IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 5 March 2023, accepted 23 April 2023, date of publication 28 April 2023, date of current version 4 May 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3271605

RESEARCH ARTICLE

A Distributed and Decentralized Certificateless Framework for Reliable Shared Data Auditing for FOG-CPS Networks

MANOHAR SAI BURRA, (Member, IEEE), AND SOUMYADEV MAITY, (Member, IEEE)
Department of Information Technology, Indian Institute of Information Technology Allahabad, Prayagraj 211015, India

Corresponding author: Manohar Sai Burra (rsi2018504@iiita.ac.in)

This work was supported in part by the IHUB NTIHAC Foundation, Indian Institute of Technology (IIT) Kanpur, under Grant IHUB-NTIHAC/2021/01/3; and in part by the Ministry of Education, Government of India.

ABSTRACT FOG computing enhances communication efficiency by processing data at the network's edge. In many critical infrastructure (CI) networks, user entities outsource data to the FOG server, but the reliability of the data is a concern. To avoid incorrect decision-making due to corrupt data, a data auditing process is mandatory. Further, a group of user entities in a CI network generate data onto a common file and outsource the file to the FOG server. Auditing such shared group data creates additional complexities. Existing shared group data auditing schemes using public key infrastructure (PKI) or identity-based cryptography (IBC) face challenges such as certificate management or key escrow problems respectively. While, certificateless cryptography (CLC) offers better security and efficiency, but many CLC-based shared group data auditing schemes lack protection against key generation center compromise. The key generation process in many of the existing shared group data auditing schemes is centralized and does not support flexible user entity admission. This paper proposes a distributed and decentralized cryptography approach to enhance the security of key generation for group members. The scheme offers flexible user admission, allowing for full or semi-membership. The proposed scheme provides efficient and secure shared group data auditing against public key replacement, metadata forgery, and integrity attacks, as shown by performance evaluation and security analysis.

INDEX TERMS Certificateless, data auditing, provable data possession, shared data.

I. INTRODUCTION

The fog computing model introduced by cisco [1] is a paradigm shift from traditional cloud computing for user data management. Both the cloud and fog models provide data storage and processing functionalities to their users. One of the main differences is that the fog service providers place the fog servers close to a user, whereas the cloud server is far away at some remote location. Hence, fog computing improves communication bandwidth and latency. Moreover, many critical infrastructure networks such as vehicular-ad-hoc networks (VANETs), smart-grid networks, wireless sensor networks (WSN), mobile ad-hoc networks (MANETs), etc. frequently require reliable data access and faster data processing; therefore, fog computing is a better option than cloud computing for such networks.

In cloud computing, the user saves the data file to the cloud server and then removes the local copy of the file. While the user trusts and assumes the confidentiality of data files at the cloud server, there is still concern about remote data file integrity. Numerous schemes in the literature [2], [3], [4], [5], [7], [9], [10], [11], [12], [14], [15], [16], [17], [18], [19], [23], [24], [25], [26], [28], [29], [30], [31], [32], [40], [41], [42] provide remote data integrity checking (RDIC) mechanisms that allow the user to verify the integrity of the remote data file. An RDIC scheme contains two main components: The first is called the preprocessing component, and the second is called the auditing component. The user generates metadata for each data block associated with the data file and uploads the metadata, data file, and some public information with

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

the cloud server in the preprocessing component [2], [3]. While in the auditing component, a third-party auditor (TPA) or the user issues a challenge to the cloud server, which subsequently carries out the computation to generate a proof and forwards the proof of data file integrity to the TPA. The TPA verifies whether the received proof of data file integrity is valid and is as per the challenge [2], [3].

There are several RDIC schemes for single-user settings [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [14], [16], [17], where a single cloud user owns a data file and performs the auditing task accordingly. In contrast, some critical infrastructure networks require multiple users to form a group, contribute, and compute on a commonly shared data file. To instantiate an RDIC for shared group data will create newer challenges because different group members calculate the metadata for the individual blocks of the shared data file. The proof generation mechanism by the cloud server and proof verification mechanism by either the group member or the TPA must be able to incorporate the aggregation of mixed information as generated by different members of the group.

There are many RDIC techniques [23], [24], [25], [28], [30], [31], [40] for shared group data in cloud computing scenarios. A few of these schemes employ the public key infrastructure (PKI) [13], [25], where users select their private-public key pair, and the public key is securely transmitted to a trusted certificate authority (CA). The CA computes a public key certificate that binds users to their public key. These certificates help prove a user's public key is legitimate to other network users. Though the PKI seems efficient for Internet protocols, the certificate generation, revocation, maintenance, and verification cause considerable communication and computation overhead. Hence, the PKI is not advisable for resource-constraint networks.

Further, a centralized certificate authority is always a cause for concern, considering the chances of a CA being compromised at some point. Identity-based cryptography (IBC) [37] is an alternative to the PKI mechanism that minimizes the overall communication and computation overhead to bind a user to its public key. Under the IBC, users select their identity and communicate it to a trusted private key generator (PKG). The PKG uses its master secret key to generate a private key for the user's chosen identity [37]. In the IBC system, a user's public key is implicitly tied to their chosen identity. Therefore, any two users of the IBC system can derive each other's public key using only their identity and some public information. However, a major disadvantage of the IBC is the key escrow problem, where the compromise of the PKG would compromise the private keys of all users of the IBC system [39]. Hence, IBC is not a suitable approach while designing an RDIC scheme since the data might be of critical importance, and failure to provide confidentiality and integrity services will have a greater impact on the users of the network and their services. Certificateless cryptography (CLC) [39] is a more secure approach for eliminating the need for certificate generation and maintenance in PKI and addressing the key escrow issue in IBC. In CLC, a key generation center (KGC) binds a user's public key and its identity but only gives the user with a partial private key. The user generates their secret-public key pair, and their private key is a combination of their secret and partial private keys, known only to the user. Hence, CLC is a better alternative to design RDIC schemes for shared group data. But, it was observed that some of the CLC-based shared group data auditing schemes are vulnerable to metadata forgery, public key replacement attacks, and collusion of data auditor and cloud, as discussed in section I-B.

Existing CLC-based RDIC schemes for shared group data auditing in fog computing environments consider the key generation center (KGC) as a curious but passive entity that sets up the network. However, it was observed in [42] that the KGC could perform metadata forgery. This potential issue can be addressed by restricting the KGC from learning information about the data blocks and their corresponding metadata. In contrast, this assumption fails when the KGC is a group manager or a member that can easily access the data blocks and their corresponding metadata. Hence, it is not advisable to give the role of the KGC to a single entity. Further, it can be noted that all the group members contribute to the shared group data, and authorizing only a single entity has a high probability of compromise.

A. MOTIVATION AND CONTRIBUTION

The emerging distributed and decentralized networks, such as wireless sensor networks (WSN), MANETs, VANETs, smart-grid networks, etc., cannot assume trust in a single entity such as the key generation center (KGC). Hence, trust and power must be distributed to the group members. Moreover, the group members must be able to distinguish and decide a new user admission into a group to become a full member (fully trusted) or semi-member (semi-trusted). Considering all the above-stated reasons, we are motivated to design an efficient and secure certificateless cryptography-based decentralized and distributed network setup for shared group data auditing without the need for a trusted central authority and that can mitigate security issues arising from a compromise of KGC as possible in the existing works [28], [30], [40] and enable secure flexible new user admission.

The contributions of this paper are as follows:

1) Our scheme provides secure new user entity admission for shared group data auditing. The new entity can become a full member with all the capabilities or a semi-member with limited capabilities.
2) A user entity can request and compute a private-public key pair associated with the group secret. The private-public key pair will be useful to generate metadata (digital signatures), pairwise symmetric keys, and decrypt messages. The scheme is secure against metadata forgery, and public key replacement attacks. It also prevents the precomputation of integrity proofs. In contrast, a lot of existing schemes suffer from the same.

3) The scheme utilizes the FOG computing capabilities to establish secure broadcast group key agreement and communication among the edge device and the group members.
4) The scheme supports aggregate auditing of shared group data by the data auditor. At the same time, it can also perform aggregate verification of the auditing proofs of the data auditor. Hence, the metadata generation mechanism's aggregate property improves the auditing task's efficiency.

The rest of the paper is organized as follows: Section I-B provides a comprehensive review of the related literature. In Section II, we explore the necessary background information, discussing the cryptographic concepts that have been used to implement the proposed protocol. The system model, including the network architecture and data storage types, is described in Section III. In Section IV, we discuss the security model. Section V presents a detailed explanation of the proposed protocol, including the specific steps and algorithms used in each phase. The security and performance of the proposed protocol are evaluated in Sections VI and VII, respectively. These sections provide a thorough analysis of the proposed protocol's strengths, and weaknesses, and suggest potential areas for future research.

B. RELATED WORK

Public key cryptography techniques are highly employed in designing and developing public data auditing schemes [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [14], [16], [17], [23], [24], [25], [28], [30], [31], [40]. The metadata generation, a core component of a data auditing scheme, employs user private keys. The user uploads the data and metadata onto the cloud. Later, the data auditor requests proof of remote data possession to determine the data integrity using some public information. Ateniese et al. [2] proposed the first probabilistic public data auditing scheme called the provable data possession (PDP), where the data auditor challenges the cloud server to provide the PDP proof of the remote file. The challenge contains randomly chosen indices of the data blocks of the remote file for which the cloud must provide integrity proof. The cloud computes and forwards the PDP proof, and then the data auditor verifies the received proof using some public information and confirms the integrity of the remote data. Sacham et al. [3] scheme based on knowledge of exponent assumption proposed the proofs of retrievability (PoR), where the user can successfully retrieve the original data blocks from the challenge response interactions of the integrity proofs generated by the cloud. Many existing public data auditing schemes are based on the PoR and PDP mechanisms.

Several RDIC schemes [2], [3], [4], [5] are based on public key infrastructure requiring certificate generation, management, and revocation. There is high computational and communication overhead in verifying the certificates. Hence, PKI-based data auditing schemes are not useful for users with resource constraints, such as in the case of CPS devices of a FOG computing architecture. An alternative to PKI-based data auditing schemes is using ID-Based Cryptography (IBC). Here, a PKG creates the user's private key using the user's chosen identity. In IBC, a user's public key binds with the user's chosen identity. Hence, any user of the IBC system can derive the public key of the other user of the system using only the identity and therefore does not need to verify the public keys or generate any certificates [6].

Zhao et al. [7] proposed an ID-based aggregate signature scheme to generate metadata and allow batch verification. Wang et. al [4] scheme considers RDIC verification for a multi-cloud scenario using IBC. The scheme in [9] utilizes IBC to provide an efficient metadata generation mechanism but was proven to be insecure in [10]. Recently, Liu et al. [11] proposed an RDIC scheme using a sanitizable ID-based signature which provides data protection against untrusted third-party auditors. Though the schemes in [4], [7], [9], [10], [11], and [12] have better performance than that of the PKI-based counterparts. But, the inherent key escrow problem in the IBC can enable strong security weaknesses in these data auditing protocols. Hence are not suitable for data auditing environments that require the confidentiality of the security keys of the users.

Certificateless cryptography (CLC) [39] on the other hand, can be the best alternative to both PKI and IBC. It eliminates both the key escrow problem inherent in IBC and certificates which are mandatory in PKI. In a CLC system [39], a private key of a user is composed of both a secret key selected by the user and a partial private key created by the Key Generation Center (KGC). Applying CLC to the data auditing protocols may seem to provide numerous benefits regarding computational efficiency and private key security. Wang et al. [14] proposed the first CLC-based RDIC scheme, whose metadata generation mechanism is a homomorphic verifiable certificateless signature. The scheme in [15], while highlighting the need for CLC-based RDIC schemes, proves [14] is insecure and proposes an improvement to [14] for application in wireless body area networks (WBANs). Kim and Jeong [16] CLC-based RDIC provides mechanisms to verify the integrity in constant time efficiently. He et al. [17] scheme claims to protect data from an untrusted third-party auditor but was proven to suffer from metadata forgery attacks as shown in [18]. Zhao et al. [19] scheme provides certificateless data auditing with a designated verifier, while such designation may not be suitable for public data auditing schemes. The schemes in [14], [15], [16], [17], [18], [19], and [29] employ certificateless cryptography technique for remote data auditing for single-user settings where a single user owns the file and generates the metadata. Whereas, in the context of fog computing architecture, a group of CPS devices dedicated to performing data collection for specific activity stores these data into a common shared file.

Remote data auditing of such shared group data brings difficulties because different group members compute the metadata for different data file blocks with their private keys.

The cloud generates integrity proof, and the data auditor performs proof verification to ensure data integrity. Both integrity-proof generation and verification must also efficiently aggregate information of different group users. The schemes in [23], [24], [25], [26], [28], [30], [31], [32], [40], and [41] support shared group data auditing. While we primarily focus on CLC-based shared group data auditing schemes considering the benefits of CLC over PKI and IBC. The CLC-based shared group data auditing schemes in [28], [30], [31], [32], and [40] are for conventional cloud computing scenarios. Hence, these schemes cannot be applied directly to shared group data auditing in fog computing scenarios, considering the distinctive properties of fog networks as mentioned below.

The computational and storage resources are closer to the user entities (CPS devices) in a fog computing architecture enabling faster communication and increased bandwidth. Hence a data auditor within the premises and part of the fog network will improve the efficiency of data auditing. Therefore, we are motivated to implement the auditing task by a group member (CPS device). But, the trustworthiness of the auditing task is a concern in two practical scenarios. When the data auditor is selfish, and when the edge device and data auditor collude. In the first case, when the data auditor is selfish and tries to save its computational power by not performing auditing duties as per the service level agreement, it can procrastinate or skip its duties without getting detected. In the second case, the data auditor reveals the challenge information to the edge device ahead of time. The edge device can precompute the data integrity proofs beforehand and forward them when challenged for proof of data possession. The work in [27] shed light on the adverse consequences of collusive behavior between the data auditor and the cloud. Most of the previous data auditing schemes were found to be vulnerable to such attacks. However, the scheme presented in [27] has limitations, as it cannot be easily adapted for shared group data auditing scenarios and does not incorporate certificateless cryptography or data privacy mechanisms against third-party auditors. It is also worth mentioning that the data auditor in fog computing environments is a CPS device, which is more susceptible to compromise, and thus the threat of collusion is of serious concern.

A robust data auditing protocol should be capable of preventing collusion between the edge device and the data auditor on the challenge information. The work presented in [42] is the first to tackle various adversarial scenarios specific to shared group data auditing in FOG-CPS networks under a certificateless security model. It was revealed that the CLC-based group data auditing protocols [28], [30] were vulnerable to metadata forgery and public key replacement attacks, as demonstrated in [42]. The schemes proposed in [40] and [41] also adopt a similar metadata generation mechanism as [28], and are therefore vulnerable to these attacks. Additionally, none of the schemes presented in [28], [30], [31], [32], and [40] provide mechanisms to prevent the collusive behavior between the data auditor and the edge device.

TABLE 1. List of acronyms.

Acronym	Abbreviation
PKI	Public Key Infrastructure
IBC	Identity-Based Cryptography
CLC	Certificateless Cryptography
PBC	Pairing-Based Cryptography
TPA	Third Party Auditor
CA	Certification Authority
RDIC	Remote Data Integrity Checking
PKG	Private Key Generator
KGC	Key Generation Center
PDP	Provable Data Possession
ECDHP	Elliptic Curve Diffie-Hellman Problem
ECDLP	Elliptic Curve Discrete Log Problem
ED	Edge Device
CPS	Cyber-Physical System
SLA	Service-Level Agreement
ROM	Random Oracle Model
chal	Challenge Set
A_1	Attacker Type-1
A_2	Attacker Type-2
A_3	Attacker Type-3
MAC_{key}	Keyed Message Authentication Code

The issue of KGC metadata forgery was addressed in [42] by restricting the KGC's access to information about the data blocks and their corresponding metadata. However, this assumption fails when the KGC is a group manager or a member, who would have easy access to the data blocks and their metadata. Hence, it is not recommended to assign the role of KGC to a single entity.

A detailed analysis of the existing shared group data auditing protocols revealed that the current approaches are insufficient for situations that require decentralization and distribution of trust and power among the group members. Moreover, the possibility of admitting new users into the group as either full or semi-members has not been fully explored. To the best of our understanding, the crucial aspects of decentralization, key distribution, and secure flexible user admission for a certificateless cryptography-based shared group data auditing remain unexplored.

II. PRELIMINARIES

This section briefly discusses the cryptographic concepts which have been used to implement the proposed protocol. The Table 1 provides list of acronyms and the Table 2 provides important notations and definitions used throughout the paper.

A. THRESHOLD SECRET SHARING SCHEME

In general, a (k, d) threshold secret sharing scheme [33] consists of a dealer who possess a secret s, and there is a group $\mathbf{E}$ with d entities where $\mathbf{E} = \{E_1, E_2, E_3, \ldots, E_d\}$. The dealer distributes the share s_i of the secret s to each entity E_i of the group using a secret univariate polynomial $f(x)$. The degree of the polynomial is selected as per the threshold

TABLE 2. Important notations and definitions.

Notation	Definition
G	Cyclic Additive Group
G_1	Cyclic Multiplicative Group
e	$G \times G \rightarrow G_1$
h	$\{0,1\}^* \rightarrow Z_q^*$
q, p	large primes
H, H_2	$\{0,1\}^* \rightarrow G$
s	secret key of group
f_α	$\{0,1\}^* \times \{0,1\}^l \rightarrow \{0,1\}^l$
key_u	shared symmetric between the user entity (E_u) and edge device
MAC_{key_u}	$\{0,1\}^* \times \{0,1\}^l \rightarrow Z_q^*$
π	$\{0,1\}^\lambda \times \{Z_{n+1} - \{0\}\} \rightarrow \{Z_{n+1} - \{0\}\}$
f	$\{0,1\}^\lambda \times \{Z_{n+1} - \{0\}\} \rightarrow Z_q^*$
pri_j	secret share of user entity E_j corresponding to group secret key (s)
P_{pub}	public key of the group
d_{E_j}	partial private key of user entity E_j
η_j	secret value of user entity E_j
$< d_{E_j}, \eta_j >$	private key of user entity E_j
P_{E_j}	public key of user entity E_j
F_{ID}	unique identifier value of a file
m_i	data block at index i of file F_{ID}
v_i	authenticating metadata of a data block m_i
σ_i	metadata for data block m_i
T_i	encrypted information of metadata σ_i

requirements i.e. let k be the threshold where at least k shares of the secret s are needed to reconstruct the secret s then the degree of the univariate polynomial will be $k-1$. It is proven that shares less than k will not be able to reconstruct the secret s.

A (k, d) threshold secret-sharing scheme employs polynomial interpolation. The dealer chooses a univariate polynomial $f(x) = s + \sum_{i=1}^{k-1} c_i \cdot x^i$, $f(0) = s$ and each of the c_i value is chosen uniformly random from Z_q^*, where q is a large prime integer. Let E_i have an identity ID_i which is mapped to a public value $b_i \in Z_q^*$. The dealer calculates the share of E_i as $f(b_i)$ and forwards it secretly to the entity E_i. To reconstruct the secret s, at least k entities must cooperate as follows: $s = \sum_{i=1}^{k} s_i \cdot l_i$; where $l_i = \prod_{j \neq i} \frac{b_j}{b_i - b_j}$ are called the Lagrange coefficients.

There are various versions of the Shamir secret-sharing scheme [33]. However, for our proposed protocol, we have chosen to use the bivariate polynomial version. This variation was inspired by the works of Blundo et al. [34] and Saxena et al. [35]. It is particularly useful for our needs as it supports distributed and decentralized system initialization. Additionally, it allows for threshold-based member role designation and authorization. This flexibility enables the admission of a user to the group either as a full member or a semi-member which is determined by the possession of both a secret share and private key or only the possession of a private key respectively, as described in section VI of the paper.

B. BILINEAR MAP

Consider a cyclic additive group G, and a multiplicative cyclic group G_1, for $|G| = |G_1| = p$, such that p is a large prime integer. A bilinear map [36] takes two elements from an algebraic structure (G) and produces an element of G_1; $\hat{e} : G \times G \rightarrow G_1$. It has the following properties:

Bilinearity: $\forall a, b \in Z_q^*, \forall A, B \in G$, $e(a \cdot A, b \cdot B) = e(A, B)^{ab} = e(b \cdot A, a \cdot B)$

Non-Degeneracy: For all $x, y \in G$, then there exists some $x', y' \in G$ such that $e(x, y) = e(x', y')$. This property states that the map e is non-degenerate, meaning that it is not the case that $e(x, y) = 1_{G_1}$ for all x and y.

Computable: A bilinear map is considered computable if an algorithm can compute the map in polynomial time.

The two well-known implementations of the bilinear maps are tate [37], and weil [38] pairing.

C. NEGLIGIBLE FUNCTION

In cryptography, a function $\phi(l) : \mathbb{N} \rightarrow \mathbb{R}$ is considered negligible if $\forall c > 0, \exists l_0 \in \mathbb{N}$ such that $\forall l > l_0, |\phi(l)| < \frac{1}{l^c}$

It states that for any positive constant c, there exists some integer l_0 such that for all $l > l_0$, the absolute value of $\phi(l)$ is less than $\frac{1}{l^c}$. In other words, as l becomes very large, the value of $\phi(l)$ becomes extremely small, approaching zero.

A scheme is "provably secure" when the security failure probability of the scheme is a negligible function of the security parameter. Generally, the length of the cryptography key (security parameter (l)) is the input to the negligible function.

D. COMPUTATIONAL ELLIPTIC CURVE DIFFIE-HELLMAN (ECDH) PROBLEM

Let G be an additive group of order p with P as the generator. An instance of computational ECDH problem in G is shown below:

Given $< p, G, P, A, B >$ for some $A, B \in G$, where $A = s \cdot P$, and $s \in Z_q^*$, calculate $s \cdot B$.

The decisional version of the ECDH problem is as follows:

Given $< p, G, P, A, B, C >$ for some A, B, and $C \in G$, where $A = \alpha \cdot P$, $B = \beta \cdot P$, and $C = \gamma \cdot P$, for $\alpha, \beta, \gamma \in Z_q^*$. Decide if $\alpha\beta \cdot P = \gamma \cdot P$.

The Elliptic Curve Decisional Diffie-Hellman (ECDDH) problem is easy to solve in a group G if there is a bilinear map present in G. However, to this day, there is no known efficient method for computing sB within polynomial time in the group G, making the computational Elliptic Curve Diffie-Hellman (ECDH) problem considered difficult to solve within G.

III. SYSTEM MODEL

A. THE NETWORK ARCHITECTURE

There are four major types of participating entities in the proposed protocol as described below:

Founding Members: These entities are responsible for setting up the distributed network. They run the setup

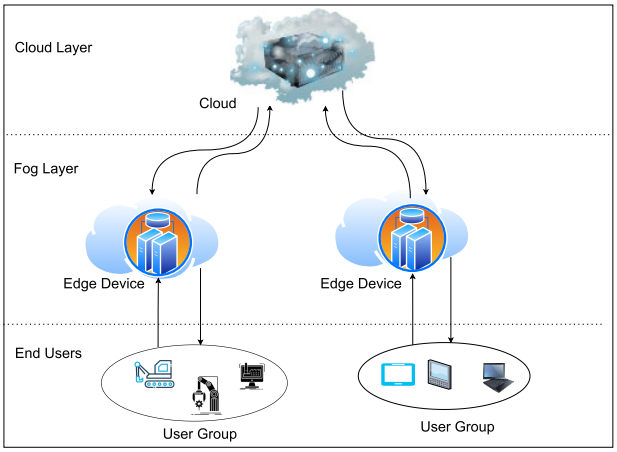


FIGURE 1. System model.

algorithm in our proposed protocol and publish the global parameters of the network. Each of the founding members is assumed to be honest.

The Edge Device (ED): In fog computing architecture, the edge device is a key component that provides storage and computing services to the end-users, also known as Cyber-Physical Systems (CPS) devices. The edge device is located closer to the source, which reduces latency and bandwidth issues associated with sending data to a centralized cloud.

The User Group: In our proposed protocol, a user entity is a CPS device. A group of users contribute data to a commonly shared file and are known as the user group. These user entities generate metadata for their data blocks of the shared file. Both data and metadata are stored on the edge device. Whenever a user entity joins the network, it also establishes a shared symmetric key with the edge device for secure data transmission.

The Data Auditor: It is responsible for auditing the PDP proofs computed by the edge device. In our proposed protocol, the role of a data auditor is designated to the CPS devices. The load of the auditing task is shared among the group members. Each of the CPS devices performs the auditing task on a turn basis. The details of the schedule of which device performs when and audit frequencies are written into an SLA document. A generic system model is presented in the fig. 1.

B. DATA STORAGE TYPES

As a part of the proposed protocol, there are multiple data storage types generated and maintained by different types of entities of the network as described below:

Data File: A set of user entities, represented by the group $\mathbb{E}$, share a common data file containing a collection of data blocks. Each of the data blocks may have been contributed by a member of the group.

Metadata File: For every data block of the shared data file there is a metadata block generated by the corresponding user entity of the group. The edge device maintains a metadata file that contains the metadata blocks corresponding to the shared data file.

Indicator Matrix: For every shared data file F that consists of n data blocks, the edge device keeps a matrix I. This matrix is a two-dimensional array of size $n \times d$, where the row index i is associated with a data block ($1 \leq i \leq n$) and the column index u is associated with a member of the group ($1 \leq u \leq d$). The notation $I_{i,u}$ is used to denote $I[i][u]$ as defined below:-

$$I_{i,u} = \begin{cases} 1 & \text{if entity } E_u \text{ is the generator of block } i \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Audit-Log: Whenever a data auditor performs an auditing task it stores and maintains the auditing results into a Audit-Log file in the following way:

The fields of a record into the above file are: $< t, t', F_{ID}, Result, SignedProof >$ for a data file with identity F_{ID}, the t and t' denote the time instances when the auditing task was initiated and at what time instance the auditing record has been updated onto a decentralized immutable ledger (such as blockchain) [21], [22] respectively. The *Result* is a 1-bit field that states if the edge device has passed the data integrity test or not. The last field *SignedProof* contains the integrity proof and the digital signature on the integrity proof by the edge device.

The SLA Document: A service level agreement (SLA) document contains important information such as who does the auditing duties, when, and their frequency. The SLA document also contains details that identify the members of the group and their privileges. The SLA is signed by the authorized members of the group before it is executed between the user group and the edge device. It is assumed that the edge device maintains the signed SLA document, and is made available only to the members of the user group.

IV. SECURITY MODEL

In order for an RDIC scheme to be secure, both the metadata generation and proof of data possession mechanisms must be unforgeable. The security of the metadata generation, which is achieved through a certificateless signature mechanism in our proposed protocol, must be proven in the certificateless cryptography (CLC) security model. There are three types of attackers in this model: Type-1, which refers to internal or external attackers other than the KGC; Type-2, which refers to a compromised but passive KGC; and Type-3, which refers to the edge device. Therefore, the proposed scheme must be analyzed in the random oracle model (ROM) against all three types of attackers in order to ensure its overall security.

The security analysis in the ROM setup is a game between an attacker and a challenger. An attacker is modeled in the game with all the real-world capabilities and is allowed to make respective queries to the challenger. Each of the below games describes the interactions of the respective attackers with the challenger.

A. TYPE-I ATTACKER ($\mathbb{A}_1$):

The attacker, referred to as $\mathbb{A}_1$, can be either an internal or external entity such as a user of the system or even the edge device, with the exception of the key generation center (KGC). The goal of $\mathbb{A}_1$ is to create metadata, which is a signature used to authenticate a data block, that is valid under a chosen user identity (E_u) for the data block (m), using authenticating information (v). The power of the attacker $\mathbb{A}_1$ is determined through a series of queries made to a challenger, $\mathbb{B}$, who holds the master private key (s) of the system and generates the public parameters (Pub_p). The queries include requests for hash functions, partial private keys, secret values, public keys, updates to public keys, and metadata.

The Game-1: The Game-1 is an interaction between the attacker $\mathbb{A}_1$ and a challenger $\mathbb{B}$. $\mathbb{A}_1$ can ask different types of queries and these queries represent the power of the attacker $\mathbb{A}_1$. The challenger provides a response for each of the query from $\mathbb{A}_1$. The Game-1 is as follows:

Setup: The challenger, referred to as $\mathbb{B}$, will first run the setup algorithm. This generates the public parameters (Pub_p) and the master private key (s) of the system. $\mathbb{B}$ will keep the master private key secret and send the public parameters to the attacker $\mathbb{A}_1$. This establishes the necessary information for $\mathbb{A}_1$ to make queries and for $\mathbb{B}$ to provide responses as part of the Game-1 process as shown below:

Queries: $\mathbb{A}_1$ makes the following types of queries to the challenger $\mathbb{B}$.

1) **Hash Query:** The hash functions which are part of the meta data generation are modelled by the challenger and $\mathbb{A}_1$ can query for any of these hash functions. $\mathbb{B}$ must return a valid hash response to $\mathbb{A}_1$.
2) **Partial.Private.Extract Query:** $\mathbb{A}_1$ can request a partial private key by submitting a user identity (E_u). $\mathbb{B}$ will then run the Partial-Private-Key-Extract algorithm of the proposed scheme and sends the partial private key to $\mathbb{A}_1$.
3) **Secret.Value Query:** $\mathbb{A}_1$ can request the secret value (β_u) by submitting a user identity (E_u). $\mathbb{B}$ will then run the Secret-Value-Generation algorithm of the proposed scheme and sends β_u to $\mathbb{A}_1$.
4) **Public.Key Query:** $\mathbb{A}1$ can request the public key (P_{E_u}) by submitting a user identity (E_u). $\mathbb{B}$ will then run the Public-Key-Generation algorithm of the proposed scheme and sends P_{E_u} to $\mathbb{A}_1$.
5) **Public.Key.Replacement Query:** $\mathbb{A}1$ can request an update to the public key for a user (E_u) by submitting the user identity and the new public key (P'_{E_u}). $\mathbb{B}$ will then update the public key for E_u to P'_{E_u}.
6) **Metadata Query:** $\mathbb{A}_1$ can request metadata by submitting a user identity (E_u), a data block (m), and authenticating information (v). $\mathbb{B}$ will then run the Metadata-Generation algorithm of the proposed scheme and sends the metadata (σ) to $\mathbb{A}_1$.

Metadata.Forgery: $\mathbb{A}_1$ submits the values (E'_u, m', v', σ') as metadata forgery proof. Here, E'_u is the user identity, m' is the data block, v' is the authenticating information of the data block m', and σ' is the corresponding metadata.

For the attacker, $\mathbb{A}_1$, to be considered successful in the Game-1 process, the following conditions must be met:

1) The metadata submitted by $\mathbb{A}_1$, referred to as σ', must be a valid metadata corresponding to the target user identity (E'_u), data block (m'), and authenticating information (v') as determined by the Metadata-Generation algorithm of the proposed scheme.
2) $\mathbb{A}_1$ must not have previously requested the secret value for the user identity (E'_u).
3) $\mathbb{A}_1$ must not have simultaneously requested both a public key replacement and partial private key for the user identity (E'_u).
4) $\mathbb{A}_1$ must not have previously requested metadata for the values (E'_u, m', v') at any point in the interaction.

B. TYPE-II ATTACKER ($\mathbb{A}_2$):

The attacker $\mathbb{A}_2$ is a compromised key generation center that is passive in nature. The goal of $\mathbb{A}_2$ is to create a valid metadata for a chosen user identity (E_u) that corresponds to a data block (m) and its authenticating information (v). The capabilities of $\mathbb{A}_2$ are outlined in the following Game-2 process.

The Game-2: The Game-2 is an interaction between the attacker $\mathbb{A}_2$ and a challenger $\mathbb{B}$. $\mathbb{A}_2$ can ask different types of queries and these queries represent the power of the attacker $\mathbb{A}_2$. The challenger provides a response for each of the query from $\mathbb{A}_2$. The Game-2 is as follows:

Setup: The challenger, $\mathbb{B}$, generates the public parameters (Pub_p) and the master private key (s) using the setup algorithm. $\mathbb{B}$ then sends both the master private key (s) and the public parameters (Pub_p) to $\mathbb{A}_2$.

Queries: The attacker $\mathbb{A}_2$ can make requests for hash values, secret value and public keys, which are identical to the queries provided in the first game.

The Partial.Private.Key Query is not required to be included in the Game-2 since $\mathbb{B}$ has already forwarded master private key s of the system to $\mathbb{A}_2$. Hence, $\mathbb{A}_2$ can use s to generate the partial private key of any user identity E_u.

Metadata.Forgery: $\mathbb{A}_2$ submits the values (E'_u, m', v', σ') as metadata forgery proof. Here, E'_u is the user identity, m' is the data block, v' is the authenticating information of the data block m', and σ' is the corresponding metadata.

A winning condition for the attacker $\mathbb{A}_2$ in the proposed scheme is established when the following criteria are met:

1) σ' must be legitimate metadata produced by the Metadata-Generation algorithm, corresponding to the target user identity E'_u for a data block m' with authentication information v'.
2) $\mathbb{A}_2$ must not have gained access to the secret value associated with user identity E'_u as part of the secret value query.
3) $\mathbb{A}_2$ must not have queried for metadata on the values (E'_u, m', v') at any phase of the interaction.

C. TYPE-III ATTACKER ($\mathbb{A}_3$):

The attacker $\mathbb{A}_3$ is a compromised edge device. The goal of $\mathbb{A}_3$ is to produce a valid proof of possession without holding at least one original data block of the corresponding challenge. The power of the attacker $\mathbb{A}_3$ is defined in the following Game-3.

Game-3: The Game-3 is an interaction between the attacker $\mathbb{A}_3$ and a challenger $\mathbb{B}$. $\mathbb{A}_3$ can ask different types of queries and these queries represent the power of the $\mathbb{A}_3$ in the real world. The challenger provides a response for each of the query from $\mathbb{A}_3$. The Game-3 is as follows:

Setup: The challenger $\mathbb{B}$ executes the setup algorithm, and generates the public parameters Pub_p and the master private key s of the system. $\mathbb{B}$ sends Pub_p to $\mathbb{A}_3$. Further, $\mathbb{B}$ maintains a data file with identity F_{ID}, the data file F_{ID} contains n number of data blocks where m_i denotes the data block at index i of the file F_{ID}.

Queries: $\mathbb{A}_3$ makes the following types of queries to the challenger $\mathbb{B}$.

1) **Data.Block Query:** The attacker $\mathbb{A}3$ can request information about a specific data block by providing the index i of the file F_{ID}. Upon receiving this request, the challenger $\mathbb{B}$ will locate and retrieve the corresponding data block m_i, and sends it to $\mathbb{A}_3$
2) **Metadata Query:** $\mathbb{A}3$ can also request metadata for a specific data block by again providing the index i of the file F_{ID}. In this case, $\mathbb{B}$ will run the Metadata-Generation algorithm of the proposed scheme and send the resulting metadata σ to $\mathbb{A}_3$. The attacker can also make requests for hash values and public keys, which are identical to the corresponding queries in the first game.

Challenge: $\mathbb{B}$ calculates the challenge vector values *chal* for the file F_{ID} as a part of the proof of possession mechanism using the Challenge algorithm of the proposed scheme. $\mathbb{B}$ sends the *chal* to $\mathbb{A}_3$

Proof.Forgery: $\mathbb{A}_3$ submits the proof of possession $proof(t)$ for the corresponding challenge vector values *chal*.

The attacker $\mathbb{A}_3$ is said to win the above game if the following conditions are satisfied:

1) The data integrity proof $proof(t)$ submitted by $\mathbb{A}_3$ must be valid according to the Proof-of-Possession algorithm of the proposed scheme.
2) $\mathbb{A}_3$ must not have queried for at least one of the data block part of the challenge vector values in the Data.Block Query.

V. THE PROPOSED PROTOCOL

A. OVERVIEW OF THE PROPOSED PROTOCOL

The proposed protocol consists of five phases, each of which plays a critical role in ensuring the security and efficiency of the system. The initial phase of the protocol is the system initialization and member authorization phase. During this phase, the founding members initialize the network and obtain their share of the group's secret key. They also authorize new trusted users to receive a share of the group's secret key. The second phase is the key generation phase, where a user generates a private and public key pair. The third phase is the service level agreement (SLA) phase, in which authorized members of the group reach a consensus on the SLA document and generate a group signature on it. The fourth phase is the data preprocessing phase, in which group members generate metadata for the data blocks they have contributed to the shared group data file, enabling future verification of shared group data file integrity. The final phase, the data auditing phase, involves verifying the integrity of the shared group data file by a data auditor. A detailed overview of the five phases is presented below.

1) SYSTEM INITIALIZATION AND MEMBER AUTHORIZATION PHASE

In a generic CLC-RDIC scheme the KGC runs the setup algorithm. But, considering the existence of security weakness due to the compromise of either passive or active KGC as shown in section I-B. We distribute the role of the KGC to the members of the group (shown in fig. 2). It can also be noted that in a shared group data auditing, the role of each member of the group is important. Hence, it is crucial to distribute the trust and key generation among members of the group itself. In the system initialization phase, a group of founding members performs the steps outlined in the setup algorithm. As a result, each founding member receives a share of the group secret key (denoted as s). A group member with a share of the group's secret key is known as an authorized member. A new member can request partial admission information from authorized members by following the steps outlined in the member authorization algorithm. Once a new member has received at least a threshold number (denoted as k) of partial admission information from authorized members (shown in fig. 3), it can generate its own share of the group secret key (s). This enables the new authorized member to provide partial admission information to other new user entities who wish to join the group. In addition, authorized members can use their share of the group secret key to generate a partial signature for a message. To generate a group signature on the message, at least k such partial signatures are required. This means that in order to create a group signature, at least k authorized members must use their share of the group secret key to generate a partial signature for the message.

2) KEY GENERATION PHASE

In this phase, a user entity runs the partial private key and secret value generation algorithms to compute its own private key. Along with this, a user also generates its corresponding public key using the public key generation algorithm. A private key is used to generate metadata for data blocks of a file. It is important to note that a user's level of access and involvement in the group is determined by their possession of both a secret share and a private key. A user who possesses both is considered a full member, while a user who only has a private key is considered a semi-member.

3) SERVICE LEVEL AGREEMENT(SLA) PHASE

This phase involves the authorized group members coming to a consensus on an SLA document. To ensure secure communication of the SLA document among the authorized members of the group and edge device, we employ broadcast encryption. We assume that the edge device acts as an authentication server and issues a broadcast group encryption key to each authorized member of the group. The SLA document contains details that identify the members of the group and their privileges. It also defines the members who are responsible for invoking the data auditing task, the time instances at which the task must be invoked, and their frequency. Once the document has been agreed upon, each of the authorized members generates a partial signature on the SLA document. One authorized member, chosen to represent the group, will then aggregate at least k of these partial signatures to generate a group signature on the SLA document. This process ensures that the final SLA document has been agreed upon by at least a threshold number of authorized members. The final SLA document, along with the group signature, is securely forwarded to the edge device, where the signature is verified by the edge device. Only after the signature has been verified, the edge device makes the SLA document accessible to all members of the group. This allows all members to have access to the agreed-upon terms and conditions outlined in the SLA document. This added step of signature verification by the edge device ensures the authenticity and integrity of the SLA document.

4) DATA PREPROCESSING PHASE

In the data preprocessing phase (shown in fig. 9), a member of the group who possesses a private key can generate the metadata corresponding to the data blocks they have contributed to the shared data file. The user then securely forwards both the data and metadata to the edge device. The edge device then verifies the validity of the metadata, if it is deemed valid, it stores the data blocks and metadata in their respective files. The metadata generation mechanism employed in this phase supports aggregate verification, which allows the edge device to verify the validity of multiple metadata blocks at once, providing an efficient and secure way of handling the data.

5) DATA AUDITING PHASE

In the data auditing phase (shown in fig. 10), a data auditor initiates the data auditing task by running the challenge algorithm. This algorithm generates a challenge set, which contains information on the random indices of the data blocks of a shared data file. The data auditor then forwards the challenge set to the edge device, which must then generate proof of data possession. The edge device runs the proof of possession algorithm and forwards its output back to the data auditor. The data auditor then validates the proof of possession and updates the auditing results accordingly. In addition, there is a verify-proof algorithm that enables the verification of multiple auditing tasks of a data auditor at once. It allows determining if the auditing tasks were performed correctly or not. This will enable the rewarding of data auditors who perform auditing tasks correctly. The reward system can also enable a semi-member to be upgraded to a full member. The reward mechanisms and member upgradation will be considered as a future scope of this work.

B. ALGORITHMS

Let us define the parameters and functions that would be utilized in the proposed protocol. A cyclic additive group G, and a multiplicative cyclic group G_1, for $|G| = |G_1| = p$, such that p is a large prime integer. A bilinear map [36] $\hat{e} : G \times G \rightarrow G_1$. The security parameter l, typically of 160 bits corresponds to the order of the group G. There are three secure hash functions $h : \{0,1\}^* \rightarrow Z_q^*$, $H : \{0,1\}^* \rightarrow G$, and $H_2 : \{0,1\}^* \rightarrow G$. There are also two secure keyed pseudo-random functions $f_\alpha : \{0,1\}^* \times \{0,1\}^l \rightarrow \{0,1\}^l$, $MAC_{key} : \{0,1\}^* \times \{0,1\}^l \rightarrow \{0,1\}^{2l}$. We employ a time-dependent pseudo-random generation system made up of two functions: π and f. One input to these functions is an λ-bit string that varies with time and serves as the seed for generating random numbers. The first function, $\pi : \{0,1\}^\lambda \times \{Z_{n+1} - \{0\}\} \rightarrow \{Z_{n+1} - \{0\}\}$. The second function, $f : \{0,1\}^\lambda \times \{Z_{n+1} - \{0\}\} \rightarrow Z_q^*$. These functions are utilized to create random challenge vectors.

The intricate details of each of the five phases of the protocol are described below:

1) SYSTEM INITIALIZATION AND MEMBER AUTHORIZATION PHASE

Setup Algorithm:

The setup algorithm is as follows:

1) Each founding member E_i of the group $\mathbb{E}$ selects a symmetric bivariate polynomial $F_i(x, y) \in \mathbb{Z}_q[x, y]$ with degree at most $k-1$, where k is the threshold parameter. Let the constant term in each of the polynomials $F_i(x, y)$ be represented by $f_{i,0}$, where $f_{i,0} = F_i(0, 0)$ and $f_{i,0}$ is secret to the corresponding entity E_i. The entities in $\mathbb{E}$ define an implicit polynomial $F(x, y) = \sum_{E_i \in \mathbb{E}} F_i(x, y)$. Let us denote s (secret key of the group) to be the constant term of the polynomial $F(x, y)$ where $s = F(0, 0) = \sum_{E_i \in \mathbb{E}} f_{i,0}$.
2) Each entity E_i calculates the value $F_{ij}(x) = F_i(x, h(E_j))$ corresponding to every other entity $E_j \in \mathbb{E}$. E_i also calculates the public information $f_{i,0} \cdot P$. E_i securely sends $F_{ij}(x)$ and $f_{i,0} \cdot P$ to the corresponding entity E_j.
3) Once the above step is complete. Every entity $E_j \in \mathbb{E}$ can calculate its private information (a univariate polynomial) as follows:
$PRI_j(x) = \sum_{E_i \in \mathbb{E}} F_{ij}(x) = \sum_{E_i \in \mathbb{E}} F_i(x, h(E_j)) = F(x, h(E_j))$.
The entity E_j can compute its private share (pri_j) of the secret key s as $pri_j = PRI_j(0) = F(0, h(E_j))$.

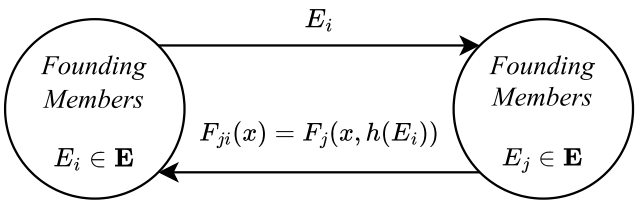


FIGURE 2. Setup.

Each entity E_j will also calculate the public key of the group as follows:
$P_{pub} = \sum_{E_i \in \mathbb{E}} f_{i,0} \cdot P = s \cdot P.$

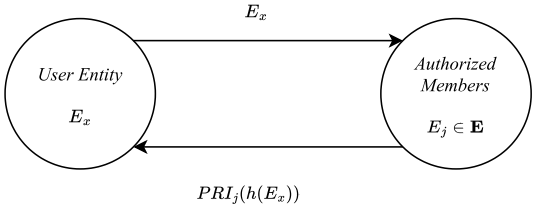
Member Authorization Algorithm: By running this algorithm, a user entity initiates the process to gain access to the partial admission information from at least a threshold number (k) of authorized members that is required to generate its own share of the group secret key.

1) Let us suppose an entity E_x wants to become a authorized member of the group $\mathbb{E}$. It should request at least k entities that are already authorized members of the group $\mathbb{E}$.
2) If an entity E_j is an authorized member of the group $\mathbb{E}$ and accepts E_x request then E_j computes the following:
$PRI_j(h(E_x)) = F(h(E_x), h(E_j)) = F(h(E_j), h(E_x)) = PRI_x(h(E_j))$
The above equation holds, considering the symmetric property of the bivariate polynomial $F(x, y)$. The entity E_j sends $PRI_j(h(E_x))$ to entity E_x.
3) Once E_x receives the values from k authorized members. It can run the following steps to generate its private information (a univariate polynomial) by using Lagrange interpolation and then it can compute its private share of the group secret s.
$\sum_{E_j \in \mathbb{E}} \prod_{E_{i,i \neq j} \in \mathbb{E}} \frac{x - h(E_i)}{h(E_j) - h(E_i)} \cdot PRI_j(h(E_x))$
$= \sum_{E_j \in \mathbb{E}} \prod_{E_{i,i \neq j} \in \mathbb{E}} \frac{x - h(E_i)}{h(E_j) - h(E_i)} \cdot F(h(E_j), h(E_x)) = F(x, h(E_x)) = PRI_x(x).$
4) Now, the entity E_x can compute its private share of the group secret key s as follows:
$pri_x = PRI_x(0) = F(0, h(E_x)).$

2) KEY GENERATION PHASE

Partial Private Key Extract Algorithm: A user entity E_x that wants to obtain its partial private key (shown in fig. 4) can run the following steps:

1) An entity E_x must contact at least k authorized members of the group $\mathbb{E}$ to obtain its partial private key.
2) If an entity E_j is an authorized member of group $\mathbb{E}$ and accepts E_x request then E_j calculates as follows:
$d_{jx} = pri_j \cdot H(E_x) = F(0, h(E_j)) \cdot H(E_x)$
The entity E_j securely sends the value d_{jx} to E_x.



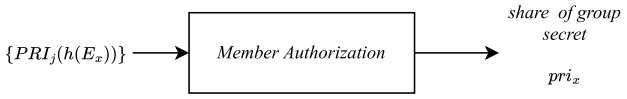
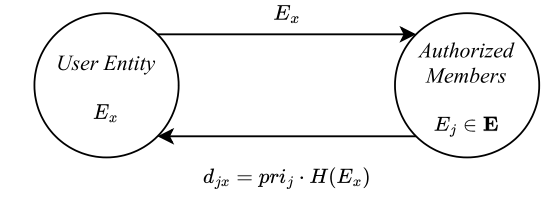


FIGURE 3. Member authorization.



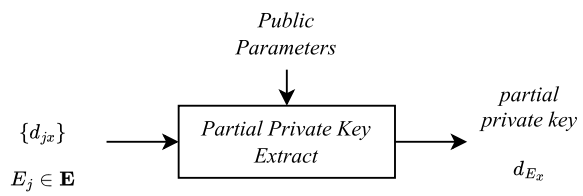


FIGURE 4. Partial private key extract.

3) Once the entity E_x receives values from k entities. It can generate its partial private key $d_{E_x} = s \cdot H(E_x)$ using Lagrange's interpolation.
$d_{E_x} = \sum_{E_j \in \mathbb{E}} d_{jx} \cdot L^{\mathbb{E}}_{E_j}(0) = H(E_x) \sum_{E_j \in \mathbb{E}} pri_j \cdot L^{\mathbb{E}}_{E_j}(0) = s \cdot H(E_x).$
Where the value $L^{\mathbb{E}}_{E_j}(0) = \prod_{E_{j,i \neq j} \in \mathbb{E}} \frac{0 - h(E_i)}{h(E_j) - h(E_i)}$
4) The entity E_x can use its partial private key d_{E_x} to establish a symmetric secret key $d_{E_{xj}}$ with an entity E_j of the group for secure data transmission as follows:
$d_{E_{xj}} = e(d_{E_x}, H(E_j)) = e(H(E_x), H(E_j))^s$
Similarly, the entity E_j can compute the symmetric key $d_{E_{jx}} = e(d_{E_j}, H(E_x)) = e(H(E_j), H(E_x))^s$.
The pairing function e is symmetric therefore $d_{E_{jx}} = d_{E_{xj}}$. Hence any two entities of the group can establish a symmetric key between them in a non-interactive way as shown above.

Secret Value Generation Algorithm: The user entity E_i randomly selects $\eta_i \in Z_q^*$ and η_i is kept as a secret to itself.

Private Key Generation Algorithm: For the proposed data auditing protocol, the entity E_i establishes its private key

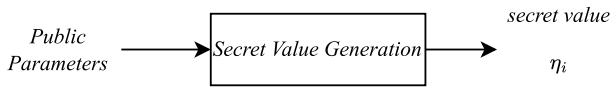


FIGURE 5. Secret value generation.

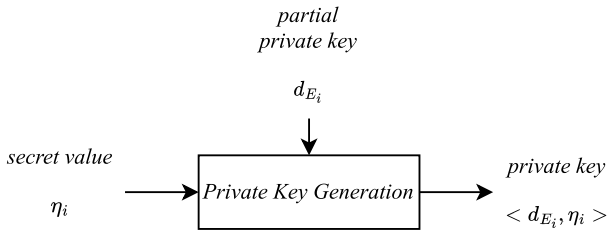


FIGURE 6. Private key generation.

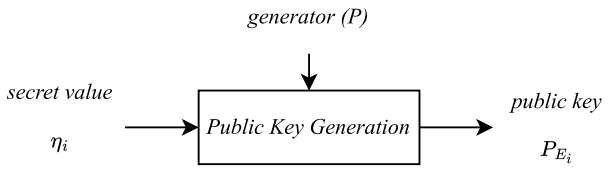


FIGURE 7. Public key generation.

as $< d_{E_i}, \eta_i >$ (shown in fig. 6). This private key is then utilized to create metadata.

Public Key Generation Algorithm: The entity E_i broadcasts its public key $P_{E_i} = \eta_i \cdot P$ (shown in fig. 7). Since the network employs CLC techniques, there is no need to attach a certificate to this public key.

3) SERVICE LEVEL AGREEMENT PHASE

Broadcast Group Encryption Key Generation Algorithm: Let the edge device have a secret key α. A user entity must forward a request for a broadcast group encryption key to the edge device (ED).

- $E_u \rightarrow ED : Request\_for\_membership$
- Edge device chooses a random unique identity ID_u for E_u.
- Computes $Key_u = f_\alpha(ID_u)$, where the function f is a keyed PRF with seed α and input ID_u.
- $FOG \rightarrow E_u : (ID_u, Key_u)$

The edge device runs the below steps to establish an SLA broadcast group encryption key:

- $\forall E_u \in \mathbb{E}$ compute:
  - $Key_u = f_\alpha(ID_u)$
  - $validity_u = exp_t||ID_u$
  - $N_u = MAC_{Key_u}(Validity_u)$
- compute $lcm_s = LCM(N_u|E_u \in \mathbb{E})$.
- choose a random value r_s and a random SLA broadcast group encryption key (K_E), $\ni (\forall E_u \in \mathbb{E}(K_E < N_u))$.
- compute $X_s = (K_E + lcm_s \cdot r_s)$.

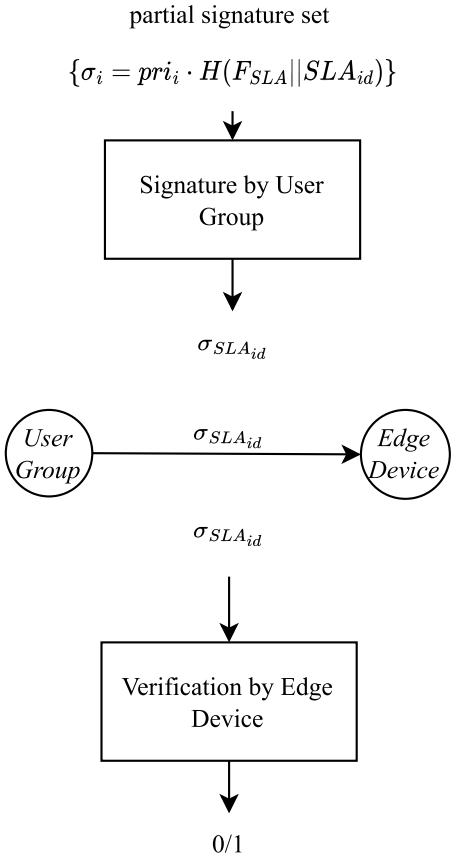


FIGURE 8. Threshold signature on SLA.

- The edge device broadcasts the $exp_t||X_s$ to the group $\mathbb{E}$.

$FOG \rightarrow \mathbb{E} : (exp_t||X_s)$

Once each group member E_u receives the broadcast information then it computes the key K_E as follows:

- $K_E = X_s mod(MAC_{Key_u}(exp_t||ID_u))$

After the above step, every member of the group has the same key K_E. Now the members can securely agree upon the SLA document (F_{SLA}) using the key K_E.

- $E_u \rightarrow \mathbb{E} : (F_{SLA}||MAC_{K_E}(F_{SLA}))$

The authenticity of the key K_E depends on the expiration time exp_t. New keys from time to time can be established as above in the future.

When a minimum of k members have reached a consensus regarding the Service Level Agreement (SLA), they will proceed to utilize their individual secret shares to produce partial signatures. Subsequently, a minimum of k of these partial signatures will be combined to form a group signature on the SLA document, providing an efficient mechanism for establishing the authenticity of the agreement.

Threshold Signature on SLA (shown in fig. 8):

Signature Algorithm: It is crucial for the *SLA* to be digitally signed by the authorized group members so that no malicious entity can tamper with the information in it. The entities can generate the signature as per the below steps:

1) Since our scheme is a (k, d) threshold scheme. At least k authorized members from $\mathbb{E}$ must perform a threshold signature on the value $H(F_{SLA}||SLA_{id})$. Here, F_{SLA} and SLA_{id} correspond to the SLA document and unique file identifier of SLA document respectively.
2) An authorized member $E_j \in \mathbb{E}$ initiates the signature process and requests at least k authorized members (including itself) to generate the partial signature on the value $H(F_{SLA}||SLA_{id})$.
3) Each entity E_i that accepts the request of entity E_j generates the partial signature σ_{SLA_i} using its private share pri_i as follows:
$\sigma_{SLA_i} = pri_i \cdot H(F_{SLA}||SLA_{id})$
The entity E_i securely forwards σ_{SLA_i} to entity E_j.
4) Once entity E_j receives at least k valid σ_{SLA_i} then it generates the final signature on the value $H(F_{SLA}||SLA_{id})$ as follows:
$\sigma_{SLA_{id}} = \sum_{E_i \in \mathbb{E}} \sigma_{SLA_i} \cdot L_{E_i}^{\mathbb{E}}(0)$
$= H(F_{SLA}||SLA_{id}) \sum_{E_i \in \mathbb{E}} pri_i \cdot L_{E_i}^{\mathbb{E}}(0)$
$= s \cdot H(F_{SLA}||SLA_{id})$.
Where the value $L_{E_i}^{\mathbb{E}}(0) = \prod_{E_{i,i \neq j} \in \mathbb{E}} \frac{0 - h(E_j)}{h(E_i) - h(E_j)}$

The entity E_j securely sends $\sigma_{SLA_{id}}$ and F_{SLA} to the edge device. Further, the entities can update the file F_{SLA} as needed in the future and perform a threshold signature as described above and send it to the edge device. The edge device will maintain only the verified latest digitally signed copy and accordingly grants access to user entities as per the SLA document (F_{SLA}).

Verification Algorithm: The edge device accepts the signature $\sigma_{SLA_{id}}$ on the file F_{SLA} only if the below equation holds:

$$e(\sigma_{SLA_{id}}, P) = e(H(F_{SLA}||SLA_{id}), P_{pub}) \quad (2)$$

where $P_{pub} = s \cdot P$ is the public key of the group. If the above equation holds then the edge device stores both the file F_{SLA} and the signature $\sigma_{SLA_{id}}$.

Correctness Proof: The edge device has the following values: $< \sigma_{SLA_{id}}, F_{SLA}, P_{pub} >$. It can compute the hash value $H(F_{SLA}||SLA_{id})$. Let us derive the left-hand side (LHS) from the right-hand side (RHS) of equation 2.

RHS of equation 2 is $e(H(F_{SLA}||SLA_{id}), P_{pub})$

$$= e(H(F_{SLA}||SLA_{id}), s \cdot P)$$
$$= e(H(F_{SLA}||SLA_{id}), P)^s$$
$$= e(s \cdot H(F_{SLA}||SLA_{id}), P)$$
$$= e(\sigma_{SLA_{id}}, P) = LHS$$

Hence proved.

4) DATA PREPROCESSING PHASE

Metadata Generation Algorithm: The following steps are run by a user entity with identity E_u to generate encrypted metadata T_i for a data block m_i corresponding to a file F_{ID}.

An entity E_u can follow these steps to create encrypted metadata T_i for a data block m_i associated with file F_{ID}

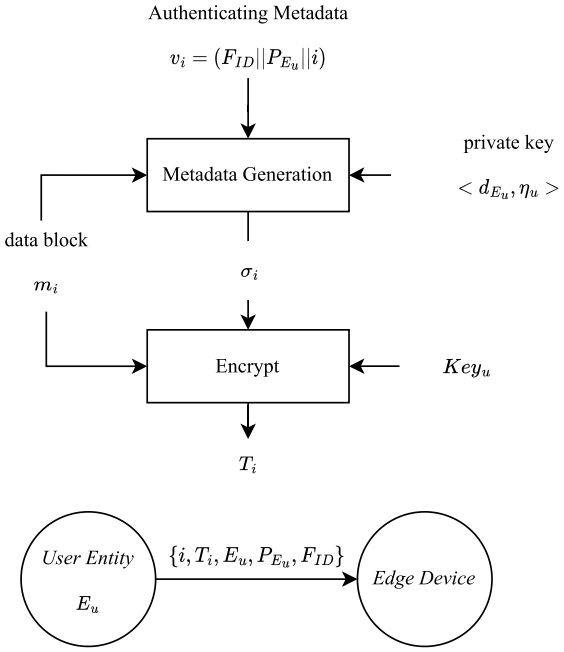


FIGURE 9. Data preprocessing phase.

- Generates the authenticating metadata $v_i = (F_{ID}||P_{E_u}||i)$ corresponding to the data block m_i. The F_{ID} value is a unique identifier value of the file F.
- Calculates the metadata for a data block m_i as $\sigma_i = \eta_u \dot{H}_2(v_i) + m_i \cdot d_{E_u}$.
- Generates the encrypted metadata $T_i = Encrypt_{Key_u}(\sigma_i||m_i)$. The key Key_u is a pre-shared symmetric key between the entity E_u and the edge device.
- The user entity E_u sends the following data to the edge device (ED) corresponding to the data block m_i.

$$E_u \rightarrow ED : \{i, T_i, E_u, P_{E_u}, F_{ID}\}$$

The edge device decrypts T_i using the key Key_u to extract σ_i and m_i. The edge device computes the authenticating metadata $v_i = (F_{ID}||P_{E_u}||i)$ and verifies the validity of σ_i w.r.t data block m_i for the user entity E_u as follows:

$$e(\sigma_i, P) \stackrel{?}{=} e(H_2(v_i), P_{E_u}) \cdot e(m_i \cdot H(E_u), P_{pub}) \quad (3)$$

If the equation 3 is valid, then the edge device updates the index matrix I of the file F_{ID} as $I_{i,u} = 1$. The edge device stores σ_i as the i^{th} metadata block into the metadata file and m_i as the i^{th} data block into the file F_{ID}.

It can also be noted that the equation 3 supports aggregate verification. For example, the user entity E_u sends the following information corresponding to n blocks of the file F_{ID} to the edge device.

$$E_u \rightarrow ED : \{\{(i, T_i)\}, E_u, P_{E_u}, F_{ID}\}$$

The edge device decrypts each T_i using the key K_u to extract every σ_i and m_i corresponding to each of the n blocks. The edge device computes the authenticating metadata $v_i = (F_{ID}||P_{E_u}||i)$ for each data block m_i and verifies the validity

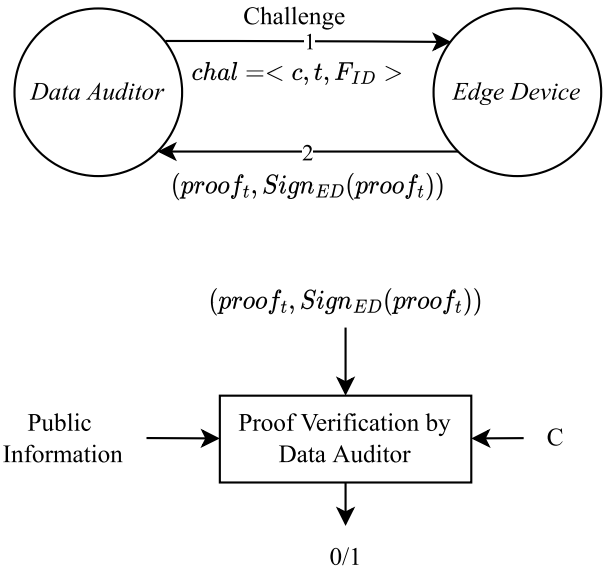


FIGURE 10. Data auditing phase.

of set of σ_i w.r.t set of data block m_i for the user entity E_u as follows:

$$\sigma = \sum_{i=1}^{n} \sigma_i; R = \sum_{i=1}^{n} m_i \cdot H_2(v_i)$$

$$e(\sigma, P) \stackrel{?}{=} e(H(E_u), P_{pub}) \cdot e(R, P_{E_u}) \quad (4)$$

5) DATA AUDITING PHASE

Challenge Algorithm: The data auditor selects a challenge set that consists of c elements chosen from the set $\{1, 2, \ldots, n\}$. Let t denote the time instant as per the SLA when the auditor initiates the auditing task. The auditor sends the challenge $chal =< c, t, F_{ID} >$ to the edge device.

$$Auditor \rightarrow ED : chal$$

Proof of Possession Algorithm: The edge device upon receiving the challenge ($chal$) extracts $\{c, t\}$. It computes a set C with c number of pseudorandom values $\{(i, w_i)\}$ using two time-dependent global pseudorandom functions. For the range of $1 \leq x \leq c$, the value of i is determined by $\pi(t, x)$ and the value of w_i is calculated by $f(t, x)$.

For every user entity E_u and the value of u belongs to the set $\{1, 2, \ldots, d\}$. The edge device computes the following values:

$$\mu_u = \sum_{i \in C} I_{i,u} \cdot m_i \cdot w_i \quad (5)$$

$$\sigma_u = \sum_{i \in C} I_{i,u} \cdot w_i \cdot \sigma_i \quad (6)$$

Finally, the edge device computes $\sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_d\}$ and $\mu = \{\mu_1, \mu_2, \ldots, \mu_d\}$ and proof of possession $proof(t) = (\sigma, \mu)$. The edge device also calculates a certificateless signature on the $proof(t)$ as $(Sign_{ED}(proof(t)))$ and sends the $proof(t)$ along with its signature on $proof(t)$ to the auditor.

$$ED \rightarrow DataAuditor : (proof(t), Sign_{ED}(proof(t)))$$

Proof Verification by Data Auditor Algorithm: The auditor runs the following steps after receiving $(proof(t), Sign_{ED}(proof(t)))$ from the edge device.

- Validates the certificateless signature $Sign_{ED}(proof(t))$ using the $proof(t)$ and the public key of the edge device. The auditor continues to the next step only if signature $Sign_{ED}(proof(t))$ is valid.
- Compute the public information $Q_u = H(E_u); \forall$ user entities $u \in \{1, \ldots, d\}$. Now aggregate values in $proof(t) =< \sigma, \mu >$ where $\sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_d\}$ and $\mu = \{\mu_1, \mu_2, \ldots, \mu_d\}$

$$\sigma_{\sum} = \sum_{u=1}^{d} \sigma_u \quad (7)$$

$$\mu_{\sum} = \sum_{u=1}^{d} \mu_u \cdot Q_u \quad (8)$$

- It computes the set $C = \{(i, w_i)\}$ with c number of pseudorandom values using two time-dependent global pseudorandom functions. For the range of $1 \leq x \leq c$, the value of i is determined by $\pi(t, x)$ and the value of w_i is calculated by $f(t, x)$. Here, (t, x) correspond to the challenge $chal$.
- For every $i \in C$, the auditor computes the authenticating metadata $v_i = (F_{ID}||P_{E_u}||i)$ by retrieving the public information of user entity E_u such that $I_{i,u} = 1$.
- The $proof(t)$ is a correct proof of possession only if the below equation is valid.

$$e(\sigma_{\sum}, P) \stackrel{?}{=} e(\mu_{\sum}, P_{pub}) \cdot \prod_{u=1}^{d} e(\sum_{i \in C}(I_{i,u} \cdot w_i \cdot H_2(v_i)), P_{E_u}) \quad (9)$$

Lemma 1: Proof of possession value generated by an honest edge device always verifies correctly as per the above equation 9.

Proof: The proof of the lemma follows the correctness guarantee of the verification equation 9 and holds always true when all the values w.r.t proof of possession are correct

$$e(\sigma_u, P) = e(\sum_{i \in C} I_{i,u} \cdot w_i \cdot \sigma_i, P) = \prod_{i \in C} e(\sigma_i, P)^{I_{i,u} \cdot w_i}$$

$$= \prod_{i \in C} e(m_i \cdot Q_u, P_{pub})^{I_{i,u} \cdot w_i} \prod_{i \in C} e(H_2(v_i), P_{E_u})^{I_{i,u} \cdot w_i}$$

[from eqn. (3)]

$$= e(\sum_{i \in C} I_{i,u} \cdot m_i \cdot w_i \cdot Q_u, P_{pub}).e(\sum_{i \in C} I_{i,u} \cdot w_i \cdot H_2(v_i), P_{E_u})$$

$$= e(\mu_u \cdot Q_u, P_{pub}).e(\sum_{i \in C} I_{i,u} \cdot w_i \cdot H_2(v_i), P_{E_u})$$

Hence,

$$e(\sigma_{\sum}, P) = e(\sum_{u=1}^{d} \sigma_u, P) = \prod_{u=1}^{d} e(\sigma_u, P)$$
$$= \prod_{u=1}^{d} e(\sum_{i \in C} I_{i,u} \cdot w_i \cdot H_2(v_i), P_u)$$
$$\prod_{u=1}^{d} e(\mu_u \cdot Q_u, P_{pub})$$
$$= e(\mu_{\sum}, P_{pub}). \prod_{u=1}^{d} e(\sum_{i \in C} I_{i,u} \cdot w_i \cdot H_2(v_i), P_{E_u})$$

□

Audit-Verification Algorithm: The audit-verification algorithm allows a member of the group to verify the correctness of multiple auditing tasks at once, related to a shared data file F_{ID}. The algorithm takes a time frame $T = [t_1, t_2]$ as input and retrieves and analyzes the challenge values, auditing results, and required information from the Audit-Log file corresponding to the time instances within the specified time frame for the data file F_{ID}. The audit-verifier then validates if the auditing instances were initiated in accordance with the SLA. Next, the audit-verifier checks for auditing results that have a value of zero in the result field. For these instances, the audit-verifier retrieves and verifies the corresponding signed proof, $(proof(t), Sign_{ED}(proof(t)))$, using the proof-verification algorithm. If the verification result is inconsistent with the corresponding auditing result, the audit-verifier informs all members of the group that the particular auditor who performed the auditing task is misbehaving else the edge device is misbehaving.

For auditing results that have a value of one in the result field, the audit-verifier follows a set of steps to ensure the integrity and accuracy of the auditing process. The notation $ProofSignature(t)$ represents the signed proof corresponding to the audit instance t within the time frame T. Given, $ProofSignature(t) = (proof(t), sign_{ED}(proof(t)))$ for, $proof(t) = (\sigma(t), \mu(t))$.

$$\sigma(t) = \{\sigma_1(t), \sigma_2(t), \ldots, \sigma_d(t)\}$$
$$\mu(t) = \{\mu_1(t), \mu_2(t), \ldots, \mu_d(t)\}$$

1) For every auditing instance t within a specified time frame T, the role of the audit-verifier is to carry out the following operations:
   - Checks the validity of the signature $Sign_{ED}(proof(t))$ using the public key of the edge device. The process proceeds only if the signature is valid.
   - Computes a set of values, denoted as $C(t) = (i, w_i)$, where $w_i = f(t, x)$ and $i = \pi(t, x)$, and aggregate information shown below for a specific audit instance t, for all values of x within a range $1 \leq x \leq c$.
     - $\sigma_{\sum}(t) = \sum_{u=1}^{d} \sigma_u(t)$
     - $\mu_{\sum}(t) = \sum_{u=1}^{d} \mu_u(t) \cdot Q_u$
2) The audit-verifier computes final aggregate values as stated below using the values obtained from each audit instance t within the specified time interval T.
   - $\sigma_{aggregate} = \sum_{\forall t \in T} \sigma_{\sum}(t)$
   - $\mu_{aggregate} = \sum_{\forall t \in T} \mu_{\sum}(t)$
   - $C_{aggregate} = \cup_{\forall t \in T} \{C(t)\}(t)$
3) The audit-verifier then proceeds to retrieve the public key P_{E_u} of each user u whose identifier i is present in the set $C_{aggregate}$ where $I_{i,u} = 1$. This information is used to compute the authenticating tag information, denoted as $w_i = (F_{ID}||P_{E_u}||i)$, by concatenating the data file identifier F_{ID}, the public key P_{E_u}, and the data block index i. If the audit verification equation below holds then the auditing tasks are deemed to have been performed correctly:

$$e(\sigma_{aggregate}), P) \stackrel{?}{=} e(\mu_{aggregate}, P_{pub}) \cdot \prod_{\forall t \in T} (\prod_{u=1}^{d} e(\sum_{i \in C(t)} (I_{i,u} \cdot w_i \cdot H_2(v_i)), P_{E_u})) \quad (10)$$

Lemma 2: The honest execution of the proof verification algorithm by the data auditor for each audit instance $t \in T$ guarantees the success of the aggregate audit-verification.

Proof: The aggregate audit-verification is only initiated when the *Result* field in the Audit-Log shows a value of 1 for each audit instance $t \in T$. This can only occur if the proof verification algorithm for each of these instances resulted in success. Therefore, each of these audit instances must satisfy the verification equation 11 as shown below:

$$e(\sigma_{\sum}(t), P) \stackrel{?}{=} e(\mu_{\sum}(t), P_{pub}) \cdot \prod_{u=1}^{d} e(\sum_{i \in C(t)} (I_{i,u} \cdot w_i \cdot H_2(v_i)), P_{E_u}) \quad (11)$$

Let us consider the LHS of equation 11 for all instances t within the time interval T.

$$\prod_{\forall t \in T} (LHS) = \prod_{\forall t \in T} (e(\sigma_{\sum}(t), P)) = e(\sum_{\forall t \in T} \sigma_{\sum}(t), P) = e(\sigma_{aggregate}, P) \quad (12)$$

The RHS of equation 11 is calculated for all $t \in T$ as shown below:

$$\begin{aligned}\prod_{\forall t\in T}(RHS) &= \prod_{\forall t\in T}(e(\mu_{\sum}(t), P_{pub})\cdot \\ &\quad \prod_{u=1}^{d} e(\sum_{i\in C(t)}(I_{i,u}\cdot w_i\cdot H_2(v_i)), P_{E_u})) \\ &= e(\sum_{\forall t\in T}\mu_{\sum}(t), P_{pub})\cdot \\ &\quad \prod_{\forall t\in T}(\prod_{u=1}^{d} e(\sum_{i\in C(t)}(I_{i,u}\cdot w_i\cdot H_2(v_i)), P_{E_u})) \\ &= e(\mu_{aggregate}, P_{pub})\cdot \\ &\quad \prod_{\forall t\in T}(\prod_{u=1}^{d} e(\sum_{i\in C(t)}(I_{i,u}\cdot w_i\cdot H_2(v_i)), P_{E_u})) \qquad (13)\end{aligned}$$

It can be noted that the equation 12 and 13 directly establish the correctness of the audit-verification equation 10. If the aggregate audit-verification equation 10 does not evaluate to true, it indicates that the data auditor has provided incorrect results and is in violation of the Service Level Agreement (SLA). In this case, the audit-verifier will notify the members of the group and report the data auditor as non-compliant. □

VI. SECURITY ANALYSIS

In this section, the robustness of the data auditing protocol against $\mathbb{A}_1$, $\mathbb{A}_2$, and $\mathbb{A}_3$ attackers is established. Initially, we examine the security of the metadata generation process against Type-I and Type-II attackers through theorem 1 and theorem 2 respectively. Subsequently, the security of the proof generation process is explored against Type-III attacker with the help of theorem 3. The definitions of the three attackers Type (I, II, III) are specified in sections IV-A, IV-B, and IV-C respectively.

The following theorem establishes the security of the proposed metadata generation mechanism against type-I adaptive chosen-message attacks in the random oracle model, assuming that the computational elliptic curve Diffie-Hellman problem (ECDHP) in G is infeasible.

Theorem 1: Suppose there is an attacker $\mathbb{A}_1$ who can forge metadata of the data auditing protocol with a probability of success = $\phi(l)$. Then, there exists a challenger $\mathbb{B}$ that uses $\mathbb{A}_1$ as a subroutine who can solve an instance of the computational elliptic curve Diffie-Hellman problem (ECDHP). Let the attacker $\mathbb{A}_1$ ask g_{md} meta data generation queries, g_{pku} Public.Key.Update queries, g_{pk} Public.Key queries, g_{sv} Secret.Value queries, g_{ppk} Partial.Private.Key queries, g_{H_2} H_2 hash queries, and g_H H hash queries where all of $g_{md}, g_{pku}, g_{pk}, g_{sv}, g_{ppk}, g_{H_2}, g_H$ are in $poly(l)$, then, $\phi'(l)$, which is the challenger $\mathbb{B}$ probability of success in solving ECDHP. The value $\phi'(l)$ is lower bounded by

$$\phi'(l) \geq \frac{\phi(l)}{e\cdot(1+g_{ppk}+q_{md})}$$

the value e is the Euler's constant. The time complexity of $\mathcal{O}(\mathcal{C}) \approx \mathcal{O}(\mathbb{A}_1)+\mathcal{O}(g_H+g_{H_2}+g_{ppk}+g_{sv}+g_{pk}+g_{pku}+g_{md})$.

Proof:

Let $\mathbb{A}_{\mathbb{1}}$ be an attacker as defined in Game-1 who can forge a metadata corresponding to a block of a file for some identity E_u. Then we design a challenger $\mathbb{B}$ that uses $\mathbb{A}_1$ as a subroutine to find a solution of the ECDHP in G. The challenger $\mathbb{B}$ runs the game-1 as below:

Setup: Given an instance of the ECDH problem $< p, G, P, A, B >, A = s\cdot P$ for some $s \in Z_q^*$. $\mathbb{B}$ selects a bilinear pairing function $\hat{e} = G\times G \rightarrow G_1$, and a cyclic multiplicative group G_1, for ($|G| = |G_1| = p$). $\mathbb{B}$ defines the the public parameters $pub_p =< G, G, p, \hat{e}, P, (P_{pub} = A), \mathbb{H}, \mathbb{H}_2 >$ and forwards pub_p to $\mathbb{A}_1$. It is important to note that the private key s is not known to $\mathbb{B}$.

$\mathbb{H}$ **Hash Query:** $\mathbb{A}_1$ sends H query adaptively for a user identity E_u of choice. $\mathbb{B}$ adds a new $\mathbb{H}$-tuple for each new H query, $\mathbb{B}$ manages a H-*Table* $= \{(E_u, D_u, a_u, \eta_u, P_{E_u}, c_u)\}$. For every repeated $\mathbb{H}$ query, $\mathbb{B}$ accesses the corresponding tuple of $\mathbb{H}$-*Table* and sends D_u to $\mathbb{A}_1$. For every new E_u query, $\mathbb{B}$ flips a coin $c_u \in \{0, 1\}$, for some probability of $c_u = 0$ is ψ and $c_u = 1$ is $1-\psi$. $\mathbb{B}$ then chooses a random value $a_u \in Z_q^*$. In case ($c_u = 1$), $\mathbb{B}$ computes $D_u = B + a_uP$, and sets $\eta_u \underset{R}{\leftarrow} Z_q^*$, and $P_{E_u} = \eta_u P$. Otherwise, if ($c_u = 0$), $\mathbb{B}$ computes $D_u = a_uP$, and sets $\eta_u \underset{R}{\leftarrow} Z_q^*$, and $P_{E_u} = \eta_u P$. $\mathbb{B}$ finally adds the tuple $(E_u, D_u, a_u, P_{E_u}, c_u)$ into H-*Table*. $\mathbb{B}$ sends D_u to $\mathbb{A}_1$.

Partial.Private.Key Query: $\mathbb{A}_1$ sends partial private key query adaptively for a user identity E_u of choice. Then, $\mathbb{B}$ accesses the $\mathbb{H}$-*Table* to check for E_u. In case E_u is absent in $\mathbb{H}$-*Table*, then $\mathbb{B}$ performs a self query to the $\mathbb{H}$ hash for the value E_u. In case ($c_u = 1$), then terminate. Otherwise, if ($c_u = 0$), $\mathbb{B}$ accesses a_u from the tuple corresponding to the user identity E_u from the $\mathbb{H}$-*Table*.. $\mathbb{B}$ computes $d_{E_u} = a_uA$, and forwards d_{E_u} to $\mathbb{A}_1$.

Secret.Value Query: $\mathbb{A}_1$ sends secret value query adaptively for a user identity E_u of choice. Then, $\mathbb{B}$ accesses the $\mathbb{H}$-*Table* to check for E_u. In case E_u is absent in $\mathbb{H}$-*Table*, then $\mathbb{B}$ performs a self query to the $\mathbb{H}$ hash for the value E_u. $\mathbb{B}$ checks if ($\eta_u = \perp$), then sets $\eta_u \underset{R}{\leftarrow} Z_q^*$ and $P_{E_u} = \eta_u P$. $\mathbb{B}$ updates the tuple corresponding to E_u of $\mathbb{H}$-*Table* with values (η_u, P_{E_u}). $\mathbb{B}$ forwards η_u to $\mathbb{A}_1$.

Public.Key Query: $\mathbb{A}_1$ sends public key query adaptively for a user identity E_u of choice. $\mathbb{B}$ accesses the $\mathbb{H}$-*Table* to check for E_u. In case E_u is absent in $\mathbb{H}$-*Table*, then $\mathbb{B}$ performs a self query to the $\mathbb{H}$ hash for the value E_u. $\mathbb{B}$ accesses P_{E_u} from the tuple corresponding to E_u and forwards it to $\mathbb{A}_1$.

Public.Key.Update Query: $\mathbb{A}_1$ sends public key update query adaptively for a user identity E_u of choice with (E_u, P'_{E_u}). $\mathbb{B}$ accesses the $\mathbb{H}$-*Table* to check for E_u. In case E_u is absent in $\mathbb{H}$-*Table*, then $\mathbb{B}$ performs a self query to the $\mathbb{H}$ hash for the value E_u. $\mathbb{B}$ updates $\eta_u = \perp$ and $P_{E_u} = P'_{E_u}$.

H_2 **Hash Query:** $\mathbb{A}_1$ sends queries for $\mathbb{H}_2$ hash adaptively for the value (v_i, P_{E_u}). $\mathbb{B}$ manages a H_2-*Table* $= \{(v_i, \mathbb{H}_2^i, a_i, P_{E_u}\}$. $\mathbb{B}$ accesses the $\mathbb{H}_2$-*Table* to check for

(v_i, P_{E_u}). In case (v_i, P_{E_u}) is absent in $\mathbb{H}_2$-*Table*, $\mathbb{B}$ selects a random value $a_i \in Z_q*$, computes $\mathbb{H}_2^i = a_i P$. $\mathbb{B}$ finally adds the tuple $\{(v_i, \mathbb{H}_2^i, a_i, P_{E_u}\}$ into H_2-*Table*. $\mathbb{B}$ sends H_2^i to $\mathbb{A}_1$.

Metadata Query: $\mathbb{A}_1$ sends queries for metadata adaptively for the value (m, v_i, E_u). $\mathbb{B}$ accesses the $\mathbb{H}$-*Table* to check for E_u. In case E_u is absent in $\mathbb{H}$-*Table*, then $\mathbb{B}$ performs a self query to the $\mathbb{H}$ hash for the value E_u. $\mathbb{B}$ accesses H-*Table* for P_{E_u},

In case $(c_u = 0)$, $\mathbb{B}$ accesses for d_{E_u} from $\mathbb{H}$-*Table*, and (v_i, P_{E_u}) from the $\mathbb{H}_2$-*Table*. In case (v_i, P_{E_u}) is absent in $\mathbb{H}_2$-*Table*, $\mathbb{B}$ performs a self query to the $\mathbb{H}_2$ hash for the value (v_i, P_{E_u}). $\mathbb{B}$ accesses η_u corresponding to E_u from the $\mathbb{H}$-*Table*. $\mathbb{B}$ generates the tag as follows:

$$\sigma = m d_{E_u} + \eta_u H_2^i$$

Otherwise, If $c_u = 1$ and (v_i, P_{E_u}) is already present in the $\mathbb{H}_2$-*Table* then terminate. Else, $\mathbb{B}$ chooses a random value $a_i \in Z_q*$, computes $\mathbb{H}_2^i = a_i P + (-mA)$ and $P_{E_u} = B$. Then computes $\sigma = a_u mA + a_i B$. $\mathbb{B}$ finally adds the tuple $\{(v_i, \mathbb{H}_2^i, \perp, P_{E_u}\}$ into H_2-*Table*. $\mathbb{B}$ forwards σ to $\mathbb{A}_1$

Metadata.Forgery: Finally, $\mathbb{A}_1$ submits a tuple $(m', \sigma', v_i', E_u', P_{E_u}')$, for σ' is a metadata forgery value on message m', for an user identity E_u with authenticating data v_i, and public key P_{E_u}.

Analysis: In case $\mathbb{A}_1$ submits a tuple $(m', \sigma', v_i', E_u', P_{E_u}')$ that is a valid forgery and if $(c_u = 0)$ then terminate. Otherwise, if $(c_u = 1)$ in the H-*Table* corresponding to the user identity E_u'. Then, the tuple must satisfy the following verification equation:

$$e(\sigma', P) = e(H_2(v_i', P_{E_u}'), P_{E_u}') \cdot e(m' H(ID_u'), P_{pub}) \quad (14)$$

$\mathbb{B}$ accesses the values $D_u = B + a_u P$, and $H_2(v_i, P_{E_u}') = a_i P$ from the H-*Table* and H_2-*Table* respectively corresponding to the tuple $(m', \sigma', v_i', E_u', P_{E_u}')$.

Using the verification equation 14, $\mathbb{B}$ uses the forgery by $\mathbb{A}_1$ to compute a solution to the instance of a ECDH problem as shown below:

$$\begin{aligned}
&Given: A = sP, B\ Compute:\ sB \\
&e(\sigma', P) = e(a_i P, P_{E_u}') \cdot e(m'(B + a_u P), P_{pub}) \\
&e(\sigma', P) = e(a_i P_{E_u}', P) \cdot e(m' s(B + a_u P), P) \\
&e(\sigma', P) = e(a_i P_{E_u}', P) \cdot e(m' s(B + a_u P), P) \\
&e(\sigma', P) = e((a_i P_{E_u}') + (m' s(B + a_u P)), P) \\
&\sigma' = a_i P_{E_u}' + m' s(B + a_u P) \\
&m' sB = \sigma' - m' a_u sP \\
&sB = (\sigma' - m' a_u P_{pub})^{(\frac{1}{m'})}
\end{aligned}$$

□

It can be noted from the above equation, that challenger $\mathbb{B}$ is able to solve for sB using attacker $\mathbb{A}_1$ as subroutine. Let success probability of challenger $\mathbb{B}$ be $\phi'(l)$, where $\phi'(l)$ depends on the probability of $\mathbb{B}$ not terminating the simulation. There are three possible stages in which the simulation can be terminated: the Metadata query, Partial.Private.Key query, and the Analysis. However, under certain circumstances, indicated by the condition $(c_u = 0)$, $\mathbb{B}$ does not terminate the simulation during the Partial.Private.Key query or the Metadata query. For $(c_u = 1)$ $\mathbb{B}$ does not terminate in the Analysis. Hence the success probability that $\mathbb{B}$ does not terminate is $(1 - \psi)\psi^{g_{ppk} + g_{md}}$ and is maximum when $\psi = \frac{g_{ppk} + g_{md}}{1 + g_{ppk} + g_{md}}$. Therefore, minimum probability of success of $\mathbb{B}$ solving ECDHP is $\phi'(l) \geq \frac{\phi(l)}{e \cdot (1 + g_{ppk} + q_{md})}$. The value $\frac{1}{1 + g_{ppk} + g_{md}}$ is non-negligible since g_{ppk}, and g_{md} are in $poly(l)$. If, success probability of attacker $\mathbb{A}_1$ is non-negligible then the success probability $\psi'(l)$ of $\mathbb{B}$ is also non-negligible.

Let the running time complexity of challenger $\mathbb{B}$ be $\mathcal{O}(B)$ which is at most the upper bound running time of all the interactions in the simulation of Game-1 and the running time of attacker algorithm $\mathbb{A}_1$ i.e. $\mathcal{O}(\mathbb{B}) = \mathcal{O}(\mathbb{A}_1) + \mathcal{O}(g_H T_H + g_{H_2} T_{H_2} + g_{ppk} T_{ppk} + g_{sv} T_{sv} + g_{pk} T_{pk} + g_{pku} T_{pku} + g_{md} T_{md})$. The values $T_H + T_{H_2} + T_{ppk} + T_{sv} + T_{pk} + T_{pku} + T_{md}$ are the running time of the respective algorithms which are polynomial time computable. Further the values $g_H, g_{H_2}, g_{ppk}, g_{sv}, g_{pk}, g_{pku}, g_{md}$ are in $poly(l)$. Hence $\mathcal{O}(\mathbb{B}) \approx \mathcal{O}(\mathbb{A}_1)$ and if $\mathcal{O}(\mathbb{A}_1)$ is in $poly(l)$ then $\mathcal{O}(\mathbb{B})$ is also in $poly(l)$.

The following theorem demonstrates that the proposed metadata generation is secure against an adaptive attacker $\mathbb{A}_2$ attempting a chosen-message attack in the random oracle model, provided that the ECDH problem in the group G is considered computationally difficult.

Theorem 2: Suppose there is an attacker $\mathbb{A}_2$ who can forge metadata of the data auditing protocol with a probability of success $= \phi(l)$. Then, there exists a challenger $\mathbb{B}$ that uses $\mathbb{A}_2$ as a subroutine who can solve an instance of the computational elliptic curve Diffie-Hellman problem (ECDHP). Let the attacker $\mathbb{A}_2$ ask g_H H hash queries, g_{pk} Public.Key queries, g_{sv} Secret.Value queries, and g_{H_2} H_2 hash queries where all of $g_H, g_{pk}, g_{sv}, g_{H_2}$ are in $poly(l)$, then, $\phi'(l)$, which is the challenger $\mathbb{B}$ probability of success in solving ECDHP. The value $\phi'(l)$ is lower bounded by

$$\phi'(l) \geq \frac{\phi(l)}{e \cdot (1 + g_{sv})}$$

the value e is the Euler's constant. The time complexity of $\mathcal{O}(\mathcal{C}) \approx \mathcal{O}(\mathbb{A}_2) + \mathcal{O}(g_{sv})$.

Proof:

Let $\mathbb{A}_2$ be an attacker as defined in Game-2 who can forge a metadata corresponding to a block of a file for some identity E_u. Then we design a challenger $\mathbb{B}$ that uses $\mathbb{A}_2$ as a subroutine to find a solution to an instance of the ECDHP in G. The challenger $\mathbb{B}$ runs the Game-2 as below:

Setup: Given an instance of the ECDH problem $< p, G, P, A, B >, A = a \cdot P$. $\mathbb{B}$ selects a bilinear pairing function $\hat{e} = G \times G \rightarrow G_1$, and a cyclic multiplicative group G_1, for $(|G| = |G_1| = p)$, and a random $s \in Z_q^*$.. $\mathbb{B}$ defines the the public parameters $pub_p =< G, G, p, \hat{e}, P, (P_{pub} =$

$sP), \mathbb{H}, \mathbb{H}_2 >$ and forwards (s, pub_p) to $\mathbb{A}_2$. The value a is not known to $\mathbb{B}$

$\mathbb{H}$ **Hash Query:** $\mathbb{A}_2$ sends H query adaptively for a user identity E_u of choice. $\mathbb{B}$ adds a new $\mathbb{H}$-tuple for each new H query, $\mathbb{B}$ manages a H-*Table* $= \{(E_u, D_u, a_u, \eta_u, P_{E_u}, c_u)\}$. For every repeated $\mathbb{H}$ query, $\mathbb{B}$ accesses the corresponding tuple of $\mathbb{H}$-*Table* and sends D_u to $\mathbb{A}_2$. For every new E_u query, $\mathbb{B}$ selects $a_u, \eta_u \overset{R}{\leftarrow} Z_p*$ and computes $D_u = a_u P$, flips a coin $c_u \in \{0, 1\}$, for some probability of $c_u = 0$ is ψ and $c_u = 1$ is $1 - \psi$. In case $(c_u = 0)$, $\mathbb{B}$ sets $P_{E_u} = \eta_u P$. Otherwise, if $(c_u = 1)$, $\mathbb{B}$ computes $P_{E_u} = \eta_u A$, and sets $\eta_u = \perp$. $\mathbb{B}$ finally adds the tuple $(E_u, D_u, a_u, \eta_u, P_{E_u}, c_u)$ into H-*Table*. $\mathbb{B}$ sends D_u to $\mathbb{A}_2$.

Secret.Value Query: $\mathbb{A}_2$ sends secret value query adaptively for a user identity E_u of choice. Then, $\mathbb{B}$ accesses the $\mathbb{H}$-*Table* to check for E_u. In case E_u is absent in $\mathbb{H}$-*Table*, then $\mathbb{B}$ performs a self query to the $\mathbb{H}$ hash for the value E_u. In case $(c_u = 1)$ then terminate. Otherwise, $\mathbb{B}$ accesses η_u from $\mathbb{H}$-*Table* and forwards it to $\mathbb{A}_2$.

Public.Key Query: $\mathbb{A}_2$ sends public key query adaptively for a user identity E_u of choice. $\mathbb{B}$ accesses the $\mathbb{H}$-*Table* to check for E_u. In case E_u is absent in $\mathbb{H}$-*Table*, then $\mathbb{B}$ performs a self query to the $\mathbb{H}$ hash for the value E_u. $\mathbb{B}$ accesses P_{E_u} from the tuple corresponding to E_u and forwards it to $\mathbb{A}_2$.

H_2 **Hash Query:** $\mathbb{A}_2$ sends queries for $\mathbb{H}_2$ hash adaptively for the value (v_i, P_{E_u}). $\mathbb{B}$ manages a H_2-*Table* $= \{(v_i, \mathbb{H}_2^i, a_i, P_{E_u}\}$. $\mathbb{B}$ accesses the $\mathbb{H}_2$-*Table* to check for (v_i, P_{E_u}). In case (v_i, P_{E_u}) is absent in $\mathbb{H}_2$-*Table*, $\mathbb{B}$ selects a random value $a_i \in Z_q*$, computes $\mathbb{H}_2^i = a_i B$. $\mathbb{B}$ finally adds the tuple $\{(v_i, \mathbb{H}_2^i, a_i, P_{E_u}\}$ into H_2-*Table*. $\mathbb{B}$ sends H_2^i to $\mathbb{A}_2$.

Metadata.Forgery: Finally, $\mathbb{A}_2$ submits a tuple $(m', \sigma', v_i', E_u', P_{E_u}')$, for σ' is a metadata forgery value on message m', for an user identity E_u with authenticating data v_i, and public key P_{E_u}.

Analysis: In case $\mathbb{A}_2$ submits a tuple $(m', \sigma', v_i', E_u', P_{E_u}')$ that is a valid forgery and if $(c_u = 0)$ then terminate. Otherwise, if $(c_u = 1)$ in the H-*Table* corresponding to the user identity E_u'. Then, the tuple must satisfy the below verification equation:

$$e(\sigma', P) = e(H_2(v_i', P_{E_u}'), P_{E_u}') \cdot e(m' H(ID_u'), P_{pub}) \quad (15)$$

$\mathbb{B}$ accesses the values $D_u = a_u P$, and $H_2(v_i, P_{E_u}') = a_i B$ from the H-*Table* and H_2-*Table* respectively corresponding to the tuple $(m', \sigma', v_i', E_u', P_{E_u}')$.

Using the verification equation 15, $\mathbb{B}$ uses the forgery by $\mathbb{A}_2$ to compute a solution to the instance of an ECDH problem as shown below:

$$\begin{aligned}
&Given: A = aP, B\ Compute:\ aB \\
&e(\sigma', P) = e(a_i B, \eta_u A) \cdot e(m'(a_u P), P_{pub}) \\
&e(\sigma', P) = e(a_i B, \eta_u aP) \cdot e(m's(a_u P), P) \\
&e(\sigma', P) = e(a_i \eta_u aB, P) \cdot e(m's(a_u P), P) \\
&e(\sigma', P) = e((a_i \eta_u aB) + (m' a_u (sP)), P) \\
&\sigma' = a_i \eta_u aB + m' a_u sP) \\
&a_i \eta_u aB = \sigma' - m' a_u sP
\end{aligned}$$

$$aB = (\sigma' - m' a_u P_{pub})^{(\frac{1}{a_i \eta_u})}$$

□

It can be noted from the above equation, that challenger $\mathbb{B}$ is able to solve for aB using attacker $\mathbb{A}_2$ as subroutine. Let success probability of challenger $\mathbb{B}$ be $\phi'(l)$, where $\phi'(l)$ depends on the probability of $\mathbb{B}$ not terminating the simulation. $\mathbb{B}$ can terminate in the Secret.Value query, and Analysis. For $(c_u = 0)$ $\mathbb{B}$ does not terminate in Secret.value query. For $(c_u = 1)$ $\mathbb{B}$ does not terminate in the Analysis. Hence the success probability that $\mathbb{B}$ does not terminate is $(1 - \psi)\psi^{q_{sv}}$ and is maximum when $\psi = \frac{q_{sv}}{1+q_{sv}}$. Therefore, the minimum probability of success of $\mathbb{B}$ solving ECDHP is $\phi'(l) \geq \frac{\phi(l)}{e \cdot (1+g_{sv})}$. The value $\frac{1}{1+g_{sv}}$ is non-negligible since g_{sv} is in $poly(l)$. If, success probability of attacker $\mathbb{A}_2$ is non-negligible then the success probability $\psi'(l)$ of $\mathbb{B}$ is also non-negligible.

Let the running time complexity of challenger $\mathbb{B}$ be $\mathcal{O}(B)$ which is at most the upper bound running time of all the interactions in the simulation of Game-2 and the running time of attacker algorithm $\mathbb{A}_2$ i.e. $\mathcal{O}(\mathbb{B}) = \mathcal{O}(\mathbb{A}_2) + \mathcal{O}(g_H T_H + g_{H_2} T_{H_2} + g_{sv} T_{sv} + g_{pk} T_{pk})$. The values $T_H + T_{H_2} + T_{sv} + T_{pk} +$ are the running time of the respective algorithms which are polynomial time computable. Further the values $g_H, g_{H_2}, g_{sv}, g_{pk}$ are in $poly(l)$. Hence $\mathcal{O}(\mathbb{B}) \approx \mathcal{O}(\mathbb{A}_2)$ and if $\mathcal{O}(\mathbb{A}_2)$ is in $poly(l)$ then $\mathcal{O}(\mathbb{B})$ is also in $poly(l)$.

The below theorem demonstrates the proposed RDIC scheme is secure against an adaptive attacker $\mathbb{A}_3$ attempting to forge an integrity proof in the random oracle model, given that the ECDHP and elliptic curve discrete log problem (ECDLP) in group G are considered computationally hard.

Theorem 3: Suppose there is an attacker $\mathbb{A}_3$ who can forge integrity proof of the data auditing protocol with a probability of success = $\phi(l)$. Then, there exists a challenger $\mathbb{B}$ that uses $\mathbb{A}_3$ as a subroutine who can solve an instance of the ECDHP or ECDLP. Let the attacker $\mathbb{A}_3$ ask g_H H hash queries, g_{pk} Public.Key queries, g_{db} Data.Block queries, and g_{mH} metadata and H_2 hash queries, where all of $g_H, g_{mH}, g_{pk}, g_{db}$ are in $poly(l)$, then, $\phi'(l)$, which is the challenger $\mathbb{B}$ probability of success in solving the above stated problem. The value $\phi'(l)$ is lower bounded by

$$\phi'(l) \geq \phi(l)(1 - \frac{1}{p})$$

for $q = |G|$. The time complexity of $\mathcal{O}(\mathcal{B}) \approx \mathcal{O}(\mathbb{A}_3) + \mathcal{O}(g_H + g_{db} + g_{mH} + g_{pk})$.

Proof:

Let $\mathbb{A}_3$ be an attacker as defined in Game-3 who can forge an integrity proof on the data auditing protocol. Then we design a challenger $\mathbb{B}$ that uses $\mathbb{A}_3$ as a subroutine to find a solution to an instance of the computational ECDHP or ECDLP in G. The challenger $\mathbb{B}$ runs the Game-3 as below:

Setup: Given an instance of the ECDL $< p, P, B, G >$ and the ECDH problem $< p, G, P, A, B >$, for $A = a \cdot P, B = b \cdot P$. $\mathbb{B}$ selects a bilinear pairing function $\hat{e} = G \times G \rightarrow G_1$, and a cyclic multiplicative group G_1, for

$(|G| = |G_1| = p)$. $\mathbb{B}$ defines the the public parameters $pub_p =< G, p, \hat{e}, P, (P_{pub} = A), \mathbb{H}, \mathbb{H}_2 >$ and forwards pub_p to $\mathbb{A}_3$. The value a, b are not known to $\mathbb{B}$

$\mathbb{H}$ **Hash Query:** $\mathbb{A}_3$ sends H query adaptively for a user identity E_u of choice. $\mathbb{B}$ adds a new $\mathbb{H}$-tuple for each new H query, $\mathbb{B}$ manages a H-*Table* $= \{(E_u, D_u, a_u, \eta_u, P_{E_u}, c_u)\}$. For every repeated $\mathbb{H}$ query, $\mathbb{B}$ accesses the corresponding tuple of $\mathbb{H}$-*Table* and sends D_u to $\mathbb{A}_3$. For every new E_u query, $\mathbb{B}$ selects $a_u, \eta_u \overset{R}{\leftarrow} Z_p*$ and computes $D_u = a_u P + \eta_u B$, and $P_{E_u} = \eta_u B$. $\mathbb{B}$ finally adds the tuple $(E_u, D_u, a_u, \eta_u, P_{E_u}, c_u)$ into H-*Table*. $\mathbb{B}$ sends D_u to $\mathbb{A}_3$.

Data.Block Query: The challenger $\mathbb{B}$ maintains a file (F_{ID}) with n data blocks and a corresponding indicator matrix $I_{u,i}$. For $F_{ID} =< m_1, m_2, m_3, \ldots, m_n >$. $\mathbb{A}_3$ sends data block query adaptively for a index i of choice. $\mathbb{B}$ accesses the F_{ID} and sends the data block in index i to $\mathbb{A}_3$.

Public.Key Query: $\mathbb{A}_3$ sends public key query adaptively for a user identity E_u of choice. $\mathbb{B}$ accesses the $\mathbb{H}$-*Table* to check for E_u. In case E_u is absent in $\mathbb{H}$-*Table*, then $\mathbb{B}$ performs a self query to the $\mathbb{H}$ hash for the value E_u. $\mathbb{B}$ accesses P_{E_u} from the tuple corresponding to E_u and forwards it to $\mathbb{A}_3$.

Metadata and $\mathbb{H}$ Query: $\mathbb{B}$ manages a $\mathbb{H}'_2$-*Table* $= \{(i, a_i, \mathbb{H}_2(F_{ID}||P_{E_u}||i)), \sigma_i\}$. The challenger $\mathbb{B}$ computes the metadata σ_i for each data block m_i with index i of the file F_{ID} w.r.t user identity E_u as maintained in index matrix I_m. First, $\mathbb{B}$ selects $a_i \overset{R}{\leftarrow} Z_p*$, where $1 \leq i \leq n$. For every i and $\mathbb{B}$ accesses the corresponding E_u as per the index matrix. $\mathbb{B}$ accesses $\mathbb{H}$-*Table*, computes $\mathbb{H}_2(F_{ID||P_{E_u}||i}) = a_i P + (-m_i A)$, and $\sigma_i = a_u m_i A + \eta_u a_i B$. It can be noted that the metadata and $\mathbb{H}$ simulation is as per the metadata generation algorithm of our proposed protocol. $\mathbb{B}$ adds the tuple $\{(i, a_i, \mathbb{H}_2(F_{ID}||P_{E_u}||i)), \sigma_i\}$ to the $\mathbb{H}'_2$-*Table*. $\mathbb{B}$ finally forwards the metadata values $(\sigma_1, \sigma_2, \ldots, \sigma_n)$ of the data blocks in file F_{ID}.

Challenge: The challenger $\mathbb{B}$ selects a value $c \in \{1, 2, \ldots, n\}$ as the size of the challenge set. $\mathbb{B}$ computes a set C with c number of pseudorandom values $\{(i, w_i)\}$ using two time dependent pseudo random functions. For the range of $1 \leq x \leq c$, the value of i is determined by $\pi(t, x)$ and the value of w_i is calculated by $f(t, x)$. $\mathbb{B}$ sends the challenge $chal =< c, t, F_{ID} >$ to the $\mathbb{A}_3$.

Proof.Forgery: Finally, $\mathbb{A}_3$ submits a integrity proof $proof(t)' = (\mu', \sigma')$ where $\sigma' = \{\sigma'_1, \ldots, \sigma'_d\}$, and $\mu' = \{\mu'_1, \ldots, \mu'_d\}$.

Analysis: $\mathbb{B}$ computes the original integrity proof $proof(t) = (\mu, \sigma)$ where $\sigma = \{\sigma_1, \ldots, \sigma_d\}$, and $\mu = \{\mu_1, \ldots, \mu_d\}$.

Additionally, $\mathbb{B}$ calculates the below values:

1) $\sigma_{\sum} = \sum_{u=1}^{d} \sigma_u$ and $\sigma'_{\sum} = \sum_{u=1}^{d} \sigma'_u$
2) $\mu_{\sum} = \sum_{u=1}^{d} \mu_u D_u$ and $\mu'_{\sum} = \sum_{u=1}^{d} \mu_u D_u$
3) For each $i \in C$ do the following:-
   - Retrieve user details for $I_{u,i} = 1$
   - Compute: $v_i = F_{ID}||P_{E_u}||i$

It can be noted that the $proof(t)$ and $proof(t)'$ correspond to the same challenge and must be valid as per the below verification equation.

$$e(\sigma_{\sum,P}) = e(\mu_{\sum}, P_{pub}).\prod_{u=1}^{d} e(\sum_{i \in C} v_i I_{i,u} \mathbb{H}_2(v_i), P_{E_u}) \tag{16}$$

$$e(\sigma'_{\sum,P}) = e(\mu'_{\sum}, P_{pub}).\prod_{u=1}^{d} e(\sum_{i \in C} v_i I_{i,u} \mathbb{H}_2(v_i), P_{E_u}) \tag{17}$$

It can be noted from equation 16 and 17 that if $\mu_{\sum} = \mu'_{\sum}$ then $proof(t) = proof(t)'$. But, this is a contradiction to our assumption that $\mu_\pi \neq \mu'_\pi$. Hence there are two scenarios as discussed below:

Scenario 1: $\sigma_{\sum} = \sigma'_{\sum}$

Dividing the equation 16 and 17 we get:

$$e(\sum_{u=1}^{d}(\mu_u D_u, P_{pub}) = e(\sum_{u=1}^{d}(\mu'_u D_u, P_{pub})$$

$$e(\sum_{u=1}^{d}(\mu_u D_u, a\,P) = e(\sum_{u=1}^{d}(\mu'_u D_u, a\,P)$$

$$e(\sum_{u=1}^{d}(a\mu_u D_u, P) = e(\sum_{u=1}^{d}(a\mu'_u D_u, P)$$

$$\sum_{u=1}^{d} a\mu_u D_u = \sum_{u=1}^{d} a\mu'_u D_u$$

$$\sum_{u=1}^{d} \Delta\mu_u a D_u = 0,$$

where $\Delta\mu_u = \mu'_u - \mu_u$.

$$\sum_{u=1}^{d} \Delta\mu_u a(a_u P + \eta_u B) = 0$$

$$\sum_{u=1}^{d} \Delta\mu_u a_u a P + \sum_{u=1}^{d} \Delta\mu_u a_u \eta_u a B = 0$$

$$aB = \frac{\sum_{u=1}^{d} \Delta\mu_{ua_u} A}{\sum_{u=1}^{d} \Delta\mu_u a_u \eta_u} \tag{18}$$

It can be noted from the equation 18 that, a solution to an instance of ECDHP is found. On further analysis, we can find a solution to the EDLP as shown below:

$$abP = \frac{\sum_{u=1}^{d} \Delta\mu_u a_u A}{\sum_{u=1}^{d} \Delta\mu_u a_u \eta_u}$$

$$bA = \frac{\sum_{u=1}^{d} \Delta\mu_u a_u A}{\sum_{u=1}^{d} \Delta\mu_u a_u \eta_u}$$

$$b = \frac{\sum_{u=1}^{d} \Delta\mu_u a_u}{\sum_{u=1}^{d} \Delta\mu_u a_u \eta_u}$$

From above, we are able to find the value b unless $\sum_{u=1}^{d} \Delta\mu_u a_u \eta_u = 0$. Further, more than one of $\Delta\mu_u$ is non-zero, and all the $\{a_u, \eta_u\}$ values are random elements from Z_p*. Therefore, there is a $\frac{1}{p}$ probability that the denominator will be zero. The value $\phi'(l)$ represents the lowest probability of success for $\mathbb{B}$ to solve an instance of the EDLP. The value of $\phi'(l)$ depends on the probability of success of the attacker algorithm $\mathbb{A}_3$ ($\phi(l)$) and the probability that $\sum_{u=1}^{d} \Delta\mu_u a_u \eta_u \neq 0$. Therefore, $\phi'(l) \geq \phi(l)(1 - \frac{1}{p})$. It can also be observed that the value $(1 - \frac{1}{p})$ is non-negligible since p is exponential in l. Further, $\phi'(l)$ is non-negligible when $\phi(l)$ is non-negligible.

Scenario 2: $\sigma_{\sum} \neq \sigma'_{\sum}$

Case 2: $\sigma_{\pi} \neq \sigma'_{\pi}$

Dividing the equation 16 by 17:

$$\begin{aligned}
e(\sigma_{\sum}, P)e(-\sigma_{\sum'}, P) &= e(\mu_{\sum}, P_{pub})e(-\mu_{\sum'}, P_{pub}) \\
\sigma_{\sum} - \sigma_{\sum'} &= a(\mu_{\sum} - \mu_{\sum'}) \\
\sum_{u=1}^{d}(\sigma_u - \sigma'_u) &= a(\sum_{u=1}^{d}(\mu_u D_u - \mu'_u D_u)) \\
\sum_{u=1}^{d}(\sigma_u - \sigma'_u) &= a(\sum_{u=1}^{d}(a_u P + \eta_u B)(\mu_u - \mu'_u)) \\
\sum_{u=1}^{d}(\Delta\sigma_u) &= a(\sum_{u=1}^{d}(a_u P + \eta_u B)(\Delta\mu_u)) \\
\sum_{u=1}^{d}(\Delta\sigma_u) &= \sum_{u=1}^{d}(a_u aP(\Delta\mu_u) + \eta_u aB(\Delta\mu_u)) \\
\eta_u aB(\Delta\mu_u) &= \sum_{u=1}^{d}(\Delta\sigma_u) - \sum_{u=1}^{d}(a_u A(\Delta\mu_u) \\
aB &= \frac{\sum_{u=1}^{d}(\Delta\sigma_u) - \sum_{u=1}^{d}(a_u A(\Delta\mu_u))}{\sum_{u=1}^{d}(\eta_u)} \qquad (19)
\end{aligned}$$

□

It can be noted from the equation 19, aB is the solution to an instance of the ECDHP unless the value of the denominator $\sum_{u=1}^{d} \eta_u = 0$. But, the value η_u is chosen randomly from Z_p^* and probability of $\sum_{u=1}^{d} \eta_u = 0$ is $\frac{1}{p^d}$. The minimum probability of success that challenger $\mathbb{B}$ solves the instance of a ECDHP is denoted by $\phi'(l)$. The value of $\phi'(l)$ depends on the probability of success of the attacker algorithm $\mathbb{A}_3$ ($\phi(l)$) and the probability that $\sum_{u=1}^{d} \eta_u \neq 0$. Therefore, $\phi'(l) \geq \phi(l)(1 - \frac{1}{p^d})$. It can also be observed that the value $(1 - \frac{1}{p^d})$ is non-negligible since p is exponential in l. Further, $\phi'(l)$ is non-negligible when $\phi(l)$ is non-negligible.

Let the running time complexity of challenger $\mathbb{B}$ be $\mathcal{O}(B)$ which is at most the upper bound running time of all the interactions in the simulation of Game-3 and the running time of attacker algorithm $\mathbb{A}_3$ i.e. $\mathcal{O}(\mathbb{B}) = \mathcal{O}(\mathbb{A}_3) + \mathcal{O}(g_H T_H + g_{db} T_{db} + g_{mH} T_{mH} + g_{pk} T_{pk})$. The values $T_H + T_{db} + T_{mH} + T_{pk}+$ are the running time of the respective algorithms which are polynomial time computable. Further the values $g_H, g_{db}, g_{mH}, g_{pk}$ are in $poly(l)$. Hence $\mathcal{O}(\mathbb{B}) \approx \mathcal{O}(\mathbb{A}_3)$ and if $\mathcal{O}(\mathbb{A}_3)$ is in $poly(l)$ then $\mathcal{O}(\mathbb{B})$ is also in $poly(l)$.

TABLE 3. Comparison of proposed scheme security properties with Yuan et al. scheme.

Security Property	Proposed Scheme	Yuan et al. 2022 [20]
Metadata Forgery Resistant (Type-I)	Yes	Yes
Metadata Forgery Resistant (Type-II)	Yes	No
Integrity Proof Forgery Resistant (Type-III)	Yes	Yes
Single Point of Failure Resistant (Decentralization)	Yes	No
Collusion Resistant	Yes	No
Authenticity and Integrity of SLA document	Yes	No
Secure Key Distribution	Yes	No

A. OTHER SECURITY ASPECTS

Threshold Signature Security: In our proposed protocol, we have employed a threshold variation of the Boneh-Lynn-Shacham (BLS) short signature scheme [36], [47] for generating a group signature on the SLA document (Section V-B3). The security of the threshold-BLS signature was first established in the random oracle model (ROM) setting by Boldyreva [47]. Subsequently, the security of the threshold-BLS signature was further strengthened by two independent works, one by Bacho et al. [48] and the other by Bellare et al. [49]. The work by Bacho et al. [48] demonstrated the security of the threshold-BLS signature against adaptive adversaries in the AGM and ROM models. On the other hand, the study by Bellare et al. [49] established the

security of the threshold-BLS signature in the generic group model (GGM). Therefore, based on the results of the works by [47], [48], and [49], it can be concluded that the security of the threshold-BLS signature scheme employed in our protocol is well established.

Broadcast Group Encryption Key Security: The proposed protocol incorporates a least common multiple (LCM)-based broadcast group encryption key (BGEK) drawing inspiration from the works of [44] and [45]. This key serves as a secure means of communicating the Service Level Agreement (SLA) document and facilitating consensus among authorized members on its content prior to generating a group signature (Section V-B3). The security of the BGEK is established through the judicious use of a standard keyed pseudorandom function (PRF), denoted by f, and its seed, denoted by α. The inclusion of an expiration time for the broadcast group encryption key serves to effectively mitigate brute-force attacks. The security of the BGEK design is rooted in the security analysis presented in [44] and [45], and follows the same lines of reasoning to establish the security of the BGEK in the proposed protocol.

The proposed scheme and a recent data auditing scheme of Yuan et al.'s are evaluated based on seven security properties as shown in Table 3, that are essential for a decentralized secure shared group data auditing scheme namely metadata forgery resistant (Type-I and Type-II), integrity proof forgery resistant (Type-III), single point of failure resistant (decentralization), collusion resistant, authenticity and integrity of SLA, and secure key distribution. The details of the security properties are discussed below:

- **Metadata forgery resistant (Type-I):** This security property ensures that a Type-I adversary (as in section IV-A) cannot forge metadata for a data block of a data file. The proposed scheme and Yuan et al. scheme are metadata forgery resistant against Type-I adversary.
- **Metadata forgery resistant (Type-II):** This security property ensures that a Type-II adversary (as in section IV-B) cannot forge metadata for a data block of a data file. The proposed scheme employs certificateless cryptography and therefore ensures security against Type-II adversary. While, Yuan et al. scheme employs ID-based cryptography which inherently has key escrow problem and is therefore susceptible to metadata forgery by Type-II adversary.
- **Integrity proof forgery resistant (Type-III):** This security property ensures that a Type-III adversary (as in section IV-C) cannot forge a data integrity proof as part of data auditing phase. The proposed scheme and Yuan et al. scheme are secure against integrity proof forgery attacks.
- **Single point of failure resistant (Decentralization)**: This property ensures that the system can function even if one or more user entities fail or compromised. The proposed scheme achieves this by using a decentralized architecture through threshold cryptography techniques that eliminates any single point of failure. While the Yuan et al. scheme has key escrow problem and therefore suffers from single point of failure.
- **Collusion resistant**: This property prevents collusion between the data auditor and the FOG server on the challenge information in advance, which restricts replaying integrity proofs that were calculated in advance. The proposed scheme achieves this by using two time dependent pseudo random functions to generate challenge information. While the data auditor in Yuan et al. scheme can generate challenge information in advance and therefore can collude with the FOG server to generate integrity proofs in advance.
- **Authenticity and integrity of SLA document**: This property ensures that the service level agreement (SLA) document is both authentic and maintains its integrity. The proposed scheme achieves this through secure threshold signature generation on the SLA document which allows the FOG server and group members to verify the authenticity and integrity of the SLA document. While, Yuan et al. scheme does not provide any mechanism to determine the authenticity and integrity of the SLA document.
- **Secure key distribution**: This property ensures that the group key is distributed securely and only to authorized group members. The proposed scheme achieves this by using a secure threshold key distribution mechanism that is based on secret sharing techniques. While, Yuan et al. scheme does not provide any such mechanism.

VII. PERFORMANCE ANALYSIS

In this section, we analyze the computation, communication, storage cost, and performance of the proposed scheme from the perspective of the user entity, the edge device, and the data auditor. The efficiency of its algorithms often determines the performance of a protocol. However, it is important to note that not all phases (algorithms) in the protocol are run with the same frequency. The system initialization and member authorization phase, and key generation phase are only run once (per user/user group) and the service level agreement phase is run a very limited number of times. In contrast, the algorithms associated with the data preprocessing phase and data auditing phase, are run frequently. In such cases, it is necessary to carefully consider these algorithms to evaluate the overall performance of the proposed protocol. We can more accurately gauge the protocol's efficiency and effectiveness by focusing on these frequently run algorithms.

A. NUMERICAL ANALYSIS

In this section, the costs associated with the communication and computation requirements are established.

Computation Cost: The following notations: point addition, hash-to-point, and point multiplication in the group G are denoted by PA_G, HTG_G, and PM_G respectively. Multiplication in G_1 is denoted by M_{G_1}. BP denotes the bilinear pairing computation. Our numerical analysis ignores ordinary

TABLE 4. Approximate running times of critical cryptographic functions.

Function	Laptop Device (millisecond)	Raspberry Pi 3 Model B Device (millisecond)
Hash-to-Point in G (HTP_G)	2	10
Bilinear Pairing (BP)	0.8	7
Point Multiplication in G (PM_G)	0.12	1.5
Multiplication in G_1 (M_{G_1})	0.3	2.8
Point Addition in G (PA_G)	0.006	0.02

TABLE 5. Numerical comparison of data auditing algorithms.

Algorithm	Proposed Scheme	Yuan et al. 2022 [20]
Metadata Generation	$n * (HTG_G + PA_G + 2 * PM_G)$	$n * (HTG_G + 2 * PA_G + 2PM_G)$
Proof Generation	$(c-1) * PA_G + c * PM_G$	$(c-1) * PA_G + c * PM_G$
Proof Verification	$(c) * HTG_G + (c-1) * PA_G + (1+c) * PM_G + 3 * BP + 1 * M_{G1}$	$c * HTG_G + 2 * c * PA_G + 2 * (c+1) * PM_G + 4 * BP + 2 * M_{G1}$

Note: n and c indicate the number of data blocks and the challenged number of data blocks, respectively.

functions such as modular multiplication and exponentiation as their cost is negligible compared to other operations. The table 4 summarizes the running time of critical cryptographic functions.

The user entity generates metadata for data blocks contributed to the shared file, this is the most expensive operation for the user entity. Still, a user can perform an offline process for most of the computation in metadata generation mechanism. An n data blocks file will cost the user entity $nHTG_G + 2nPM_G + (n-1)PA_G$. A major computational expense for an edge device is verifying the metadata and generating proof of possession. While verification of metadata for n blocks is, $nHTG_G + 3BP + nPM_G + M_{G_1} + 3(n-1)PA$ and the cost for proof of possession for c challenged data blocks is cM_G. The main cost for the data auditor is checking the validity of proof and is $cHTG_G + 3BP + (c+1)PM_G + M_{G_1} + (c-1)PA_G$ where c is the number of challenge blocks of the file. All the above values correspond to a single-user setting, while the cost for multiple-user settings is evaluated in the implementation results.

Communication Cost: During the challenge phase, the data auditor sends the edge device a set of parameters, including c, t, F_{ID}. We employ two global time-dependent pseudo-random functions that take the time instant t and provide random challenge vectors $\in Z_q*$ and random indices from $\{1, 2, 3, \ldots, n\}$ corresponding to the time t. In this case, the communication cost is of binary length $log_2(c) + log_2(t) + log_2(F_{ID})$. In the proof of possession, the edge device sends the data auditor a response of $< proof(t), Sign_{ED}(proof(t))$. For a single-user setting, the communication cost would be $log_2(\mu) + 3|G|$. While the multi-user setting for the case of d user entities of the shared group data is $log_2(\mu)d + (d+2)|G|$. Where the signature $Sign_{ED}(proof(t))$ size is $2|G|$.

Storage Cost: Our scheme supports public data auditing, and therefore the user entity and the data auditor do not need to store any message blocks or metadata. While the edge device should maintain both the message blocks and metadata without which the edge device will not be able to provide correct proof of possession. Additionally, the edge device also stores the index matrix and the signed SLA by the group members. Considering the storage capabilities of the edge device we assume the edge device scales well to accommodate any storage requirements.

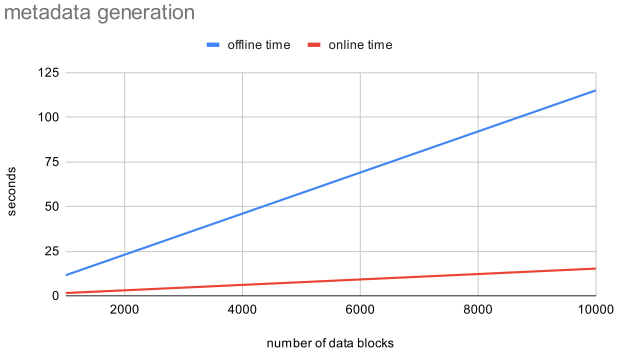


FIGURE 11. Metadata generation by user entity.

B. IMPLEMENTATION RESULTS

To measure the running time of various operations presented in Table 4, we used a simulation environment consisting of two devices, a GMP-6.2.1 [43] and a PBC-0.5.14 [46] libraries, which are pairing-based cryptography software. Specifically, we used a laptop with an Intel Core i5 6200U CPU @ 2.80 GHz and 16 GB RAM to simulate the edge device, and a Raspberry Pi 3 Model B with a quad-core Broadcom BCM2837 64 bit CPU @ 1.2 GHz and 1 GB RAM to simulate the user entity. We also used the standard security parameter *a.param* of the PBC library to initialize the system. This parameter gives the fastest speed among the default symmetric pairing options. The *a.param* has a base field size of 512 bits that provides a Dlog security of 1024 bits. The order of the elliptic curve group is 160 bit and its embedding degree is 2.

The metadata generation algorithm is run on the second device to simulate the user entity. Though the overall running time of the algorithm is high. The costly operations such as the HTP_G can be run offline since the HTP_G of the authenticating information is independent of the data blocks. Therefore, it can be noted from fig. 11 the offline time for 5000 blocks is 57.5 seconds while online is just 7.6 seconds. Further, fig. 11 shows that the running time of metadata generation is a linear function of the number of data blocks.

TABLE 6. Running time comparison of data auditing algorithms.

	Running Time (seconds)	
Algorithm	Proposed Scheme	Yuan et al. 2022 [20]
Metadata Generation	64.9	65.2
Proof Generation	0.12	0.12
Proof Verification	2.12	2.25

Note: The above running times are for n = 5000 and c = 1000, where n, c indicate the number of data blocks and the challenged number of data blocks, respectively.

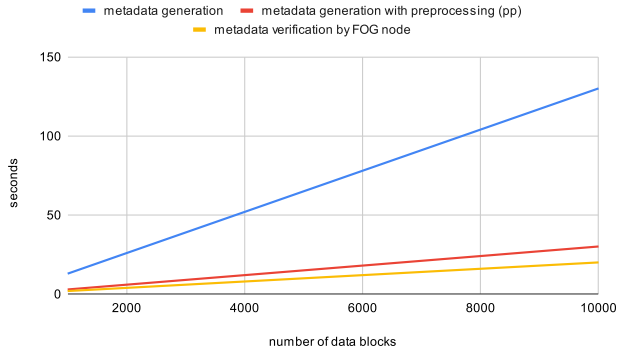


FIGURE 12. Metadata generation vs. metadata verification.

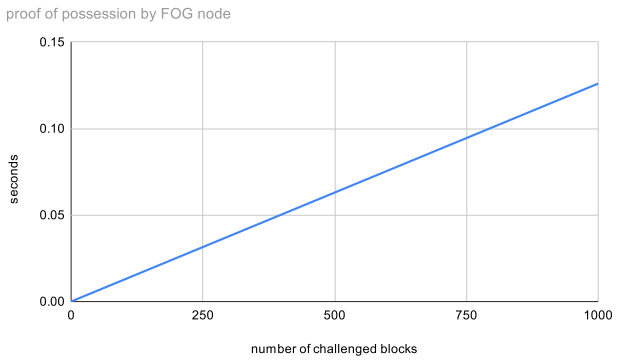


FIGURE 13. Proof of possession by the edge device.

The overall metadata generation time, the online metadata generation time, and the metadata verification time are compared to understand the computational load on the user entity and the edge device. The metadata generation is run by a user entity, while the edge device verifies the metadata. From fig 12, it is evident that metadata verification is comparable to a user entity's online computation cost. Further, the edge device can scale well to verify metadata for multiple users considering its computational capabilities.

The proof of possession generation by the edge device is a linear function of the number of challenged blocks. At the same time, the running time is proportional to the computing capabilities of the FOG device. From fig. 13, it can be noted that the edge device takes 62 ms for 500 challenged blocks

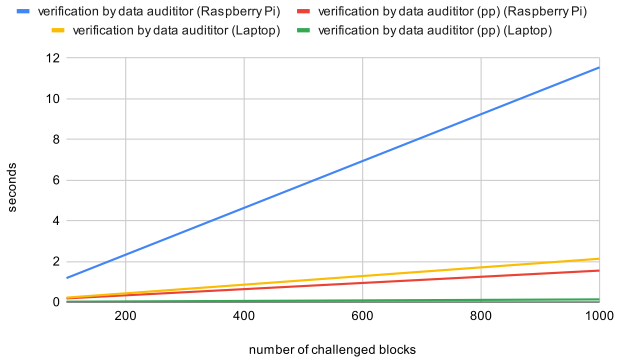


FIGURE 14. Proof verification.

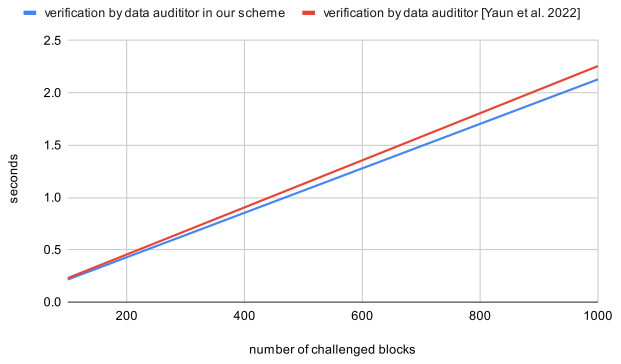


FIGURE 15. Proof verification comparison.

while it takes 125 ms for 1000 challenged blocks. Our proposed scheme follows a similar probabilistic guarantee of the scheme in [2] and, therefore, can detect the data corruption with 99% probability by only challenging 460 data blocks when the edge device corrupts 1% of the data blocks. Further, it can be noted that it only takes 57ms for the edge device to generate proof of possession for 460 challenged blocks.

Our proposed protocol supports public data auditing; therefore, the proof verification can be done by the user entity or a trusted third-party auditor. We implemented the proof verification algorithm in four ways. The first and second are on the Raspberry Pi device without preprocessing and with preprocessing, respectively. The third and fourth are on the Laptop device without preprocessing and with preprocessing, respectively. It can be noted from fig. 14 that to detect the data corruption with 99% probability when the challenged blocks are 500, the time for each of the four variations is 5.78 s, 0.78 s, 1.06 s, and 65 ms, respectively.

Further, the preprocessing task can be sourced to a third party without data privacy issues. The user entity (Raspberry Pi) performs data auditing tasks equally better against the third-party auditor (Laptop without preprocessing).

Table 5 and 6 compares the the proposed scheme and a recent data auditing scheme of Yuan et al [20] in terms of numerical analysis and concrete running time respectively.

It can be noted from Table 6 and fig. 15 that the cost of metadata generation and proof verification is better in our proposed scheme. While the proof generation in both schemes have almost the same cost. We have implemented the metadata generation on Raspberry Pi device, proof generation and proof verification on a Laptop device without the preprocessing for both schemes.

VIII. CONCLUSION

The proposed work has successfully mitigated the implications of key compromise of the KGC through distributed and decentralized key generation mechanisms specifically suited to CLC-based shared group data auditing for the FOG-CPS scenarios. Further, our flexible approach to carefully admitting a new user entity to the network with full or partial privileges can enable a secure and robust functioning of the group tasks. The metadata generation mechanism efficiently supports the aggregation of group users' information in the proof generation, proof verification, and audit verification. The implementation results and security analysis have demonstrated the proposed work's feasibility, efficiency, and security. The future work of this research aims to further enhance the performance of shared group data auditing, providing scalable protocols for diverse, resource-constrained devices in the fog computing architecture. Additionally, mechanisms will be established to incentivize group members who successfully comply with the agreed service level agreement and correctly perform the auditing duties.

REFERENCES

[1] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Thing," in *Proc. 1st MCC Workshop Mobile Cloud Comput.*, Helsinki, Finland, 2012, pp. 13–16.

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

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

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

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

[6] T. Shang, F. Zhang, X. Chen, J. Liu, and X. Lu, "Identity-based dynamic data auditing for big data storage," *IEEE Trans. Big Data*, vol. 7, no. 6, pp. 913–921, Dec. 2021.

[7] J. Zhao, C. Xu, F. Li, and W. Zhang, "Identity-based public verification with privacy-preserving for data storage security in cloud computing," *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, vol. E96.A, no. 12, pp. 2709–2716, 2013.

[8] H. Wang, "Identity-based distributed provable data possession in multicloud storage," *IEEE Trans. Services Comput.*, vol. 8, no. 2, pp. 328–340, Mar. 2015.

[9] J. Zhang and Q. Dong, "Efficient ID-based public auditing for the outsourced data in cloud storage," *Inf. Sci.*, vols. 343–344, pp. 1–14, May 2016.

[10] D. He, H. Wang, J. Zhang, and L. Wang, "Insecurity of an identity-based public auditing protocol for the outsourced data in cloud storage," *Inf. Sci.*, vol. 375, pp. 48–53, Jan. 2017.

[11] Z. Liu, L. Ren, R. Li, Q. Liu, and Y. Zhao, "ID-based sanitizable signature data integrity auditing scheme with privacy-preserving," *Comput. Secur.*, vol. 121, Oct. 2022, Art. no. 102858.

[12] G. Bian, R. Zhang, and B. Shao, "Identity-based privacy preserving remote data integrity checking with a designated verifier," *IEEE Access*, vol. 10, pp. 40556–40570, 2022.

[13] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, document RFC 5280, Internet Engineering Task Force, Fremont, CA, USA, 2008.

[14] B. Wang, B. Li, H. Li, and F. Li, "Certificateless public auditing for data integrity in the cloud," in *Proc. IEEE CNS*, National Harbor, MD, USA, Oct. 2013, pp. 136–144.

[15] D. He, S. Zeadally, and L. Wu, "Certificateless public auditing scheme for cloud-assisted wireless body area networks," *IEEE Syst. J.*, vol. 12, no. 1, pp. 64–73, Mar. 2018.

[16] D. Kim and I. R. Jeong, "Certificateless public auditing protocol with constant verification time," *Secur. Commun. Netw.*, vol. 2017, pp. 1–14, Jan. 2017.

[17] D. He, N. Kumar, H. Wang, L. Wang, and K.-K. R. Choo, "Privacy-preserving certificateless provable data possession scheme for big data storage on cloud," *Appl. Math. Comput.*, vol. 314, pp. 31–43, Dec. 2017.

[18] Y. Liao, Y. Liang, A. W. Oyewole, and X. Nie, "Security analysis of a certificateless provable data possession scheme in cloud," *IEEE Access*, vol. 7, pp. 93259–93263, 2019.

[19] Y. Zhao and J. Chang, "Certificateless public auditing scheme with designated verifier and privacy-preserving property in cloud storage," *Comput. Netw.*, vol. 216, Oct. 2022, Art. no. 109270.

[20] Y. Yuan, J. Zhang, W. Xu, and Z. Li, "Identity-based public data integrity verification scheme in cloud storage system via blockchain," *J. Supercomput.*, vol. 78, no. 6, pp. 8509–8530, Apr. 2022.

[21] K. Lei, M. Du, J. Huang, and T. Jin, "Groupchain: Towards a scalable public blockchain in fog computing of IoT services computing," *IEEE Trans. Services Comput.*, vol. 13, no. 2, pp. 252–262, Mar. 2020.

[22] 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, Sep. 2020.

[23] B. Wang, B. Li, and H. Li, "Knox: Privacy-preserving auditing for shared data with large groups in the cloud," in *Applied Cryptography and Network Security*. Berlin, Germany: Springer, 2012, pp. 507–525.

[24] Y. Zhang, J. Yu, R. Hao, C. Wang, and K. Ren, "Enabling efficient user revocation in identity-based cloud storage auditing for shared big data," *IEEE Trans. Depend. Sec. Comput.*, vol. 17, no. 3, pp. 608–619, May/Jun. 2020.

[25] K. He, J. Chen, Q. Yuan, S. Ji, D. He, and R. Du, "Dynamic group-oriented provable data possession in the cloud," *IEEE Trans. Depend. Sec. Comput.*, vol. 18, no. 3, pp. 1394–1408, May/Jun. 2021.

[26] Y. Li, Y. Yu, B. Yang, G. Min, and H. Wu, "Privacy preserving cloud data auditing with efficient key update," *Future Gener. Comput. Syst.*, vol. 78, pp. 789–798, Jan. 2018.

[27] F. Armknecht, J.-M. Bohli, G. Karame, and W. Li, "Outsourcing proofs of retrievability," *IEEE Trans. Cloud Comput.*, vol. 9, no. 1, pp. 286–301, Jan. 2021.

[28] 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, p. 49, Jun. 2018.

[29] Y. Ming and W. Shi, "Efficient privacy-preserving certificateless provable data possession scheme for cloud storage," *IEEE Access*, vol. 7, pp. 122091–122105, 2019.

[30] J. R. Gudeme, S. Pasupuleti, and R. Kandukuri, "Certificateless privacy preserving public auditing for dynamic shared data with group user revocation in cloud storage," *J. Parallel Distrib. Comput.*, vol. 156, pp. 163–175, Oct. 2021.

[31] J. R. Gudeme, S. K. Pasupuleti, and R. Kandukuri, "Certificateless multi-replica public integrity auditing scheme for dynamic shared data in cloud storage," *Comput. Secur.*, vol. 103, Apr. 2021, Art. no. 102176.

[32] H. Yan, Y. Liu, Z. Zhang, and Q. Wang, "Efficient privacy-preserving certificateless public auditing of data in cloud storage," *Secur. Commun. Netw.*, vol. 2021, pp. 1–11, May 2021.

[33] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979.

[34] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, "Perfectly secure key distribution for dynamic conferences," *Inf. Comput.*, vol. 146, no. 1, pp. 1–23, Oct. 1998.
[35] N. Saxena, G. Tsudik, and J. H. Yi, "Efficient node admission for short-lived mobile ad hoc networks," in *Proc. 13th IEEE Int. Conf. Netw. Protocols (ICNP)*, Boston, MA, USA, Nov. 2006, p. 278.
[36] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," *J. Cryptol.*, vol. 17, no. 4, pp. 297–319, Sep. 2004.
[37] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," *SIAM J. Comput.*, vol. 32, no. 3, pp. 586–615, 2003.
[38] M. Scott, "Implementing cryptographic pairings," in *Proc. 1st Int. Conf. Pairing-Based Cryptogr.*, vol. 4575, 2007, pp. 177–196.
[39] S. S. Al-Riyami and K. G. Paterson, "Certificateless public key cryptography," in *Advances in Cryptology—ASIACRYPT 2003*. Berlin, Germany: Springer, 2003, pp. 452–473.
[40] J. Li, H. Yan, and Y. Zhang, "Certificateless public integrity checking of group shared data on cloud storage," *IEEE Trans. Serv. Comput.*, vol. 14, no. 1, pp. 71–81, Jan./Feb. 2018.
[41] 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 Trans. Veh. Technol.*, vol. 69, no. 12, pp. 15815–15827, Dec. 2020.
[42] M. S. Burra and S. Maity, "Certificateless reliable and privacy-preserving auditing of group shared data for FOG-CPSs," *Secur. Commun. Netw.*, vol. 2022, pp. 1–32, Feb. 2022.
[43] T. Granlund. *The GNU Multiple Precision Arithmetic Library*. Accessed: Aug. 5, 2022. [Online]. Available: https://gmplib.org
[44] R. K. Balachandran, B. Ramamurthy, X. Zou, and N. V. Vinodchandran, "CRDTH: An efficient key agreement scheme for secure group communications in wireless ad hoc networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, vol. 2, May 2005, pp. 1123–1127.
[45] S. H. Talawar, S. Maity, and R. C. Hansdah, "Secure routing with an integrated localized key management protocol in MANETs," in *Proc. IEEE 28th Int. Conf. Adv. Inf. Netw. Appl.*, Victoria, BC, Canada, May 2014, pp. 605–612.
[46] *The PBC Library*. Accessed: Aug. 5, 2022. [Online]. Available: http://crypto.standford.edu/pbc/
[47] A. Boldyreva, "Threshold signatures, multisignatures and blind signatures based on the gap-Diffie–Hellman-group signature scheme," in *Public Key Cryptography—PKC 2003*. Berlin, Germany: Springer, 2003, pp. 31–46.
[48] R. Bacho and J. Loss, "On the adaptive security of the threshold BLS signature scheme," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Los Angeles, CA, USA, Nov. 2022, pp. 193–207.
[49] M. Bellare, S. Tessaro, and C. Zhu, "Stronger security for non-interactive threshold signatures: BLS and FROST," *Cryptol. ePrint Arch.*, 2022. Accessed: Nov. 20, 2022. [Online]. Available: https://eprint.iacr.org/2022/833.pdf

MANOHAR SAI BURRA (Member, IEEE) received the B.E. degree in electronics and communication engineering from Osmania University, Telangana, India, in 2015, and the M.Tech. degree in computer networks and information security from JNTU Hyderabad, Telangana, in 2017. He is currently pursuing the Ph.D. degree with the Department of Information Technology. He is also associated with the FOG-CPS Security Laboratory, Indian Institute of Information Technology Allahabad (IIIT-A), India. His research interests include applied cryptography, cyber-physical systems security, and network security.

SOUMYADEV MAITY (Member, IEEE) received the B.E. degree in information technology from the Dr. B. C. Roy Engineering College, Durgapur, West Bengal, in 2004, the M.Tech. degree in information technology from the University of Calcutta, Kolkata, in 2007, and the Ph.D. degree from the Department of Computer Science and Automation, Indian Institute of Science, Bengaluru, in 2015. He secured the FIRST position. He continued his academic pursuits. Currently, he is an Assistant Professor in information technology with the Indian Institute of Information Technology, Allahabad (IIIT-A), Prayagraj, India. He has been with institute, since 2017, and is also associated with the FOG-CPS Security Laboratory, IIIT-A. Prior to joining IIIT-A, he was a Visiting Scientist with the Advanced Computing and Microelectronics Unit, Indian Statistical Institute, Kolkata, and a Postdoctoral Fellow with the National Institute of Technology, Rourkela. He has also a full-time Lecturer with the Guru Nanak Institute of Technology, Kolkata, for more than a year. He has published numerous articles in reputed international journals, including the IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING, *Vehicular Communications*, and *Computer Networks*. He has also authored more than 30 international conference papers. His research interests include applied cryptography, quantum and post-quantum cryptography, network security, cyber-physical systems security, cloud data security, software security, and anomaly detection systems. He is also a reviewer for reputed international journals, such as the IEEE SENSORS JOURNAL, IEEE ACCESS, and *Journal of Information Security and Applications*.