Journal of Systems Architecture xxx (xxxx) xxx

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

Edge-based auditing method for data security in resource-constrained Internet of Things

Tian Wang [a,b], Yaxin Mei [a], Xuxun Liu [c], Jin Wang [d], Hong-Ning Dai [e], Zhijian Wang [f,*]

[a] College of Computer Science and Technology, Huaqiao University, Xiamen, China
[b] Artificial Intelligence and Future Networks, Beijing Normal University & UIC, Zhuhai, Guangdong, China
[c] School of Electronic and Information Engineering, South China University of Technology, Guangzhou, China
[d] School of Computer and Communication Engineering, Changsha University of Science & Technology, Changsha, China
[e] Faculty of Information Technology, Macau University of Science and Technology, Macau
[f] Information Science School, Guangdong University of Finance & Economics, Guangzhou, China

ARTICLE INFO

Keywords:
Data security
Audit model
Edge computing
Resource-constrained terminal
Binary tree

ABSTRACT

The explosive generation of Internet of Things (IoT) data calls for cloud service providers (CSP) to further provide more secure and reliable transmission, storage, and management services. This requirement, however, goes against the honest and curious nature of CSP, to the extent that existing methods introduce the third-party audit (TPA) to check data security in the cloud. TPA solves the problem of unreliable CSP but puts a heavy burden on lightweight users because of the sheer amount of the pre-audit data processing work. In this paper, we establish an audit model based on a designed binary tree assisted by edge computing, which provides computing capability for the resource-constrained terminals. The data pre-processing task is offloaded to the edge, which reduces computing load and improves the efficiency of task processing. We propose an improved correlation mechanism between data blocks and nodes on the binary tree so that all nodes on the binary tree can be fully utilized while existing methods use only leaf nodes and thus are required to establish multiple binary trees. Moreover, to improve audit efficiency, the binary tree in the audit process is designed to be self-balanced. In experiments, we compare our methods with the traditional method and experimental results show that the proposed mechanism is more effective to store and manage big data, which is conducive to providing users with more secure IoT services.

1. Introduction

Internet of things (IoT) has attracted the attention of academia and industry. One of its most important features is the mass of nodes. In addition to people and servers, devices and sensor networks are all components of IoT. At full working condition, the amount of data captured by machines and sensors increases explosively [1,2]. According to IDC's prediction, by 2020, we will generate more than 40 ZB of data. Due to the explosive growth of data volume, the traditional storage mode can no longer meet people's needs, which leads to the emergence of cloud storage. Cloud storage service is a derivative of cloud computing [3]. With this service, users could outsource data to the cloud service provider (CSP), which provides the infrastructure for effective maintenance of the IoT system [4–6]. Based on the cloud services, the quality of service (QoS) enjoyed by users has been effectively improved, including savings in computing resources, communications resources, and other resources [7,8].

What cannot be ignored is that while enjoying the benefits brought by cloud services, users also lose the physical control over their data, and data security issues frequently erupt [9,10]. One survey found that 43 percent of respondents had lost their outsourced data and had to recover it through other means [11,12]. In addition to data loss, data tampering also occurs frequently in reality. Data integrity has been considered as one of the key security issues in cloud storage. In the IoT system, data loss and tampering are small compared to the total amount of data stored, but researchers have found terrible consequences of data integrity being compromised, including huge property losses and even life threats [13,14]. Therefore, cloud security plays an important role in cloud services, and users need to ensure the integrity of their outsourced data. As users have lost control over the data, the method of encryption for data processing cannot be directly adopted for data security [15–17]. To prevent data security from being compromised by third parties, many secure storage schemes have been proposed,

* Corresponding author.
E-mail address: zjian@gdufe.edu.cn (Z. Wang).

https://doi.org/10.1016/j.sysarc.2020.101971
Received 30 June 2020; Received in revised form 20 October 2020; Accepted 7 December 2020
Available online 11 December 2020

among which one method that has aroused concern is to introduce the concept of third-party audit (TPA) to audit outsourced data. The TPA accepts the mandate of the data owner (DO), replaces the audit authority, and relieves the pressure on users [18]. Since then, many TPA-based auditing mechanisms have been proposed to meet different practical needs, such as dynamic auditing, privacy protection, etc. Erway et al. [19] proposed an outsourcing data audit mechanism that supports fully dynamic updates based on rank-based authentication skip lists. Wang et al. [11] proposed another verification scheme based on Merkel hash tree (MHT), which also supports public audit and dynamic data update with better efficiency. Based on the above two methods, there are many improved solutions, including [20–22].

Given the huge scale of the outsourced data and the limited resource capabilities of users, the task of auditing the integrity of data in the cloud is daunting and expensive. We can conclude that it is necessary to reduce the cost of CSP and users. To solve the problem that the existing scheme is too expensive to support the fully dynamic update, we designed an audit mechanism based on edge computing that has stronger computing power than the terminal [23–25]. Also, we remove the restriction of using leaf nodes only to store data blocks, which is a clear disadvantage in MHT. This method results in a longer depth of the tree. It is not conducive to search, and a node branch is too long after several new nodes are inserted. In this scheme, each file corresponds to a binary tree, and each node corresponds to a data block. After operating on data and changing its corresponding node, the height of the tree is readjusted according to the data block index number and the property of the self-balancing of the tree [26]. Compared with the existing schemes, our mechanism improves the QoS of users, reduces the storage overhead of CSP, and improves the efficiency of the audit process.

The contributions of this paper are mainly summarized as follows:

- Edge computing is integrated into the designed method, which shares the task of tag generation with the terminal and greatly shortens the generation time. Besides, audit binary trees are stored and maintained on the edge, which reduces the communication cost during the audit process.
- A correlation mechanism between binary tree nodes and data blocks is provided, which improves the utilization of nodes and reduces the number of trees that need to be constructed, thus greatly decreasing the storage cost of CSP.
- The element of binary tree self-balancing is added in the audit process. After the dynamic operation of data blocks, the search rate of the corresponding nodes of data blocks is accelerated due to the balance of the tree, and then the audit efficiency is improved.

The rest of this paper is organized as follows. In Section 2, some related work about the existing auditing methods is proposed. Then, to give readers a better understanding of our audit method, some preliminary knowledge is introduced in Section 3. The proposed model and design goals are proposed in Section 4 to clarify the significance of this paper. Details of the proposed method are introduced in Sections 5 and 6, which describes the dynamic structure and the process of auditing respectively. To demonstrate the validity of our method, security and experiment analysis are presented in Sections 7 and 8 respectively. Finally, the conclusion is shown in Section 9.

2. Related work

The development of cloud storage has aroused heated discussions in its various branches, among which the research on storage security has gradually become the mainstream. In the past few years, various methods have been proposed for security. Sangaiah et al. [27] proposed a method for privacy protection with machine learning techniques, but with high energy consumption. To improve energy consumption, they also proposed an energy-aware green adversary model, which reduces overall energy consumption by minimizing the communication and computing costs of each interaction [28]. These schemes only ensure the privacy of the data but ignore integrity. Audit schemes efficiently compensate for this shortcoming, which can be simply divided into private methods and public methods. The private method is that the audit is done by the users themselves, and the entities in the process are only data owners (DO) and CSP. Users complete the whole process and have a unique private key. Similar work can be found in [29–33] and so on.

The traditional method of data integrity audit is not adapted to the cloud storage environment. The audit work done by the local consumes too many resources, which is against the limited computing power of the terminal. Therefore, most of the current research programs support public auditing, which allows TPA to check data integrity. Besides, there is another way to check the integrity of outsourced data called “retrievable of proof" which has been widely used [34]. In practice, the user’s private information may be exposed to TPA. In other words, TPA may be dishonest [35]. Fu et al. [36] observed this fact and proposed a public audit mechanism based on constructing homomorphic verifiable group signatures and introduced a new entity, group manager. Multiple group managers restrict each other to ensure security. However, in this method, management resources are consumed too much and audit efficiency is sacrificed. To improve weaknesses, this method has been re-discussed in [37]. Wu et al. combined the group signature with an authentication message on the same platform to generate data block signatures. Random masking technology is introduced to ensure the anonymity of user identity and not affected by TPA. Unlike [36], there is only one group manager in this mechanism. It is trusted by all groups and generates key pairs for each group. Compared with the research of Fu et al. the audit efficiency has been improved to some extent, which is reflected in the experiment. Nevertheless, based on previous studies, it is difficult to find such an entity trusted by all groups.

In reality, besides paying attention to the protection of privacy, another important application is the audit of dynamic data. In the audit system, users cannot only access data but also update data and ensure data integrity by the audit system. To achieve lightweight operation in the dynamic audit process, Yeh et al. proposed a fine-grained access control framework in [38], and improved MHT to make its variants suitable for fine-grained access control. This method not only supports dynamic auditing but also supports batch auditing. Tian et al. [39] proposed a tailor-made public audit scheme for data storage in a cloud-to-fog based Internet of Things scenario. To reduce communication during the verification phase, they designed a tag-transforming strategy based on the bilinear mapping technique to convert the tags generated by mobile sinks to the ones created by the fog nodes. The public-key encryption techniques they use, however, are so complex that the computing resources remain high and inefficient. Similar work is can be found in [29,40,41], etc., but the overall audit efficiency of this work still needs to be improved, and there is still some computational pressure for users locally.

As far as we know, the current audit work is mainly carried out between three entities, namely, DO, TPA and CSP, and edge computing is seldom used for assistance. In most of the work introducing edge computing, it is mostly used to fundamentally solve the problem of unreliable cloud storage, such as establishing a cloud edge collaborative storage model [42–44] and encrypting the data shared to the cloud [45,46]. Some other work in the state of the art that makes use of the edge to optimize auditing protocols is also distinguishing from our work, specifically, the phrase applied to audit is different. Although the efficiency of audit has improved in these works, the overhead of computing resources at the terminal is still linear with the amount of audit. When one faces with large-scale data, the resource-constrained terminal will incur the huge burden [47]. How to achieve a secure and efficient design for users to integrate these two important components for audit service remains an open challenging task. In this paper, we focus on edge computing, transfer most of the computing overhead to the edge and optimize MHT, which improves the overall audit efficiency and reduces the storage cost generated in the audit process, thus improving QoS for users.

3. The basic scheme

In this section, we firstly introduce the general data integrity detection system model. Then, the detailed description of the relevant concepts involved is given. Finally, some preliminary knowledge is given in the next discussion.

3.1. Basic model of detection

In this paper, the method of data integrity detection is improved on the traditional audit model. It consists of four elements, namely, DO, cloud service provider (CSP), third-party auditor (TPA), and the edge, distinct from the traditional 3-entities model as shown in Fig. 1. DO holds data and can choose CSP independently. CSP is the provider of cloud storage services. Meanwhile, it has powerful data processing capability to provide DO computing and storage services. TPA is a third-party organization that performs specific audit tasks instead of users to reduce burden. In the process of running the traditional system model, there are five major algorithms: $KenGen$, $TagGen$, $ChalGen$, $ProofGen$ and $Verify$.

- $KeyGen(\theta) \rightarrow (sk_F, ssk, spk)$. The key generation algorithm uses the non-public security parameter θ as input and outputs a pair of secret-public key (spk, ssk) of tags and file information encryption key sk_F. Then randomly select a number $\alpha \in \mathbb{Z}_p^*$ (specific explanations will be given later) to calculate $v = g^{\alpha}$ and $sk = (\alpha, ssk), pk = (v, spk)$.
- $TagGen(F, sk) \rightarrow Tag$. The tag generation algorithm takes the file F and the secret tag key sk as input and outputs Tag of F. Notice that, the tag generation algorithm takes effect for each file block m_i and generates a set of tags, that is, $Tag = \{t_i\}_{i\in[1,n]}$. n represents the total number of data blocks in file F.
- $ChalGen(F_{info}) \rightarrow Chal$. The challenge generation algorithm is a link in the Challenge–Response mechanism, which takes the abstract information of F as input, marked as F_{info} (it refers to the one-way hash function algorithm which calculates the output of fixed bits from any length of input message). Based on F_{info}, it outputs the challenge information $Chal$.
- $ProofGen(F, Tag, Chal) \rightarrow Proof$. The proof generation algorithm is the other link in Challenge–Response mechanism. The file F, a set of tags Tag and the challenge information $Chal$ as taken as input and the proof information $Proof$ is outputted.
- $Verify(Chal, Proof, pk, F_{info}, sk_F) \rightarrow 0$ or 1. The verification algorithm is based on the results of the previous four algorithms. It takes $Chal$, $Proof$, pk, F_{info}, and sk_F as input and outputs the final audit result 0 or 1 (normally, 0 stands for incomplete data and 1 stands for the opposite).

As shown in Fig. 1, the main audit process is divided into four steps. Firstly, DO submits the audit request to TPA and sends (F_{info}, sk, sk_F) to TPA and (F, Tag) to CSP. Secondly, TPA uses $ChalGen$ to challenge CSP. Thirdly, CSP sends proof information to TPA with $ProofGen$ and TPA validates $Proof$ information with $VerifyGen$. Finally, TPA feeds back the results to DO.

3.2. Basic definitions in the model

In this subsection, some preliminary cryptography knowledge applied to the audit process is reviewed [11].

1. **Bilinear Map**
   An important tool is bilinear pairing mapping, which is the basic function for data integrity verification. It can be expressed in this way: $e : G_1 \times G_1 \rightarrow G_2$. G_1 and G_2 are two multiplication groups with Prime Order P. e satisfies the following properties:
   • Computability. Mapping e is effective and there must be an algorithm to calculate the result.

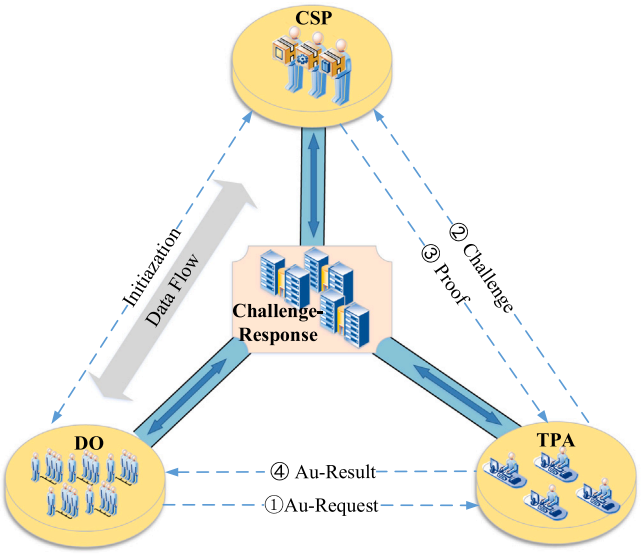

Fig. 1. Traditional system model.

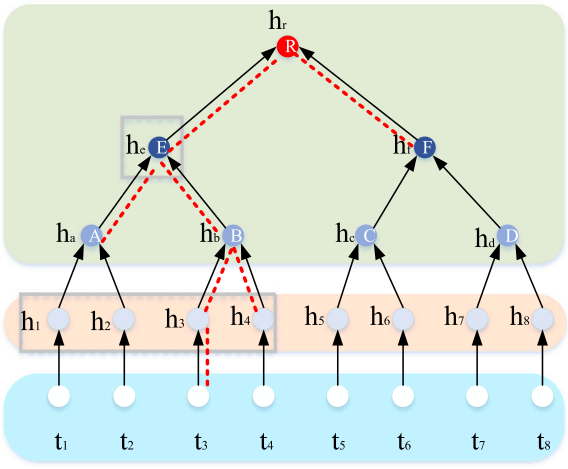

Fig. 2. The traditional MHT structure.

• Bilinearity. For $\forall h_1, h_2 \in G$, $a, b \in \mathbb{Z}_p^*$ ($\mathbb{Z}_p^* = x | x \leq \lfloor p \rfloor \cup x \geq 0$), the mapping e satisfies the relationship:

$$e(g_1^b, g_2^a) = e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}. \quad (1)$$

• Non-degeneracy. $e(g_1, g_2) \neq 1$, where g is a generator of group G.

2. **Computational Diffie–Hellman (CDH) Problem**
   For $x, y \in \mathbb{Z}_p^*$, $g^{xy} \in G_1$ is output based on the given g and the inputted g^x and g^y. The CDH assumption supports if the CDH problem in G_1 is computationally infeasible, it in G_1 is valid.
3. **Discrete Logarithm (DL) Problem**
   For $x \in \mathbb{Z}_p$, x is output based on the given g and the inputted g^x. The DL assumption supports if the DL problem in G_1 is computationally infeasible, it in G_1 is valid.

3.3. MHT

Merkle hash tree (MHT) is another important method to detect data integrity, and in this paper, we improve this method. In MHT, each leaf node represents the hash value of the data block tag, and the left and right leaf nodes are connected in two levels. According to this rule, we

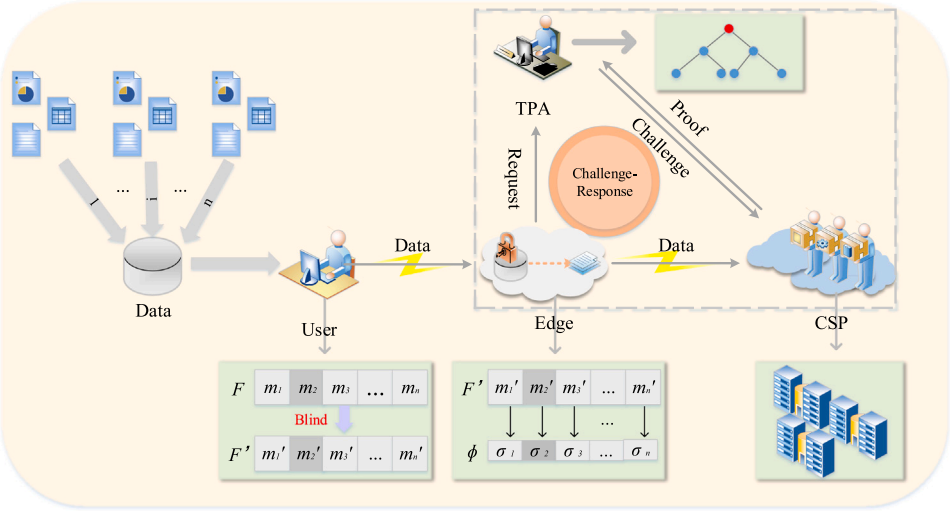

Fig. 3. The architecture of our proposed scheme.

cascade operations from the bottom of MHT, and finally, we can get the value of the root node, which we mark R. It can be seen that the tag value of the root node is based on the calculation results of the leaf nodes. When we verify the data integrity, if the data integrity, the value of the leaf node marker remains unchanged, and the root node R remains unchanged. For a deeper understanding, we give an example. As shown in Fig. 2, the hash value of E node is $h_e = h(h(h(t_1)\|h(t_2)) \| h(h(t_3)\|h(t_4)))$. If we want to verify the integrity of data block m_3, first we get the tag t_3 of m_3. Second, we need to calculate the corresponding leaf node h_3 of t_3 and the values of all sibling nodes between h_3 and root node, namely h_4, h_a and h_f. These values called AAL(auxiliary authentication information) are used to verify whether the value of root node R has changed, to verify the integrity.

4. System model and design goals

4.1. System model

The improved system model involves four different kinds of entities: the user (namely DO), the edge, the cloud and the TPA, which are shown in Fig. 3.

1. User: The entity that has a large amount of data but exceeds its storage capacity needs to store data in the cloud.
2. Edge: The edge is the entity trusted by other entities. It is in charge of interacting with the TPA and the cloud on behalf of the user and sending audit requests regularly to TPA.
3. Cloud: The cloud is responsible for providing the data storage service to the user. Based on the provided service, the user can outsource and share data.
4. TPA: TPA is the entity that checks data integrity on behalf of the user.

The real data is not available to any entity other than the user because it is blinded at the edge of the data upload phase, which ensures the security of intimate information. After receiving the blinded file from the user, the edge servers come into play, which is generating corresponding authenticators instead of users, which reduces the burden of users. Notice that, the extra transmission between edge servers and local devices is based on WLAN (Wireless Local Area Network) whose transmission rate is far higher than WAN (Wide Area Network). It can be seen that, the extra overhead is very little and almost negligible. Then the corresponding data generated and the blinded file are uploaded to the cloud. In our model, one of the highlights is that the user who does not send audit requests to TPA, but the edge who sends requests to TPA regularly, eliminating the workload of users and improving the overall efficiency. After receiving the request, the TPA sends the challenge to the cloud, and then, the cloud returns the proof of data possession. The result of the validity of the proof will be sent to the user finally.

4.2. Design goals

To improve audit efficiency and save entity computing and storage costs, our scheme needs to achieve the following goals:

1. Data security: to ensure that real data is not illegally accessed by entities other than users and cannot recover the real data from corresponding data blocks.
2. Storage cost reduction of AAL: to ensure the cloud does not need to store too much unnecessary information in the process of audit and the validity of proof can be still guaranteed.
3. Light-weight computation to the user: to support the low workload to the user, which guarantees the user does not need to perform high-computing operations, and the main work is to blind the data to prevent other entities from obtaining the real data and leaking it.
4. Efficient data dynamic operation: to support the cloud searches nodes at a low frequency when the user performs block-level operations on the file, which reduces the computation cost.
5. Auditing soundness: to guarantee whether data is stored correctly in the cloud is equivalent to TPA audit results.

5. Dynamic MHT-based structure

In the past work, MHT only uses leaf nodes to store corresponding block tags, but this method wastes resources greatly and makes the tree height too high. It increases the overhead of the authentication path in integrity verification and is not conducive to the maintenance of the tree in the dynamic operation of files. Thus, in this section, we propose two new detection methods based on MHT, which combine AVL and RBT respectively.

5.1. Hash Adelson–Velsky–Landis Tree

Hash Adelson–Velsky–Landis Tree (HAVL) is the method that based on Adelson–Velsky–Landis Tree (AVL) [48]. It is a self-balanced binary search tree, in which each node is balanced and can be described as

the depth difference between the left subtree and the right subtree is no more than 1. When adding, deleting, and modifying AVL, it may cause AVL to lose balance, but AVL can restore balance by rotating trees. This makes the structure of AVL greatly improve the efficiency of data block operation and the equilibrium of unbalanced binary trees can be restored by a finite number of rotations.

Also, we apply Henon chaotic mapping to the stored values of nodes, which can be expressed as

$$x_{n+1} = 1 - ax_n^2 + y_n \tag{2}$$

$$y_{n+1} = bx_n \tag{3}$$

Further, the formula can be rewritten to

$$x_{n+1} = 1 - ax_n^2 + bx_{n-1} \tag{4}$$

When a = 1.4 and b = 0.3, the mapping is chaotic, which is sensitive to initial values [49]. As long as the initial conditions are different, the later behavior of the mapping will change dramatically with iteration. Furthermore, unlike MHT, where only leaf nodes store block signatures. However, each node in HAVL stores block tags and we redefine the tag in the node next. The computation of tags is based on two kinetic equations, which belong to Henon Chaotic Mapping. Thus, it is suitable for us to detect the integrity of data tags.

We apply the Henon chaotic computation method to AVL tag computation, which can be divided into two cases: leaf nodes and non-leaf nodes.

- The storage value of leaf nodes: leaf nodes have no child nodes, so the stored value is still $hash(t_{m_i})$ like MHT, that is, the storage value of leaf nodes $X(m_i) = hash(t_{m_i})$.
- The storage value of non-leaf nodes: the storage value of the non-leaf node is calculated in two steps. Firstly, the transformation value of the sub-node of the non-leaf node is calculated, which is denoted as $Tran(m_i)$.

$$Tran(m_i) = 1 - aX_l^2 + bX_r \tag{5}$$

where X_l is the storage value of left subnode and X_r is the storage value of the right subnode. Next, the storage value of the node is calculated as

$$X(m_i) = hash(1 - aTran(m_i)^2 + bhash(t_{m_i})) \tag{6}$$

Thus, each node of HAVL is related to each other, and the root node is globally related. Using this structure, data integrity can be effectively checked. For the structure is balanced, AAI can be effectively reduced, but it is also difficult to maintain and suitable for checking the integrity of infrequently accessed files.

5.2. Hash Red Black Tree

Hash Red Black Tree (HRBT) is similar to HAVL in that it combines Henon chaotic map but based on Red Black Tree (RBT) [50]. Unlike AVL, RBT does not pursue complete balance, which can be interpreted as the height difference of nodes does not have to be within 1. It only requires partial balance but proposes to add color to nodes. RBT uses a non-strict balance to reduce the number of rotations when adding or deleting nodes. In other words, RBT is a black balanced binary tree, which maintains the balance of the binary tree through color constraints. AVL is a strictly balanced tree, so when adding or deleting nodes, according to different circumstances, the number of rotations is more than that of RBT. At worst, AVL trees rotate at most $O(logN)$ times, while RBT rotates at most three times. Therefore, RBT is easier to maintain than AVL for files with more operations, which is a weakly balanced binary tree. The method of integrity verification is similar to that of HAVL.

Table 1
Definitions of symbols.

Symbol	Definitions
ssk	System secret key
spk	System public key
sk	Secret key for the user
pk	Public key for the user
β_i	Blinding factor
ID	File identity
n	The total number od data blocks of the file
H	Hash mapping function
Θ	The set of tags
Ω	The set of challenge data blocks
μ	The number in G
λ_i	The elements in $\mathbb{Z}_p^*$
m_i	The data block of the file
m_i'	The blinded data block of the file
$m_i^{*'}$	The blinded data block that needed to be updated
F'	The set of m_i'

6. The proposed method

The drawbacks of the existing detection models are large amounts of computation, high consumption of resources and low efficiency, etc. To solve these problems, our improved audit model based on the edge layer in this section is proposed.

6.1. Notions

The symbols involved in the method and their descriptions are shown in Table 1.

6.2. Description of architecture

The effective implementation of the mechanism requires the guarantee of algorithms. In order to achieve the goals we designed, the audit process mainly consists of the following six algorithms and shown as Fig. 4.

(1) **Setup phase**

(i) Algorithm $KeyGen(\theta)$ is first executed to get the key, and then the blinded file is needed. $KeyGen(\theta)$ is the only algorithm that the user needs to execute, which realizes the lightweight calculation of the user. Specific implementation steps are described below. The user selects a key pair (spk, ssk) randomly and random number $\alpha \in \mathbb{Z}_p^*$. $\lambda = g^\alpha$ is needed to compute to get $sk = (\alpha, ssk)$ and $pk = (\lambda, spk)$. In the process of blinding data, the user selects a random seed $k_1 \in \mathbb{Z}_p^*$ to compute the blinding factor $\beta_i = f_{k_1}(i, ID)$. Thus, the blinded data block $m_i' = m_i + \beta_i$. Finally, the user sends the blinded data block m_i' to the edge.

(ii) After receiving m_i', $TagGen(F', sk)$ is executed in the edge, which can be used to construct HAVL/HRBT. Then the edge device randomly selects $\mu \in G$ so that $t = h(ID\|n\|\mu) \| Tag_{ssk}(h(ID\|n\|\mu))$ is marked as the tag for the file F', where ID represents the file identity and n represents the total number of data blocks in the file. Besides, the tag of the data block m_i' is also needed and can be worked out based on the formulation $Tag_i = (H(m_i' \times u^{m_i}))^\alpha$. The set of Tag_i is marked as Θ, namely, $\Theta = \{Tag_i\}(1 \le i \le n)$. With the result of $h(H(m_i))(1 \le i \le n)$ and Henon chaotic mapping, HAVL/HRBT could be constructed. Notice that, the root node need to be signed with the private key α, which can be expressed as $Tag_{sk}(R) = (R)^\alpha$. When all the above work are finished,

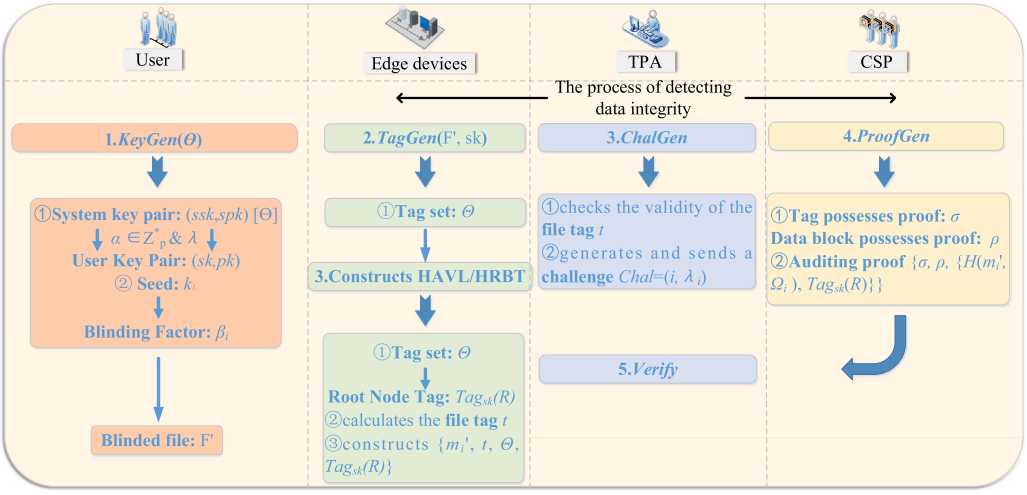

Fig. 4. The flowchart of architecture.

the edge constructs $\{m_i', t, \Theta, Tag_{sk}(R)\}$, transfers to the cloud and deletes the local file. Loop the files in the file list sequentially to get tag sets, and confirm the information to ensure that the files and information are stored correctly in the cloud.

(2) **Verification phase**

(i) *ChalGen*. Before TPA initiates challenge information to CSP, TPA first requires CSP to send the file tag t, verifies it with the public key pk generated by the user in the setup phase, returns 'True' if successful, and returns 'False' if failed ('True' means the file tag is intact and 'False' is on the contrary). *ChallGen* is performed based on 'True'. TPA selects c elements of the data block set randomly, that is, $I = (Q_1, Q_2, Q_3, \dots, Q_c), Q_1 \le \cdots \le Q_c$. For each $i \in I$, TPA chooses a random number $\lambda_i \in \mathbb{Z}_p^*$ to construct challenge information $Chal = \{(i, \lambda_i)\}_{Q_1 \le i \le Q_c}$ and sends to CSP. $\{i\}$ represents the location index of c challenge blocks and $\{\lambda_i\}$ enables the CSP to generate validation information.

(ii) *ProofGen*. When CSP receives the challenge information *Chal* from TPA, it executes *ProofGen* to generate evidence. For challenging data blocks, CSP calculates two values, namely, tag possesses proof σ and data block possesses proof ρ:

$$\sigma = \prod_{i=Q_1}^{Q_c} \sigma_i^{\lambda_i} \in G, \rho = \sum_{i=Q_1}^{Q_c} \lambda_i m_i' \in \mathbb{Z}_p^* \tag{7}$$

where m_i' is the ith blinded data block of file F' and σ_i is the corresponding tag of m_i'. In addition, the CSP also needs to provide the value of the cascade node associated with the challenge block, which is AAI and marked as $\{\Omega_i\}_{Q_i \le i \le Q_c}$. Then CSP returns proof to TPA, that is, $Proof\{\sigma, \rho, \{H(m_i'), \Omega_i\}_{Q_1 \le i \le Q_c}, Tag_{sk}(R)\}$.

(iii) *Verify*. After receiving the *Proof* from CSP, TPA executes *Verify* to check the validity. According to AAI, TPA reconstructs the binary tree firstly. Then the follows are checked:

$$e(Tag_{sk}(R), g) \stackrel{?}{=} e(R, g^{\alpha}) \tag{8}$$

$$e(\sigma, g) \stackrel{?}{=} e(\prod_{i=Q_1}^{Q_c} H(m_i')^{\lambda_i} \cdot \mu^{\rho}, \lambda) \tag{9}$$

The validation of Eq. (8) can be shown as follows:

$$\begin{aligned} e(Tag_{sk}(R), g) &= e(R^{\alpha}, g) \\ &= e(R, g^{\alpha}) \end{aligned} \tag{10}$$

The validation of Eq. (9) can be shown as follows:

$$\begin{aligned} e(TP, g) &= e(\prod_{i=Q_1}^{Q_c} \sigma_i^{\lambda_i}, g) \\ &= e(\prod_{i=Q_1}^{Q_c} (H(m_i') \cdot \mu^{m_i'})^{\alpha} \lambda_i, g) \\ &= e(\prod_{i=Q_1}^{Q_c} (H(m_i')^{\lambda_i} \cdot \mu^{m_i' \lambda_i})^{\alpha}, g) \\ &= e(\prod_{i=Q_1}^{Q_c} (H(m_i')^{\lambda_i} \cdot \mu^{\sum_{i=Q_1}^{Q_c} m_i' \lambda_i}), g^{\alpha}) \\ &= e(\prod_{i=Q_1}^{Q_c} H(m_i')^{\lambda_i} \cdot \mu^{\rho}, \lambda) \end{aligned} \tag{11}$$

If Eq. (8) fails in verification, 0 will be returned and it shows that $Tag_{sk}(R)$ given by CSP matches R under the condition of validation. What is more, AAI can also be determined to be perfect. Eq. (9) is used to verify whether the file is damaged. Thus we can get the truth that TPA only returns 1 when all the above two validations pass, and the rest returns 0, which confirms auditing soundness.

(3) **Update phase**

Obviously, updating the data includes modifying, deleting, and adding. It can be seen that modifying data does not affect the tree structure but adding and deleting data will. Thus, through left-handed and right-handed, HAVL can maintain balance and HRBT can maintain week balance, which improves the efficiency of checking data integrity. The specific steps of data modification are shown below:

(i) *Prepare the information needed to be updated*. Assuming that the user wants to update m_i to m_i^*. The task the user needs to do is to blind m_i^* and transfers the relevant information to the edge. After receiving the information, the edge generates the tag of m_i^*, $\sigma_i = (H(m_i^{*'}) \cdot \mu^{m_i^{*'}})^{\alpha}$ and modification information $update = (i, m_i^{*'}, \sigma_i^*)$, which will be sent to the cloud.

(ii) *ExeUpdate*. CSP updates the modified data block $m_i^{*'}$, the corresponding tag σ_i^* and HAVL/HRBT. With the modified information, the new root node R^* could be generated. During the update process, if the CSP itself caches some required nodes, the update efficiency of the binary tree

will be improved. Then, CSP regenerates the verification information $Proof_{update} = (\Omega_i, H(m_i'), Tag_{sk}(R), R')$ and sends to TPA. TPA recalculates the root node R^* with $\{\Omega_i, H(m_i')\}$ and verify $e(Tag_{sk}(R), g) \stackrel{?}{=} e(R, g^{\alpha})$. If the result is true, TPA verifies the operation of updating based on $H(m_i^{*'})$ that computed in the edge. After confirming the node update, the edge signs the root node R^* and sends it to the CSP update.

7. Security analysis

The security of our proposed method is built on the truth that there is no valid method to solve the discrete logarithm problem on elliptic curves and the features of CDH. In our proposed method, violations of CSP can be effectively prevented. Assuming the situation that instead of deleting outdated information, the CSP returns outdated $\{H(m_i'), \Omega_i\}_{Q_1 \le i \le Q_c}$ and $Tag_{sk}(R)$ to TPA when performing GenProof. TPA could determine whether the CSP returned outdated validation information in the phase of $Verify$. The method is to work out whether the reconstructed root node R is consistent with the R^* stored at TPA. Compared to the traditional method, the edge uses data blocks m_i' directly to calculate $\sigma = (H(m_i') \cdot u^{m_i'})^{\alpha})$ instead of using the BLS scheme to get $\sigma_i = H(m_i)^{\alpha}$, which is based on the consideration that the file may have been deleted by the CSP but retains the data block tags. Notice that, the storage cost of tags of data blocks is smaller than data blocks. This method can save costs and avoid integrity detection. Moreover, there is a situation like this that the TPA cannot control the behavior of the CSP for the reason that in the phase of $Verify$, TPA does not require any data blocks to participate in and only needs tags of data blocks to operate. For the reason that maybe tags of data are in integrity, and the verification operation can still proceed smoothly, but data blocks have been deleted by the CSP. Thus, in the phase of $TagGen$, data blocks must be incorporated into the calculation. Based on this premise, in the phase of $ProofGen$, $\{m_i'\}_{Q_1 \le i \le Q_c}$ also needs to be plugged in to compute $\rho = \sum_{i=Q_1}^{Q_c} \lambda_i m_i' \in \mathbb{Z}_p^*$. At this point, CSP cannot delete data block freely.

We can also consider a worst-case scenario in which TPA colludes with CSP. For the reason that: (1) the data blocks CSP received are blinded; (2) the edge avoids users from contacting with other entities and the user does not need to process the validation results returned by TPA; (3) the user's blinding factor and private key are not exposed to TPA. It can be seen that our method is security based on the premise.

8. Experiments evaluations

In this section, we compare the computational cost, communication cost, and storage cost of the proposed method with the traditional MHT detection method, and prove the feasibility of the method.

8.1. Experimental environment

We evaluate the performance of the proposed scheme by several experiments and these experiments run on a Linux OS machine with an Intel Pentium 2.30 GHz processor and equipped with 8GB of RAM. In the course of our research, C language is used with the free Pairing-Based Cryptography (PBC) Library [51] and the GNU Multiple Precision Arithmetic (GMP) [52]. Notice that, the base field size is set to 512 bits, $|p| = 160$ bits (an element in $\mathbb{Z}_p^*$) and the size of test file is set to be 20 MB.

8.2. Experimental performance analysis

In this subsection, the experimental performance of our proposed schemes is introduced. Firstly, we give the analysis of the computational and communication overhead of the proposed method and compare it with MHT in storage cost, which is the most representative among the tree-based auditing methods. Then we give the analysis of the experimental results.

Table 2
The computation complexity in different phases.

	DO	Edge	TPA	CSP
Data blinding	$O(n)$	–	–	–
TagGen	–	$O(n)$	–	–
ChalGen	–	–	$O(c)$	–
ProofGen	–	–	–	$O(c)$
Verify	–	–	$O(c)$	–

8.2.1. Computation cost and computation complexity

In the proposed method, there are four entities, namely, DO, TPA, the edge, and CSP. For DO, the main computation overhead is concentrating on the phase of setup. The $KeyGen$ costs $\mathbb{E} + \mathbb{M}$, in which $\mathbb{E}$ denotes the computation of exponentiation operation in G_1 and $\mathbb{M}$ has the same meaning but refers to multiplication operation. For the edge, the $TagGen$ costs $n(\mathbb{H} + \mathbb{M} + 2\mathbb{E})$, in which $\mathbb{H}$ denotes the computation of $Hsah$ operation in G_1.

For TPA, the main computational overhead is generating challenge information and verifying the validity of the equation. However, generating challenge information requires only a random selection of values, which consumes a little computing resources, so we ignore it here. The other costs $2Pair + (c+1)\mathbb{E} + c\mathbb{M} + c\mathbb{H}$, in which $Pair$ denotes the pairing operation. For CSP, the computation overhead is $(c-1)\mathbb{M} + c\mathbb{E} + c\mathbb{M}_{\mathbb{p}} + (c-1)\mathbb{A}_{\mathbb{p}}$, where $\mathbb{M}_{\mathbb{p}}$ and $\mathbb{A}_{\mathbb{p}}$ denote the multiplication operation and the addition operation in $\mathbb{Z}_{\mathbb{p}}$ respectively.

The computation complexity of four entities in different phases is shown in Table 2. Notice that, the total number of data blocks is marked as n and c is the number of challenged data blocks. Clearly, it turns out that computation complexities $Datablinding$ and $TagGen$ are both $O(n)$. The remaining steps $ChalGen$, $Verify$ and $ProofGen$ are all $O(c)$.

8.2.2. Communication cost

According to the previous description, we can easily get that communication overhead mainly comes from the challenge–response mechanism, which is focusing on TPA and CSP. In this phase, TPA first needs to generate challenge information $Chal = (i, \lambda_i)$, including $c \cdot (|n| + |p|)$ bits, in which $|p|$ is the length of one element of $\mathbb{Z}_{\mathbb{p}}$. Then, TPA sends challenge information to CSP, which generates $Proof$ when it receives challenge information and the size of $Proof$ is $|p| + |q|$, in which $|q|$ is the size of G_1. Thus, based on the above analysis, the communication of challenge–response mechanism is $c \cdot |n| + (c+1) \cdot |p| + |q|$.

8.2.3. Storage cost

Compared with other MHT-based schemes, our scheme changed the audit model. In improved structure, each node in the tree corresponds to a data block, while only leaf nodes correspond to the corresponding data block in the traditional scheme. For example, when we need to store information about n data blocks, HAVL or HRBT only contains n nodes, while other MHT-based schemes need $2n-1$ nodes. From this, we can conclude that the proposed scheme can greatly reduce the storage overhead.

8.3. Experimental performance analysis

In this subsection, the performance of our proposed scheme is evaluated by computing resource consumption in different steps and methods. Different from other existing schemes, the task of generating tags in our scheme is accomplished by edge devices and the only thing that DO needs to do is blindly processing the data before it is sent to the edge. Our experiments show the process of data preprocessing. As shown in Fig. 5(a), it can be seen that the time cost of tag generation is much higher than the time cost of data blindness. Furthermore, with the increase of data blocks, the time gap between the generation of tags and data blindness gradually increases, which demonstrates the

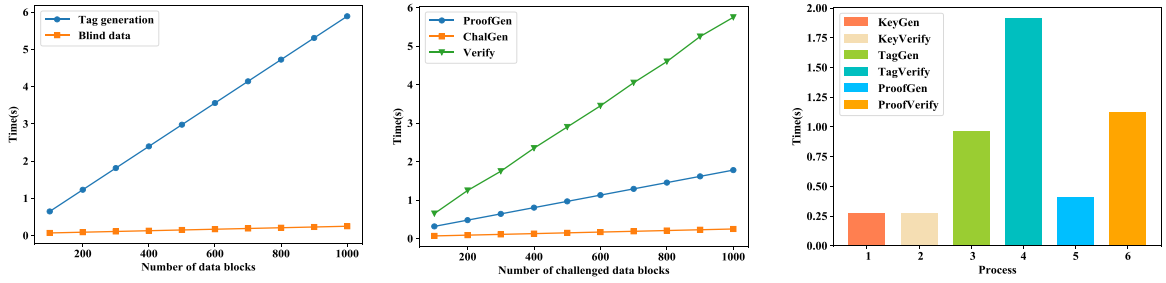

Fig. 5. (a) Comparison of the time cost for DO between blinding data and generating tags. (b) The computation cost of the challenge–response mechanism with different number of challenged blocks. (c) Performance of different processes.

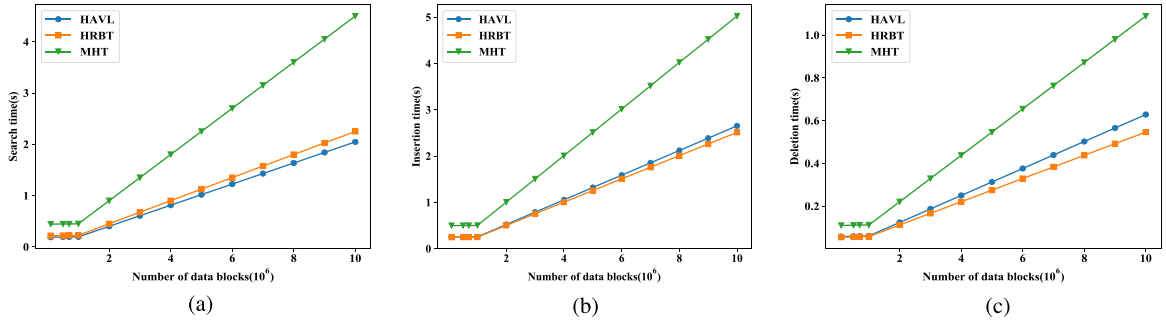

Fig. 6. (a) Search time under different data blocks. (b) Insertion time under different data blocks. (c) Deletion time under different data blocks.

effectiveness of sharing tasks to the edge that consumes less computing resources to DO and overcomes the shorting of resource-limited terminals.

We know that the audit process is mainly a challenge–response mechanism. In order to effectively evaluate the mechanism we proposed, as shown in Fig. 5(b), we show the computational cost of three steps performed on TPA and CSP respectively in the audit phase. In the experiment, the change in the number of challenge blocks is tested and the range is from 100 to 1000. From the experimental results, we can conclude that the computational cost of generating challenge information can be neglected, which just like what we analyzed in the last part. However, the process of generating authentication messages by CSP is slightly longer, and the computing cost increases from 0.18 s to 1.79 s as the number of challenge blocks increases. It is easy to see that the computation time of verification is much longer than the first two stages. Therefore, handling over the most time-consuming process to TPA greatly saves the cost of DO.

In addition to the audit process, in order to evaluate the performance of different processes more effectively, we compared them in the experiment. Notice that, in our experiment, the number of data blocks is set to 100. As shown in Fig. 5(c), the key generation time and verification time are almost the same, but the generation and verification time of tags and the proof are quite different, among which the tag authentication time is the longest, so shortening the tag authentication time as far as possible is one of our next work.

In the next experiment, as shown in Fig. 6, we compared the proposed methods to the time consumption of the typical MHT in dynamic operations. Specifically, we evaluate the computation time on the Y-axis of a given block relative to the number of blocks on the X-axis. Fig. 6(a) shows the searching time relative to the block. Obviously, our method has greatly improved MHT, and HAVL is the most effective because a fully balanced binary tree is applied in this method, the distribution of child nodes is more regular and the depth is smaller, which is easy to search. In addition, we can see that as the number of blocks increases, the gap between several methods becomes larger, which is largely due to the effect of binary tree depth. Fig. 6(b) shows another operation, inserting new data, in which several different forms of line representation are the same as in the previous experiment. Obviously, our method still has a great improvement on MHT, but the difference is that HRBT performs better in the case of insertion. During the data update phase, HRBT is more flexible with respect to the requirement of the balance of HAVL, resulting in less time spent on tree adjustment when new data is inserted. Similarly, Fig. 6(c) shows that as the number of data blocks increases, so does the time required for deletion, and HRBT performs better than HAVL.

9. Conclusion

For the frequent occurrence of outsourcing data security problems, we propose a lightweight and trustworthy audit method based on edge computing in this paper, which not only addresses the contradiction between the limited computing power of the terminal and the huge computing tasks but also protects the data security in the audit process and improves the audit efficiency. In our method, the edge plays a central role, which processes the blind files uploaded by users to ensure the privacy of the files, undertakes the initial phrase of tag generation with high overhead, and reduces the resource overhead of lightweight user devices. Besides, in our method, the traditional MHT is extended and multi-node storage is introduced to save the storage cost. The experimental performance analysis demonstrates that our proposed edge-based method achieves safety and efficiency as expected.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Above work was supported in part by Natural Science Foundation of Fujian Province of China (No. 2020J06023 and No. 2018J01092), the

Major Project of Basic and Applied Research in Guangdong Universities under Grant 2017WZDXM012, National Natural Science Foundation of China (NSFC) under Grant No. 61872154 and No. 61772148 and the Fujian Provincial Outstanding Youth Scientific Research Personnel Training Program.

References

[1] X. Liu, P. Lin, T. Liu, T. Wang, A. Liu, W. Xu, Objective-variable tour planning for mobile data collection in partitioned sensor networks, IEEE Trans. Mob. Comput. (2020) http://dx.doi.org/10.1109/TMC.2020.3003004.

[2] T. Wang, L. Qiu, A.K. Sangaiah, A. Liu, M.Z.A. Bhuiyan, Y. Ma, Edge-computing-based trustworthy data collection model in the internet of things, IEEE Internet Things J. 7 (5) (2020) 4218–4227.

[3] Z. Gu, M. Qiu, Introduction to the special issue on embedded artificial intelligence and smart computing, J. Syst. Archit. 84 (2018) http://dx.doi.org/10.1016/j.sysarc.2018.01.004.

[4] J. Wang, Y. Yang, T. Wang, R.S. Sherratt, J. Zhang, Big data service architecture: A survey, J. Internet Technol. 21 (2) (2020) 393–405, http://dx.doi.org/10.3966/160792642020032102008.

[5] Y. Luo, K. Yang, Q. Tang, J. Zhang, P. Li, S. Qiu, An optimal data service providing framework in cloud radio access network, EURASIP J. Wireless Commun. Networking (2016) http://dx.doi.org/10.1186/s13638-015-0503-2.

[6] T. Wang, Z. Cao, S. Wang, J. Wang, L. Qi, A. Liu, M. Xie, X. Li, Privacy-enhanced data collection based on deep learning for internet of vehicles, IEEE Trans. Ind. Inf. 16 (10) (2020) 6663–6672.

[7] W. Li, Z. Chen, X. Gao, W. Liu, J. Wang, Multimodel framework for indoor localization under mobile edge computing environment, IEEE Internet Things J. 6 (3) (2019) 4844–4853.

[8] Y. Wu, H. Huang, Q. Wu, A. Liu, T. Wang, A risk defense method based on microscopic state prediction with partial information observations in social networks, J. Parallel Distrib. Comput. 131 (2019) 189–199, http://dx.doi.org/10.1016/j.jpdc.2019.04.007.

[9] M. Du, K. Wang, Y. Chen, X. Wang, Y. Sun, Big data privacy preserving in multi-access edge computing for heterogeneous internet of things, IEEE Commun. Mag. 56 (8) (2018) 62–67.

[10] M. Du, K. Wang, Z. Xia, Y. Zhang, Differential privacy preserving of training model in wireless big data with edge computing, IEEE Trans. Big Data 6 (2) (2020) 283–295.

[11] Q. Wang, C. Wang, K. Ren, W. Lou, J. Li, Enabling public auditability and data dynamics for storage security in cloud computing, IEEE Trans. Parallel Distrib. Syst. 22 (5) (2011) 847–859, http://dx.doi.org/10.1109/TPDS.2010.183.

[12] Q. Tang, K. Yang, D. Zhou, Y. Luo, F. Yu, A real-time dynamic pricing algorithm for smart grid with unstable energy providers and malicious users, IEEE Internet Things J. 3 (4) (2016) 554–562.

[13] D. Yu, Y. Jin, Y. Zhang, X. Zheng, A survey on security issues in services communication of microservices-enabled fog applications, Concurr. Comput.: Pract. Exper. 31 (22) (2019) e4436, http://dx.doi.org/10.1002/cpe.4436.

[14] J. Zhang, W. Wang, X. Wang, Z. Xia, Enhancing security of fpga-based embedded systems with combinational logic binding, J. Comput. Sci. Tech. 32 (2) (2017) 329–339.

[15] J. Zhang, S. Zhong, T. Wang, H.-C. Chao, J. Wang, Blockchain-based systems and applications: A survey, J. Internet Technol. 21 (1) (2020) 1–14.

[16] S. He, W. Zeng, K. Xie, H. Yang, M. Lai, X. Su, Ppnc: Privacy preserving scheme for random linear network coding in smart grid, KSII Trans. Internet Inf. Syst. 11 (3) (2017) 1510–1532.

[17] K. Gu, L. Yang, B. Yin, Location data record privacy protection based on differential privacy mechanism, Inf. Technol. Control 47 (4) (2018) 639–654.

[18] Y. Wu, H. Huang, N. Wu, Y. Wang, M.Z.A. Bhuiyan, T. Wang, An incentive-based protection and recovery strategy for secure big data in social networks, Inform. Sci. 508 (2020) 79–91, http://dx.doi.org/10.1016/j.ins.2019.08.064.

[19] C.C. Erway, A. Küpçü, C. Papamanthou, R. Tamassia, Dynamic provable data possession, ACM Trans. Inf. Syst. Secur. 17 (4) (2015) http://dx.doi.org/10.1145/2699909.

[20] J. Xu, L. Wei, Y. Zhang, A. Wang, F. Zhou, C. zhi Gao, Dynamic fully homomorphic encryption-based merkle tree for lightweight streaming authenticated data structures, J. Netw. Comput. Appl. 107 (2018) 113–124, http://dx.doi.org/10.1016/j.jnca.2018.01.014.

[21] W. Shen, J. Qin, J. Yu, R. Hao, J. Hu, Enabling identity-based integrity auditing and data sharing with sensitive information hiding for secure cloud storage, IEEE Trans. Inf. Forensics Secur. 14 (2) (2019) 331–346, http://dx.doi.org/10.1109/TIFS.2018.2850312.

[22] M. Long, F. Peng, H. Li, Separable reversible data hiding and encryption for hevc video, IEEE Trans. Syst. Man Cybern.: Syst. 14 (1) (2018) 171–182.

[23] Y. Wang, K. Wang, H. Huang, T. Miyazaki, S. Guo, Traffic and computation co-offloading with reinforcement learning in fog computing for industrial applications, IEEE Trans. Ind. Inf. 15 (2) (2019) 976–986.

[24] K. Gu, W. Jia, C. Jiang, Efficient identity-based proxy signature in the standard model, Comput. J. 58 (4) (2015) 792–807.

[25] B. Xiong, K. Yang, J. Zhao, K. Li, Robust dynamic network traffic partitioning against malicious attacks, J. Netw. Comput. Appl. 87 (2017) 20–31, http://dx.doi.org/10.1016/j.jnca.2016.04.013.

[26] J.J. Hasbestan, I. Senocak, Binarized-octree generation for cartesian adaptive mesh refinement around immersed geometries, J. Comput. Phys. 368 (2018) 179–195, http://dx.doi.org/10.1016/j.jcp.2018.04.039.

[27] A.K. Sangaiah, D.V. Medhane, T. Han, M.S. Hossain, G. Muhammad, Enforcing position-based confidentiality with machine learning paradigm through mobile edge computing in real-time industrial informatics, IEEE Trans. Ind. Inf. 15 (7) (2019) 4189–4196.

[28] A.K. Sangaiah, D.V. Medhane, G. Bian, A. Ghoneim, M. Alrashoud, M.S. Hossain, Energy-aware green adversary model for cyberphysical security in industrial system, IEEE Trans. Ind. Inf. 16 (5) (2020) 3322–3329.

[29] K. Gu, W. Jia, G. Wang, S. Wen, Efficient and secure attribute-based signature for monotone predicates, Acta Inform. 54 (5) (2017) 521–547.

[30] T. Wang, H. Luo, X. Zeng, Z. Yu, A. Liu, A.K. Sangaiah, Mobility based trust evaluation for heterogeneous electric vehicles network in smart cities, IEEE Trans. Intell. Transp. Syst. (2020) http://dx.doi.org/10.1109/TITS.2020.2997377.

[31] S. Wan, R. Gu, T. Umer, K. Salah, X. Xu, Toward offloading internet of vehicles applications in 5g networks, IEEE Trans. Intell. Transp. Syst. (2020) http://dx.doi.org/10.1109/TITS.2020.3017596.

[32] A.K. Sangaiah, A.A.R. Hosseinabadi, M.B. Shareh, S.Y. Bozorgi Rad, A. Zolfagharian, N. Chilamkurti, Iot resource allocation and optimization based on heuristic algorithm, Sensors 20 (2) (2020) http://dx.doi.org/10.3390/s20020539.

[33] S. Wan, X. Xu, T. Wang, Z. Gu, An intelligent video analysis method for abnormal event detection in intelligent transportation systems, IEEE Trans. Intell. Transp. Syst. (2020) http://dx.doi.org/10.1109/TITS.2020.3017505.

[34] C.B. Tan, M.H.A. Hijazi, Y. Lim, A. Gani, A survey on proof of retrievability for cloud data integrity and availability: Cloud storage state-of-the-art, issues, solutions and future trends, J. Netw. Comput. Appl. 110 (2018) 75–86, http://dx.doi.org/10.1016/j.jnca.2018.03.017.

[35] R. Zhang, G. Zhang, L. Liu, C. Wang, S. Wan, Anomaly detection in bitcoin information networks with multi-constrained meta path, J. Syst. Archit. 110 (2020) 101829, http://dx.doi.org/10.1016/j.sysarc.2020.101829, URL http://www.sciencedirect.com/science/article/pii/S1383762120301211.

[36] A. Fu, Y. Shui, Y. Zhang, H. Wang, C. Huang, Npp: A new privacy-aware public auditing scheme for cloud data sharing with group users, IEEE Trans. Big Data (2017) http://dx.doi.org/10.1109/TBDATA.2017.2701347.

[37] L. Wu, J. Wang, S. Zeadally, D. He, Privacy-preserving auditing scheme for shared data in public clouds, J. Supercomput. 74 (11) (2018) 6156–6183, http://dx.doi.org/10.1007/s11227-018-2527-y.

[38] L. Yeh, P. Chiang, Y. Tsai, J. Huang, Cloud-based fine-grained health information access control framework for lightweightiot devices with dynamic auditing andattribute revocation, IEEE Trans. Cloud Comput. 6 (2) (2018) 532–544, http://dx.doi.org/10.1109/TCC.2015.2485199.

[39] H. Tian, F. Nan, C.-C. Chang, Y. Huang, J. Lu, Y. Du, Privacy-preserving public auditing for secure data storage in fog-to-cloud computing, J. Netw. Comput. Appl. 127 (2019) 59–69, http://dx.doi.org/10.1016/j.jnca.2018.12.004.

[40] D. He, S. Zeadally, L. Wu, Certificateless public auditing scheme for cloud-assisted wireless body area networks, IEEE Syst. J. 12 (1) (2018) 64–73, http://dx.doi.org/10.1109/JSYST.2015.2428620.

[41] X. Liu, M.S. Obaidat, C. Lin, T. Wang, A. Liu, Movement-based solutions to energy limitation in wireless sensor networks: State of the art and future trends, IEEE Netw. (2020) http://dx.doi.org/10.1109/MNET.011.2000445.

[42] J. Xia, G. Cheng, S. Gu, D. Guo, Secure and trust-oriented edge storage for internet of things, IEEE Internet Things J. 7 (5) (2020) 4049–4060, http://dx.doi.org/10.1109/JIOT.2019.2962070.

[43] Y. Tao, P. Xu, H. Jin, Secure data sharing and search for cloud-edge-collaborative storage, IEEE Access 8 (2020) 15963–15972, http://dx.doi.org/10.1109/ACCESS.2019.2962600.

[44] T. Wang, Y. Mei, W. Jia, X. Zheng, G. Wang, M. Xie, Edge-based differential privacy computing for sensor-cloud systems, J. Parallel Distrib. Comput. 136 (2020) 75–85, http://dx.doi.org/10.1016/j.jpdc.2019.10.009.

[45] W. Wang, P. Xu, D. Liu, L.T. Yang, Z. Yan, Lightweighted secure searching over public-key ciphertexts for edge-cloud-assisted industrial iot devices, IEEE Trans. Ind. Inf. 16 (6) (2020) 4221–4230, http://dx.doi.org/10.1109/TII.2019.2950295.

[46] Y. Li, Z. Dong, K. Sha, C. Jiang, J. Wan, Y. Wang, Tmo: Time domain outsourcing attribute-based encryption scheme for data acquisition in edge computing, IEEE Access 7 (2019) 40240–40257, http://dx.doi.org/10.1109/ACCESS.2019.2907319.

[47] J. Kang, R. Yu, X. Huang, M. Wu, S. Maharjan, S. Xie, Y. Zhang, Blockchain for secure and efficient data sharing in vehicular edge computing and networks, IEEE Internet Things J. 6 (3) (2019) 4660–4670, http://dx.doi.org/10.1109/JIOT.2018.2875542.

[48] A. Abdi, N. Idris, R.M. Alguliyev, R.M. Aliguliyev, Pdlk: Plagiarism detection using linguistic knowledge, Expert Syst. Appl. 42 (22) (2015) 8936–8946, http://dx.doi.org/10.1016/j.eswa.2015.07.048.

[49] T.S. Ali, R. Ali, A novel medical image signcryption scheme using tlts and henon chaotic map, IEEE Access 8 (2020) 71974–71992.

[50] Y. Chen, L. Li, Very fast decision tree classification algorithm based on red-black tree for data stream with continuous attributes [j], J. Nanjing Univ. Posts Telecommun. (Nat. Sci. Ed.) 37 (2) (2017) 86–90.

[51] M. Morales-Sandoval, J.L. Gonzalez-Compean, A. Diaz-Perez, V.J. Sosa-Sosa, A pairing-based cryptographic approach for data security in the cloud, Int. J. Inf. Secur. 17 (2) (2017) 441–461, http://dx.doi.org/10.1007/s10207-017-0375-z.

[52] F. Kirchner, N. Kosmatov, V. Prevosto, J. Signoles, B. Yakobowski, Frama-c: A software analysis perspective, Form. Asp. Comput. 27 (3) (2015) 573–609, http://dx.doi.org/10.1007/s00165-014-0326-7.