\prod_{i \in C_x} H_2(w_i)^{\nu_i I_{i,u} \gamma_u(x)}, P_u\right). \tag{47}$$

But $\forall t \in T$ instances of the data possession proofs pass the verification test as per the *Verify-Proof* algorithm, which implies the following:

$$\prod_{\forall t \in T} \prod_u R_u(t) \cdot e\left(\prod_{\forall t \in T} \prod_u \Sigma_u(t)^{\gamma_u(t)}, g\right) = \prod_{\forall t \in T} \left(\left(\prod_u e\left(\prod_{i \in C_t} H_2(w_i)^{\nu_i I_{i,u} \gamma_u(t)}, P_u\right)\right)\right) \cdot e\left(\prod_{\forall t \in T} \prod_u Q_u^{\mu_u'(t)}, P_{\text{pub}}\right) \cdot e(g_1, g)^{\sum_{\forall t \in T} \sum_u \varepsilon_u(t)}. \tag{48}$$

The above also implies that $\forall t \in T \& t \neq x$ instances of the data possession proofs also pass the verification test as per the *Verify-Proof* algorithm, and hence, we get

$$\prod_{\forall t (\neq x) \in T} \prod_u R_u(t) \cdot e\left(\prod_{\forall t (\neq x) \in T} \prod_u \Sigma_u(t)^{\gamma_u(t)}, g\right) = \prod_{\forall t (\neq x) \in T} \left(\left(\prod_u e\left(\prod_{i \in C_t} H_2(w_i)^{\nu_i I_{i,u} \gamma_u(t)}, P_u\right)\right)\right) \cdot e\left(\prod_{\forall t (\neq x) \in T} \prod_u Q_u^{\mu_u'(t)}, P_{\text{pub}}\right) \cdot e(g_1, g)^{\sum_{\forall t (\neq x) \in T} \sum_u \varepsilon_u(t)}. \tag{49}$$

Dividing (48) by (49), we get

$$\prod_u R_u(x) \cdot e\left(\prod_u \Sigma_u(x)^{\gamma_u(x)}, g\right) = \prod_u \left(e\left(Q_u^{\mu_u'(x)}, P_{pub}\right) \cdot e\left(g_1^{\varepsilon_u(x)}, g\right)\right) \cdot e\left(\prod_{i \in C_x} H_2(w_i)^{\nu_i I_{i,u} \gamma_u(x)}, P_u\right). \tag{50}$$

Equation (50) shows that the $x^{\text{th}}$ instance passes the verification test of *Verify-Proof* algorithm which contradicts with equation (47) derived from the assumption. Hence, the theorem is proved.

In the following two subsections, we explain in brief how the mechanisms adopted in the proposed CRPPA scheme can prevent and detect type-V and type-VI attacks, respectively. □

*8.2.1. Security against Type-V Attack.* Type-V attack can be launched in two different ways as discussed in Section 7.1.2. Hence, we need to detect an incorrect data auditing report generated by the auditor and also restrict the auditor from disclosing the random values of the challenge vectors in advance. Using the *Verify-Proof* algorithm of the proposed CRPPA scheme, any user, acting as audit-verifier, can verify *correctness* of the audit reports. We have established the completeness and the soundness of this algorithm in Lemma 5 and Theorem 6, respectively. Second, the CRPPA protocol ensures that all parties, including the data auditor, CSP, and audit-verifier, generate the challenge vector chal from the hash-block of a public blockchain corresponding to the audit time, which is specified in the SLA. The hash-block values in the blockchain continuously get updated, and it is not feasible to guess what would be the value of the block for a future instance of time. Hence, the auditor cannot disclose to the CSP in advance the challenge vectors for a future time instance.

*8.2.2. Security against Type-VI Attack.* Type-VI attack by the compromised selfish auditor can only be detected if (i) there is a timestamp associated for each audit report generated by the auditor, (ii) the auditor is not able to modify once the report is timestamped, and (iii) the timestamped report is publicly available for verification. In our proposed CRPPA scheme, the hash digests of the audit reports need to be inserted as a transaction into the public blockchain. A transaction in the blockchain is immutable, timestamped, and verified by the blockchain. Hence, a user, acting as audit-verifier, can always verify the *timeliness* of the auditing task performed by the auditor, as explained in the *Verify-Proof* algorithm of the CRPPA protocol.

## 9. Performance Analysis and Comparisons

In this section, we evaluate the performance of the proposed CRPPA scheme on two factors. First, we compare our CRPPA scheme with the recent CLS-based group shared data auditing schemes which find their applications in FOG-CPS environments. We specifically point to the concerns of their application to FOG-CPS environments based on the necessary cryptographic requirements and security against six types of attacks possible in a FOG-CPS. Second, we provide the computational cost comparisons with the well-known schemes in the remote data integrity checking field to understand the performance of the CRPPA scheme in the broad area of RDIC.

*9.1. A Qualitative Comparison of the Proposed CRPPA Scheme.* Table 3 shows the comparison of the proposed CRPPA scheme with the schemes of Jaya et al. [23], Yang et al. [24], Hongbin et al. [39], Cui et al. [21], Armknecht et al. [48], Zhang et al. [25], Yu et al. [12], and Li et al. [62] w.r.t the necessary security property requirements of a FOG-CPS which include group shared data, the cryptographic techniques employed for generating security credentials of the FOG-CPS users, the tag-unforgeability of the metadata

Table 3: Comparison of the proposed CRPPA scheme with the schemes in [12, 21, 23–25, 39, 48, 62] against six types of attacks.

| | Proposed CRPPA scheme | Jaya et al. [23] | Yang et al. [24] | Hongbin et al. [39] | Cui et al. [21] | Armknecht et al. [48] | Zhang et al. [25] | Yu et al. [12] | Li et al. [62] |
|---|---|---|---|---|---|---|---|---|---|
| Cryptography used | CLPKC | CLPKC | CLPKC | CLPKC | CLPKC | PKI | CLPKC | ID-PKC | CLPKC |
| Group shared data | Yes | Yes | Yes | Yes | Yes | No | No | No | Yes |
| Type-I secure | Yes | No | Yes | No | No | — | Yes | No | No |
| Type-II secure | Yes | No | Yes | No | No | — | Yes | No | No |
| Type-III secure | Yes | No | — | No | No | Yes | Yes | No | No |
| Type-IV secure | Yes | No | Yes | Yes | No | No | No | Yes | No |
| Type-V secure | Yes | No | No | No | No | Yes | Yes | No | No |
| Type-VI secure | Yes | No | No | No | No | No | Yes | No | No |

generation mechanism to protect from forged signature (type I and type II) and integrity proof forgery (type III), the data privacy of sensitive data of the edge user against the semi-trusted data auditor (type IV), reliability on the correctness (type V), and timeliness of the auditing tasks by the semi-trusted data auditor (type VI).

We discuss the specific security limitations of the schemes in [12, 21, 23–25, 39, 48, 62] against our proposed CRPPA scheme. The schemes in [12, 21, 23–25, 39, 62] are susceptible to tag forgery and integrity proof forgery attacks (type I, type II, and type III). The auditor in the schemes in [12, 21, 23–25, 39, 48, 62] after executing a large number of challenge-response instants can extract data file blocks or tags, i.e., these schemes are susceptible to type-IV attack and do not provide zero-knowledge data privacy against the auditor. Schemes in [12, 21, 23–25, 39, 48, 62] are susceptible to type-V attack and do not provide any mechanism to prevent the collusion of auditor and CSP on the random challenge vectors in advance used as a part of data auditing. Type-VI attack is possible in the schemes in [12, 21, 23–25, 39, 48, 62] where the auditor can delay or skip auditing tasks without even getting detected. Thus, these schemes in [12, 21, 23–25, 39, 48, 62] lack correctness and timeliness reliability of data auditing tasks by the auditor. Our proposed CRPPA is secure against all the six types of attacks and hence provides better security for FOG-CPS than the schemes in [12, 21, 23–25, 39, 48, 62].

Further, Table 4 presents various difficulty assumptions employed in the proposed CRPPA scheme and the schemes in [12, 21, 23–25, 39, 48, 62] to provide security against the six types of attack scenarios. The security against type-I and type-II attacks in [21, 23–25, 39, 48, 62] and our proposed CRPPA scheme is based on CDH assumption. The security against type-III attack in the schemes [23, 39] is based on the DL assumption, that in scheme [25] is based on CDH assumption, and that in scheme [48] and our CRPPA scheme is based on the CDH and DL assumptions. The security against type-IV attack in scheme [23] is based on CDH assumption and that in schemes [12, 39] is based on the one-way property of a hash function. In our proposed CRPPA scheme and in scheme [24], the security against type-IV attack is based on DL assumption. The security against type-V attack in scheme [48] and our CRPPA scheme is based on the consensus mechanism of blockchain and that in scheme [25] is based on proof-of-work mechanism in blockchain. Finally, our proposed CRPPA scheme and the scheme in [25] provide security against type-VI based on the immutability property of blockchain.

*9.2. Computation Cost.* To understand the performance of the proposed CRPPA scheme in the broad area of RDIC, we consider three well-known RDIC schemes. The first is Li et al.'s CLS-based group shared data auditing scheme [62], and we refer to this scheme as certificateless-RDIC scheme. The second scheme is a ID-based data privacy-preserving scheme proposed by Yu et al. [12], and we refer to this scheme as ID-RDIC scheme. The third is Zhang et al.'s scheme [25] that employs blockchain technology for verifying data auditing tasks by the auditor, and we refer to this scheme as blockchain-RDIC.

A detailed comparison of cryptographic operations cost is performed on the proposed CRPPA scheme w.r.t to schemes in [12, 25, 62]. Tables 5–7 show the cryptographic operation cost at the auditor, the CSP, and the user, respectively. We consider only the operations that have a significant impact on the performance of the schemes, and these operations are hash-to-element in $G_1$, exponentiation in $G_1$, multiplication in $G_1$ and $G_2$, and bilinear pairing operation. Let us denote their costs as $T_H, T_{\exp_{G_1}}, T_{\text{mul}_{G_1}}, T_{\text{mul}_{G_2}}$, and $T_p$, respectively.

The value $c$ in the tables corresponds to the number of challenged blocks in one challenge instance, the value $b$ corresponds to the number of blocks for which tags are generated, and the value $t$ corresponds to the number of challenge instances.

We have derived the cryptographic operations cost for each of the schemes in [12, 25, 62] by studying and analyzing each algorithm corresponding to the tag generation by the user, the proof generation by the CSP, and the proof verification by the auditor, and accordingly the significant cryptographic operation costs for each of the schemes are provided in Tables 5–7. The cryptographic operation costs in Tables 5–7 are for a single user, and the cryptographic operation costs for group of users can be calculated accordingly.

*9.3. Experimentation Results.* In Section 9.1, through a qualitative comparison of our proposed CRPPA scheme w.r.t the existing schemes in [12, 21, 23–25, 39, 48, 62], we have shown that the proposed protocol achieves a unique combination of security objectives, which has not been achieved by any of the existing protocols. Table 3 clearly summarizes the additional security objectives that we gain with respect to the existing approaches. In Section 8, through rigorous mathematical analysis, we have established the claimed security properties of the proposed protocol. In this

Table 4: Comparison of the proposed CRPPA scheme with the schemes in [12, 21, 23–25, 39, 48, 62] on the difficulty assumption employed against six types of attacks.

| | Proposed CRPPA scheme | Jaya et al. [23] | Yang et al. [24] | Hongbin et al. [39] | Cui et al. [21] | Armknecht et al. [48] | Zhang et al. [25] | Yu et al. [12] | Li et al. [62] |
|---|---|---|---|---|---|---|---|---|---|
| Type I | CDH | CDH | CDH | CDH | CDH | — | CDH | — | CDH |
| Type II | CDH | CDH | CDH | CDH | CDH | — | CDH | — | CDH |
| Type III | CDH & DL | DL | — | DL | — | CDH & DL | CDH | — | DL |
| Type IV | DL | CDH | DL | One-way property of hash function | — | — | — | One-way property of hash function | — |
| Type V | Consensus mechanism of blockchain | — | — | — | — | Consensus mechanism of blockchain | Proof-of-work mechanism in blockchain | — | — |
| Type VI | Immutability property of blockchain | — | — | — | — | — | Immutability property of blockchain | — | — |

*Note.* "—" indicates not applicable.

Table 5: Cryptographic operations cost at the auditor.

| Scheme | Proof verification cost |
|---|---|
| CRPPA | $cT_H + (c+2)T_{\exp_{G_1}} + (c-1)T_{\text{mul}_{G_1}} + T_{\text{mul}_{G_2}} + 3T_p$ |
| Certificateless-RDIC [62] | $cT_H + (c+1)T_{\exp_{G_1}} + (c-1)T_{\text{mul}_{G_1}} + T_{\text{mul}_{G_2}} + 3T_p$ |
| ID-RDIC [12] | $cT_H + (c)T_{\exp_{G_1}} + (c-1)T_{\text{mul}_{G_1}} + 1T_p$ |
| Blockchain-RDIC [25] | $cT_H + (c+4)T_{\exp_{G_1}} + (c+1)T_{\text{mul}_{G_1}} + 2T_{\text{mul}_{G_2}} + 4T_p$ |

subsection, through experimental evaluations, our objective is to demonstrate that we achieve these additional security objectives with almost no or marginal additional cost. We evaluate the performance of the proposed CRPPA scheme based on the computation cost of tag generation by the user, proof generation by the CSP, and proof verification by the data auditor against the schemes in [12, 25, 62]. The cost for the tag-generation mechanism for the user can be split into two types. The first is an offline tag-generation mechanism where some of the operations in the tag-generation mechanism can be calculated offline, independent of the file's data blocks. The second is the online tag-generation cost dependent on the file's data blocks.

We have performed the experimental computation cost analysis with the help of the PBC library [76] on an intel i5-6200 CPU@2.30 GHz and 4 GB DDR4 RAM for simulating the CSP and with the help of Android PBC [77] on a Quad-core Cortex-A7 mobile processor@1.6 GHz and 2 GB RAM for simulating the edge user device and edge data auditor device. The benchmark execution time in milliseconds for each cryptographic operation $T_H, T_{\exp_{G_1}}, T_{\text{mul}_{G_1}}, T_{\text{mul}_{G_2}}$, and $T_p$ is calculated on both the Linux OS PC device and Android OS mobile device and is shown in Table 8. Using the benchmark execution times in Table 6 and the number of cryptographic operations from Tables 5–7, we derive the computation cost in seconds for online and offline tag generation by the user for different file sizes, proof generation by the CSP, and proof verification by the data auditor based on the number of challenged blocks.

Figure 1 shows the results obtained from the above experimentation. From this figure, we can observe that the computation cost, in seconds, of tag generation (offline and online) is higher than that of proof generation and verification. This is obvious since the tag generation is executed for a complete data file and by a resource-constrained edge user device, whereas the proof generation and verification are executed for only the challenged number of blocks. It can also be observed that the online tag-generation cost is approximately 75% lower compared to offline cost. For instance, for a filesize of 1000 blocks, the offline cost is 10 seconds, whereas the online cost is only 2.5 seconds. The reason is that the users can precompute some operations beforehand to reduce the computational load required for online tag generation. It can be noted that although the tag-generation cost by an edge user device is higher, it is only a one-time process for a file, unlike the proof generations and verifications which are executed periodically. The proof generation cost by the cloud is comparatively very low compared to proof-verification cost by the data auditor since the cloud has higher computational power. In contrast, the data auditor is an edge device with low computational resources. This is the reason for which we have outlined few mechanisms in Section 5 to choose the data auditor from the group of edge devices to distribute the computational load among the edge devices.

Overall, from Figure 1, we can conclude that the computation costs of our proposed CRPPA scheme are comparable to the costs of the existing schemes in [12, 25, 62]. This is because the number of major

Table 6: Cryptographic operations cost at the CSP.

| Scheme | Proof generation cost |
|---|---|
| CRPPA | $cT_{\exp_{G_1}} + cT_{\mathrm{mul}_{G_1}}$ |
| Certificateless-RDIC [62] | $(c)T_{\exp_{G_1}} + (c-1)T_{\mathrm{mul}_{G_1}}$ |
| ID-RDIC [12] | $(c+1)T_{\exp_{G_1}} + (c-1)T_{\mathrm{mul}_{G_1}} + T_{\mathrm{mul}_{G_2}} + 2T_p$ |
| Blockchain-RDIC [25] | $cT_{\exp_{G_1}} + (c-1)T_{\mathrm{mul}_{G_1}}$ |

Table 7: Cryptographic operations cost at the user.

| Scheme | Tag generation cost |
|---|---|
| CRPPA | $bT_H + 2bT_{\exp_{G_1}} + bT_{\mathrm{mul}_{G_1}}$ |
| Certificateless-RDIC [62] | $bT_H + 2bT_{\exp_{G_1}} + bT_{\mathrm{mul}_{G_1}}$ |
| ID-RDIC [12] | $bT_H + 2bT_{\exp_{G_1}} + bT_{\mathrm{mul}_{G_1}}$ |
| Blockchain-RDIC [25] | $(b+3)T_H + (2b+5)T_{\exp_{G_1}} + 2(b+1)T_{\mathrm{mul}_{G_1}}$ |

Table 8: Execution times of various cryptographic operations.

| Operation | Linux PC device (millisecond) | Android mobile device (millisecond) |
|---|---|---|
| Bilinear pairing ($T_p$) | 3.25 | 18.42 |
| Hash-to-element in $G_1$ ($T_H$) | 0.6 | 6.78 |
| Multiplication in $G_1$ ($T_{\mathrm{mul}_{G_1}}$) | 0.05 | 0.1 |
| Exponentiation in $G_1$ ($T_{\exp_{G_1}}$) | 0.25 | 2.87 |
| Multiplication in $G_2$ ($T_{\mathrm{mul}_{G_2}}$) | 1.45 | 8.76 |

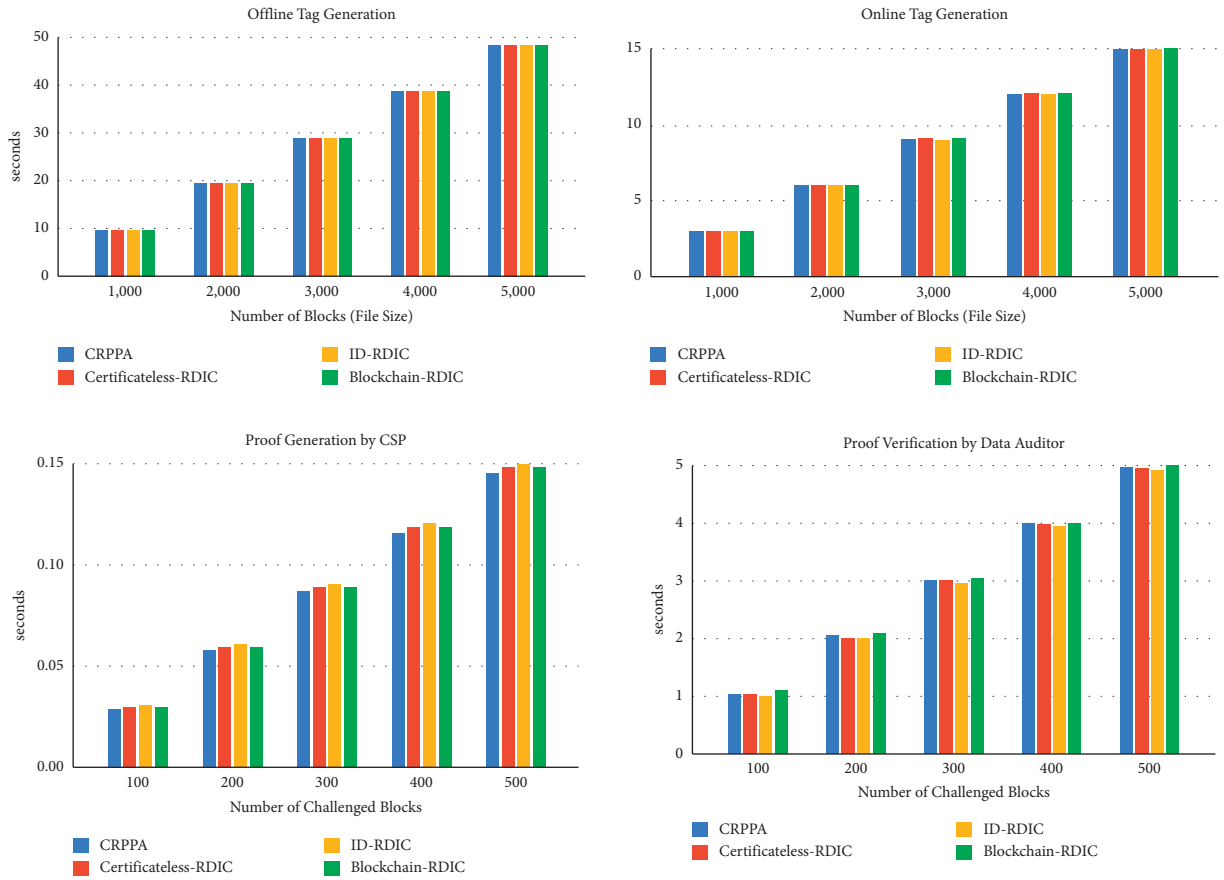


Figure 1: Performance comparison of various computation costs of our proposed CRPPA scheme with other related schemes.

cryptographic operations, such as $T_H, T_{\exp_{G_1}}, T_{\text{mul}_{G_1}}, T_{\text{mul}_{G_2}}$, and $T_p$, required for implementing each of the offline and online tag-generation algorithms executed by an edge user device, the proof-generation algorithm executed by the CSP, and the proof-verification algorithm executed by the data auditor in the proposed scheme and in the compared schemes is almost the same, as listed in Tables 5–7. Hence, the results confirm that the proposed protocol achieves its additional security objectives with marginal increase in the cost. Besides, the aggregate verification cost of verifying $n$ number of audit reports by an audit-verifier in our proposed CRPPA scheme, as shown in the *Verify-Proof* algorithm of Section 7.2, has only $(1/3)^{\text{rd}}$ of the computation cost required by an auditor to verify the same $n$ number of data possession proofs.

## 10. Conclusions and Future Work

In this paper, first, we have performed a detailed cryptanalysis of two CLPKC-based privacy-preserving shared data auditing schemes to pinpoint the exact vulnerabilities in their metadata generation mechanisms. Second, we have proposed a novel group shared data auditing protocol tailored to the specific security requirements of a FOG-CPS. The proposed data auditing protocol takes advantage of the localized storage and computing facilities available at the edge of the FOG-CPS network by delegating an edge device geographically close to the storage resources to perform the data auditing task. Our proposed protocol ensures zero-knowledge data privacy against the data auditor and can provide the reliability of the data auditing tasks done by a data auditor. We ensured the reliability of auditing tasks in two ways: enabling any user to verify the data auditing reports generated by the auditor and detect any procrastination of auditing tasks done by the auditor and ensuring that the challenge vectors are generated from a time-varying and verifiable universal source of randomness to restrict auditor and CSP collusion on the challenge vectors in advance. We have proved that the metadata generated by our protocol are unforgeable and are secure against both type-I and type-II attacks in the random oracle model setting. We have also established proof in support of the claimed zero-knowledge privacy and the reliability properties of our proposed protocol. The comparative performance evaluations establish the efficiency of our proposed protocol. As future work, we plan to incorporate user revocation, support data dynamics, and establish a collective auditing mechanism among the edge devices in a distributed way.

## Data Availability

The data used to support the findings of this study are included within the article.

## Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

## Acknowledgments

The authors acknowledge the support of Ministry of Education, Government of India, for partially funding the research.

## References

[1] F. Bonomi, R. Milito, Z. Jiang, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, pp. 13–16, Association for Computing Machinery, NY, USA, 2012.

[2] I. Stojmenovic and S. Wen, "The fog computing paradigm: scenarios and security issues," in *Proceedings of the 2014 Federated Conference on Computer Science and Information Systems*, pp. 1–8, IEEE, Warsaw, Poland, September 2014.

[3] Junejo, A. Kanwal, and N. Komninos, "A lightweight Attribute-based security scheme for fog-enabled cyber physical systems," *Wireless Communications and Mobile Computing*, vol. 2020, Article ID 2145829, 2020.

[4] X. Liu, W. Chen, Y. Xia, and C. Yang, "SEVFC: Secure and Efficient Outsourcing Computing in Vehicular Fog Computing," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, 2021.

[5] C. Badii, P. Bellini, A. Difino, and P. Nesi, "Smart city IoT platform respecting GDPR privacy and security aspects," *IEEE Access*, vol. 8, pp. 23601–23623, 2020.

[6] X. Shen, L. Zhu, C. Xu, K. Sharif, and R. Lu, "A privacy-preserving data aggregation scheme for dynamic groups in fog computing," *Information Sciences*, vol. 514, pp. 118–130, 2020.

[7] O. Yildirim, Feyza, S. Ozdemir, and Y. Xiao, "Fog computing-based privacy preserving data aggregation protocols," *Transactions on Emerging Telecommunications Technologies*, vol. 31, no. 4, Article ID e3900, 2020.

[8] Bo Li, Q. He, F. Chen, H. Jin, X. Yang, and Y. Yang, "Auditing cache data integrity in the edge computing environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 5, pp. 1210–1223, 2020.

[9] G. Ateniese, R. Burns, R. Curtmola et al., "Provable data possession at untrusted stores," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, pp. 598–609, ACM, Alexandria, VA, USA, October 2007.

[10] V. Arulkumar, R. Lathamanju, and A. Sandanakaruppan, "Assurance on data integrity in cloud data centre using PKI built RDIC method," *Recent Trends in Communication and Electronics*, CRC Press, Boca Raton, Florida, USA, pp. 98–102, 2021.

[11] M. J. Jeyasheela Rakkini and K. Geetha, "Secure decentralized public key infrastructure with multi-signature in blockchains," in *Lecture Notes in Networks and Systems Inventive Communication and Computational Technologies*, vol. 145, pp. 451–461, Springer, Singapore, 2021.

[12] Y. Yu, M. Ho Au, G. Ateniese et al., "Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 4, pp. 767–778, 2016.

[13] Y. Zhang, Yu Jia, H. Rong, C. Wang, and K. Ren, "Enabling efficient user revocation in identity-based cloud storage auditing for shared big data," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 3, pp. 608–619, 2018.

[14] J. F. Tian and H. N. Wang, "An efficient and secure data auditing scheme based on fog-to-cloud computing for

Internet of things scenarios," *International Journal of Distributed Sensor Networks*, vol. 16, no. 5, Article ID 1550147720916623, 2020.

[15] L. Huang, G. Zhang, and A. Fu, "Privacy-preserving public auditing for non-manager group shared data," *Wireless Personal Communications*, vol. 100, no. 4, pp. 1277–1294, 2018.

[16] Y. Zhang, C. Xu, S. Yu, H. Li, and X. Zhang, "SCLPV: secure certificateless public verification for cloud-based cyber-physical-social systems against malicious auditors," *IEEE Transactions on Computational Social Systems*, vol. 2, no. 4, pp. 159–170, 2015.

[17] S. Li, J. Cui, H. Zhong, and L. Liu, "Public auditing with privacy protection in a multi-user model of cloud-assisted body sensor networks," *Sensors*, vol. 17, no. 5, p. 1032, 2017.

[18] G. Wu, Y. Mu, W. Susilo, F. Guo, and F. Zhang, "Privacy-preserving certificateless cloud auditing with multiple users," *Wireless Personal Communications*, vol. 106, no. 3, pp. 1161–1182, 2019.

[19] 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.

[20] 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.

[21] M. Cui, D. Han, J. Wang, K.-C. Li, and C.-C. Chang, "ARFV: an efficient shared data auditing scheme supporting revocation for fog-assisted vehicular ad-hoc networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 15815–15827, 2020.

[22] H. Yan, Y. Liu, Z. Zhang, and Q. Wang, "Efficient privacy-preserving certificateless public auditing of data in cloud storage," *Security and Communication Networks*, vol. 2021, Article ID 6639634, 2021.

[23] G. R. Jaya, S. Pasupuleti, and K. Ramesh, "Certificateless privacy preserving public auditing for dynamic shared data with group user revocation in cloud storage," *Journal of Parallel and Distributed Computing*, vol. 156, pp. 163–175, 2021.

[24] X. Yang, M. Wang, X. Wang, G. Chen, and C. Wang, "Stateless cloud auditing scheme for non-manager dynamic group data with privacy preservation," *IEEE Access*, vol. 8, pp. 212888–212903, 2020.

[25] Y. Zhang, C. Xu, X. Lin, and X. S. Shen, "Blockchain-based public integrity verification for cloud storage against procrastinating auditors," *IEEE Transactions on Cloud Computing*, vol. 9, no. 3, pp. 923–937, 2019.

[26] T. Shang, F. Zhang, X. Chen, J. Liu, and X. Lu, "Identity-based dynamic data auditing for big data storage," *IEEE Transactions on Big Data*, vol. 7, no. 6, 2019.

[27] Y. Li, Y. Yu, G. Min, W. Susilo, J. Ni, and K. K. R. Choo, "Fuzzy identity-based data integrity auditing for reliable cloud storage systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 1, pp. 72–83, 2017.

[28] N. Garg and S. Bawa, "ID-PAPC: identity based public auditing protocol for cloud computing," in *Proceedings of the 2017 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE)*, pp. 14–17, IEEE, Dehradun, India, December 2017.

[29] Y. Yang, Y. Sun, Q. Huang, W. Yin, and F. Chen, "RLWE-Based ID-DIA protocols for cloud storage," *IEEE Access*, vol. 7, pp. 55732–55743, 2019.

[30] H. Wang and J. Li, "Private certificate-based remote data integrity checking in public clouds," in *International Computing and Combinatorics Conference*, pp. 575–586, Springer, Salmon Tower Building, NY, USA, 2015.

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

[32] H. Shacham and B. Waters, "Compact proofs of retrievability," *Journal of Cryptology*, vol. 26, no. 3, pp. 442–483, 2013.

[33] J. Zhao, C. Xu, F. Li, and W. Zhang, "Identity-based public verification with privacy-preserving for data storage security in cloud computing," *IEICE-Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E96.A, no. 12, pp. 2709–2716, 2013.

[34] H. Wang, "Identity-based distributed provable data possession in multicloud storage," *IEEE Transactions on Services Computing*, vol. 8, no. 2, pp. 328–340, 2014.

[35] C. Sasikala and C. Shoba Bindu, "Certificateless remote data integrity checking using lattices in cloud storage," *Neural Computing and Applications*, vol. 31, no. 4, pp. 1–7, 2018.

[36] C. Lan, H. Li, and C. Wang, "Cryptanalysis of "Certificateless remote data integrity checking using lattices in cloud storage," in *Proceedings of the 2020 10th International Conference on Information Science and Technology (ICIST)*, pp. 134–138, IEEE, Plymouth, UK, September 2020.

[37] Y. Yu, M. H. Au, Y. Mu et al., "Enhanced privacy of a remote data integrity-checking protocol for secure cloud storage," *International Journal of Information Security*, vol. 14, no. 4, pp. 307–318, 2015.

[38] D. He, N. Kumar, S. Zeadally, and H. Wang, "Certificateless provable data possession scheme for cloud-based smart grid data management systems," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 3, pp. 1232–1241, 2017.

[39] H. Yang, S. Jiang, W. Shen, and Z. Lei, "Certificateless provable group shared data possession with comprehensive privacy preservation for cloud storage," *Future Internet*, vol. 10, no. 6, pp. 49–6, 2018.

[40] S. Peng, F. Zhou, Q. Wang, Z. Xu, and J. Xu, "Identity-based public multi-replica provable data possession," *IEEE Access*, vol. 5, pp. 26990–27001, 2017.

[41] F. Zafar, A. Khan, S. U. R. Malik et al., "A survey of cloud computing data integrity schemes: design challenges, taxonomy and future trends," *Computers & Security*, vol. 65, pp. 29–49, 2017.

[42] K. He, J. Chen, Y. Quan, S. Ji, D. He, and R. Du, "Dynamic group-oriented provable data possession in the cloud," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 3, 2019.

[43] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," in *Proceedings of the Annual International Cryptology Conference*, pp. 213–229, Springer, Berlin, Heidelberg, Germany, August 2001.

[44] A. Fu, S. Yu, Y. Zhang, H. Wang, and C. Huang, "NPP: A new privacy-aware public auditing scheme for cloud data sharing with group users," *IEEE Transactions on Big Data*, vol. 8, no. 1, 2017.

[45] S. S. Al-Riyami, K. G. Paterson, and K. G. Paterson, "Certificateless public key cryptography," in *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security*, pp. 452–473, Springer, Berlin, Heidelberg, Germany, November 2003.

[46] M. Aazam, S. Zeadally, and K. A. Harras, "Deploying fog computing in industrial internet of things and industry 4.0," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4674–4682, 2018.

[47] X. Li, J. Wan, H.-N. Dai, M. Imran, M. Xia, and A. Celesti, "A hybrid computing solution and resource scheduling strategy for edge computing in smart manufacturing," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 4225–4234, 2019.

[48] F. Armknecht, J. M. Bohli, G. Karame, and W. Li, "Outsourcing Proofs of Retrievability," *IEEE Transactions on Cloud Computing*, vol. 9, no. 1, pp. 286–301, 2018.

[49] U. Feige, A. Fiat, and A. Shamir, "Zero-knowledge proofs of identity," *Journal of Cryptology*, vol. 1, no. 2, pp. 77–94, 1988.

[50] J. Ni, K. Zhang, X. Lin, and X. Shen, "Securing fog computing for internet of things applications: challenges and solutions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 601–628, 2017.

[51] R. Rezapour, P. Asghari, H. H. S. Javadi, and S. Ghanbari, "Security in fog computing: a systematic review on issues, challenges and solutions," *Computer Science Review*, vol. 41, no. 2021, Article ID 100421.

[52] B. Chen, T. Xiang, M. Ma, D. He, and X. Liao, "CL-ME: efficient certificateless matchmaking encryption for internet of things," *IEEE Internet of Things Journal*, vol. 18, no. 9, 2021.

[53] J. Shen, T. Zhou, D. He, Y. Zhang, X. Sun, and X. Yang, "Block design-based key agreement for group data sharing in cloud computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 6, pp. 996–1010, 2017.

[54] A. Karati, C. I. Fan, and E. S. Zhuang, "Reliable data sharing by certificateless encryption supporting keyword search against vulnerable KGC in industrial internet of things," *IEEE Transactions on Industrial Informatics*, 2021.

[55] M. H. Ziegler, M. Grossmann, and U. R. Krieger, "Integration of fog computing and blockchain technology using the plasma framework," in *Proceedings of the 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pp. 120–123, Seoul, South Korea, May 2019.

[56] O. Bouachir, M. Aloqaily, L. Tseng, and A. Boukerche, "Blockchain and fog computing for cyberphysical systems: the case of smart industry," *Computer*, vol. 53, no. 9, pp. 36–45, 2020.

[57] K. Lei, M. Du, J. Huang, and T. Jin, "Groupchain: towards a scalable public blockchain in fog computing of IoT services computing," *IEEE Transactions on Services Computing*, vol. 13, no. 2, pp. 252–262, 2020.

[58] S. Misra, A. Mukherjee, A. Roy, N. Saurabh, Y. Rahulamathavan, and M. Rajarajan, "Blockchain at the edge: performance of resource-constrained IoT networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 1, pp. 174–183, 2020.

[59] M. Blum, W. Evans, P. Gemmell, S. Kannan, and M. Naor, "Checking the correctness of memories," *Algorithmica*, vol. 12, no. 2, pp. 225–244, 1994.

[60] A. Juels and B. S. Kaliski, "PORs: proofs of retrievability for large files," *Proceedings of the 14th ACM Conference on Computer and Communications Security*, vol. 2007, pp. 584–597, 2007.

[61] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," *Journal of Cryptology*, vol. 17, no. 4, pp. 297–319, 2004.

[62] J. Li, H. Yan, and Y. Zhang, "Certificateless public integrity checking of group shared data on cloud storage," *IEEE Transactions on Services Computing*, vol. 14, no. 1, pp. 71–81, 2018.

[63] J. R. Gudeme, S. K. Pasupuleti, and R. Kandukuri, "Certificateless multi-replica public integrity auditing scheme for dynamic shared data in cloud storage," *Computers & Security*, vol. 103, Article ID 102176, 2021.

[64] D. He, S. Zeadally, and L. Wu, "Certificateless public auditing scheme for cloud-assisted wireless body area networks," *IEEE Systems Journal*, vol. 12, no. 1, pp. 64–73, 2015.

[65] B. Kang, J. Wang, and D. Shao, "Certificateless public auditing with privacy preserving for cloud-assisted wireless body area networks," *Mobile Information Systems*, vol. 2017, Article ID 2925465, 2017.

[66] L. Zhou, A. Fu, G. Yang, H. Wang, and Y. Zhang, "Efficient certificateless multi-copy integrity auditing scheme supporting data dynamics," *IEEE Transactions on Dependable and Secure Computing*, 2020.

[67] M. Bellare and G. Neven, "Multi-signatures in the plain public-key model and a general forking lemma," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, pp. 390–399, Alexandria, VA, USA, November 2006.

[68] S. D. Galbraith, K. Harrison, and D. Soldera, "Implementing the tate pairing," in *Proceedings of the International Algorithmic Number Theory Symposium*, pp. 324–337, Sydney, Australia, July 2002.

[69] A. Joux and K. Nguyen, "Separating decision diffie-hellman from computational diffie-hellman in cryptographic groups," *Journal of Cryptology*, vol. 16, no. 4, pp. 239–247, 2003.

[70] M. Bellare and P. Rogaway, "Random oracles are practical: a paradigm for designing efficient protocols," in *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pp. 62–73, VA, USA, November 1993.

[71] B. Liu, Y. Zhang, G. Zhang, and P. Zheng, "Edge-cloud orchestration driven industrial smart product-service systems solution design based on CPS and IIoT," *Advanced Engineering Informatics*, vol. 42, Article ID 100984, 2019.

[72] J. Kang, Y. Rong, X. Huang, and Y. Zhang, "Privacy-preserved pseudonym scheme for fog computing supported internet of vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 8, pp. 2627–2637, 2017.

[73] A. Bounceur, M. Bezoui, R. Euler, N. Kadjouh, and F. Lalem, "Brogo: a new low energy consumption algorithm for leader election in wsns," in *Proceedings of the 10Th International Conference on Developments in Ecosystems Engineering (DESE)*, pp. 218–223, IEEE, Paris, France, June 2017.

[74] L. C. Guillou and J. J. Quisquater, "A "paradoxical" indentity-based signature scheme resulting from zero-knowledge," in *Proceedings of the Conference on the Theory and Application of Cryptography*, pp. 216–231, Springer, NY, USA, August 1988.

[75] C. Wang, S. M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for secure cloud storage," *IEEE Transactions on Computers*, vol. 62, no. 2, pp. 362–375, 2011.

[76] B. Lynn, "Crypto.stanford.edu. n.d. PBC Library - Pairing-Based Cryptography," 2013, https://crypto.stanford.edu/pbc/download.html.

[77] W. Liu, J. Liu, Q. Wu, and Bo Qin, "Android PBC: a pairing based cryptography toolkit for android platform," in *Proceedings of the 2014 Communications Security Conference (CSC 2014)*, p. 14, Beijing, China, May 2014.