

Received January 9, 2020, accepted January 23, 2020, date of publication February 4, 2020, date of current version February 11, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2971519

Distributed Machine Learning Oriented Data Integrity Verification Scheme in Cloud Computing Environment

XIAO-PING ZHAO^{ID} AND RUI JIANG^{ID}

School of Cyber Science and Engineering, Southeast University, Nanjing 210096, China

Corresponding author: Rui Jiang (r.jiang@seu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61372103, in part by the Key Project of Chinese National Programs for Fundamental Research and Development (973 program) under Grant 2013CB338003, in part by the Key Laboratory of Information Network Security of Ministry of Public Security under Grant C19607, and in part by the Key Laboratory of Computer Network Technology of Jiangsu Province.

ABSTRACT Distributed Machine Learning (DML) is one of the core technologies for Artificial Intelligence (AI). However, in the existing distributed machine learning framework, the data integrity is not taken into account. If network attackers forge the data, modify the data, or destroy the data, the training model in the distributed machine learning system will be greatly affected, and the training results are led to be wrong. Therefore, it is crucial to guarantee the data integrity in the DML. In this paper, we propose a distributed machine learning oriented data integrity verification scheme (DML-DIV) to ensure the integrity of training data. Firstly, we adopt the idea of Provable Data Possession (PDP) sampling auditing algorithm to achieve data integrity verification so that our DML-DIV scheme can resist forgery attacks and tampering attacks. Secondly, we generate a random number, namely blinding factor, and apply the discrete logarithm problem (DLP) to construct proof and ensure privacy protection in the TPA verification process. Thirdly, we employ identity-based cryptography and two-step key generation technology to generate data owner's public/private key pair so that our DML-DIV scheme can solve the key escrow problem and reduce the cost of managing the certificates. Finally, formal theoretical analysis and experimental results show the security and efficiency of our DML-DIV scheme.

INDEX TERMS Distributed machine learning, public auditing, data integrity, bilinear mapping, identity-based cryptography.

I. INTRODUCTION

Artificial Intelligence (AI) has become a hot research spot in academia and IT industry in recent years. AI can help to solve various problems in people's real life, such as shopping recommendation, navigation, face recognition, and automatic driving. Hence, the research on artificial intelligence has important theoretical value and practical significance.

Machine Learning (ML), as the core technology of AI, is the fundamental way to make computers and networks intelligent. The application of machine learning spans all fields of artificial intelligence. For example, face recognition, navigation, and automatic driving can be realized through machine learning technology.

The associate editor coordinating the review of this manuscript and approving it for publication was Chi-Yuan Chen^{ID}.

Due to the poor efficiency of traditional machine learning technology, this technology cannot deal with large data, especially when the training data reaches the Peta Byte (PB) level or even larger. In order to solve the problem, some well-known companies such as Google and Microsoft have set up large-data-based machine learning and artificial intelligence research institutions to do the further research on distributed machine learning technology. Meanwhile, the Chinese Computer Society also treats the distributed machine learning technology as an important research area and the trend of big data.

A. RELATED WORK

1) DISTRIBUTED MACHINE LEARNING

Recently, the framework of distributed machine learning technology mainly includes the MapReduce-based system,

the graph model-based abstraction system, and the parameter server system. MapReduce-based systems, which include Hadoop [1] and Spark [2], are widely implemented with their practical algorithms. However, the two systems run slowly than other specially designed distributed machine learning algorithms. Graph model-based abstraction systems, which include GraphLab [3] and Pregel [4], apply better parallel machine learning algorithms and flexible computation scheduling. However, in the systems, it is difficult to abstract the representation of the graph and rewrite the pattern of the graph. Also, the graph model-based abstraction system is more complicated than other systems. Compared with the previous two types of systems, the parameter server system architecture [5]–[9] can be efficiently and flexibly adopted to the parallel strategy of multiple machine learning algorithms. Moreover, with the open source, more and more people and companies use the structure of parameter server system.

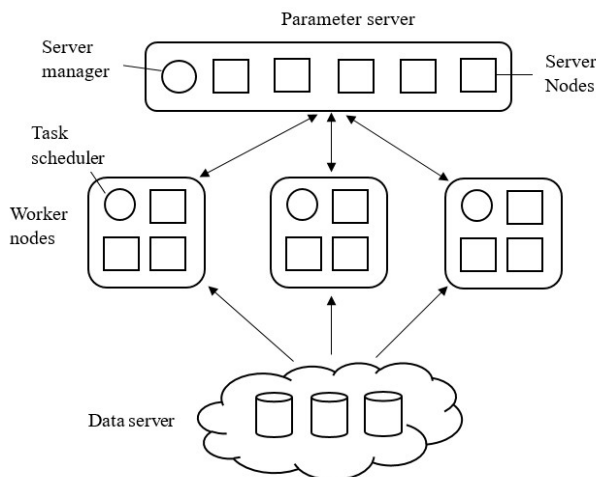


FIGURE 1. System model of the parameter server.

As shown in Fig. 1, the parameter server system consists of three entities, which are the Data server (DS), Worker nodes (WNs), Parameter server (PS).

In the parameter server system, multiple worker nodes perform calculations at the same time to ensure the high parallel efficiency for the system. Moreover, the training data are stored in the data server so as to reduce worker nodes' overhead of local storage. Hence, the parameter server system has become the mainstream framework for distributed machine learning.

However, in the parameter server system, it is not taken into account for the integrity of training data in DS. If network attackers forge the training data, modify the training data, or destroy the training data, the training model in the parameter server system will be greatly affected, and the training results are led to wrong. Therefore, it is extremely important to protect the integrity of training data.

2) DATA INTEGRITY PROTECTION SCHEME IN CLOUD COMPUTING

In cloud environment, the Provable Data Possession (PDP) integrity verification scheme is most effective. The PDP scheme adopts the sampling auditing, so it does not need to download all the data when performing data integrity verification. Thus, communication overhead is greatly reduced. In 2007, Atenies *et al.* [10] proposed first public sampling auditing scheme, named Provable Data Possession (PDP), with Homomorphic tag based on RSA. However, the first PDP scheme did not support dynamic auditing and data privacy protection. In 2008, Atenies *et al.* [11] further proposed their dynamic auditing scheme. However, their scheme was only partially dynamic, and did not provide data privacy protection. In 2009, full dynamic PDP scheme was proposed by Erway *et al.* [12].

However, in the PDP scheme, there is a data privacy protection problem, which means the privacy of data will be disclosed in the TPA verification process. In order to achieve privacy protection, researchers proposed many privacy-preserving public auditing schemes based on PDP. In 2010, first privacy preserving PDP scheme was proposed by Wang *et al.* [13]. In their scheme, the authors applied integrating Homomorphic authenticator with the random masking technique to ensure the data privacy protection. In 2012, Zhu *et al.* [14] proposed a privacy-preserving public auditing scheme to support dynamic data operation. In 2013, Wang *et al.* [15] proposed a privacy-preserving public auditing scheme with a random masking technique to blind the response data. In 2017, Yu *et al.* [22] applied the concept of perfect data privacy-preserving for remote cloud storage auditing and proposed a new construction of ID-based protocol with key-homomorphic cryptographic primitive. Yan *et al.* [16] proposed a Remote Data Possession Checking (RDPC), which could withstand forgery attack, replace attack, and replay attack. However, the RDPC scheme did not support privacy protection. In 2018, Sookhak *et al.* [17] proposed a different authentication data structure with D&CT. In their D&CT structure, the authors introduced the logical subscript (LI) and version number (VN) of the data block to proficiently support dynamic data for normal file sizes. The authors also applied large-scale data storage to greatly reduce the computation and communication overhead for auditors and cloud servers. However, the authors did not consider the data privacy in the auditing process. In 2019, Zhao *et al.* [18] proposed a user stateless privacy-preserving public auditing scheme with the rank-based authenticated skip list. Yan *et al.* [19] proposed a public auditing scheme with the designated verifier. The data owner could designate one trusted person to check data. However, the scheme did not support privacy protection.

Also, most PDP schemes rely on Public Key Infrastructure (PKI), which may cause the key escrow problem and the certificate management overhead. In PKI, the Key Generate Center (KGC) is required to manage and store all users'

public/private key pairs, hence the KGC may control all public/private key pairs, which may cause the key escrow problem. To solve the key escrow problem and reduce the cost of managing the certificates, in 2015, Wang [20] proposed an identity-based distributed provable data possession scheme in multi-cloud storage to implement data integrity verification. In 2016, Wang *et al.* [21] proposed a proxy-oriented data uploading and remote data integrity checking model with identity-based public key cryptography. In 2016, Zhang *et al.* [22] proposed an ID-based public auditing scheme in multi-user scenario. However, Zhang's scheme did not resolve key escrow problem, and could not support privacy protection. In 2017, Yu *et al.* [23] proposed a remote data integrity auditing scheme with perfect data privacy preserving with identity-based cryptosystems. However, Yu's scheme relied on PKI. In 2018, Li *et al.* [24] proposed a certificateless public integrity checking scheme. The scheme released certificate management, and resolved key escrow problem by adopting certificateless signature and two-part private key. However, in the process of user revocation, computation cost was relatively huge. In 2019, Shen *et al.* [25] proposed their scheme to solve the problem of sensitive information sharing in the process of data integrity auditing. The proposed auditing scheme could realize the secure sharing of sensitive information. However, the scheme relied on PKI. Li *et al.* [26] proposed an efficient identity-based provable multi-copy data possession scheme to release heavy communication and computational cost caused by PKI. Zhu *et al.* [27] proposed a secure and efficient data integrity verification scheme for Cloud-IoT to reduce the computational overhead. However, the scheme did not resolve the key escrow problem.

B. OUR CONTRIBUTIONS

In order to protect the integrity of training data in distributed machine learning system, in this paper, we propose the distributed machine learning oriented data integrity verification scheme (DML-DIV). To the best of our knowledge, our DML-DIV scheme is the first scheme in distributed machine learning area to apply public sampling auditing algorithm and ensure the integrity of training data. The main contributions of our DML-DIV scheme are as follows.

(1) In our DML-DIV scheme, we propose a data integrity verification scheme based on distributed machine learning. Our DML-DIV scheme can ensure the integrity of training data, resist forgery attack and tampering attack. In DML-DIV scheme, we first employ data owner's private key to construct data signature, then apply sampling technique and PDP technology to generate proof, finally apply bilinear mapping algorithm to verify the proof.

(2) Our DML-DIV scheme can guarantee the privacy of training data during the public sampling auditing process. Firstly, the data server adds a blinding factor to proof. Then the data server encrypts blinding factor, and sends the proof and blinding factor to the TPA. Finally, based on the discrete logarithm problem, the TPA and Network Attackers cannot

decrypt the blinding factor to disclose the secret of training data.

(3) Our DML-DIV scheme can solve the key escrow problem and reduce the cost of managing the certificates. In DML-DIV scheme, we adopt identity-based cryptography algorithm to construct data owner's public/private key pair, so that all entities can verify the data owner's public key without certificates. Firstly, the KGC generates partial long-term private key for the data owner. Then, the data owner generates his private key with partial long-term private key. Finally, the KGC cannot figure out the data owner's private key with partial long-term private key.

(4) We formally prove DML-DIV scheme is correct, can resist forgery and tampering attack, ensure the privacy protection in the TPA verification process, and solve the key escrow problem. In addition, we evaluate the performances of our DML-DIV scheme and show the efficiency of our DML-DIV scheme.

The rest of paper is organized as follows. In section 2, we introduce some basic knowledge and assumptions for this paper. Our DML-DIV scheme is proposed in section 3. In section 4, we give formal proof on our scheme to ensure the security goals. In section 5, we make performance analysis on our scheme. Finally, we make a conclusion in section 6.

II. PRELIMINARIES

In this section, we introduce the basic mathematic algorithms and problems applied in the following paper.

A. BILINEAR MAPPING [28]

Definition 1: Let G_1 and G_2 be two multiplicative cyclic groups of a large prime order p , a pairing is a bilinear map $e : G_1 \times G_1 \rightarrow G_2$. It satisfies the following properties:

Bilinear: $e(u^a, v^b) = e(u, v)^{ab}$, $\forall u, v \in G_1$; $a, b \in \mathbb{Z}_p$.

Non-Degeneracy: $\exists u, v \in G_1$, thus $e(u, v) \neq 1 \in G_1$.

Computability: $\forall u, v \in G_1$, there is a polynomial time algorithm to calculate $e(u, v)$.

Safety: It is difficult to calculate the discrete logarithm problem in G_1 and G_2 .

B. DISCRETE LOGARITHM PROBLEM(DLP) [28]

Definition 2: Let $\alpha \in \mathbb{Z}_p^*$, and G_1 be a multiplicative cyclic group, known $g, g^\alpha \in G_1$, for any polynomial time adversary, the advantage of solving the value α is negligible. The probability of Probabilistic Polynomial Time (PPT) algorithm A successfully solving the DLP is $Adv^{DL} = \Pr[A(g, g^\alpha) = \alpha : \alpha \in \mathbb{Z}_p^*]$, which is negligible. The probability comes from the random selection of α on \mathbb{Z}_p^* and the random selection of algorithm A .

C. COMPUTATIONAL DIFFIE-HELLMAN PROBLEM(CDHP) [29]

Definition 3: Let $a, b \in \mathbb{Z}_p^*$, and G_1 be a multiplicative cyclic group, given $P, P^a, P^b \in G_1$, for any polynomial time adversary, it is feasible in calculation to solve $P^{ab} \in G_1$. The probability of Probabilistic Polynomial Time (PPT)

algorithm A successfully solving the CDH problem is $Adv_{CDHP} = \Pr[A(P, P^a, P^b) = P^{ab} : a, b \in Z_p^*]$, which is negligible. The probability comes from the random selection of a, b on Z_p^* and the random selection of algorithm A .

D. CO-COMPUTATIONAL BILINEAR DIFFIE-HELLMAN PROBLEM(CO-CDH) [30]

Definition 4: Let $a \in Z_p^*$, and G_1, G_2 be two multiplicative cyclic groups, given $P, P^a \in G_1, Q \in G_2$, for any polynomial time adversary, it is feasible in calculation to solve $Q^a \in G_2$. The probability of Probabilistic Polynomial Time (PPT) algorithm A successfully solving the Co-CDH problem is $Adv_{Co-CDH} = \Pr[A(P, P^a, Q) = Q^a : a \in Z_p^*]$, which is negligible.

E. RANDOM ORACLE MODEL [30]

Definition 5: Existential unforgeability of data owner's private key under security game is defined as follows:

Setup: The challenger runs Join algorithm and sends public key to the adversary A , and keeps secret key.

Queries: The adversary A can ask two queries to the challenger.

a) Hash Queries: The adversary A requests hash value based on an identity at most p_H times of his choice $ID_1, \dots, ID_{p_H} \in \{0, 1\}^*$. The challenger computes Q_{ID} , and sends hash value to A .

b) Private key queries: The adversary A requests private key based on an identity at most times of his choice $ID_1, \dots, ID_{p_s} \in \{0, 1\}^*$. The challenger computes S_{ID} , and sends private key to A .

Output: The adversary A outputs a pair (S_{ID}, Pk, ID) , where ID is not any of ID_1, \dots, ID_{p_H} . The adversary A wins the game if $Verify(S_{ID}, Pk, ID) = valid$ holds.

We define Adv_A to be the probability that A wins in the above game.

Definition 6: The privacy protection of training data under security game is defined as follows:

Setup: The challenger C runs the Setup algorithm to generate public parameter $u \in G_1$, where G_1 is a multiplicative cyclic group. We assume $e(H_1(X), syPQ)$ is the data server's private key, and $R = u^r$ is the corresponding public key.

Queries: The adversary A queries the public key of the random number $c \in Z_p$.

Output: The adversary A can obtain the data server's private key r' .

We let $Adv = \Pr[r' = r] - 1/2$. We say the TPA and Network attacker cannot obtain the data if the function Adv is negligible for any polynomial-time adversary A .

III. OUR DML-DIV SCHEME

A. SYSTEM MODEL

In our DML-DIV scheme, as shown in Fig.2, the system model consists of four entities which are the Data Server (DS), Third Party Auditor (TPA), Data Owner (DO) and Key Generate Center (KGC).

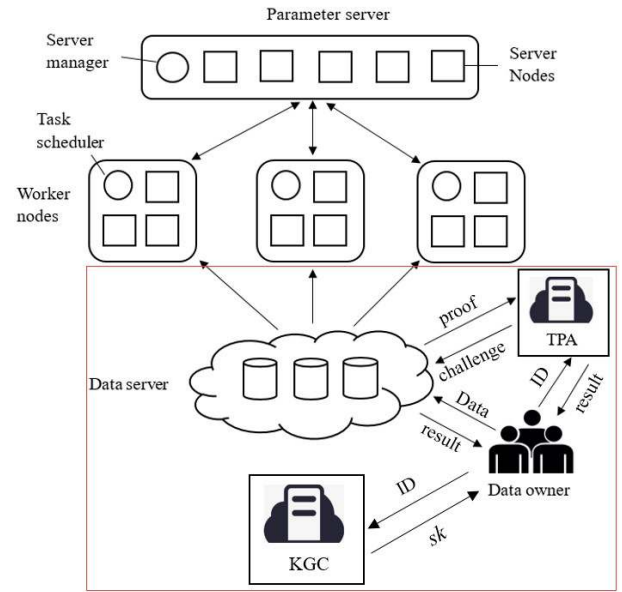


FIGURE 2. System model of our DML-DIV Scheme.

KGC: The KGC is an authority that manages the data owner's partial secret key. First, the KGC generates master secret key and public key. After receiving ID from the data owner, the KGC generates corresponding partial secret key for the data owner with the data owner's ID and master key.

DO: The data owner is responsible for collecting training data and uploading them to the data server. The training data may come from computers or mobile terminals.

DS: The data server is a cloud service provider that stores the training data from data owner. The data server accepts the challenge initiated by the TPA, generates proof, and sends proof to the TPA. The data server runs the protocol faithfully and proves that training data is stored correctly and completely.

TPA: Third party auditor (TPA) can verify the integrity of the training data stored in the data server. The TPA initiates a challenge to the data server, and receives a response from the data server. The TPA can carry out public auditing with the data owner's public key.

B. AN APPLICATION EXAMPLE

Based on the system model, our DML-DIV scheme can be applied in distributed machine learning algorithms for prediction, recommendation and classification. One of the examples is advertising recommendation application, which is illustrated in Fig.3.

In advertising recommendation application, the users are data owners who register a web application, data server is the Amazon service provider who stores the user's data, the KGC is an application developer who generates the user's key, and the TPA is a trusted third party who verifies the integrity of data stored in Amazon.

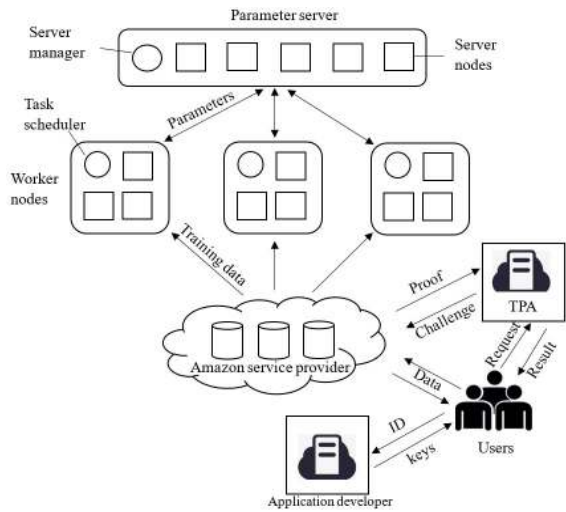


FIGURE 3. Advertising recommendation application.

Firstly, the application developer runs Setup algorithm to generate and distribute private/public keys, and determine initial parameters. Secondly, the users generate signatures of advertisement clicking data, and upload data and signatures to Amazon service provider. Thirdly, the TPA sends challenge to Amazon service provider, and Amazon service provider sends proof to the TPA to verify the integrity of data. Fourthly, the TPA sends the results to users and worker nodes. Once the data is lost or damaged, worker nodes stop working. Then, worker nodes load data from Amazon, run Distributed Delayed Block Proximal Gradient Algorithm to obtain parameters, and push parameters to parameter server. Hence, the parameter server updates the original parameters with parameters from worker nodes. Finally, the application recommends favorite advertisement to the users based on the training results.

The distributed machine learning algorithm [8] that we use to train model is as follow:

C. THREAT MODEL

In our DML-DIV scheme, we define the threat model in terms of the Key Generate Center (KGC), Data Owner (DO), Data Server (DS), Third Party Auditor (TPA), Network Attackers (NA).

KGC: The KGC is semi-honest, which means it will generate the data owner's partial key, but it may figure out data owner's partial key.

DO: The data owner honestly collects training data and uploads training data to the data server.

DS: The data server is semi-honest. That is, it performs operations in accordance with the agreement with the data owner. But in order to maintain the reputation, it may conceal to data owner when the training data is lost or corrupted. At that time, the data server may attempt to forge proof to pass the auditing of the TPA.

TPA: The TPA is credible, and faithfully enforces the agreement. However, the TPA is curious about training data and tries to obtain the training data in the verification process.

Algorithm 1 Delayed Block Proximal Gradient

Task Scheduler:

- 1: part features into b ranges R_1, \dots, R_b
- 2: issue LoadData() to all workers
- 3: for iteration $t = 1, \dots, T$ do
- 4: pick random range R_i
- 5: issue WorkerIterate(t) to all workers
- 6: end for

Worker $r = 1, \dots, m$:

- 1: wait until all iterations before $t - \tau$ are finished
- 2: function LoadData()
- 3: load a part of training data
- 4: end function
- 5: function WorkerIterate(t)
- 6: compute first-order gradient $g_r^{(t)}$ and diagonal second
- 7: order gradient $u_r^{(t)}$ on range R_i
- 8: push $g_r^{(t)}$ and $u_r^{(t)}$ to parameter servers
- 9: pull $w_r^{(t+1)}$ from parameter servers
- 10: end function

Parameter Servers:

- 1: function ServerIterate(t)
- 2: aggregate $g^{(t)} \leftarrow \sum_{r=1}^m g_r^{(t)}$ and $w^{(t)} \leftarrow \sum_{r=1}^m w_r^{(t)}$
- 3: $\omega^{(t+1)} \leftarrow \arg \min_u \Omega(u) + \frac{1}{2\eta} \|w^{(t)} - \eta g^{(t)} + u\|_H^2$
- 4: where $H = \text{diag}(h^{(t)})$ and $\|x\|_H^2 = x^T H x$
- 5: end function

NA: Network Attackers attempt to obtain the private key of the data owner, and disclose the secret of training data by intercepting the proof. In addition, Network Attackers attempt to tamper with and forge data proof and signature proof, and they may launch the tempering attack and forgery attack.

D. NOTATION

The symbols in our DML-DIV scheme are shown in the Table 1.

TABLE 1. Symbols explanation.

Symbol	Quantity
m	the number of training data blocks
i	the index of training data blocks
p	a large prime
I	the subset of index of training data blocks
$name$	the name of training data
P	an arbitrary generator of multiplicative cyclic group G_1
s	the master key
r	a random element from G_1

E. DETAILS OF OUR DML-DIV SCHEME

In this section, we propose the detail construction of our DML-DIV scheme. The scheme includes six steps: Setup,

Key extra, Tag generation, Challenge, Proof generation, and Proof verification.

1) SETUP

Let G_1 and G_2 be two multiplicative cyclic groups with the same prime order p , $e : G_1 \times G_1 \rightarrow G_2$ is an efficiently computable bilinear map, and u is an element randomly selected from G_1 . Two one-way hash functions are $H_1(\cdot) : \{0, 1\}^* \rightarrow Z_p^*$ and $H_2(\cdot) : \{0, 1\}^* \rightarrow G_1$. In RSA algorithm, d is the data owner's private key, c is the data owner's public key, and n is the modulus.

The KGC chooses an arbitrary generator $P \in G_1$, randomly selects a master key $s \in Z_p^*$, and computes $P_{pub} = P^s \in G_1$ treated as a master public key. The system parameters are $params = \{G_1, G_2, e, u, P, P_{pub}, H_1, H_2, (c, n)\}$.

2) KEY EXTRA

The data owner sends its identity $ID \in \{0, 1\}^*$ to the KGC to generate partial private key. Firstly, the KGC computes $Q = H_2(ID)$, and applies the master key s to compute the data owner's partial private key $D = Q^s$. The KGC sends the partial private key D to the data owner.

Secondly, the data owner can verify the correctness of partial private key from the KGC, as follows:

$$e(D, P) \stackrel{?}{=} e(Q, P_{pub}) \quad (1)$$

If the above equation (1) is correct, it indicates the partial private key received by the data owner from the KGC is correct. Otherwise, the data owner discards it and terminates the operation.

Thirdly, if the partial private key is correct, the data owner randomly selects a secrete value $x \in Z_p^*$. Then, the data owner computes $sk = H_1(D^x)$ as its private key.

Finally, the data owner computes $Z = P_{pub}^x = P^{sx}$, $X = P^{sk}$ and $Y = ((H_2(X||Z)) \cdot Q)^x$. The data owner sets $pk = \langle X, Y, Z \rangle$ as its public key.

All entities can use public parameters $params$ and the data owner's identity ID to verify the public key of the data owner, as follows:

$$e(Y, P_{pub}) \stackrel{?}{=} e(H_2(X||Z) \cdot Q, Z) \quad (2)$$

If the equation (2) holds, it is true that $pk = \langle X, Y, Z \rangle$ is the correct public key for the data owner.

3) TAG GENERATION

Firstly, the data owner collects training data, and generates corresponding signatures. The training data is $F = \{F_1, F_2, \dots, F_m\}$, and the corresponding signature for data F_i is $\sigma_i = (H_2(W_i) \cdot u^{F_i})^{sk}$, where $W_i = name||i$ ($i = 1, 2, \dots, m$), and the $name$ is chosen by the data owner uniformly at random from Z_p . Meanwhile, the data owner applies RSA signature algorithm to compute $SIG_{sk}(name||m) = (name||m)^d \bmod n$, where d is the data owner's private key in RSA signature algorithm, and n is the

modulus in RSA signature algorithm. The data owner computes $Sig_F = name||m||SIG_{sk}(name||m)$ as the signature of training data.

Secondly, the data owner uploads training data and signatures $\{F, \Psi, Sig_F\}$ to the data server and deletes them in local storage, where $\Psi = \{\sigma_i\}_{i=1,2,\dots,m}$.

Thirdly, after receiving training data from the data owner, the data server generates proof based on the uploaded data and signatures, and returns the proof to the data owner to prove training data is received and stored integrally. The proof is as follows:

$$\sigma = \prod_{i \in [1, m]} \sigma_i^{v_i}$$

$$\mu = \sum_{i \in [1, m]} v_i F_i + r$$

where $v_i, i = 1, 2, \dots, m$ is a random value in Z_p . Simultaneously, the data server calculates $R_1 = u^r$, and sends R_1 to the data owner, where $r \in Z_p$ is a random number (blinding factor). The data server will pass $\{\sigma, \mu, R_1, H_2(W_i), v_i\}_{i \in [1, m]}$ back to the data owner. And the data owner computes $R = R_1^{sk} = u^{r \cdot sk}$.

Finally, the data owner verifies the proof returned by the data server, as follows:

$$e(\sigma \cdot R, P) \stackrel{?}{=} e\left(\prod_{i \in [1, m]} H_2(W_i)^{v_i} \cdot u^\mu, X\right) \quad (3)$$

When the above Equation (3) is established, it means that the data server receives and stores the training data completely.

4) CHALLENGE

Firstly, the TPA verifies the integrity of the training data signature Sig_F by checking whether $SIG_{sk}(name||m)$ is a valid signature with the data owner's public key pk . That is, the TPA judges whether the equation $SIG_{sk}(name||m)^c \bmod n \stackrel{?}{=} name||m$ is correct, where c is the data owner's public key in RSA signature algorithm, and n is the modulus in RSA signature algorithm. The TPA aborts the message if the verification fails. Otherwise, the TPA recovers $name||m$, and further generates the following auditing challenge message.

Secondly, the TPA randomly selects challenge subscript set I of training data, where $I \subset [1, m]$. The TPA also chooses a set of random numbers $v_i, i \in I$, where $v_i \in Z_p$.

Finally, the TPA generates the challenge $Q = \{I, v_i, i \in I\}$ and sends the challenge to the data server.

5) PROOF GENERATION

In response to the TPA's challenge, the data server generates the proof based on the challenge Q to ensure the correctness as follows:

$$\sigma = \prod_{i \in I} \sigma_i^{v_i}$$

$$\mu = \sum_{i \in I} v_i F_i + r$$

where $I \subset [1, m]$ and $v_i, i \in I$ come from the challenge Q , and r is a random value in Z_p . Then the data server calculates $R_1 = u^r$, and sends R_1 to the data owner, where $r \in Z_p$ is a random number (blinding factor). The data owner computes $R = R_1^{sk} = u^{r \cdot sk}$, and sends R to the data server. Finally, the data server sends the proof $\{\sigma, \mu, R, H_2(W_i)_{i \in I}\}$ to the TPA and ensures that it stores the training data integrally.

6) PROOF VERIFICATION

After receiving the proof generated by the data server, the TPA verifies the proof to judge whether the training data is intact as follows:

$$e(\sigma \cdot R, P) \stackrel{?}{=} e\left(\prod_{i \in I} H_2(W_i)^{v_i} \cdot u^\mu, X\right) \quad (4)$$

If the above Equation (4) is true, it means that the data server completely stores the training data. Otherwise, it indicates that training data stored in the data server is lost or damaged for some reason.

F. CORRECTNESS

Theorem 1: The data owner can correctly verify the correctness of partial private key generated by the KGC.

Proof: In order to prove that the data owner can verify the correctness of partial private key generated by the KGC, it is equivalent to prove the correctness of Equation (1). Based on the properties of bilinear map, the correctness of Equation (1) can be proved as follows:

$$e(D, P) = e(Q^s, P) = e(Q, P^s) = e(Q, P_{pub})$$

□

Theorem 2: Any entity can verify the data owner's public key with public parameters $params$ and the data owner's identity ID .

Proof: To prove the correctness of the data owner's public key, it is equivalent to prove the correctness of verification Equation (2). Based on the properties of bilinear map, the correctness of Equation (2) can be proved as follows:

$$e(Y, P_{pub}) = e((H_2(X||Z) \cdot Q), P^{xs}) = e((H_2(X||Z) \cdot Q), Z),$$

where $pk = \langle X, Y, Z \rangle$ is the data owner's public key, and P_{pub} is the master public key. □

Theorem 3: In our DML-DIV scheme, the TPA can correctly verify the integrity of training data stored in the data server.

Proof: In order to prove that the data server integrally stores the training data, it is equivalent to prove the correctness of Equation (4). Based on the properties of bilinear map, the correctness of Equation (4) can be proved as follows:

$$\begin{aligned} e(\sigma \cdot R, P) &= e\left(\prod_{i \in I} \sigma_i^{v_i} \cdot R_1^{sk}, P\right) \\ &= e\left(\prod_{i \in I} ((H_2(W_i) \cdot u^{F_i}))^{sk \cdot v_i} \cdot R_1^{sk}, P\right) \end{aligned}$$

$$\begin{aligned} &= e\left(\left(\prod_{i \in I} (H_2(W_i) \cdot u^{F_i})^{sk \cdot v_i} \cdot u^{sk \cdot r}, P\right)\right) \\ &= e\left(\left(\prod_{i \in I} (H_2(W_i) \cdot u^{F_i})^{v_i} \cdot u^r, P^{sk}\right)\right) \\ &= e\left(\left(\prod_{i \in I} H_2(W_i)^{v_i} \cdot u^{\sum_{i \in I} v_i F_i} \cdot u^r, X\right)\right) \\ &= e\left(\left(\prod_{i \in I} H_2(W_i)^{v_i} \cdot u^{\sum_{i \in I} v_i F_i + r}, X\right)\right) \\ &= e\left(\left(\prod_{i \in I} H_2(W_i)^{v_i} \cdot u^\mu, X\right)\right) \end{aligned}$$

□

Theorem 1, Theorem 2, and Theorem 3 prove the correctness of our DML-DIV scheme. Theorem 1 proves the data owner can verify the correctness of partial private key generated by the KGC. Theorem 2 proves any entity can verify the correctness of the data owner's public key. Theorem 3 proves the TPA can correctly verify the integrity of training data stored in the data server.

IV. SECURITY ANALYSIS

A. FORMAL PROOF

In this section, we formally prove that our DML-DIV scheme can resist forgery attack and tampering attack, solve the key escrow problem and prevent the TPA and Network Attackers from disclosing the secret of training data.

Theorem 4: In our DML-DIV scheme, it is computationally infeasible for the data server and Network Attackers to tamper with training data and signatures.

Proof: If any of the challenged training data blocks F_i or corresponding σ_i is corrupted or lost in the data server, the data server cannot pass the TPA's integrity verification.

When the data server wants to launch the tampering attack and tries to pass the TPA's integrity verification, the data server may use other pair of training data block and signature (F_t, σ_t) to replace the chosen one (F_i, σ_i) . Then, the signature proof σ becomes:

$$\sigma^* = \prod_{i \in I, i \neq t} (\sigma_i)^{v_i} \cdot (\sigma_t)^{v_t}.$$

The data proof μ becomes:

$$\mu^* = \sum_{i \in I, i \neq t} v_i F_i + v_t F_t + r$$

Then, the verification equation (4) can be expressed as:

$$e(\sigma^* \cdot R, P) \stackrel{?}{=} e\left(\prod_{i \in I} H_2(W_i)^{v_i} \cdot \left(\frac{(H_2(W_t))^{v_t}}{(H_2(W_i))^{v_i}}\right) \cdot u^{\mu^*}, X\right)$$

Due to the collision resistance of hash function, $\left(\frac{(H_2(W_t))^{v_t}}{(H_2(W_i))^{v_i}}\right)$ cannot be equal to 1 in the oracle model. Thus, the integrity verification equation (4) does not hold, so that the data server cannot pass the integrity verification. Hence, our DML-DIV scheme can resist the tampering attack launched by the data

server. In the same way, our DML-DIV scheme can also resist the tampering attack launched by Network Attackers. \square

Theorem 5: In our DML-DIV scheme, it is computationally infeasible for the data server and Network Attackers to forge a proof to pass the TPA's auditing.

Proof: According to literature [31], we can get the following proof form. If the data server could pass our verification by forging an auditing proof, this means that we can solve a Co-CDH problem (Definition 4), which contradicts the Co-CDH hypothesis.

Given the same challenge $Q = \{I, v_{i,i \in I}\}$ from the TPA, the correct proof should be $P = \{\sigma, \mu\}$. However, the data server can forge an incorrect proof $P' = \{\sigma', \mu'\}$, where $\sigma \neq \sigma', \mu \neq \mu'$. If proof forged by the data server can pass TPA's verification, according to Equation (4), we can get:

$$e(\sigma' \cdot R, P) = e\left(\prod_{i \in I} H_2(W_i)^{v_i} \cdot u^{\mu'}, X\right)$$

As $P = \{\sigma, \mu\}$ is a correct proof, we can get:

$$e(\sigma \cdot R, P) = e\left(\prod_{i \in I} H_2(W_i)^{v_i} \cdot u^{\mu}, X\right)$$

Based on the properties of bilinear map, we can learn that:

$$e(\sigma' / \sigma, P) = e(u^{\mu' - \mu}, X)$$

The above formula can also be rewritten as:

$$e(\sigma' / \sigma, P) = e(u^{\Delta\mu}, P^{sk})$$

where $\Delta\mu = \mu' - \mu$. Clearly, we can get:

$$\sigma' / \sigma = u^{\Delta\mu \cdot sk}$$

At the same time, we can rewrite the above formula to get following equation:

$$u^{sk} = (\sigma' / \sigma)^{1/\Delta\mu}$$

That is, for given u, P, P^{sx}, P^{sk} , we can figure out u^{sk} . Hence, we can find an algorithm to solve the Co-CDH problem (Definition 4), which contradicts the Co-CDH hypothesis. Therefore, it is computationally infeasible for the data server to forge an auditing proof to pass the TPA's verification. In the same way, Network Attackers also cannot forge a proof to pass the TPA's data integrity verification in our DML-DIV scheme. \square

According to Theorem 4 and Theorem 5, we know that the data server can't pass the TPA's integrity verification if it doesn't store the training data completely. That is, our DML-DIV scheme can resist forgery attack and tampering attack by the data server and Network Attackers.

Theorem 6: In our DML-DIV scheme, we can solve the key escrow problem. That is, the KGC and Network Attackers cannot forge the user's private key. Suppose an (t, ε) -algorithm can fake the identity ID . Then, we can find an (t', ε') adversary A that can solve the CDHP problem with $t' < t + P_m(p_H + p_s + 1)$ and $\varepsilon' \geq \frac{\varepsilon}{(p_H + p_s)}$, where P_m refers to the time required for a scalar point multiplication in G_1 .

Proof: Let P be a generator of G_1 , giving a CDHP triple (P, P^a, P^b) , where $a, b \in \mathbb{Z}_p^*$, and noting $(P, A, B) \in G_1$, the challenger C is to figure out P^{ab} , which is solving the CDHP problem, and in contradiction with the Definition 3.

Let $P_{pub} = P^a$, the game between the challenger C and the adversary A can be described as follows: First, the challenger C sets up game, and input is 1^k . Then, the adversary A gets the system parameter $params$, and starts the game.

1) PUBLIC KEY QUERIES

The challenger C owns a table PT , which includes $(ID_i, c_i, x_i, y_i, Q_{ID_i})$, and allows the adversary A to perform public key queries based on an identity ID_i . Firstly, the challenger C checks whether the questioned ID_i is in the table PT . If the ID_i is not in the table PT , the challenger C randomly selects two elements $x_i, y_i \in G_1$. Secondly, the challenger C flips a coin $c_i \in \{0, 1\}$. We assume $\Pr(c_i = 0) = \delta$, and $\Pr(c_i = 1) = 1 - \delta$. If $c_i = 0$, the challenger C computes $P^{y_i} = H_1(ID_i) = Q_{ID_i}$, otherwise, the challenger C computes $B^{y_i} = H_1(ID_i) = Q_{ID_i}$. Then, the challenger C responds to A with $H_1(ID_i) = Q_{ID_i}$. Finally, the challenger C adds $(ID_i, c_i, x_i, y_i, Q_{ID_i})$ into table PT . If the ID_i is in the table PT , the challenger C finds the Q_{ID_i} corresponding to the ID_i in the table PT and sends Q_{ID_i} to A.

2) PRIVATE KEY QUERIES

The challenger C owns a table ST , which includes (x_i, S_{ID_i}) , and allows the adversary A to perform private key queries based on an identity ID_i . Firstly, the challenger C checks whether the questioned ID_i is in the table ST . If the ID_i is not in the table ST , the game returns to the public key queries phase. Otherwise, the challenger C finds the c_i corresponding to the ID_i in the table PT . If $c_i = 0$, the challenger C computes $A^{y_i} = S_{ID_i}$ and adds (x_i, S_{ID_i}) into the table ST . If $c_i = 1$, the challenger C cannot generate the partial private key that can satisfy the verification equation.

3) OUTPUT QUERIES

The adversary A queries the ID that is not queried in the private key queries phase. The adversary A fakes the identity ID successfully and can get a record $(ID, c, x, y, Q_{ID}, S_{ID})$. Since the adversary A successfully fakes the identity ID , $e(Y, P_{pub}) = e(H_2(X||Z) \cdot Q, Z)$ is right. The challenger C views the c_i corresponding to the ID . If $c_i = 1$, then C fails. Otherwise, $c_i = 0$ and thus $e(Y, P_{pub}) = e((H_2(X||Z) \cdot Q)^{yb}, P^a)$, $e(H_2(X||Z) \cdot Q, Z) = e(H_2(X||Z) \cdot Q, P^{sy})$. Since $e((H_2(X||Z) \cdot Q)^{yb}, P^a) e((H_2(X||Z) \cdot Q), P^{aby}) = e((H_2(X||Z) \cdot Q), P^{sy})$, $P^{sy} = P^{aby}$ is the solution to the CDHP problem owned by the challenger C.

Let the number of public key queries be $p_H \leq p$ and the number of private key queries be $p_s \leq p$. The adversary A cannot repeatedly query the same ID , can only query up to $p_H + p_s$ times. The challenger C gives the different query responses according to different values of the coin result c_i .

TABLE 2. Security comparison.

	Scheme[15]	Scheme[19]	Scheme[27]	Scheme[31]	Our DML-DIV
Key escrow problem	No	No	No	No	Yes
Privacy protection	Yes	No	Yes	No	Yes
Forgery attack	Yes	Yes	Yes	Yes	Yes
Tampering attack	Yes	Yes	Yes	Yes	Yes

Let $\Pr(c_i = 0) = \delta$, we know that the number of $c_i = 0$ is $(p_H + p_s)\delta$ and the number of $c_i = 1$ is $(p_H + p_s)(1 - \delta)$. In private key queries, if $c_i = 1$, the challenger C cannot generate the partial private key that can satisfy the verification equation. Thus, the probability of failure is $(1 - \delta)$ and the probability of success is δ . Both the public key query and the private key query are successful, so that the challenger C can successfully solve the CDHP problem (refer to Definition 3). Thus, the probability of the challenger's success is $\varepsilon' > \frac{\delta}{(1-\delta)(p_H+p_s)}\varepsilon$. If and only if $\delta = \frac{1}{2}$, the maximum of $\frac{\delta}{(1-\delta)(p_H+p_s)}\varepsilon$ is $\frac{\varepsilon}{(p_H+p_s)}$. Thus, the probability of the challenger's success is $\varepsilon' \geq \frac{\varepsilon}{(p_H+p_s)}$. It is assumed that after $p_H + p_s$ queries, the adversary A can fake identity ID, that is, adversary A outputs a set of valid $(ID, c, x, y, Q_{ID}, S_{ID})$ and $e(Y, P_{pub}) = e(H_2(X||Z) \cdot Q, Z)$ is true. Therefore, the challenger C can solve the CDHP problem on group G_1 , which is in contradiction with the Definition 3. \square

Theorem 6 proves that our DML-DIV scheme solves the key escrow problem, that is the KGC and Network Attackers cannot forge the user's private key.

Theorem 7: In our DML-DIV scheme, according to the data server's response $\{\sigma, \mu, R\}$, the TPA and Network Attackers cannot disclose the training data blocks. Suppose an (t, ε) -algorithm can obtain the training data blocks in the auditing process. Then, we can find an (t', ε') adversary A that can solve the DLP problem with $t' \approx 2t$ and $\varepsilon' \geq \frac{\varepsilon^2}{p}$.

Proof: First, the adversary A initiates challenge to the data server to obtain data proof μ .

Then, we assume that $r \in Z_p$ be the data server's private key and $R_1 = u^r$ be the corresponding public key. In the first step, the challenger C picks a random $x \in Z_p$, computes $R' = u^x$ and sends R' to adversary A. Upon receiving R' , the adversary picks a random element $c \in Z_p$ and sends c to C. Upon receiving c , the challenger C computes $y = x + rc$ and sends y to A. If $u^y = R'R_1^c$, the adversary A accepts y , otherwise rejects it. Thus, the adversary A can obtain r . Furthermore, according to $\mu = \sum_{i \in I} v_i F_i + r$, the adversary A can obtain: $\sum_{i \in I} v_i F_i = \mu - r$.

If the adversary A performs the above operations $|I|$ times, it can obtain a linear system of equations. Then, the adversary A will obtain training data blocks by solving the linear equations.

Through the above proof process, the adversary A can obtain the solution of DLP problem (Definition 2). Since the conditions for success is (t, ε) , the adversary A can solve the DLP problem within $t' \approx 2t$ and with non-negligible

probability $\varepsilon' \geq \frac{\varepsilon^2}{p}$, which is in contradiction with the Definition 2. \square

Theorem 7 proves that our DML-DIV scheme can achieve privacy protection, that is, during the public auditing process, the TPA and Network Attackers cannot obtain the secret of training data.

B. SECURITY COMPARISON

In this section, we compare the security of our DML-DIV scheme with the schemes [15], [19], [27], and [31] in Table 2. The comparison is under the key escrow problem, the data server's forgery attack and tampering attack and privacy protection problem. According to Table 2, our DML-DIV scheme can realize all of the security goals mentioned above, but other schemes cannot.

V. PERFORMANCE ANALYSIS

In this section, we will analyze computation cost and communication overhead in the process of integrity verification for training data, and evaluate the performance of our DML-IDV scheme in an example. We compare computation cost and communication overhead among Yan's scheme [19], Zhu's scheme [27], Wang's scheme [15], He's scheme [31] and our DML-DIV scheme.

A. COMPUTATION COST

We evaluate the whole verification process on a desktop with Intel Core at 2.40GHz and 8G of RAM. The basic pairing algorithm is conducted through GUN Multiple Precision Arithmetic (GMP) library version 6.1.2. We choose type d MNT curve from PBC library. We set that the length of G_1 is 175. All experimental results represent the mean of 20 trials.

In our experiment, L denotes the length of each data block, E_G denotes a power operation on group G_1 and G_2 , E_Z denotes a power operation on the number field Z_p , M_G indicates a multiplication operation on group G_1 and G_2 , M_Z indicates a multiplication operation on the number field Z_p , A_Z indicates an addition operation on the number field Z_p , and H denotes an operation of calculating the hash value for a number.

In the public sampling auditing scheme, the computation cost mainly happens on data server, TPA, and data owner, respectively.

1) COMPUTATION COSTS FOR DATA SERVER

In the whole verification process, the data server executes computation to generate proof. We simulate the proof

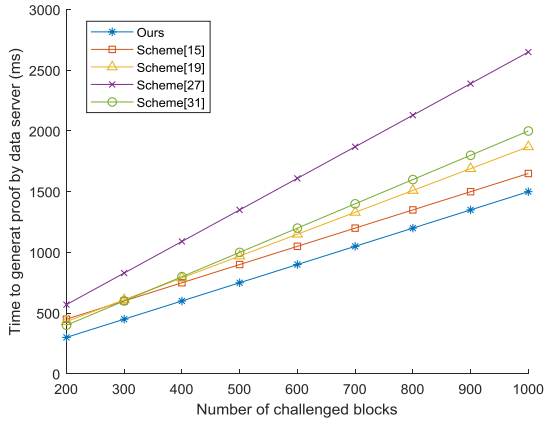


FIGURE 4. Comparison on time that data server generates proof.

generation time during data integrity verification on data server. We assume the number of blocks for file is 10000, that is $m = 10000$. The result is shown in Fig.4.

In Fig.4, the proof generation time is linearly increasing with number of challenged blocks. The reason is that data server only accumulates the challenged data blocks and multiplies corresponding signatures.

As shown in Fig.4, the data server's computation cost in our scheme is smaller than that in schemes [15], [19], [27], and [31]. In the scheme [19], the data server needs to compute random subscripts and random numbers with challenge information from the TPA. In scheme [27], the data server needs to compute more signature proof. In the scheme [15], the system needs to compute a blind factor, with a bilinear mapping operation, an operation of hash value, and a power operation. In the scheme [31], the system needs to split the data block twice and needs to aggregate data and signatures twice to obtain proof. In our scheme, the system only needs to compute a blind factor and aggregate training data and signatures to obtain proof. Therefore, in the process of generating proof, our scheme is more efficient than schemes [15], [19], [27], and [31].

2) COMPUTATION COSTS FOR TPA

In the whole verification process, the TPA executes computation to verify the proof. We simulate the proof verification time during data integrity verification on the TPA. We assume the number of data blocks for file is 10000, that is $m = 10000$. The result is shown in Fig.5.

In Fig.5, the verification time is linearly increasing with number of challenged blocks. The reason is that the calculation for Verification Equation (4) is only related to the number of challenged blocks.

As shown in Fig.5, the TPA's computation cost in our scheme is smaller than that in schemes [15], [19], and [31]. And the TPA's computation cost in our scheme is larger than that in scheme [27]. In scheme [19], the TPA needs to calculate the random subscripts and random numbers based on two functions. In the scheme [15], the system needs to introduce

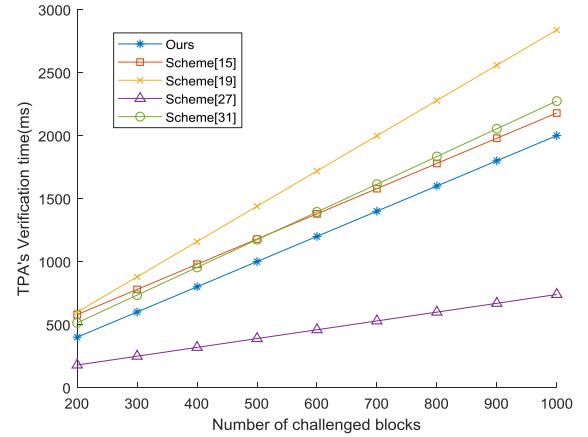


FIGURE 5. Comparison on TPA's Verification time.

the new exponential operations and multiplication operations to implement privacy protection. In the scheme [31], the algorithm needs to aggregate proof from multiple servers into verification equation. In scheme [27], there are only two pairing operations, one multiplication operation and one addition operation to verify the integrity of data. In our scheme, the TPA only needs to aggregate data block information $H_2(W_i)$ and calculate two bilinear mapping pairs. Therefore, in the process of integrity verification, our scheme is more efficient than the schemes [15], [19], and [31], and is worse than the scheme [27].

3) COMPUTATION COSTS FOR DATA OWNER

In the whole verification process, the data owner executes computation to generate signatures. In Table 3, we analyze and compare the data owner's computation cost in computing the signatures among schemes [15], [19], [27], [31] and our DML-IDV scheme. According to Table 3, we can see that in terms of the computation cost for generating signatures, our DML-DIV scheme is equal to the schemes [15], [19], and [31], and is slightly larger than the scheme [27].

In our scheme, the computation cost of each signature is $2M_G + E_G + H$, and the number of data blocks is $size\ of(F)/L$ (where L represents the number of bits of a data block). Thus, for data file F , computation cost of generating signatures is $\frac{size\ of(F)}{L}(M_G + 2E_G + H)$. As shown in Table 3, the data owners' computation in our scheme is equal to that in schemes [15], [19], and [31]. The reason is the signatures of data blocks in schemes [15], [19], [31] are same as in our scheme. According to $A_Z + M_Z < M_G + E_G$, we can know that the data owners' computation in our scheme is slightly larger than that in scheme [27]. The reason is that in scheme [27], data's signatures are in form of addition and multiplication.

B. COMMUNICATION OVERHEAD

In this section, we compare the communication overhead among Yan's scheme [19], Zhu's scheme [27], Wang's scheme [15], He's scheme [31] and our DML-DIV scheme in Table 4. In Table 4, $|Z_p|$ represents the size of Z_p , and

TABLE 3. Data owner's computation comparison.

	Our scheme	[15]	[19]	[27]	[31]
Data Owner	$\frac{\text{sizeof}(F)}{L} \cdot (M_G + 2E_G + H)$	$\frac{\text{sizeof}(F)}{L} \cdot (M_G + 2E_G + H)$	$\frac{\text{sizeof}(F)}{L} \cdot (M_G + 2E_G + H)$	$\frac{\text{sizeof}(F)}{L} \cdot (M_G + M_Z + H + A_Z)$	$\frac{\text{sizeof}(F)}{L} \cdot (M_G + 2E_G + H)$

TABLE 4. Comparison of communication.

	Our scheme	[15]	[19]	[27]	[31]
Challenge	$2 \cdot I \cdot Z_p $	$2 \cdot I \cdot Z_p $	$3 \cdot Z_p $	$2 \cdot I \cdot Z_p $	$2 \cdot I \cdot J \cdot Z_p $
Proof	$(I+1) G + 2 Z_p $	$(I+1) G + 2 Z_p $	$(I+1) G + Z_p $	$(I+3) G $	$J \cdot I \cdot G + 2I Z_p $

$|G|$ represents the size of group G_1 . In schemes [15], [19], [27] and our DML-DIV scheme, I represents the number of challenged data blocks. In the scheme [31], I represents the number of challenged data servers and J represents the number of data blocks.

1) COMMUNICATION OVERHEAD FOR CHALLENGE

We compare the communication overhead for challenge among schemes [15], [19], [27], [31], and our DML-DIV scheme. In our DML-DIV scheme, the challenge is $Q = \{I, v_{i,i \in I}\}$, which costs $2 \cdot I \cdot |Z_p|$. In scheme [19], the challenge is $Q = \{c, k_1, k_2\}$, which costs $3 \cdot |Z_p|$. In scheme [27], the challenge is $Q = \{i, v_i\}_{i \in I}$, which costs $2 \cdot I \cdot |Z_p|$. In scheme [15], the challenge is $Q = \{i, v_i\}_{i \in I}$, which costs $2 \cdot I \cdot |Z_p|$. In scheme [31], the challenge is $Q_i = \{j, v_{ij}\}_{j \in J} (i \in I)$, which costs $2 \cdot I \cdot J \cdot |Z_p|$.

The communication overhead for challenge in scheme [19] is smaller than that in our DML-DIV scheme. The reason is that in the scheme [19], the communication overhead for challenge is $3 \cdot |Z_p|$, and in our DML-DIV scheme, the communication overhead for challenge is $2 \cdot I \cdot |Z_p| (I > 2)$.

The communication overhead for challenge in scheme [27] is equal to that in our DML-DIV scheme. The reason is that the communication overhead for challenge in scheme [27] and our DML-DIV scheme is same as $2 \cdot I \cdot |Z_p|$.

The communication overhead for challenge in our scheme is smaller than that in the scheme [31]. The reason is that in the scheme [31], the system needs to store same data blocks in multiple servers, which may cause the challenge to be sent to multiple servers. In our DML-DIV scheme, we separate the training data into n blocks and store data blocks in the data server, thus the challenge is only sent to one data server.

The PDP idea adopted by the scheme [15] is same as in our DML-DIV scheme, so the communication overhead for the challenge in scheme [15] is same as that in our DML-DIV scheme.

2) COMMUNICATION OVERHEAD FOR PROOF

We compare the communication overhead for proof among schemes [15], [19], [27], [31], and our scheme.

In our DML-DIV scheme, the proof sent by the data server is $\{\sigma, \mu, R, H_2(W_i)_{i \in I}\}$, which costs $(I+1)|G| + 2|Z_p|$. In scheme [19], the proof sent by cloud server is $\{\bar{T}, \bar{F}, H_2(\alpha || F_{id} || v_i)_{i \in I}\}$, which costs $(I+1)|G| + |Z_p|$. In scheme [27], the proof sent by cloud server is $\{R, \mu, \eta, H(m_{ij})_{i \in I}\}$, which costs $(I+3)|G|$. In scheme [15], the proof sent by cloud server is $\{\sigma, \mu, R, h(W_i)_{i \in I}\}$, which costs $(I+1)|G| + 2|Z_p|$. In scheme [31], the proof sent by cloud server is $\{\sigma_i, \mu_i, h(W_{ij})_{j \in J}\}_{i \in I}$, which costs $J \cdot I \cdot |G| + 2I \cdot |Z_p|$.

As shown in table 4, the communication overhead for proof in scheme [19] is slightly smaller than that in our DML-DIV scheme. The reason is that, in the scheme [19], the communication overhead for proof is $(I+1)|G| + |Z_p|$, and in our DML-DIV scheme, the communication overhead for proof is $(I+1)|G| + 2|Z_p|$.

The communication overhead for proof in scheme [27] is approximately equal to that in our DML-DIV scheme. Since the order of group G_1 is equal to the order of Z_p , we can get $|Z_p| \approx |G|$. Hence, $(I+3)|G| \approx (I+1)|G| + 2|Z_p|$, which means the communication overhead for proof in scheme [27] is approximately equal to that in our DML-DIV scheme.

The communication overhead for proof in our scheme is smaller than that in scheme [31]. The reason is that, in the scheme [31], multiple servers are required to generate proof and send the proof to the TPA to prove that they are storing the data together. In our DML-DIV scheme, we separate the training data into n blocks and store data blocks in the data server, thus, only one server needs to send proof to the TPA.

The PDP idea adopted by the scheme [15] is same as our DML-DIV scheme, so the communication overhead for the proof in scheme [15] is same as that in our DML-DIV scheme.

C. EVALUATION OF OUR DML-DIV SCHEME

In this section, we apply the application of advertising recommendation given in part B of section 3 to evaluate the efficiency of our DML-DIV scheme.

We collect an advertisement click prediction dataset with 1.7 billion examples and 0.65 billion unique features. This dataset is 1.41TB, which is stored in Amazon. We run

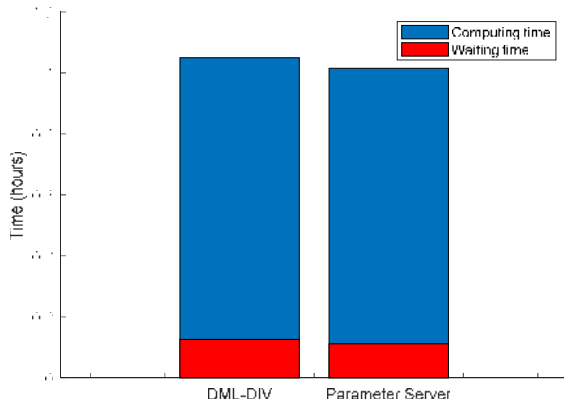


FIGURE 6. Timer per worker node spent on computing and waiting.

the Parameter Server framework on 12 machines, each with 8 physical cores. Nine machines act as worker nodes, and three machines act as parameter servers.

We apply a distributed machine learning algorithm [8] to evaluate the efficiency of our DML-DIV scheme. The comparison for computing and waiting time between general Parameter Server without data integrity protection and our DML-DIV schemes is showed in Fig.6.

According to Fig.6, in general Parameter Server framework without data integrity protection, the computing time is 0.903 hours, and the waiting time is 0.112 hours. In our DML-DIV scheme, the computing time is 0.924 hours, and the waiting time is 0.126 hours. Hence, our DML-DIV scheme is almost same as general Parameter Server framework in waiting and computing time. Although in our DML-DIV scheme, the system needs to verify the integrity of training data, the added computing and waiting time for each work node is almost negligible. Therefore, in waiting and computing time, our DML-DIV scheme is almost same as general Parameter Server system.

VI. CONCLUSION

In this paper, we propose a distributed machine learning oriented data integrity verification scheme (DML-DIV) for the parameter server framework. Our DML-DIV scheme can ensure the integrity of the training data stored in the data server, and resist forgery attack and tampering attack. Additionally, our DML-DIV scheme provides privacy protection, solves the key escrow problem, and reduces the cost of managing the certificates. Finally, the simulation results show our DML-DIV scheme performs more efficiently than other schemes.

REFERENCES

- [1] J. Dean and S. Ghemawat, "MapReduce: A flexible data processing tool," *Commun. ACM*, vol. 53, no. 1, pp. 72–77, 2010.
- [2] M. Zaharia, M. Chowdhury, and M. Franklin, "Spark: Cluster computing with working sets," in *Proc. 2nd USENIX Conf. Hot Topics Cloud Comput.*, 2010, pp. 1–7.
- [3] Y. Low, J. E. Gonzalez, and A. Kyrola, "GraphLab: A new framework for parallel machine learning," *Comput. Sci.*, vol. 31, no. 1, pp. 1–4, 2004.
- [4] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski, "Pregel: A system for large-scale graph processing," in *Proc. ACM SIGMOD Int. Conf. Oil Manage. Data*, 2010, pp. 135–146.
- [5] A. Smola and S. Narayanamurthy, "An architecture for parallel topic models," *Proc. VLDB Endow.*, vol. 3, nos. 1–2, pp. 703–710, Sep. 2010.
- [6] J. Dean, G. S. Corrado, and R. Monga, "Large scale distributed deep networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.* Red Hook, NY, USA: Curran Associates, 2013, pp. 1223–1231.
- [7] (2018). *Douban Paracel*. [Online]. Available: <http://paracel.io/>
- [8] M. Li, "Scaling distributed machine learning with the parameter server," in *Proc. Int. Conf. Big Data Sci. Comput. (BigDataSci)*, 2014, pp. 583–598.
- [9] M. Li, Z. Li, and A. Smola, "Parameter server for distributed machine learning," in *Proc. Big Learn. NIPS Workshop*, 2013, pp. 1–10.
- [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. 1–25.
- [11] Q. Zheng, S. Xu, and G. Ateniese, "Efficient query integrity for outsourced dynamic databases," in *Proc. ACM Workshop Cloud Comput. Secur. Workshop (CCSW)*, 2012, pp. 71–82.
- [12] C. C. Erway, A. Küpçü, and C. Papamanthou, "Dynamic provable data possession," *ACM Trans. Inf. Syst. Secur.*, vol. 17, no. 4, pp. 1–29, 2009.
- [13] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 525–533.
- [14] Y. Zhu, H. Hu, G.-J. Ahn, and S. S. Yau, "Efficient audit service outsourcing for data integrity in clouds," *J. Syst. Softw.*, vol. 85, no. 5, pp. 1083–1095, May 2012.
- [15] C. Wang, S. S. M. Chow, and Q. Wang, "Privacy-preserving public auditing for secure cloud storage," *IEEE Trans. Comput.*, vol. 62, no. 2, pp. 362–375, Dec. 2013.
- [16] H. Yan, J. Li, J. Han, and Y. Zhang, "A novel efficient remote data possession checking protocol in cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 1, pp. 78–88, Jan. 2017.
- [17] M. Sookhak, F. R. Yu, and A. Y. Zomaya, "Auditing big data storage in cloud computing using divide and conquer tables," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 5, pp. 999–1012, May 2018.
- [18] H. Zhao, X. Yao, X. Zheng, T. Qiu, and H. Ning, "User stateless privacy-preserving TPA auditing scheme for cloud storage," *J. Netw. Comput. Appl.*, vol. 129, pp. 62–70, Mar. 2019.
- [19] H. Yan, J. Li, and Y. Zhang, "Remote data checking with a designated verifier in cloud storage," *IEEE Syst. J.*, to be published, doi: 10.1109/jsyst.2019.2918022.
- [20] H. Wang, "Identity-based distributed provable data possession in multi-cloud storage," *IEEE Trans. Services Comput.*, vol. 8, no. 2, pp. 328–340, Mar. 2015.
- [21] A. A. Hussain and R. Subashini, "Identity-based proxy-oriented data uploading and remote data integrity checking in public cloud," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 6, pp. 1165–1176, Jan. 2019.
- [22] J. Zhang and Q. Dong, "Efficient ID-based public auditing for the outsourced data in cloud storage," *Inf. Sci.*, vols. 343–344, pp. 1–14, May 2016.
- [23] Y. Yu, M. H. Au, G. Ateniese, X. Huang, W. Susilo, Y. Dai, and G. Min, "Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 4, pp. 767–778, Apr. 2017.
- [24] J. Li, H. Yan, and Y. Zhang, "Certificateless public integrity checking of group shared data on cloud storage," *IEEE Trans. Serv. Comput.*, to be published, doi: 10.1109/tsc.2018.2789893.
- [25] W. Shen, J. Qin, J. Yu, R. Hao, and J. Hu, "Enabling identity-based integrity auditing and data sharing with sensitive information hiding for secure cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 2, pp. 331–346, Feb. 2019.
- [26] J. Li, H. Yan, and Y. Zhang, "Efficient identity-based provable multi-copy data possession in multi-cloud storage," *IEEE Trans. Cloud Comput.*, to be published, doi: 10.1109/tcc.2019.2929045.
- [27] H. Zhu, Y. Yuan, Y. Chen, Y. Zha, W. Xi, B. Jia, and Y. Xin, "A secure and efficient data integrity verification scheme for cloud-IoT based on short signature," *IEEE Access*, vol. 7, pp. 90036–90044, 2019.
- [28] H. H. Krawczyk, "A high-performance secure Diffie–Hellman protocol," in *Advances in Cryptology*. Berlin, Germany: Springer, 2005, pp. 1–62.
- [29] F. Bao, R. H. Deng, and H. Zhu, "Variations of Diffie–Hellman problem," in *Proc. Int. Conf. Inf. Commun. Secur.*, Huhehaote, China, Oct. 2003, pp. 301–312.

- [30] A. Hu, R. Jiang, and B. Bhargava, "Identity-preserving public integrity checking with dynamic groups for cloud storage," *IEEE Trans. Serv. Comput.*, to be published.
- [31] K. He, C. Huang, J. Shi, and J. Wang, "Public integrity auditing for dynamic regenerating code based cloud storage," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jun. 2016, pp. 581–588.



XIAO-PING ZHAO received the bachelor's degree in information and computing science from the Nanjing University of Posts and Telecommunications (NUPT). He is currently pursuing the master's degree with the School of Cyber Science and Engineering, Southeast University, Nanjing. His research interests include data integrity verification, and data privacy protection in distributed machine learning and cloud environments.



RUI JIANG received the Ph.D. degree from Shanghai Jiaotong University, Shanghai, China, in 2005. He is currently an Associate Professor with Southeast University, China. His current research interests include secure analysis and design of communication protocols, secure cloud computing and big data, secure network and systems communications, mobile voice end-to-end secure communications, and applied cryptography.

...