Privacy Preserving Public Auditing with Data Deduplication for Secure Data Storage in Fog to Cloud based IoT

Ruchi Saha[a], P Syam Kumar[b]

[a]*Indian Institute of Information Technology Design and Manufacturing Kancheepuram, 600127, India*
[b]*Institute for Development and Research in Banking Technology,500057, Hyderabad, India*

Abstract

The public auditing with deduplication schemes in cloud have been studied to ensure the integrity of the data and save storage space by eliminating duplicate data from multiple users in cloud storage. However, the existing public auditing with deduplication schemes are not suitable for fog-cloud based IoT scenario where data generated by IoT is stored in the cloud through fog node. Fog has been introduced as a new layer in the cloud storage hierarchy to boost performance of the cloud based IoT. In this paper, we propose a privacy preserving public integrity auditing with data deduplication for secure data storage in fog-cloud based IoT scenario. Our scheme achieves data deduplication through fog based block level deduplication and fog verifies proof of ownership on the behalf of cloud which reduces communication and computation overhead. Moreover, our scheme enables public auditor to perform public integrity auditing using homomorphic verifiable tags and protects data privacy from auditor during auditing process. Security analysis prove that our scheme is secure against different types of attacks. Performance analysis and experimental results shows that our scheme is efficient.

Keywords: Public auditing, Deduplication, Fog computing, Cloud storage, IoT, privacy-preserving

1. Introduction

Internet of Things (IoT) refers to the network of physical objects (called IoT devices) capable of transmitting data over a network. Due to limited computing resources, data generated by IoT devices are handled in cloud [1]. With its growing popularity, centralized cloud is unable to provide satisfactory services to the users. But IoT demands latency sensitive computation for constant application handling. To address this problem, Fog computing was introduced between cloud and users [2]. The Fog devices located in vicinity to users are responsible for intermediate computation and storage, hence reducing communication time for low latency IoT devices [3]. Although fog computing conveniently resolves latency challenges, it fails to provide some important requirements for fog based cloud data storage such as the abilities of integrity auditing of data in cloud by users and identifying duplicated data by cloud servers [4]. We illustrate both problems below in detail.

The first issue is integrity auditing of data in cloud due to the following reasons: i) Once data uploaded into the cloud, users lose control over their data. ii) Sometimes, cloud service providers may discard rarely accessed data of ordinary users to save storage space or may hide the data loss incidents to build their reputation [5]. Second problem is detecting and removing duplicate data in cloud. i.e. due to rapid development of cloud storage service, different users may try to upload a same data file, which will increase the extra storage costs. A survey from EMC shows that 75% of recent digital data is duplicated [6]. Data De-duplication plays a crucial role, which allow to store only a single copy of the same file or block from different users and make a provide data access link for every user. Hence, it is necessary to design integrity auditing with deduplication mechanism to assure the users about correctness of their data and save storage space in the cloud.

To address above issues, public auditing and deduplication schemes [7-12] have been proposed in literature. These schemes verifies integrity of data under the condition that the cloud stores only one copy of the same file from multiple users. However, these schemes are not directly applicable to fog-cloud based IoT devices because in IoT scenario, the data is generated in blocks by different devices (IoT) and gathered in a single place (cloud) for storage. Hence, data uploaded in the form of blocks instead of single file, in such cases, duplication will also occur for the blocks rather than the the file. Moreover, they did not consider the fog node, which is crucial for cloud based IoT devices for providing satisfactory services to the end users.

Email addresses: ruchisaha0909@gmail.com (Ruchi Saha), psyamkumar@idrbt.ac.in (P Syam Kumar)

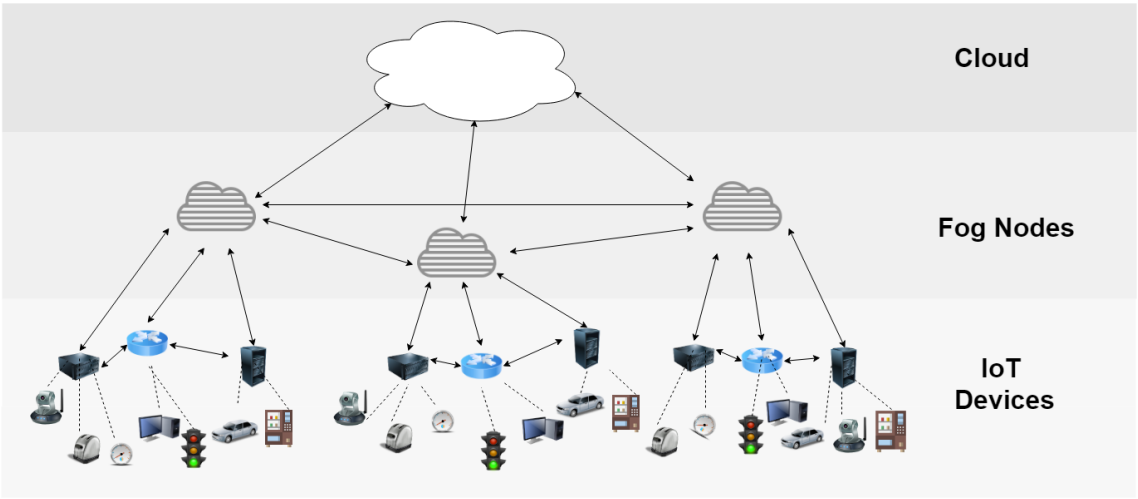

Figure 1: Fog Architecture

*1.1. **Contributions***

In this paper, we propose a privacy preserving public auditing with deduplication scheme to address both integrity auditing and data deduplication. We believe that this is the first scheme for public integrity auditing with deduplication in fog based IoT. The main contributions of our scheme are:

1. Our scheme achieves fog based data deduplication at block level that stores only single block instead of multiple duplicated blocks, which are coming from different IoT devices
2. Our scheme reduce communication overhead of resource limited IoT devices with help of fog node i.e. fog node acts as a proxy for cloud and performs deduplication check prior to cloud
3. Our scheme provides proof of Ownership(PoW) whenever duplication happens in cloud, which can help to detect unauthorized user who is trying to store same copy of data to cloud.
4. Our scheme enables public third party auditor perform integrity auditing to reduce burden of users and also protects data privacy against the curious third party auditor by using hash of the data in the proof.
5. Security analysis proves that our scheme is secure against different types of attacks. The experimental and theoretical analysis shows that the proposed scheme is efficient and suitable for fog-cloud based IoT environments.

2. Related Work

In this section, we briefly discuss some previous novel works on the given challenges with respect to generic cloud and fog and state their limitations in fog to cloud based IoT.

2.1. Integrity Auditing

To address integrity auditing, in 2007 Provable Data Possession (PDP) was proposed by Ateniese et al. [13] where users stores the data in cloud and can audit the integrity of data without downloading entire file back. Based on PDP, several proposed Public auditing schemes which addresses different issues such as public auditing with data dynamics [14-18], privacy preserving public auditing [19-22] and public integrity auditing of shared data [23-28].

Although these schemes provide novel and efficient techniques to verify the integrity of data, they did not address the data deduplication.

2.2. Secure Deduplication

With increasing popularity of cloud storage, duplicate data is also becoming one of the biggest concern. Deduplication is a technique which removes duplicate data and stores only single copy of file (file-level deduplication) or block (block-level deduplication). For secure storage deduplication, Halevi et al. [29] firstly designed a proof of ownership(PoW), which allow a user to prove to cloud that he has has the file. Later, Pietro et al. [30] enhanced a proof of ownership scheme to reduce computation overhead of the user. However, these schemes do not consider the data privacy. Ng et al. [31] introduced a secure de-duplication on encrypted data using convergent encryption to protect data privacy while while eliminating duplicate data, but it is suffering from brute force attack. To address this attack, Bellare et al. [32] proposed DupLESS scheme using messagelocked encryption and explored in efficient and secure storage. Since then several secure deduplication over encrypted data schemes have been proposed [33-40]. Although these schemes supports deduplication on plain text as well as for the encrypted files, they did not consider the integrity auditing.

2.3. Integrity auditing with Deduplication

All the schemes above consider either integrity auditing or data deduplication, In order to address both issues together, public integrity auditing with deduplication schemes have proposed [7-12]. Zheng et al. [7] first proposed a integrity auditing with deduplication scheme, but it creates storage overhead for each file, because cloud requires to maintain n authenticators for each block in the file. To address this issue, Yuan et al. [8] designed a "public constant auditing scheme with secure deduplication (PCAD)" with constant storage overhead. In this scheme, cloud keeps only one authenticator of each file block from multiple users who has the same file by aggregating the authenticators of the same file blocks. Although, this scheme supports both public auditing and deduplication, it does not support the data privacy. Naelah et al.[9] proposed a two-level privacy-preserving public auditing scheme that supports cross-user data deduplication. They proved that the TPA would not get the content of user's data while achieving integrity and deduplication. Later, Li et al.[10] proposed two secure schemes namely SecCloud and SecCloud+. SecCloud introduces an auditing entity with an additional MapReduce cloud, which helps users to generate tags for data blocks before uploading to cloud and audit the integrity of blocks that stored in cloud. SecCloud+ is designed secure deduplication, which supports data integrity auditing and deduplication over encrypted data. Hou et al.[11] proposed an identity-protected secure auditing and deduplicating data scheme. It achieves the constant storage. Secondly, it achieves the constant verification time. i.e. the first user require to compute the authenticator for block of each file where as other users not require to generate it. In their subsequent work, How et al.[12] explored cloud storage auditing with deduplication supporting different security levels according to data popularity. This scheme proved the semantic security for unpopular data and the secure deduplication for popular data concurrently. Further, TPA still can verify the data integrity even if data popularity changes.

Although above schemes achieves integrity auditing with data deduplication, they cannot applied directly for fog to cloud based IoT devices, because deduplication check is done at the file-level, which is not suitable for IoT. Second they did not consider the fog node. Therefore, exploring how to efficiently achieve privacy preserving public storage integrity auditing with deduplication in fog-cloud based IoT scenarios is challenging task.

3. System Architecture, Threats, Goals and Preliminaries

3.1. System Architecture

As shown in Figure 1, the following entities are present in fog based cloud computing for IoTs.

(a) *Users:* We coin this term to the devices which collects IoT data using sensors, organize them into blocks and send them to fog for further storage. We consider them as sources of data in fog based cloud computing.

(b) *Fog Nodes:* They can be seen as a small cloud nearby. They are placed between IoT devices and cloud to avoid huge network latency, but not necessarily present at the edge of the network. Fog nodes receive data from the user using any protocol in real time and provides transient storage which is often 1–2 hours. After that, fog sends the data to cloud for permanent storage.

(c) *Cloud:* Cloud provides scalable and on-demand computer resources such as storage and computation.

(d) *Third Party Auditor(TPA):* TPA checks the correctness of the data based upon user request. TPAs are considered as trusted but curious.

3.2. Threats

We define following possible attacks on our scheme in the light of integrity auditing and data data deduplications:

(a) *Forge attack (By cloud):* In order to bypass authentication, cloud may forge the proof.

(b) *Replace attacks:* In order to bypass the verification, cloud may replace the damaged data block and tag pair with another valid data block and tag pair.

(c) *Replay attacks:* To bypass the verification, cloud may reuse the previously generated proof.

(d) *Forge attack (By adversary):* An adversary may access to the data by forging a valid Proof of Ownership.

3.3. Design Goals

To maintain the integrity of data and data deduplication in fog-cloud based IoT environments, proposed scheme should achieve following goals:

(a) *Public Auditing:* Any approved auditor is capable of checking the integrity of data stored in cloud.

(b) *Blockless Verification and Privacy Preserving:* The auditor can validate the correctness of the data without actually having the data itself as well as the auditor cannot obtain any information regarding the data from the proof given.

(c) *Auditing Soundness*: Cloud should not be able to generate a valid proof without actually storing the data.

(d) *Data Deduplication:* Cloud must store only a single copy of data when it detects redundant upload data.

(e) *Proof of Ownership:* To ensure that user cannot generate data proof unless it has the data.

(f) *Efficiency:* Minimum computation and communication overhead should be incurred by the scheme in fog based cloud storage.

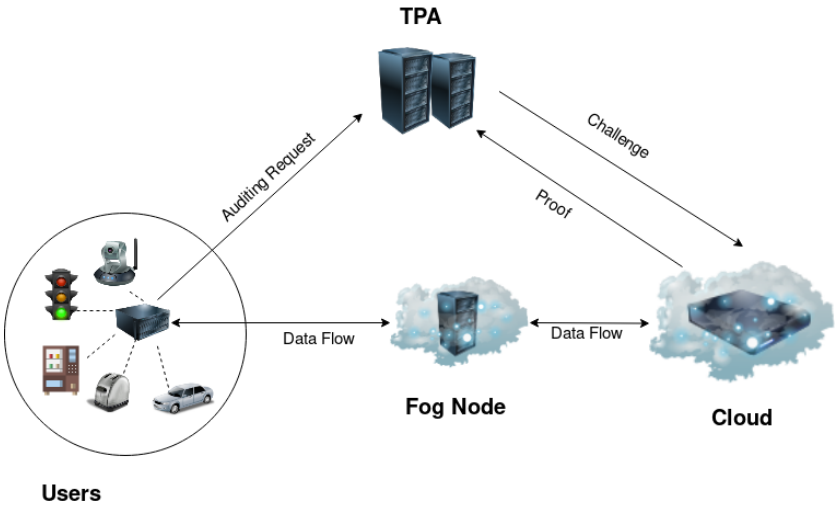

Figure 2: System Model

3.4. ***Background and Preliminaries***

3.4.1. ***Bilinear Map***

Let G_1 and G_T be two multiplicative cyclic groups whose order is a large prime p. A bilinear map is a function $e : G_1 \times G_1 \rightarrow G_T$ with the following properties:

(a) *Computability:* An efficient algorithm must exist for computing the bilinear map e.
(b) *Bilinearity:* $\forall \alpha, \beta \in G_1$ and $m, n \in Z_p^*$, $e(\alpha^m, \beta^n) = e(\alpha, \beta)^{m \cdot n}$
(c) *Non-degeneracy:* $e(\alpha, \alpha) \neq 1$

3.4.2. ***Computational Diffie-Hellman Assumption (CDH)***

Let W be a cyclic group of prime order r, for an arbitrarily chosen generator α and arbitrary numbers $m, n \in Z_r^*$, given $(\alpha, \alpha^m, \alpha^n) \in G$, it is computationally infeasible to determine the value $\alpha^{m \cdot n}$.

3.4.3. ***Discrete Logarithm Assumption (DL)***

Let W be a cyclic multiplicative group of prime order r, with a generator α. For a given $\beta \in W$, it is computationally infeasible to determine the value $m \in Z_r^*$ such that $\beta = \alpha^m$

3.4.4. ***Homomorphic Verifiable Tags (HVT)***

The Homomorphic Verifiable Tags (HVTs) introduced by Ateniese et al., (2007) is a popular building block for public auditing. Along with unforgeability, HVTs display below properties:

(a) *Blockless verification:* the public auditor should verify the integrity of data without having any knowledge on data.
(b) *Homomorphism:* Given two tags σ_{m1} and $sigma_{m2}$ for two messages m_1 and m_2, it can be combined to the value $T_{m_1+m_2}$ corresponding to the message $m_1 + m_2$

3.4.5. ***Notations***

Notations	Meaning
e	A bilinear pairing
G_1, G_T	Two cyclic multiplicative groups of large prime order
H_1, H_2	Two different secure cryptographic hash functions
g	Generator of G_1
Z_p^*	A prime field with non-zero elements
h	A secure Indexing function
sk	secret key of the user
pk	public key of the user
m_i	i^{th} message block
σ_i	Tag of m_i
t_i	Hash index of m_i
x_i	message index of m_i
n	Number of blocks in shared message
w	Number of challenged blocks
$chal$	Challenge sent to cloud
Λ	Proof from cloud

Table 1: Notations

4. Construction

In this section, we present the overview and detail construction of proposed scheme.

4.1. ***Overview***

In our scheme, the user generates hash indices for each data block and sends indices to the fog for duplication check. Then Fog checks duplication locally and sends the hash indices to cloud for checking global duplication, if duplication does not exist in fog and cloud, fog permits user to upload data blocks along with tags. Upon receiving it, fog verifies tags and upload both to the cloud. In case duplication in exists in fog/cloud, then fog performs

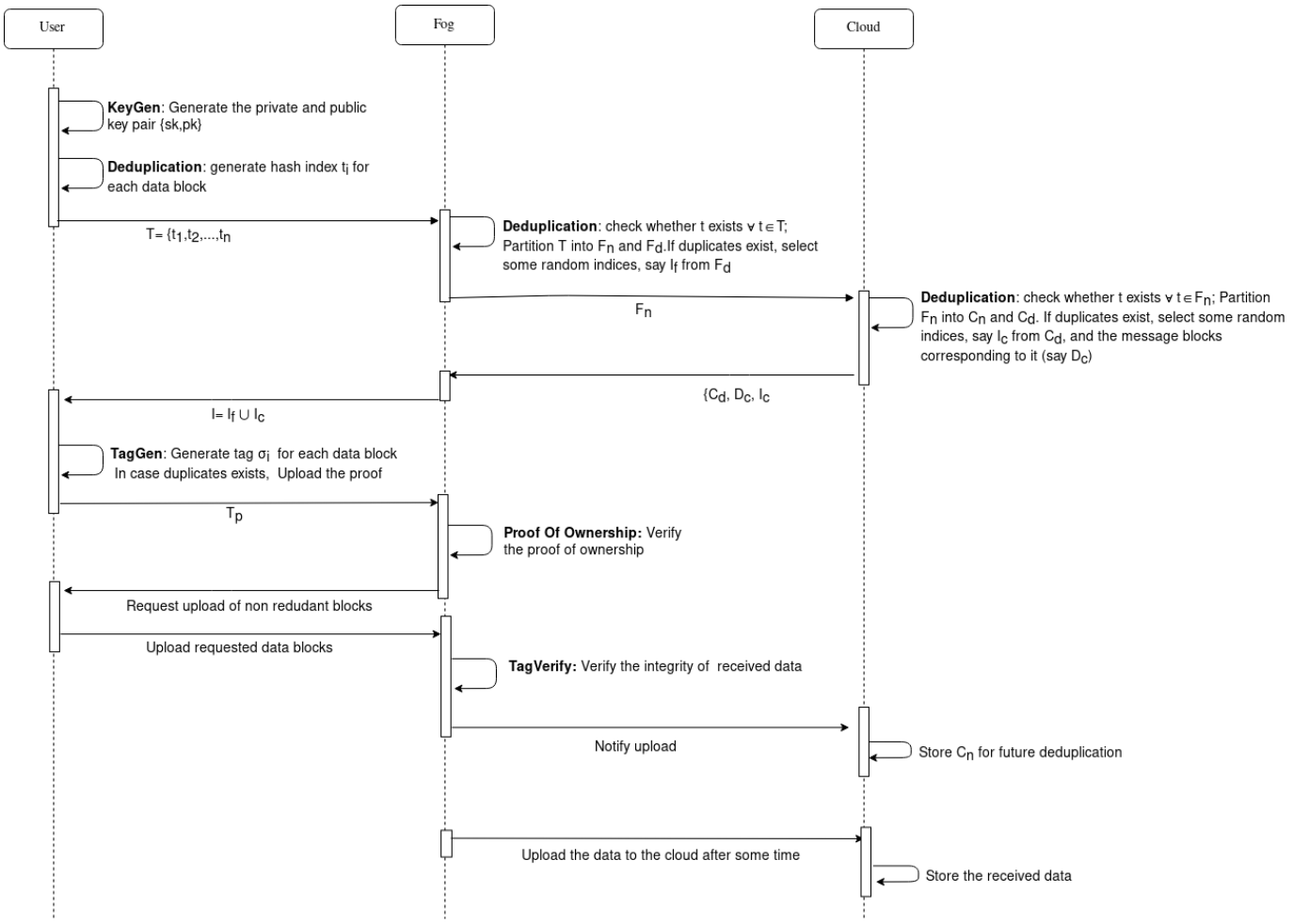

Figure 3: Deduplication Workflow

Proof of Ownership on the behalf of cloud (if required) to identify unauthorised users and sends the final report to cloud. Cloud stores the received information for future duplication's check. This is helpful in the case when some other user tries to upload same data to a different fog node. Further, to validate integrity of the data in cloud, TPA generates the arbitrary challenge and sends it to the cloud. Cloud generates and sends the proof to TPA for verification. finally verifies the integrity and send auditing results to user.

4.2. ***Detailed Construction***

Our proposed scheme construction consists of ten algorithms *Setup, KeyGen, IndexGen DeduplicationCheck, TagGen, TagVerify, Proof of Ownership, ChallengeGen, ProofGen* and *ProofVerify*.

1. **Setup:** Let G_1 and G_2 be two cyclic multiplicative groups of a large prime order and p be a large prime. Let $e : G_1 \times G_1 \rightarrow G_T$ be a bilinear map. Choose a generator g $\in$ G_1. Select two secure cryptographic hash function H_1: $\{0,1\}^* \rightarrow G_1$, H_2: $\{0,1\}^* \rightarrow Z_p^*$ and a secure indexing hash function h.
2. **KeyGen:** User selects a random number b $\in Z_p^*$ as it's private key (i.e., $sk = b$) and computes it's public key as $pk = g^b$
3. **IndexGen:** For each data block m_i, user computes the indices t_i as
$$t_i = h(m_i)$$
Then, user uploads index $T = \{t_1, t_2, ..., t_n\}$ to the fog for duplication check.
4. **DeduplicationCheck:**
After receiving indices T from user, Fog check for redundant blocks i.e. $\forall t \in T$, fog checks whether t already present or not in fog node or local network. In this duplication checking process two cases might occur:
   A **Duplicate not exist in Fog:** If duplication not exist in fog, it sends $F_n = T$ to cloud. Cloud receives F_n and $\forall t \in F_n$ then verifies whether t is already present or not in cloud. This will give arise to two cases:
      i **Duplicates not exists in cloud:** If duplicate data does not exists, cloud temporarily stores $C_n = F_n$ and notifies fog. Then Fog

asks the user to upload of all the data blocks along with their tags.

ii **Duplicates exists in cloud:** If duplicates exist in cloud, cloud segregates duplicate indices and non-duplicate indices as C_d and C_n respectively. Then cloud chooses some random indices $I_c = \{q_1, q_2, ..., q_x\}$ from C_d and sends $\{C_d, D_c, I_c\}$ to the fog for Proof of Ownership verification, where $D_c = \{m_i\}$, m_i is $q_i's$ corresponding data, $q_i \in I_c$. Cloud temporarily stores C_n for further use. Fog receives $\{C_d, D_c, I_c\}$ and forwards $I = I_c$ to the user for proof of corresponding blocks. Upon receiving I from fog, call proof of ownership algorithm.

B **Duplicate exist in Fog:** If duplication exists in fog, it segregates duplicate indices and non-duplicate indices as F_d and F_n respectively. It randomly chooses some indices $I_f = \{p_1, p_2, ..., p_y\}$ from F_d and $\forall p_i \in I_f$, it fetches $D_f = \{m_i\}$, m_i is p_i's corresponding data. Then Fog sends F_n to cloud for global duplication check. Cloud receives F_n and $\forall t \in F_n$, it verifies whether t is already present or not. Once again this will lead to two cases:

i **Duplicates not exists in cloud** Cloud temporarily stores $C_n = F_n$ and notifies fog. Fog sends $I = I_f$ to the user and asks proof for corresponding blocks.

ii **Duplicates exists in cloud:** If duplication exists in cloud also, Cloud segregates duplicate indices and non-duplicate indices as C_d and C_n respectively. Then cloud chooses some random indices $I_c = \{q_1, q_2, ..., q_x\}$ from C_d and sends $\{C_d, D_c, I_c\}$ to the fog for Proof of Ownership verification, where $D_c = \{m_i\}$, m_i is $q_i's$ corresponding data, $\forall q_i \in I_c$. Cloud temporarily stores C_n for further use. Fog receives $\{C_d, D_c, I_c\}$ and combine $I = I_f \cup I_c$ and send to the user for proof of corresponding blocks.

5. **TagGen:** In case duplication does not exist in fog or cloud, user generates tags for all the blocks. Here, we define message index x_i, $x_i > 0$ a non-random value which is unique to each message block m_i for a given user. We say, for the very first message block for the given user, the value of message index will be one i.e. $x_i = 1$. For the next block, the value of x_i will be 2 i.e. $x_i = 2$ and so on. For each block m_i, user compute tag using equation (1).

$$\sigma_i = (H_1(x_i) \cdot g^{H_2(m_i)})^{sk} \tag{1}$$

User uploads $\{\sigma_i, x_i, m_i\}$ to the fog.

6. **TagVerify:** For each received data block m_i and signatures σ_i, the fog first check it's integrity by checking the equation (2).

$$e(\sigma_i, g) \stackrel{?}{=} e(H_1(x_i) \cdot g^{H_2(m_i)}, pk) \tag{2}$$

If the result is *False*, the fog node will ask for retransmission of tag σ_i and data block m_i. Otherwise, Fog uploads data along with tags to cloud. Cloud permanently stores C_n for future deduplications.

7. **Proof of Ownership:**
In case duplicate exists, fog sends I to the user and asks proof for corresponding blocks.
Then, user sends $T_I = \{(\sigma_i, x_i)\}$ $\forall b_i \in I$ to fog, where σ_i is the corresponding tag and x_i is the corresponding index for b_i. Then fog verifies corresponding tags using equation (2).
If all the tags are verified, fog accepts proof and asks the user to upload only non-duplicate blocks i.e. blocks corresponding to $N = F_n - C_d$ indices along with all the tags. In case verification fails, fog requests upload of all the data blocks along with the tags. Cloud permanently stores C_n for future deduplications.

8. **ChallengeGen:** After uploading data blocks to cloud, user sends it's public key and maximum index value (index value of the message block last uploaded) to TPA. TPA generates the following parameters for integrity auditing

$$C = \text{Set of randomly selected } w \text{ indices}$$
$$R = \{r_1, r_2, ..., r_w\} \ ; \ 1 \leq j \leq w \ ; \ r_j \in Z_p^*$$
$$\theta = e(g, pk)$$

TPA sends $chal = \{C, R\}$ to cloud as challenge.

9. **ProofGen:** On receiving $chal = \{C, R\}$, cloud will generate proof as follows ($r_j \in R$):

$$\kappa = \prod_{j \in C} \sigma_j^{r_j}$$
$$\chi = \sum_{j \in C} (H_2(m_j) \cdot r_j)$$

Cloud sends the proof $\Lambda = \{\kappa, \chi\}$ to TPA for verification.

10. **ProofVerify:** After receiving the proof from cloud, TPA verifies the proof using the following equation

$$e(\kappa, g) \cdot (\theta^{-\chi}) \stackrel{?}{=} e\Big(\prod_{j \in C} H_1(j)^{r_j}, pk\Big) \tag{3}$$

5. Security Analysis

We provide proofs to justify the security of our algorithm and prove that our scheme is secure against forge, replace and replay attacks.

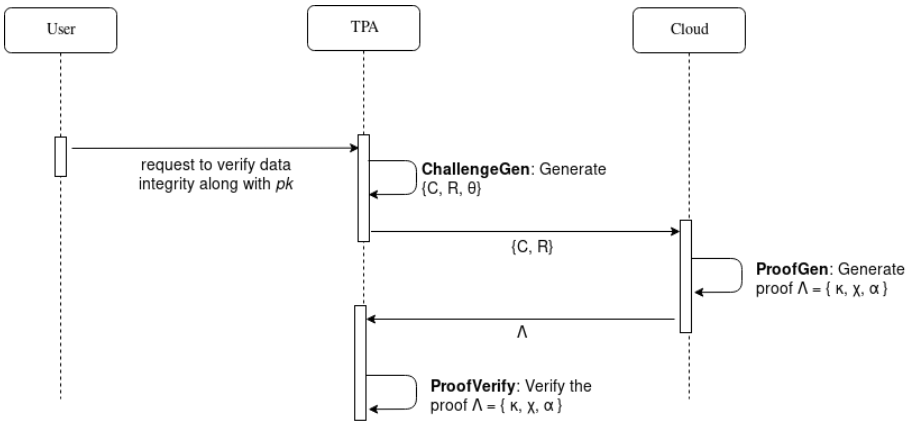

Figure 4: Integrity Auditing Workflow

(i) **(Theorem 1) Correctness:** *The construction of our scheme is correct*
Proof: We prove correctness of our scheme in two parts.

(a) *Correctness of tag verification:*
The proof of correctness of equation (2) is as follows:

$$\text{LHS} = e(\sigma_i, g)$$

$$= e\Big(\big(H_1(x_i) \cdot g^{H_2(m_i)}\big)^{sk}, g\Big)$$

$$= e\big(H_1(x_i) \cdot g^{H_2(m_i)}, g^{sk}\big)$$

$$= e\big(H_1(x_i) \cdot g^{H_2(m_i)}, pk\big)$$

$$= \text{RHS}$$

(b) *Correctness of proof verification by TPA:*
The correctness of the equation (3) can be demonstrated as follows:

$$\text{LHS} = e(\kappa, g) \cdot (\theta^{-\chi})$$

$$= e\Big(\prod_{j\in C} \sigma_j^{r_j}, g\Big) \cdot e\Big(g, g^b\Big)^{-\sum_{j\in C}(H_2(m_j)\cdot r_j)}$$

$$= \left(\frac{e\Big(\prod_{j\in C}(H_1(x_i) \cdot g^{H_2(m_j)})^{sk\cdot r_j}, g\Big)}{e\Big(g^{\sum_{j\in C}(H_2(m_j)\cdot r_j)}, g^b\Big)}\right)$$

$$= \left(\frac{e\Big(\prod_{j\in C} H_1(x_j)^{r_j} \cdot g^{H_2(m_j)\cdot r_j}, g^b\Big)}{e\Big(\prod_{j\in C} g^{(H_2(m_j)\cdot r_j)}, g^b\Big)}\right)$$

$$= e\Big(\prod_{j\in C} H_1(x_j)^{r_j}, g^b\Big)$$

Here, we know that x_i is the index of the message m_i. We also know that C is the set of randomly selected indices by the TPA, hence j ,$j \in C$ is also an index. As per our definition, index is unique for every message block, hence $j = x_i$. Therefore, the above equation could be reduced as

$$= e\Big(\prod_{j\in C} H_1(j)^{r_j}, pk\Big)$$

$$= \text{RHS}$$

(ii) **(Theorem 2) Unforgeability of Tags:** *It is computationally infeasible for any adversary A to forge tags in order to bypass any verification*
Proof: The tags in our proposal are generated using BLS signature scheme. We know that BLS signatures based HVT cannot be forged under CDH assumption, hence we claim that it is computationally infeasible for any adversary to forge tags as long as CDH assumption holds.

(iii) **(Theorem 3) Privacy preserving:** *From the given from cloud, TPA cannot get any content of the data.*
Proof: TPA receives two parameters $\{\kappa, \chi\}$ as proof. Since κ is computed using tags and tags are generated using secret key of user, TPA cannot obtain any information about message from tags. Also we know that

$$\chi = \sum_{j\in C}(H_2(m_j) \cdot r_j)$$

Here, we can see that cloud computes hash of the message blocks to generate χ. According to the property of collision resistance for secure hash function,

it is computationally infeasible to find an x such that $H(x) = H(m)$ for any data block m and any secure hash function H. Hence from the given proof, TPA cannot get any content from data stored in cloud i.e. TPA can know nothing about m.

(iv) **(Theorem 4) Immunity of forged attacks (Cloud):** *Cloud cannot forge proof in case it does not have data*
Proof: Cloud produces two parameters $\{\kappa, \chi\}$ to generate proof. In order to forge proof, cloud has to forge at least one of them. For computing κ, cloud uses tags, i.e.

$$\kappa = \prod_{j \in C} \sigma_j^{r_j}$$

Since tags cannot be forged according to Theorem 2, we argue that κ is unforgeable.
We prove unforgeability of χ by contradiction. Say cloud successfully forged $\hat{\chi}$ such that

$$\sum_{j \in C} (H_2(m_j) \cdot r_j) = \chi = \hat{\chi} = \sum_{j \in C} (H_2(\hat{m}_j) \cdot r_j)$$

where $\exists j \in C$ such that $m_j \neq \hat{m}_j$. Then,

$$\sum_{j \in C} (H_2(m_j) \cdot r_j) = \sum_{j \in C} (H_2(\hat{m}_j)$$

$$\implies \sum_{j \in C} (H_2(m_j) \cdot r_j) - \sum_{j \in C} (H_2(\hat{m}_j) \cdot r_j) = 0$$

$$\implies \sum_{j \in C} (H_2(m_j) - H_2(\hat{m}_j)) \cdot r_j = 0$$

which means $\forall j \in C$, $m_j = \hat{m}_j$ which contradicts our assumption. Hence we claim that χ is unforgeable.

(v) **(Theorem 5) Immunity to Replace Attacks** *Cloud cannot replace a corrupted data block and it's corresponding tag to bypass the verification*
Proof: Here, for generation of tag, we use index x_i which is unique to every message block for a given user. Say the cloud replaces a corrupted message block m_k and it's tag σ_k with a valid message block $\hat{m_k}$ and it's tag $\hat{\sigma_k}$, then according to ProofVerify formula,
LHS $= e(\kappa, g) \cdot (\theta^{-\chi})$

$$= e\Big(\prod_{j \in C} \sigma_j^{r_j}, g\Big) \cdot e\Big(g, g^b\Big)^{-\sum_{j \in C}(H_2(m_j) \cdot r_j)}$$

$$= \left(\frac{e\Big(\prod_{j \in C} H_1(x_j)^{r_j} \cdot g^{H_2(m_j) \cdot r_j}, g^b\Big)}{e\Big(\prod_{j \in C} g^{(H_2(m_j) \cdot r_j)}, g^b\Big)} \right)$$

$$= e\Big(\prod_{j \in C} H_1(x_j)^{r_j}, g^b\Big)$$

$$= e\Big(H_1(\hat{x_k})^{r_k} \prod_{j \in C, j \neq k} H_1(x_j)^{r_j}, g^b\Big)$$

$$= e\Big(H_1(\hat{x_k})^{r_k} \prod_{j \in C, j \neq k} H_1(j)^{r_j}, g^b\Big)$$

For Cloud to bypass authentication, LHS should be equal to RHS i.e.

LHS = RHS

$$\implies e\Big(H_1(\hat{x_k})^{r_k} \prod_{j \in C, j \neq k} H_1(j)^{r_j}, g^b\Big) = e\Big(\prod_{j \in C} H_1(j)^{r_j}, g^b\Big)$$

$$\implies e\Big(H_1(\hat{x_k})^{r_k} \prod_{j \in C, j \neq k} H_1(j)^{r_j}, g^b\Big) = e\Big(H_1(k)^{r_k} \prod_{j \in C, j \neq k} H_1(j)^{r_j}, g^b\Big)$$

$$\implies e\Big(H_1(\hat{x_k})^{r_k}, g^b\Big) = e\Big(H_1(k)^{r_k}, g^b\Big)$$

$$\implies H_1(\hat{x_k})^{r_k} = H_1(k)^{r_k}$$

$$\implies H_1(\hat{x_k}) = H_1(k)$$

We know that H_1 is a secure cryptographic hash function, hence it is safe to conclude that, $\hat{x_k} = k$. But, this is a contradiction as $\hat{x_k} \neq k$ since we replaced the message as well as tag, and $\hat{x_k}$ belongs to the replaced message (it is unique to every message block).
Hence our scheme is immune to replace attacks.

(vi) **(Therorem 6) Immunity to Replay attacks** *Cloud cannot pass verification using the proof generated previously*

Proof: For every challenge, TPA generates a different set of random numbers i.e. $R = \{r_1, r_2, ..., r_w\}$. then, cloud uses these random numbers in order to construct proof. Hence, cloud cannot reuse any generated proof for future verifications.

(vii) **(Theorem 7) Immunity to forged attacks (Adversary):** *Any adversary cannot get access to the data by forging Proof of Ownership*
Proof: We are using tags as a Proof of Ownership. According to Theorem 2, tags are unforgeable. Hence Proof of Ownership is unforgeable i.e. adversary cannot get access to data without owning it.

6. Performance Analysis

We have analyzed the performance of our scheme theoretically and analytically in this section.

6.1. Theoretical Analysis

6.1.1. Communication Costs:

Table 2 demonstrate the communication overheads of proposed scheme. For deduplication, user has to send all n hash indices to the fog. In the worst case, fog will send all the received indices to cloud (in case of no duplication in fog) for further deduplication. For proof of ownership, user will have to send proof i.e. tags for requested duplicate blocks which will be n in the worst case. During challenge, TPA sends w random indices and numbers as a challenge, which contributes to it's communication cost whereas cloud only sends three parameters $\{\kappa, \chi, \alpha\}$ as proof and hence achieves O(1) communication.

6.1.2. Computational Costs

Theoretical analysis for the computational costs for different algorithms of our scheme is shown in Table 3. Here, we can see that TagVerify and Proof of Ownership Verification have same cost per message block which is $(\hat{H}_1 + \hat{M} + \hat{H}_2 + 2\hat{E} + 2T_e)$. Cost of TagGen is less than cost for TagVerify by two bilinear pairing operation cost. Also, it can be concluded that ProofVerify overheads are significantly lower than ProofGen.

Table 2: Communication Costs

Deduplication	Challenge	Response
O(n)	O(w)	O(1)

Note: Here n is the total number of blocks to be uploaded; w number of challenged blocks

6.2. Experimental evaluation

We implement the prototype of our scheme with the help of Charm-Crypto library v0.43 which is python version of Pairing based Cryptograpgy (PBC). SS512 was used as the pairing group. Each node (user and fog) is simulated by Intel(R) Core(TM)i76500U CPU @ 2.50 GHz with 16.0 GB (15.9 GB usable) RAM. All the experimental values are average of 20 trials.

6.2.1. Computational costs for Tag generation and verification

Figure 5 and 6 demonstrates time for generating tags at users and verifying tags at fog nodes based on different number of data blocks and different block sizes respectively. We can clearly see that time required for tag verification in fog is more. This is due to the pairing operation used in the TagVerify.

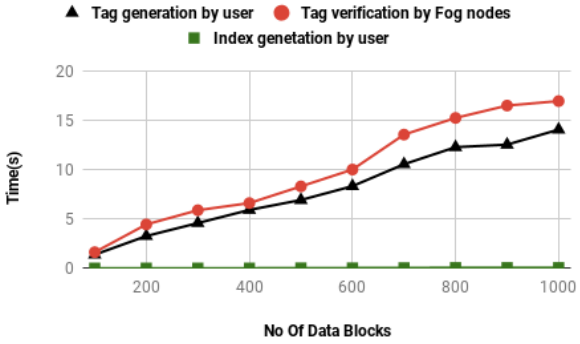

Figure 5: *Time for tag and index generation by users and verification for different blocks(block size 4kB)*

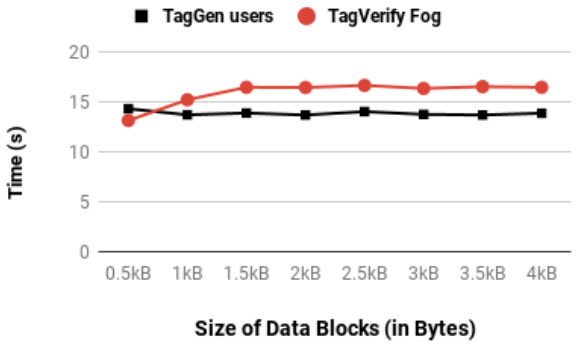

Figure 6: *Time for TagGen and TagVerify for different sizes of data blocks (No of blocks = 1000)*

6.2.2. Computational cost for deduplication

Experimental results for deduplication can be seen in Figure 7. It demonstrates lookup costs against number of blocks in fog and cloud for deduplication. The deduplication only requires computation of hashes by user and lookup of those hashes in cloud and fog. Clearly, cloud has significantly high lookup time than fog since it stores much

Table 3: Computational Costs

Function	Time
TagGen	$n \cdot (\hat{H_1} + \hat{M} + \hat{H_2} + 2\hat{E})$
Deduplication	$n \cdot (\hat{(h)} + \hat{L_f} + \hat{L_c})$
Proof of Ownership Verification	$n_2 \cdot (\hat{H_1} + \hat{M} + \hat{H_2} + 2\hat{E} + 2T_e)$
TagVerify	$n \cdot (\hat{H_1} + \hat{M} + \hat{H_2} + 2\hat{E} + 2T_e)$
ProofGen	$n_1 \cdot (2\hat{M} + \hat{E} + \hat{H_2} + \hat{A})$
ProofVerify	$3T_e + \hat{E} + n_1 \cdot (\hat{E} + \hat{H_1} + \hat{M})$

Note: Here n, n_1, n_2 are number of blocks to be uploaded, number of blocks for proof of ownership and number of challenged blocks by TPA respectively. $\hat{H_1}$ **and** $\hat{H_2}$ **are the average operation time for** H_1 **and** H_2 **respectively.** $\hat{M}$ and $\hat{A}$ are average time to multiply and add two elements from G_1 respectively. $\hat{E}$ is the average time to perform an exponentiation in G_1. $\hat{L_f}$ and $\hat{L_c}$ are the average lookup time in fog and cloud respectively. T_e is the average operation time for a bi-linear pairing.

larger amount of data (hence more number of indices to lookup). We conclude that deduplication cost for the our scheme depends upon the lookup algorithm used.

For our evaluation, storage was simulated with a python dictionary (*dict()*) and hash indices were used as keys. Proof of Ownership for our scheme is as good as verifying tags in fog. Hence we omit it's evaluation here.

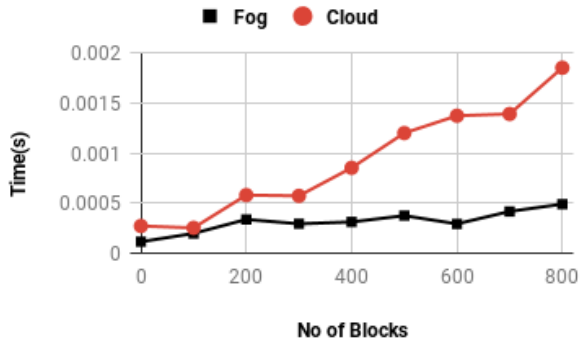

Figure 7: *Time for deduplication for different numbers of data blocks to be uploaded (block size 4kB)*

6.2.3. Computational costs for generation and verification of challenge

Figure 8 shows experimental results of the proof generation time in fog and proof verification time in TPA. Since cloud has to do a lot more computation than TPA, clearly the time taken by cloud increases significantly more than TPA as number of challenged blocks increases.

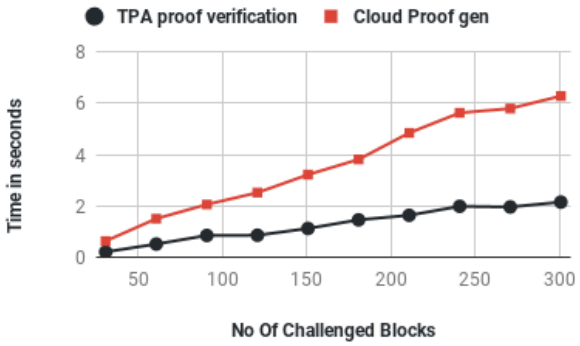

Figure 8: *Time for ProofGen by cloud and ProofVerify by TPA versus number of data blocks (block size 4kB)*

7. Conclusion and Future Works

In this paper, we achieved data deduplication using block level technique and data integrity through homomorphic verifiable tags. We used fog as a proxy for cloud to perform deduplication, and hence reduced communication overheads for users and computation overhead for the central cloud. Also, our scheme provides Proof of Ownership where it uses tags as a Proof of Ownership token, and successfully reduces further deduplication overheads. Our scheme also supports public integrity auditing with the help of trusted Third Party Auditor. Moreover, we masked the user data while construction of proof by cloud to ensure data and user privacy against TPA. Finally, we formally proved the security of our scheme and performance analysis demonstrated that our scheme is efficient.

Though our work provides an excellent solution for both integrity auditing and deduplication for fog based cloud storage in IoT, there are still some significant problems which needs further research. For example, supporting data privacy along with integrity and deduplication in IoT fog based cloud computing.

References

[1] P. P. Ray, "A survey of IoT cloud platforms," Future Comput. Inform. J., vol. 1, nos. 1–2, pp. 35–46, 2017

[2] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, Fog computing and its role in the Internet of things, in: MCC Workshop on Mobile Cloud Computing (MCC), 2012, pp. 13–16.

[3] Zhang, T., 2016. Fog Boosts Capabilities to Add More Things Securely to the Internet. http://blogs.cisco.com/innovation/fog-boosts-capabilities-to-add-more-thingssecurely-to-the-internet

[4] Roman, R., Lopez, J., Mambo, M., 2018. Mobile edge computing, fog et al.: a survey and analysis of security threats and challenges. Future Generat. Comput. Syst. 78, 680–698.

[5] Wang, C., Ren, K., Lou, W., Li, J., 2010. Toward publicly auditable secure cloud data storage services. IEEE Netw. 24, 19–24

[6] Gantz, J., Reinsel, D.: The digital universe decade-are you ready? http://www.emc.com/collateral/analyst-reports/idc-digital-uni verse-are-you-ready.pdf (2010)

[7] Zheng, Q., Xu, S.: Secure and efficient proof of storage with deduplication. In: Proceeding of ACM Conference on Data and Application Security and Privacy, pp. 1–12 (2012)

[8] Yuan, J.,Yu, S.: Secure and constant cost public cloud storage auditing with deduplication. In: IEEE Conference on Communications and Network Security, pp. 145–153 (2013)

[9] Alkhojandi, N., Miri, A., 2015. Privacy-preserving public auditing in cloud computing with data deduplication. In: 7th International Symposium on Foundations and Practice of Security, pp. 35–48

[10] J. Li, J. Li, D. Xie and Z. Cai, "Secure Auditing and Deduplicating Data in Cloud," in IEEE Transactions on Computers, vol. 65, no. 8, pp. 2386-2396, 1 Aug. 2016.

[11] Hou, Huiying , Yu, Jia , Zhang, Hanlin , Xu, Yan , Hao, Rong. (2018). Enabling secure auditing and deduplicating data without owner-relationship exposure in cloud storage. Cluster Computing.

[12] Hou, H., Yu, J., and Hao, R. (2019). Cloud storage auditing with deduplication supporting different security levels according to data popularity. Journal of Network and Computer Applications.

[13] Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., and Song, D. (2007). Provable data possession at untrusted stores. Proceedings of the 14th ACM Conference on Computer and Communications Security - CCS '07. doi:10.1145/1315245.1315318

[14] Erway, C., Küpçü, A., Papamanthou, C., Tamassia, R., 2009. Dynamic provable data possession. In: Proc. 16th ACM Conference on Computer and Communications Security. ACM, New York, NY, USA, pp. 213–222, https://doi.org/10.1145/1653662.1653688.

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

[16] Yang, K., Jia, X., 2013. An efficient and secure dynamic auditing protocol for data storage in cloud computing. IEEE Trans. Parallel Distr. Syst. 24, 1717–1726, https://doi.org/10.1109/TPDS.2012.278.

[17] Zhu, Y., Ahn, G.J., Hu, H., Yau, S.S., An, H.G., Hu, C.J., 2013. Dynamic audit services for outsourced storages in clouds. IEEE Trans. Serv. Comput. 6, 227–238

[18] Tian, H., Chen, Y., Chang, C.C., Jiang, H., Huang, Y., Chen, Y., Liu, J., 2017. Dynamic-hash-table based public auditing for secure cloud storage. IEEE Trans. Serv. Comput. 10, 701–714, https://doi.org/10.1109/TSC.2015.2512589.

[19] Wang, C., Chow, S.S.M., Wang, Q., Ren, K., Lou, W., 2013. Privacy-preserving public auditing for secure cloud storage. IEEE Trans. Comput. 62, 362–375.

[20] Z. Hao, S. Zhong, N. Yu, A privacy-preserving remote data integrity checking protocol with data dynamics and public verifiability, IEEE Trans. Knowl. Data Eng. 23 (9) (2011) 1432–1437.

[21] Y. Yu, et al., "Enhanced privacy of a remote data integrity-checking protocol for secure cloud storage," Int. J. Inf. Secur., vol. 14, no. 4, pp. 307–318, 2015

[22] Yu, Y., Au, M.H., Ateniese, G., Huang, X., Susilo, W., Dai, Y., Min, G., 2017. Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage. IEEE Trans. Inf. Forensics Secur. 12, 767–778.

[23] Wang, B., Li, B., Li, H., 2014. Oruta: privacy-preserving public auditing for shared data in the cloud. IEEE Trans. Cloud Comput. 2, 43–56.

[24] Wang, B., Li, B., Li, H., 2015. Panda: public auditing for shared data with efficient user revocation in the cloud. IEEE Trans. Serv. Comput. 8, 92–106.

[25] Yuan, J., Yu, S., 2015. Public integrity auditing for dynamic data sharing with multiuser modification. IEEE Trans. Inf. Forensics Secur. 10, 1717–1726.

[26] T. Jiang, X. Chen, J. Ma, Public integrity auditing for shared dynamic cloud data with group user revocation, IEEE Trans. Comput. 65 (8) (2016) 2363–2373.

[27] Tian, H., Nan, F., Jiang, H., Chang, C.C., Ning, J., Huang, Y., 2019. Public auditing for shared cloud data with efficient and secure group management. Inf. Sci. 472, 107–125,

[28] Tian, H., Nan, F., Chang, C.-C., Huang, Y., Lu, J., and Du, Y. (2019). Privacy-preserving public auditing for secure data storage in fog-to-cloud computing. Journal of Network and Computer Applications, 127, 59–69.

[29] Halevi, S., Harnik, D., Pinkas, B., Shulman-Peleg, A., 2011. Proofs of ownership in remote storage systems. In: 18th ACM Conference on Computer and Communications Security, pp. 491–500

[30] Pietro, R., Sorniotti, A., 2012. Boosting efficiency and security in proof of ownership for deduplication. In: 7th ACM Symposium on Information, Computer and Communications Security, pp. 81–82.

[31] .K. Ng, Y. Wen, H. Zhu, Private data deduplication protocols in cloud storage, in: Annual ACM Symposium on Applied Computing (SAC), 2012, pp. 441–446.

[32] M. Bellare, S. Keelveedhi, and T. Ristenpart, "Message-lockedencryption and secure deduplication," in Proc. Adv. Cryptol, 2013,pp. 296–312.

[33] J. Xu, E.-C. Chang, J. Zhou, Weak leakage-resilient client-side deduplication of encrypted data in cloud storage, in: ACM Symposium on Information, Computer and Communications Security (ASIACCS), 2013, pp. 195–206.

[34] M. Abadi, D. Boneh, I. Mironov, A. Raghunathan, and G. Segev, "Message-locked encryption for lock-dependent messages," in Proc. Adv. Cryptol., 2013, pp. 374–391.

[35] S. Keelveedhi, M. Bellare, T. Ristenpart, Dupless: Server-aided encryption for deduplicated storage, in: USENIX Security Symposium, 2013, pp. 179–194.

[36] J. Li, X. Chen, M. Li, J. Li, P. Lee, and W. Lou, "Secure deduplication with efficient and reliable convergent key management," IEEE Trans. Parallel Distrib. Syst., vol. 25, no. 6, pp. 1615–1625,Jun. 2014

[37] P. Puzio, R. Molva, M. Onen, S. Loureiro, Cloudedup: Secure deduplication with encrypted data for cloud storage, in: IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Vol. 1, 2013, pp. 363–370.

[38] P. Puzio, R. Molva, M. Önen, S. Loureiro, Block-level deduplication with encrypted data, Open J. Cloud Comput. (OJCC) 1 (1) (2014) 10–18.

[39] Yan, Z., Ding, W., Yu, X., Zhu, H., and Deng, R. H. (2016). Deduplication on Encrypted Big Data in Cloud. IEEE Transactions on Big Data, 2(2), 138–150.

[40] Koo, D., and Hur, J. (2018). Privacy-preserving deduplication of encrypted data with dynamic ownership management in fog computing. Future Generation Computer Systems, 78, 739–752.