Received July 25, 2020, accepted August 11, 2020, date of publication August 14, 2020, date of current version August 27, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3016760

An Authorized Public Auditing Scheme for Dynamic Big Data Storage in Cloud Computing

HAN YU[1,2], XIUQING LU[1,3], AND ZHENKUAN PAN[1]
[1]College of Computer Science and Technology, Qingdao University, Qingdao 266071, China
[2]College of Computer, National University of Defense Technology, Changsha 410073, China
[3]Department of Management Science and Engineering, College of Business, Qingdao University, Qingdao 266071, China

Corresponding author: Xiuqing Lu (luxiuqing@qdu.edu.cn)

This work was supported in part by the Natural Science Foundation of China under Grant 61772294 and Grant 61602214, and in part by the Natural Science Foundation of Shandong Province under Grant ZR2019LZH002.

ABSTRACT Nowadays, to utilize the abundant resources of cloud computing, most enterprise users prefer to store their big data on cloud servers for sharing and utilization. However, storing data in remote cloud servers is out of user's control and exposes to lots of security problems such data availability, unauthorized access and data integrity, among which data integrity is a challenging and urgent task in cloud computing. Many auditing schemes have been proposed to check the integrity of data in cloud, but these schemes usually have some disadvantages. One is that these auditing schemes cannot check which block is corrupt when the data is not integrated. The other is that there's no efficient authenticated data structure helping to achieve accurate auditing when the data needs to update frequently. To solve the problems, we propose a public auditing scheme for dynamic big data storage in cloud computing. Firstly, we design a dynamic index table, in which no elements need to be moved in insertion or deletion update operations. Secondly, when data in cloud is not integrated, the third-party auditor can detect which block is corrupt. Finally, an authorization is employed between the third party and cloud servers to prevent denial of service attack. The theoretical analysis and the simulation results demonstrate that our scheme is more secure and efficient.

INDEX TERMS Dynamic auditing, authorization, cloud storage, data security.

I. INTRODUCTION

Cloud computing is a fast-developing business computing model and has many advantages, such as large storage, low cost, and scalability. More and more enterprise users are apt to outsource their big data to cloud servers for storage and processing. After outsourcing, the enterprise users usually choose to delete their original big data from their local storage servers for saving storage space. Despite of the convenience of cloud computing, storing the data on cloud servers without possessing the original copy may bring about many security problems. Cloud servers are subject to hardware or software failures and malicious attacks occasionally [1]–[3]. And for their own benefits, cloud service providers (CSP) are reluctant to tell users the truth when the failures or attacks occur. Even worse, CSP might discard the data that the users are not or rarely accessed for saving maintenance cost or cloud storage space [4]–[6]. So CSP must provide proof to ensure the data is correctly stored on cloud servers before the data is utilized by users. How to ensure the integrity of the data in cloud servers is a challenging and urgent task in cloud computing. Data auditing schemes can enable users to verify the integrity of their data in remote cloud servers without retrieving the data. Based on the role of verifier, data auditing schemes can be divided into two categories: private auditing and public auditing. In private auditing schemes [7]–[9], the data integrity is only verified by the users and accordingly increases the overhead of users that they cannot afford. While public auditing schemes allow any public verifier who has the user's public key to execute the auditing process. Commonly, a third-party auditor (TPA) who has expertise and capabilities is involved and executes the verification performance. Many auditing schemes have been proposed to check the data integrity in cloud. However, these schemes cannot check which block is corrupt when the data is modified. Furthermore, there's no efficient authenticated data structure helps to achieve accurate auditing when the data needs to update frequently. Therefore, it is essential to propose an efficient public auditing scheme for dynamic big data storage in cloud computing.

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

In this paper, we propose an authorized dynamic public auditing scheme by introducing a new data structure named dynamic index table (DIT). Through the DIT, our scheme can achieve dynamic updating without the elements' adjustments. Additionally, our scheme can judge which block is lost or corrupt when data integrity fails. Our contributions are summarized as follows.

1) We propose an authorized dynamic public auditing scheme that can check which block is corrupt.
2) We design an efficient authenticated data structure named dynamic index table (DIT), which is used to store block properties to help TPA achieve data auditing and can be updated without element moving.
3) We prove the security of the proposed scheme and evaluate the performance of computation and communication cost. The results show that our scheme is more efficient than the other ones.

The rest of the paper is organized as follows. Section II introduces the related works on integrity verification. Section III describes the system model, threat model and the design goals of the scheme. Section IV addresses the preliminaries of the scheme. In section V, we present the proposed scheme in detail. Section VI and section VII demonstrates the security analysis and the performance of the scheme in computation and communication cost. Finally, we conclude this paper in section VIII.

II. THE RELATED WORKS

So far, many typical public auditing schemes have been proposed to verify the integrity of data stored in remote untrusted servers. In 2007, Ateniese *et al.* [10] proposed the first public auditing scheme which proposed provable data possession (PDP). This scheme allows any public verifier to check the data integrity without retrieving the data. However, this scheme can only verify static data integrity. Later Ateniese *et al.* [11] proposed another scheme based on the symmetric key PDP scheme to audit the dynamic data in cloud servers. This scheme supports dynamic modification and deletion operations, but does not support the insertion operation. To improve update efficiency, an authenticated data structure is always introduced. Erway *et al.* [12] introduced an authenticated skip list in his dynamic provable data possession (DPDP) scheme. Later, Wang *et al.* [13] proposed a dynamic public auditing scheme based on Merkle Hash Tree (MHT). The scheme can achieve dynamic data operations, but it would incur multitude computation and communication overhead during the verification process. In scheme [14], Zhu *et al.* introduced an index-hash table (IHT) stored at TPA side to help dynamic verification. Compared with other schemes, it is more efficient in computation and communication costs. However, in updating process, as IHT is a sequence data structure, it would cause an average of half adjustment of elements in IHT, resulting in the decrease of the system efficiency. In 2013, Yang and Jia [15] proposed an index table (ITable) to store the abstract information of blocks, including the current and original index number, current version number and time stamp of each block. It is efficient to prevent the replay attack, but in insert and delete operations, all the tags of blocks after deleted or inserted need to be recomputed as the indexes of these blocks are changed. Liu *et al.* [16] put forward an authorized public auditing scheme for big data with efficient verifiable fine-grained updates. Later in 2017, Tian *et al.* [17] proposed a Dynamic-Hash-Table based auditing scheme for cloud storage. In 2018, Gan *et al.* [18] designed an efficient and secure auditing scheme for outsourced big data with algebraic signature. Zhang *et al.* [19] proposed a cloud storage auditing for shared big data. In 2020, Lu *et al.* [20] propose an integrity verification scheme for Internet of Things (IoT) mobile terminal devices. In the scheme, block-tag generation and integrity verification operations are executed at third-party auditor (TPA) side, which achieves lightweight operations of data owners. However, the employed data structures in all the schemes cannot ensure the replay attack during integrity verification process. Therefore, it is crucial to develop a more secure auditing scheme for achieving dynamic integrity verification services. Table. 1 compares the scheme with other typical schemes in terms of dynamic auditing, batch auditing, data structure and authorized auditing. Nowadays, many other integrity verification schemes have been proposed and prompted the security development of cloud computing. Because the data stored in cloud for sharing can face many privacy challenge such as identity privacy and sharing data privacy. Scheme [2], [21]–[28] proposed privacy preserving auditing protocols to prevent privacy leaking. At the same time, with the development of Internet of things and mobile devices, lightweight schemes [29]–[35] are proposed to satisfying the efficiency needs of auditing process. In recent years, many schemes [36]–[40] based on identity encryption and attribute encryption are put forward to realize data sharing with other authorized users in cloud.

TABLE 1. Comparison of integrity verification schemes.

Scheme	Dynamic Auditing	Batch Auditing	Data Structure	Authorized Auditing
Scheme [11]	Yes	No	No	No
Scheme[12]	Yes	No	Yes	No
Scheme[13]	Yes	Yes	Yes	No
Scheme[14]	Yes	Yes	Yes	No
Scheme[15]	Yes	Yes	No	No
Scheme[16]	Yes	No	Yes	Yes
Scheme[17]	Yes	Yes	Yes	No
Our Scheme	Yes	Yes	Yes	Yes

III. SYSTEM MODEL, SECURITY REQUIREMENT AND DESIGN GOALS

A. SYSTEM MODEL

We describe the system model as illustrated in fig. 1. It involves four entities named enterprise user (user), Cloud Server Provider (CSP) and the Third-Party Auditor (TPA). The user generates and outsource massive amount of data to Cloud Servers (CS) which has large capacity to maintain the user's data. CSP manages the cloud servers and gives user access anywhere with an Internet service. TPA is an entity that is authorized by user and has much expertise and resources to verify data integrity efficiently.

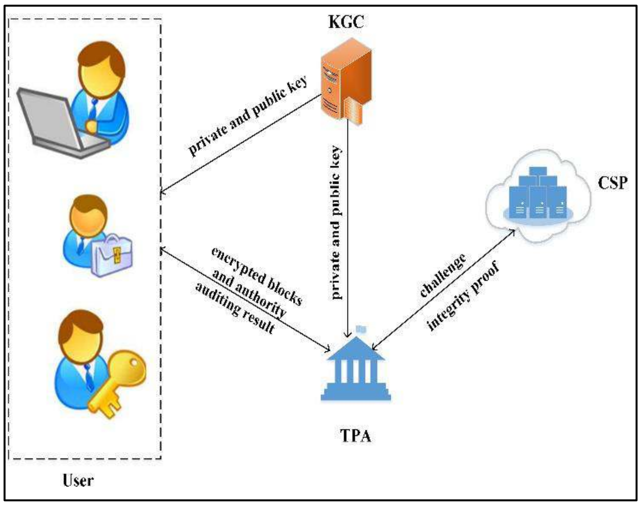


FIGURE 1. System model.

In the system, we assume that both TPA and CSP are semi-trusted. TPA is semi-trusted because he may be curious about user's data. The scheme must preserve the outsourced data privacy from TPA. CSP is semi-trusted because when some data in cloud servers is corrupt or lost, CSP may launch forge attack or replace attack to TPA for economic reasons.

B. SECURITY REQUIREMENT

Public auditing. TPA can publicly verify the integrity of outsourced data for user.

Authorized auditing. Only the authorized TPA can launch auditing challenge to avoid replay attack.

Data Privacy. TPA cannot learn the content of data stored in cloud servers in public auditing process.

Unforgeability. Only the user can generate the block tags for auditing.

Storage integrity. The integrity verification can be achieved only if CSS correctly stores data blocks and the corresponding block tags.

C. DESIGN GOALS

Based on the system model and security requirements, our scheme should achieve the following properties.

Security requirements. The scheme should satisfy the security requirements including data privacy, authorization and unforgeability during integrity verification process.

Lightweight operations. Both the computation and communication costs of user are greatly reduced in our auditing scheme because TPA is responsible for generating block tags and managing DIT.

Effectiveness. The scheme should effectively achieve data auditing process under user's authorization.

IV. PRELIMINARIES

A. NOTATIONS

The notations in this paper are described in Table. 2.

TABLE 2. Main notations of the scheme.

Notation	Meaning	Notation	Meaning
$\mathbb{G}_1, \mathbb{G}_2$	multiplicative group	F	plaintext of file
e	bilinear map	M	encrypted file
$H, h. \pi$	secure hash functions	m_i	encrypted blocks
p	prime order of group	σ_i	block tag
g	generator of $\mathbb{G}_1$	P	integrity proof
sk	secret key of user	Uid	user identity
pk	public key of user	sig	authorization
α	secret key of TPA	$Chall$	challenge
w	public key of TPA	Θ	group system

B. BILINEAR MAPS

Suppose $\mathbb{G}_1$,$\mathbb{G}_2$ are two multiplicative groups with same large prime order q, and $\mathbb{G}$ is a generator of $\mathbb{G}_1$. A bilinear map e is a map function e:$\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_1$ with the following properties: i) Computability. $\forall u, v \in \mathbb{G}_1$, an efficient algorithm exists to compute $e(u, v)$. ii) Binearity. $\forall a, b \in Z_q$, $\exists e\left(u^a, v^b\right) = e\,(u,v)^{ab}$. iii) Nondegeneracy. $e\,[g,g] \neq 1$. iv) Security. It is hard to compute Discrete Logarithm (DL) in $\mathbb{G}_1$.

C. COMPLEXITY ASSUMPTIONS

1) **Discrete Logarithm (DL) Assumption.** Suppose g is a generator of multiplicative cyclic group $\mathbb{G}$ with prime order q. On input y $\in \mathbb{G}$, there does not exist probabilistic polynomial time algorithm that outputs a value $x \in Z_q^*$ such that $g^x = y$ with non-negligible probability.
2) **Computational Diffie-Hellman (CDH) Assumption.** Suppose g is a generator of multiplicative cyclic group $\mathbb{G}$ with prime order q. On input $g^x, g^y \in \mathbb{G}$, there does not exist probabilistic polynomial time algorithm that outputs $g^{xy} \in \mathbb{G}$ with non-negligible probability.

V. CONSTRUCTIONS OF SECURE DATA SHARING SCHEME

A. DYNAMIC INDEX TABLE

To achieve the public integrity verification efficiently, an authenticated data structure named Dynamic Index Table (DIT) is employed. To avoid the elements in DIT moving when blocks are inserted or deleted, we use static linked list to construct DIT. DIT is a one-dimensional structural array and includes five structural members: block number (Bid_i), hash value of each blind block ($Hash_i$), time stamp of block (T_i), version number of block (V_i) and the static pointer pointing to the subordinate of next block ($Next_i$). $Hash_i$ is mainly used to check which block is corrupt when data is not integrated. T_i and V_i are used to avoid attacks from adversaries. $Next_i$ points to subordinate of next block for connecting the file together. For example, $Next_1$ is 2 means the next block data of m_1 is m_2. $Next_n$ is 0 means m_n is the final block data of the file. The initial DIT information is described as Table. 3.

TABLE 3. Initial DIT.

Subordinate	Bid_i	$Hash_i$	T_i	V_i	$Next_i$
1	1	$H(m_1)$	T_1	1	2
2	2	$H(m_2)$	T_2	1	3
3	3	…	…	…	…
…	…	…	…	…	…
$i-1$	$i-1$	$H(m_{i-1})$	T_{i-1}	1	i
i	i	$H(m_i)$	T_i	1	$i+1$
$i+1$	$i+1$	$H(m_{i+1})$	T_{i+1}	1	$i+2$
…	…	…	…	…	…
$n-1$	$n-1$	$H(m_{n-1})$	T_{n-1}	1	n
n	n	$H(m_n)$	T_n	1	0

When block m_i is deleted, $Next_{i-1}$ is set to value $i+1$ from the obvious value i, which means the next block data of m_{i-1} is m_{i+1}. Moreover, $Next_i$ is set to value -1, indicating that m_i is deleted from the file F and a new element information can be stored here. Table. 4. Describes modified DIT after m_i is deleted.

TABLE 4. Modified DIT after m_i is deleted.

Subordinate	Bid_i	$Hash_i$	T_i	V_i	$Next_i$
1	1	$H(m_1)$	T_1	1	2
2	2	$H(m_2)$	T_2	1	3
3	3	…	…	…	…
…	…	…	…	…	…
$i-1$	$i-1$	$H(m_{i-1})$	T_{i-1}	1	$i+1$
i	i	$H(m_i)$	T_i	1	-1
$i+1$	i	$H(m_{i+1})$	T_{i+1}	1	$i+2$
…	…	…	…	…	…
$n-1$	$n-1$	$H(m_{n-1})$	T_{n-1}	1	n
n	n	$H(m_n)$	T_n	1	0

When a new block m_i' is inserted after m_{i-1}, $Next_{i-1}$ is changed to i and previous $Next_i$ is changed to $i+1$. If there are no ineffective lines, the information of m_i can be added in the last position only with the corresponding static pointer are updated. Table. 5. Describes modified DIT after m_i' is inserted after block data m_{i-1}.

B. DETAILED INTEGRITY VERIFICATION SCHEME

The efficient and secure auditing scheme consists of three phases including setup phase, integrity verification phase and dynamic update phase. The three phases are described in detail as follows.

TABLE 5. Modified DIT after m_i' is inserted after m_{i-1}.

Subordinate	Bid_i	$Hash_i$	T_i	V_i	$Next_i$
1	1	$H(m_1)$	T_1	1	2
2	2	$H(m_2)$	T_2	1	3
3	3	…	…	…	…
…	…	…	…	…	…
$i-1$	$i-1$	$H(m_{i-1})$	T_{i-1}	1	i
i	i	$H(m_i')$	T_i	1	$i+1$
$i+1$	$i+1$	$H(m_{i+1})$	T_{i+1}	1	$i+2$
…	…	…	…	…	…
$n-1$	$n-1$	$H(m_{n-1})$	T_{n-1}	1	n
n	n	$H(m_n)$	T_n	1	0

1) **Setup phase**

In this phase, KGC generate system parameters and keys for user and TPA in algorithm *Initial*. The user is responsible to divide big data into blocks and blinds each in algorithm *BlockBlind*. TPA is in charge of generating block tags in algorithm *TagGen* and deriving DIT in algorithm *DITGen*. The user computes challenge authority for TPA in algorithm *AuthorityGen*. The dataflow in each algorithm of this phase is described in fig. 2.

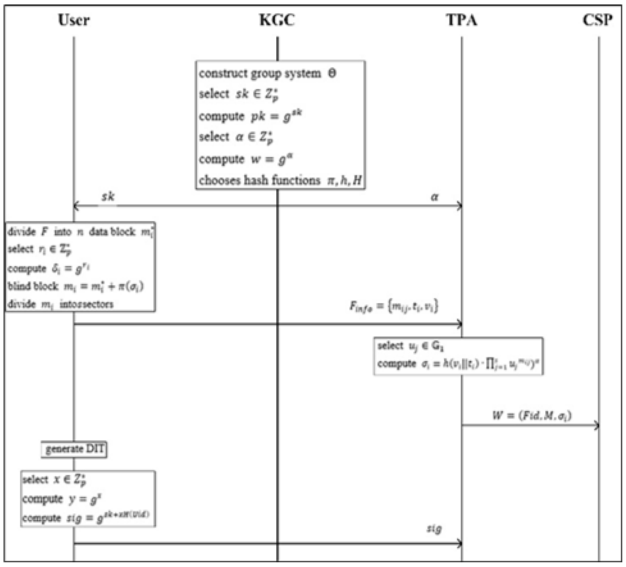

User
KGC
TPA
CSP
construct group system Θ
select $sk \in Z_p^*$
compute $pk = g^{sk}$
select $\alpha \in Z_p^*$
compute $w = g^{\alpha}$
chooses hash functions π, h, H
sk
α
divide F into n data block m_i^*
select $r_i \in Z_p^*$
compute $\delta_i = g^{r_i}$
blind block $m_i = m_i^* + \pi(\sigma_i)$
divide m_i intossectors
$F_{info} = \{m_{ij}, t_i, v_i\}$
select $u_j \in G_1$
compute $\sigma_i = h(v_i \| t_i) \cdot \prod_{j=1}^{s} u_j^{m_{ij}})^{\alpha}$
$W = (Fid, M, \sigma_i)$
generate DIT
select $x \in Z_p^*$
compute $y = g^x$
compute $sig = g^{sk+xH(Uid)}$
sig

FIGURE 2. Dataflow in setup phase.

Initial **(λ) $\rightarrow$ {Θ,α, *sk*}.** Given system security parameter λ, KGC constructs the bilinear map group system $\Theta = (\mathbb{G}_1, \mathbb{G}_2, p, g, e)$ where $\mathbb{G}_1, \mathbb{G}_2$ are multiplicative groups with prime order p, g is a generator of $\mathbb{G}_1$ and e is a bilinear map $e{:}\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. KGC selects $sk \in Z_p^*$ as user's private key and computes $pk = g^{sk}$ as the corresponding public key. Then KGC selects $\alpha \in Z_p^*$ as TPA's private key and computes $w = g^{\alpha}$ as the corresponding public key. Next, KGC chooses secure one-way hash functions $\pi : \mathbb{G}_1 \rightarrow \mathbb{G}_1, h : \{0, 1\}^* \rightarrow \mathbb{G}_1, H : \mathbb{G}_1 \rightarrow Z_p^*$. Finally, KGC sends sk

to user with identity Uid, α to TPA securely and make $\{\mathbb{G}_1, \mathbb{G}_2, p, g, e, \pi, h, H, w, pk\}$ public.

BlockBlind $(\boldsymbol{F}, \boldsymbol{SK}) \rightarrow \boldsymbol{M}$. The user first divides F with identifier Fid into n data block named m_i^* by erasure code algorithm. To keep the data private to others, user blinds each block before outsourcing F to CSS. The user selects $r_i \in \mathbb{Z}_p^*$, $i \in [1,n]$ randomly and computes $\delta_i = g^{r_i}$. Then user blinds each block as $m_i = m_i^* + \pi(\sigma_i)$ and denotes $M = \{m_i\}_{i \in [1,n]}$. Furtherly, user divides each block m_i into s sectors. That means $M = \{m_i\}, m_i = \{m_{ij}\}, i \in [1,n], j \in [1,s]$. Finally, user transfers $F_{info} = \{m_{ij}, t_i, v_i\}$ to TPA, where t_i and v_i is the timestamp and version of each block.

TagGen $(\boldsymbol{M}, \boldsymbol{\alpha}) \rightarrow \boldsymbol{\sigma_i}$. TPA selects $u_j \in \mathbb{G}_1, j \in [1, s]$ and computes block tags σ_i for each block m_i, $i \in [1, n]$ as follows.

$$\sigma_i = h(v_i||t_i) \cdot \prod_{j=1}^{s} u_j^{m_{ij}})^{\alpha} \tag{1}$$

Then TPA sends $W = (Fid, M, \sigma_i)$ to CSP.

DITGen $(\boldsymbol{F_{info}}) \rightarrow \boldsymbol{DIT}$. TPA generates DIT including $Bid_i, Hash_i, T_i, V_i, Next_i$ and stores it locally for dynamic updates later. To save space, then TPA deletes m_i from local server.

AuthorityGen $(\boldsymbol{sk}) \rightarrow \boldsymbol{sig}$. Only authorized TPA can launch auditing challenge to prevent malicious attackers from generating denial-of-service attacks on CSP. The user with identity Uid randomly selects $x \in Z_p^*$ and computes $y = g^x$. Then user generates authorization for TPA to launch auditing challenge as follows.

$$sig = g^{sk + xH(Uid)} \tag{2}$$

Finally, the user sends sig to TPA.

2) **Integrity verification phase** In this phase, TPA first generates a challenge and sends it to CSP in algorithm *ChallGen*. Next, CSP computes integrity proof and sends it to TPA for verification in algorithm *ProofGen*. Then TPA verifies whether the data is intact through the proof in algorithm *ProofVerify*. The dataflow in each algorithm of this phase is described in fig. 3.

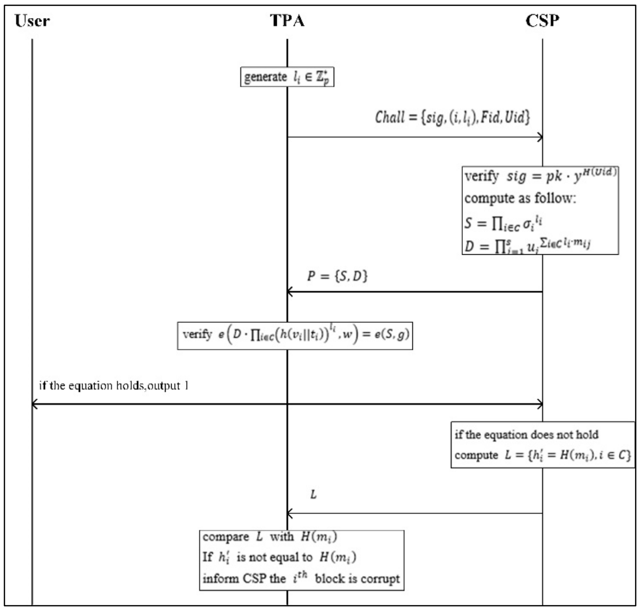
ChallGen $(\boldsymbol{F_{info}})$. When TPA gets the verification delegation from the user, he selects some blocks to construct a random c-element subset C from set [1, n] and generates random numbers $l_i \in \mathbb{Z}_p^*$, $i \in C$. Then TPA sends the challenge $Chall = \{sig, (i, l_i), Fid, Uid\}, i \in C$ to CSP.

ProofGen $(\boldsymbol{F}, \boldsymbol{T}, \boldsymbol{Chall})$: On receiving the challenge, CSP verifies the equation $sig = pk \cdot y^{H(Uid)}$. If it fails, it outputs *NO*, otherwise, CSP computes tag proof and data proof as follows.

$$S = \prod_{i \in C} \sigma_i^{l_i} \tag{3}$$

$$D = \prod_{j=1}^{s} u_j^{\sum_{i \in C} l_i \cdot m_{ij}} \tag{4}$$

Then CSP sends $P = \{S, D\}$ to TPA.



User
TPA
CSP
generate $l_i \in \mathbb{Z}_p^*$
$Chall = \{sig, (i, l_i), Fid, Uid\}$
verify $sig = pk \cdot y^{H(Uid)}$
compute as follow:
$S = \prod_{i \in C} \sigma_i^{l_i}$
$D = \prod_{j=1}^{s} u_j^{\sum_{i \in C} l_i \cdot m_{ij}}$
$P = \{S, D\}$
verify $e\left(D \cdot \prod_{i \in C}(h(v_i||t_i))^{l_i}, w\right) = e(S, g)$
if the equation holds,output 1
if the equation does not hold
compute $L = \{h_i' = H(m_i), i \in C\}$
L
compare L with $H(m_i)$
If h_i' is not equal to $H(m_i)$
inform CSP the i^{th} block is corrupt

FIGURE 3. Dataflow of dynamic integrity verification phase.

ProofVerify $(\boldsymbol{P}, \boldsymbol{w}) \rightarrow \{\boldsymbol{1}, \boldsymbol{0}\}$. After receiving the proof P from CSP, TPA verifies the proof P as follows.

$$e\left(D \cdot \prod_{i \in C} (h(v_i||t_i))^{l_i}, w\right) = e(S, g). \tag{5}$$

If the equation holds, the algorithm outputs 1. Otherwise, the algorithm checks which block is not correctly stored as follows. TPA sends check request to CSP. Then CSP computes $L = \{h_i' = H(m_i), i \in C$, where m_i is block data stored on CSS, and transfer L to TPA. Next, TPA compares L with $H(m_i)$ stored in DIT sequentially. If h_i' is not equal to $H(m_i)$, TPA informs CSP the i^{th} block is corrupt and CSP recovers it.

3) **Integrity verification phase**
The user can update the data outsourced to the cloud whenever needed. The user can execute insertion, deletion and modification operations on block level. Algorithm *BlockInsert* executes block insertion *BlockDelete* realizes block deletion. Block modification can be executed with algorithm *ockModify*. The dataflow in this phase is described in fig. 4.

BlockInsert $\left(\boldsymbol{m_i'}, \boldsymbol{i}, \boldsymbol{SK}\right)$. Suppose a new block m' is to be inserted after block m_i. The user first calls algorithm *BlockBlind* to blind the block as $m^* = m' + \pi(\sigma_i)$. Then TPA calls algorithm *TagGen* to compute a new tag σ' for m' and sends $\{m^*, \sigma'\}$ to CSP. Meanwhile, TPA computes $H(m_i')$ and adds a new item $(i + 1, H(m'), t', v')$ at the position where $Next_i$ is -1 or at the last position of DIT.

BlockDelete $(\boldsymbol{m_i})$. Assume block m_i is to be deleted. CSP deletes m_i and σ_i from CSS. Then TPA finds the position of m_i based on block number and modifies $Next_i$ to -1, indicating new item can be inserted her.

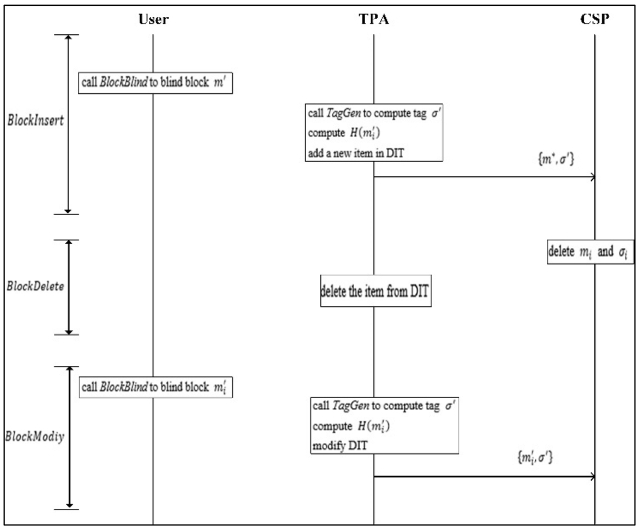

User
TPA
CSP
BlockInsert
call BlockBlind to blind block m'
call TagGen to compute tag σ'
compute $H(m_i')$
add a new item in DIT
$\{m^*, \sigma'\}$
BlockDelete
delete m_i and σ_i
delete the item from DIT
BlockModiy
call BlockBlind to blind block m_i'
call TagGen to compute tag σ'
compute $H(m_i')$
modify DIT
$\{m_i', \sigma'\}$

FIGURE 4. **Dataflow of dynamic update phase.**

BlockModify $\left(\boldsymbol{m_i'}, \boldsymbol{\alpha}, \boldsymbol{sk}\right)$. Assume block m_i is modified to m_i'. The user first calls algorithm *BlockBlind* to blind the block as $m_i' = m_i' + \pi(\sigma_i)$. Then TPA calls algorithm *TagGen* to compute a new tag σ' for m_i' and sends $\left\{m_i', \sigma'\right\}$ to CSP. Meanwhile, TPA computes $H\left(m_i'\right)$ and modifies the previous element in DIT to new element $\left(i, H\left(m_i'\right), t', v'\right)$ at the correct position. After each updating, the user delegate TPA to verify the update block. When the verification is passed, the user chooses to delete the local data.

C. BATCH AUDITING FROM MULTIUSERS

Batch auditing can concurrently process multiple verifications from different users. Suppose U is collection of k different users. When receiving k challenges from k users, CSP computes tag proof $S_i, i \in [1, k]$ and data proof $D_i, i \in [1, k]$. Then CSP gets S_U and D_U by aggregating S_i and D_i respectively according to the following equations:

$$S_U = \prod_{i=1}^{k} S_i \tag{6}$$

$$D_U = \prod_{i=1}^{k} D_i \tag{7}$$

When receiving the proof S_U and D_U from CSP, TPA checks the proof through the following verification equation:

$$e\left(D_U \cdot \prod_{i=1}^{k} \left(\prod_{j \in C} \left(h\left(v_{i,j} || t_{i,j}\right)\right)^{l_{i,j}}\right), w_i\right) = e\left(S_U, g\right). \tag{8}$$

If the equation holds, it outputs YES, meaning all the files of the k users are correctly stored on cloud servers. Otherwise, it outputs NO, meaning one or more files are corrupt.

VI. SECURITY ANALYSIS

In this section, the security of the proposed scheme, including correctness, unforgeability and privacy is analyzed.

Theorem 1: An authorized public verifier can correctly verify the integrity of the file stored in cloud servers in our scheme.

Proof: Theorem 1 can be proved through verifying the correctness of eq. (5). The proof is as follows:

$$\begin{aligned} & e\left(D \cdot \prod_{i \in C} \left(h\left(v_i || t_i\right)\right)^{l_i}, w\right) \\ & = e\left(\prod_{j=1}^{s} u_j^{\sum_{i \in C} l_i \cdot m_{ij}} \cdot \prod_{i \in C} \left(h\left(v_i || t_i\right)\right)^{l_i}, w\right) \\ & = e\left(\prod_{i \in C} \prod_{j=1}^{s} u_j^{l_i \cdot m_{ij}} \cdot \prod_{i \in C} \left(h\left(v_i || t_i\right)\right)^{l_i}, w\right) \\ & = e\left(\prod_{i \in C} \prod_{j=1}^{s} u_j^{l_i \cdot m_{ij}} \cdot \prod_{i \in C} \left(h\left(v_i || t_i\right)\right)^{l_i}, g^{\alpha}\right) \\ & = e\left(S, g\right) \end{aligned}$$

From the proof of eq. (5), TPA can verify the integrity of the file outsourced to the CSP.

Theorem 2: It is computationally impossible for CSP to forge an integrity proof to pass the public verification, if the Computational Diffie-Hellman (CDH) problem is hard in bilinear group.

Proof: After CSP receives the challenge $Chall = \{sig, (i, l_i), Fid\}, i \in C$ from TPA, he should send the correct proof $P = \{S, D\}$ to TPA. Instead, suppose CSP generates an incorrect proof $P' = \left\{S, D'\right\}$ to TPA, where $D' = \prod_{j=1}^{s} u_j^{\lambda_j'}, \lambda_j' = \sum_{i \in C} l_i \times m_{ij}', j \in [1, s]$. Define $\lambda_j = \sum_{i \in C} l_i \times m_{ij}, \Delta\lambda_j = \lambda_j - \lambda_j'$. It is obvious at least one $\Delta\lambda_j$ is nonzero. If CSP can pass the verification with P', the CSP wins the game, otherwise, it fails.

Suppose CSP can win the game, the following equation can be inferred according to eq. (5).

$$e\left(\prod_{j=1}^{s} u_j^{\lambda_j'} \cdot \prod_{i \in C} \left(h\left(v_i || t_i\right)\right)^{l_i}, w\right) = e\left(S, g\right).$$

Furthermore, $P = \{S, D\}$ is the correct proof, so the following equation also satisfies.

$$e\left(\prod_{j=1}^{s} u_j^{\lambda_j} \cdot \prod_{i \in C} \left(h\left(v_i || t_i\right)\right)^{l_i}, w\right) = e\left(S, g\right)$$

From the above two equations and the properties of bilinear maps, it can be concluded that $\prod_{j=1}^{s} u_j^{\lambda_j'} = \prod_{j=1}^{s} u_j^{\lambda_j} \Rightarrow \prod_{j=1}^{s} u_j^{\Delta\lambda_j} = 1$. Because $\mathbb{G}_1$ is a cyclic group, then for two elements $b_1, b_2 \in \mathbb{G}_1, \exists x \in Z_p$ such that $b_2 = b_1^x$. Furthermore, given b_1, b_2, u_j can be generated as $u_j = b_1^{\mu_j} b_2^{\nu_j} \in G1$, where $\mu_j, \nu_j \in Z_p$. Then we have the following. $\prod_{j=1}^{s} u_j^{\Delta\lambda_j} = \prod_{j=1}^{s} \left(b_1^{\mu_j} b_2^{\nu_j}\right)^{\Delta\lambda_j} = b_1^{\sum_{j=1}^{s} \mu_j \Delta\lambda_j} \cdot b_2^{\sum_{j=1}^{s} \nu_j \Delta\lambda_j} = 1$. Obviously, a solution to the DL problem can be found. The value x can be computed as follows unless $\Delta\lambda_j$ is zero.

$$\begin{aligned} b_2 &= b_1^x = b_1^{\frac{\sum_{j=1}^{s_{max}} \mu_j \Delta\lambda_j}{\sum_{j=1}^{s_{max}} \nu_j \Delta\lambda_j}}, \\ x &= \frac{\sum_{j=1}^{s_{max}} \mu_j \Delta\lambda_j}{\sum_{j=1}^{s_{max}} \nu_j \Delta\lambda_j}. \end{aligned}$$

However, at least one $\Delta\lambda_j$ is defined nonzero and v_j is a random element of Z_p, which means the probability of v_j being equal to zero is $1/p$. Therefore, we can find a solution to the DL problem with a probability of $1 - 1/p$, which is conflict with the suppose that the DL problem is hard in $\mathbb{G}_1$. This is the proof of the theorem 2.

Theorem 3: As long as the DL assumption holds, it is computationally infeasible for TPA to get any private data during the integrity verification.

Proof: After CSS gets the challenge *Chall* from TPA, he sends $D = \prod_{j=1}^{s} u_j^{\sum_{i \in C} l_i \cdot m_{ij}}$ to TPA as the data proof. Because $\sum_{i \in C} l_i \cdot m_{ij}$ is at the exponent position of D, according to DL assumption, TPA cannot get any information on the user's private data.

VII. PERFORMANCE EVALUATION

A. COMMUNICATION COSTS

According to the proposed scheme, in setup phase, the main communication cost is generated between user and TPA and between TPA and CSP. Suppose an element's size of $\mathbb{Z}_p$ is $|p|$. In algorithm *BlockBlind*, after user blinds each block, he sends $F_{info} = \{m_{ij}, t_i, v_i\}$ to TPA. Therefore, the communication cost is $n|p| + n(|t_i| + |v_i|)$, where $|t_i|$, $|v_i|$ are size of t_i and v_i. In algorithm *TagGen*, TPA sends $W = (Fid, M, \sigma_i)$ to CSP. Therefore, the communication cost is $2n|p| + 1$. In integrity verification phase, the main communication cost is mainly generated between TPA and CSP. When launching a challenge in algorithm *ChallGen*, TPA sends $Chall = \{sig, (i, l_i), Fid, Uid\}$ to CSP and the main communication cost is $c(|i| + |p|)$ bits, where $|i|$ is the size of the block index. In algorithm *Proof*Gen, CSP sends $P = \{S, D\}$ to TPA and the communication is $2|p|$, which is constant and can be ignored. In updating phase, the communication cost between the user and TPA and between the TPA and CSP is a constant. We compare our scheme with scheme [12]–[14], [17] in the complexity of communication costs as Table. 6. From the table, it can be concluded that the communication cost of our scheme is more efficient than Wang's and the same as Zhu's.

TABLE 6. Comparison of communication costs.

Scheme	Setup Phase	Verification Phase	Update Phase
Scheme [12]	$O(n)$	$cO(\log n)$	$O(\log n)$
Scheme [13]	$O(n)$	$cO(\log n)$	$O(\log n)$
Scheme [14]	$O(n)$	$O(c)$	$O(1)$
Scheme [17]	$O(n)$	$O(c)$	$O(1)$
Our Scheme	$O(n)$	$O(c)$	$O(1)$

B. STORAGE COSTS

Our scheme consists of three phases and the storage costs mainly generate in setup phase. Suppose the file outsourced to cloud named F includes n data blocks and each block size is $|p|$. In algorithm *BlockBlind*, user transfers $F_{info} = \{m_{ij}, t_i, v_i\}$ to TPA. In algorithm *DITGen*, TPA generates DIT including $Bid_i, Hash_i, T_i, V_i, Next_i$. To save space, TPA deletes m_i from the local server. Therefore, the total storage cost of TPA in setup phase is nl_3, where $l_3 = |Bid_i| + |t_i| + |v_i| + |Hash_i| + |Next_i|$ and indicates the bit size of each element of DIT. In algorithm *TagGen*, TPA sends $W = (Fid, M, \sigma_i)$ to CSP and CSP stores W. Therefore, the main storage cost of CSP in setup phase is $2n|p|$ which is mainly generated by block data M and block tag σ_i. In scheme [14], Index Hash Table (IHT) is used to indicate the changes of blocks and generate hash block value during integrity verification process. In IHT, B_i, V_i and R_i respectively represent block number, version number and random value. Therefore, the total storage cost of TPA is nl_1, where $l_1 = |B_i| + |V_i| + |R_i|$ indicating the bit size of each element of IHT. In scheme [17], Each block element is one node of the file list, including the block version v_i, time stamp t_i and a pointer indicating the next node $next_i$. Accordingly, the total storage cost of TPA is nl_2, where $l_2 = |v_i| + |t_i| + |next_i|$ representing bit size of each element of Dynamic hash table (DHT). The storage costs of the scheme is evaluated and compared with scheme [12]–[14], [17] as described in Table. 7. Although the size of l_3 is a bit larger than l_1 and l_2, DIT is more secure than IHT and DHT because of the employment of hash value of each block.

TABLE 7. Comparison of storage costs.

Scheme	TPA side	CSP side
Scheme [12]	0	$|p|(2^{logn+1} + n - 1)$
Scheme [13]	0	$|p|(2^{logn+1} + n - 1)$
Scheme [14]	$(n+1)l_1$	$2n|p|$
Scheme [17]	$(n+1)l_2$	$2n|p|$
Our Scheme	nl_3	$2n|p|$

C. COMPUTATION COSTS

In this section, we will evaluate the computation time of the scheme with experiments and compare it with the Zhu's scheme [14]. The scheme simulates on a Linux system with an Intel Core i5 1.60GHz processor and 1G RAM. The Pairing based Cryptography (PBC) library of version 0.5.14 is used to implement our simulation. Furtherly, in the experiment, an MNT d159 curve with 160-bit group order is utilized. All the experiment results represent the average of 20 trials.

1) Computation time of the user in setup phase
   In our experiment of setup phase, the computation time by different numbers of blocks is tested with the max block size of 1KB. From fig. 5, it can be concluded that the user's computation time is proportional to the number of blocks, and the computation cost of our protocol is lower than Zhu's.

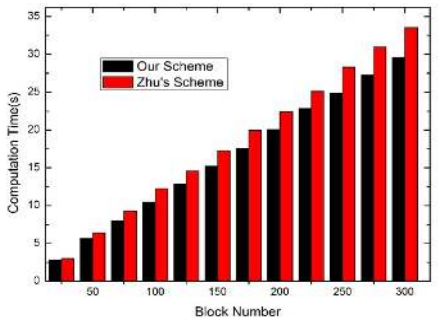


FIGURE 5. Computation time with block number in setup phase.

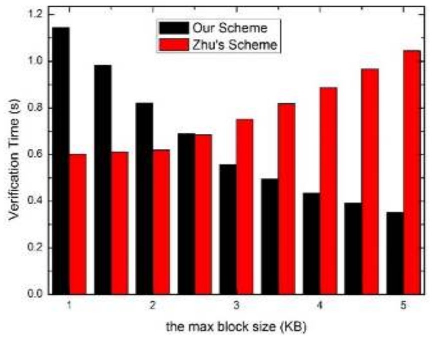


FIGURE 6. Verification time with different block size.

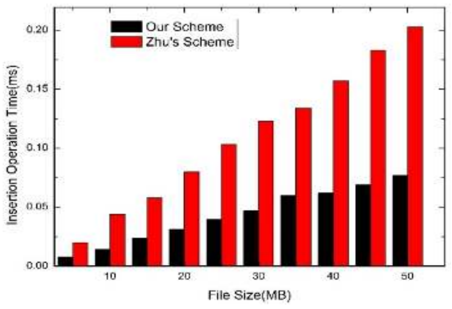


FIGURE 7. Insertion time with file size.

2) Computation Cost in Verification Phase
In verification phase, the relationship between computation costs and the block size is tested with the same file size of 1MB. In the simulation, the challenged block number accounts for 20% of the total block number. From fig. 6, it can be concluded that with the block size increasing, the verification cost of our scheme is decreasing. However, the verification time in Zhu's scheme is increasing, because the verification equation in Zhu's scheme has relation with the sectors of each block.

3) Computation cost in update phase
In the experiment of update phase, it supposes that the max block size is 1KB. The update time with file size from 1MB to 50MB is tested respectively. From Fig. 7 and Fig. 8, it can be conclude that either in insertion operation or in deletion operation, our scheme is more efficient. In Zhu's scheme, as IHT is a sequence data structure, it would cause an average of half adjustment of elements, resulting in the update operation efficiency's decrease. While in our scheme, only the static pointer needs to be changed without any moving of elements.

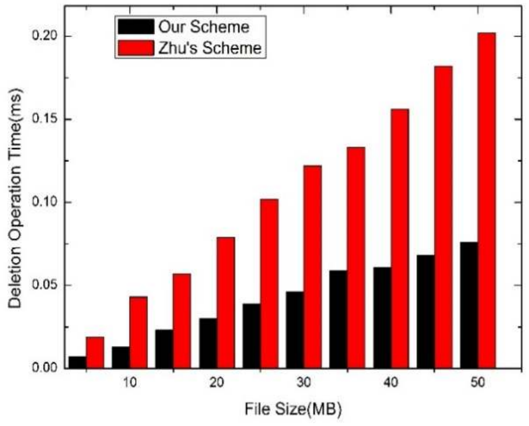


FIGURE 8. Deletion time with file size.

VIII. CONCLUSION

The paper proposes an efficient dynamic auditing scheme for outsourced data in cloud servers. In the scheme, a dynamic index table (DIT) where no elements need to be moved in insertion or deletion update operations is designed to improve data update efficiency. Furtherly, when file in cloud is not integrated, TPA can detect and recover the corrupt block. Moreover, an authorization is used between users and cloud servers to prevent denial of service attack. The scheme can achieve authorized and efficient secure integrity verification for big data in clouds and the simulation results demonstrate that the scheme costs less communication and computation than the previous schemes.

For further work, we should point out that the efficiency and security of the integrity verification scheme can be furtherly developed, because they are most important issues in cloud storage of big data. For efficiency, we should minimize the communication costs between users and cloud servers to improve integrity verification speed. Moreover, storage cost in cloud server should also be considered. For security, the privacy of user data should be emphasized, because privacy is another key point in data security of cloud computing. Efficiency and security are two important directions of our future work.

REFERENCES

[1] M. Armbrust, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and I. Stoica, "Above the clouds: A berkeley view of cloud computing," Univ. California at Berkeley, Berkeley, CA, USA, Tech. Rep. UCBEECS, Feb. 2009, pp. 1–23, vol. 28.

[2] X. Lu and X. Cheng, "A secure and lightweight data sharing scheme for Internet of medical things," *IEEE Access*, vol. 8, pp. 5022–5030, 2020, doi: 10.1109/ACCESS.2019.2962729.

[3] Y. Zhang, M. Qiu, C.-W. Tsai, M. M. Hassan, and A. Alamri, "Health-CPS: Healthcare cyber-physical system assisted by cloud and big data," *IEEE Syst. J.*, vol. 11, no. 1, pp. 88–95, Mar. 2017.

[4] Z. Guan, Z. Lv, X. Du, L. Wu, and M. Guizani, ''Achieving data utility-privacy tradeoff in Internet of medical things: A machine learning approach,'' *Future Gener. Comput. Syst.*, vol. 98, pp. 60–68, Sep. 2019.

[5] V. Chang, ''Towards data analysis for weather cloud computing,'' *Knowl.-Based Syst.*, vol. 127, pp. 29–45, Jul. 2017.

[6] A. B. Lewko and B. Waters, ''Decentralizing attribute-based encryption,'' in *Proc. 30th Annu. Int. Conf. Theory Appl. Cryptogr. Techn.*, Tallinn, Estonia, May 2011, pp. 568–588.

[7] Y. Deswarte, J. J. Quisquater, and A. Saïdane, ''Remote integrity checking,'' in *Proc. 6th Working Conf. Integrity Internal Control Inf. Syst. (IICIS)*, Nov. 2004, pp. 1–11.

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

[9] G. Yamamoto, S. Oda, and K. Aoki, ''Fast integrity for large data,'' *Proc. ECRYPT Workshop Softw. Perform. Enhancement Encryption Decryption*, pp. 21–32, 2007.

[10] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, ''Provable data possession at untrusted stores,'' in *Proc. 14th ACM Conf. Comput. Commun. Secur. CCS*, 2007, pp. 598–610.

[11] G. Ateniese, R. Di Pietro, L. V. Mancini, and G. Tsudik, ''Scalable and efficient provable data possession,'' in *Proc. 4th Int. Conf. Secur. Privacy Commun. Netowrks SecureComm*, 2008, pp. 1–10.

[12] C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, ''Dynamic provable data possession,'' in *Proc. 16th ACM Conf. Comput. Commun. Secur. CCS*, 2009, pp. 213–222.

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

[14] Y. Zhu, G.-J. Ahn, H. Hu, S. S. Yau, H. G. An, and C.-J. Hu, ''Dynamic audit services for outsourced storages in clouds,'' *IEEE Trans. Services Comput.*, vol. 6, no. 2, pp. 227–238, Apr. 2013.

[15] K. Yang and X. Jia, ''An efficient and secure dynamic auditing protocol for data storage in cloud computing,'' *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 9, pp. 1717–1726, Sep. 2013.

[16] C. Liu, J. Chen, L. T. Yang, X. Zhang, C. Yang, R. Ranjan, and K. Rao, ''Authorized public auditing of dynamic big data storage on cloud with efficient verifiable fine-grained updates,'' *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 9, pp. 2234–2244, Sep. 2014.

[17] H. Tian, Y. Chen, C.-C. Chang, H. Jiang, Y. Huang, Y. Chen, and J. Liu, ''Dynamic-Hash-Table based public auditing for secure cloud storage,'' *IEEE Trans. Services Comput.*, vol. 10, no. 5, pp. 701–714, Sep. 2017, doi: 10.1109/TSC.2015.2512589.

[18] Q. Gan, X. Wang, and X. Fang, ''Efficient and secure auditing scheme for outsourced big data with dynamicity in cloud,'' *Sci. China Inf. Sci.*, vol. 61, no. 12, pp. 93–107, Dec. 2018.

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

[20] X. Lu, Z. Pan, and H. Xian, ''An integrity verification scheme of cloud storage for Internet-of-things mobile terminal devices,'' *Comput. Secur.*, vol. 92, May 2020, Art. no. 101686, doi: 10.1016/j.cose.2019.101686.

[21] Q. Zhang, L. T. Yang, and Z. Chen, ''Privacy preserving deep computation model on cloud for big data feature learning,'' *IEEE Trans. Comput.*, vol. 65, no. 5, pp. 1351–1362, May 2016.

[22] Y. Ming and T. Zhang, ''Efficient privacy-preserving access control scheme in electronic health records system,'' *Sensors*, vol. 18, no. 10, p. 3520, Oct. 2018.

[23] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, ''Privacy-preserving public auditing for secure cloud storage,'' *IEEE Trans. Comput.*, vol. 62, no. 2, pp. 362–375, Feb. 2013.

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

[25] J. Yu and R. Hao, ''Comments on 'SEPDP: Secure and efficient privacy preserving provable data possession in cloud storage,''' *IEEE Trans. Serv. Comput.*, early access, Mar. 29, 2019, doi: 10.1109/TSC.2018.2820713.

[26] Q. Zhou, C. Tian, H. Zhang, J. Yu, and F. Li, ''How to securely outsource the extended Euclidean algorithm for large-scale polynomials over finite fields,'' *Inf. Sci.*, vol. 512, pp. 641–660, Feb. 2020, doi: 10.1016/j.ins.2019.10.007.

[27] D. Halperin, T. Kohno, T. S. Heydt-Benjamin, K. Fu, and W. H. Maisel, ''Security and privacy for implantable medical devices,'' *IEEE Pervasive Comput.*, vol. 7, no. 1, pp. 30–39, Jan./Mar. 2008.

[28] Y. Li, H. Xia, R. Zhang, B. Hu, and X. Cheng, ''A novel community detection algorithm based on paring, splitting and aggregating in Internet of Things,'' *IEEE Access*, vol. 8, pp. 123938–123951, 2020.

[29] G. Yang, L. Xie, M. Mantysalo, X. Zhou, Z. Pang, L. D. Xu, S. Kao-Walter, Q. Chen, and L.-R. Zheng, ''A health-IoT platform based on the integration of intelligent packaging, unobtrusive bio-sensor, and intelligent medicine box,'' *IEEE Trans. Ind. Informat.*, vol. 10, no. 4, pp. 2180–2191, Nov. 2014.

[30] J. Zhang, F. Ren, S. Gao, H. Yang, and C. Lin, ''Dynamic routing for data integrity and delay differentiated services in wireless sensor networks,'' *IEEE Trans. Mobile Comput.*, vol. 14, no. 2, pp. 328–343, Feb. 2015.

[31] J. Lai, R. H. Deng, C. Guan, and J. Weng, ''Attribute-based encryption with verifiable outsourced decryption,'' *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 8, pp. 1343–1354, Aug. 2013.

[32] H. Shacham and B. Waters, ''Compact Proofs of Retrievability,'' *Proc. 14th Int. Conf. Theory Appl. Cryptol. Inf. Secur., Adv. Cryptol.*, 2008, pp. 90–107.

[33] D. Cash, A. Kupcu, and D. Wichs, ''Dynamic proofs of retrievability via oblivious RAM,'' in *Proc. EUROCRYPT*, 2013, pp. 279–295.

[34] K. S. Kim and I. R. Jeong, ''Efficient verifiable data streaming,'' *Secur. Commun. Netw.*, vol. 8, no. 18, pp. 4013–4018, Dec. 2015.

[35] H. Cai, B. Xu, L. Jiang, and A. V. Vasilakos, ''Iot-based big data storage systems in cloud computing: Perspectives and challenges,'' *IEEE Internet Things J.*, vol. 4, no. 1, pp. 75–87, Jan. 2017.

[36] C. Hu, W. Li, X. Cheng, J. Yu, S. Wang, and R. Bie, ''A secure and verifiable access control scheme for big data storage in clouds,'' *IEEE Trans. Big Data*, vol. 4, no. 3, pp. 341–355, Sep. 2018.

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

[38] Y. Zhang, D. Zheng, and R. H. Deng, ''Security and privacy in smart health: Efficient policy-hiding attribute-based access control,'' *IEEE Internet Things J.*, vol. 5, no. 3, pp. 2130–2145, Jun. 2018.

[39] X. Liu, J. Ma, J. Xiong, T. Zhang, and Q. Li, ''Personal health records integrity verification using attribute based proxy signature in cloud computing,'' in *Internet and Distributed Computing Systems* (Lecture Notes in Computer Science), vol. 8223, 2013, pp. 238–251.

[40] H. Jin, K. Zhou, H. Jiang, D. Lei, R. Wei, and C. Li, ''Full integrity and freshness for cloud data,'' *Future Gener. Comput. Syst.*, vol. 80, pp. 640–652, Mar. 2018.

HAN YU received the master's degree in cyberspace security from the National University of Defense Technology, China, in 2019. His research interests include information security and deep learning.

XIUQING LU received the M.S. degree from the College of Computer Science, Shandong University, China. She is currently an Assistant Professor with the Computer Science Technology College, Qingdao University, China. Her current research interests include security of cloud computing and privacy of big data.

ZHENKUAN PAN received the M.S. and Ph.D. degrees from Shanghai Jiao Tong University. He is currently a Professor with the Computer Science Technology College, Qingdao University, China. His main research interests include virtual reality technology and computer vision.