IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 8, 2019, accepted August 27, 2019, date of publication August 30, 2019, date of current version September 11, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2938528

Efficient Privacy-Preserving Certificateless Provable Data Possession Scheme for Cloud Storage

YANG MING (Member, IEEE) AND WENCHANG SHI
School of Information Engineering, Chang'an University, Xi'an 710064, China
Corresponding author: Yang Ming (yangming@chd.edu.cn)

This work was supported in part by the Natural Science Foundation of Shaanxi Province under Grant 2018JM6081, and in part by the Fundamental Research Funds for the Central Universities, CHD, under Grant 300102249204.

ABSTRACT Cloud computing is a burgeoning paradigm that offers reliable and adaptive infrastructure to the data owner who shares his data with the data user through the cloud server. In the data sharing process, the integrity of the data stored in the cloud faces serious threats. The provable data possession schemes ensure that a cloud server provider proves to a third party auditor that it is truthfully storing data from the data owner. However, the PKI-based provable data possession scheme suffers the management issue of the certificates and the identity-based provable data possession scheme causes an inherent key escrow problem. In this paper, we put forward an efficient privacy-preserving certificateless provable data possession scheme based on certificateless cryptography and elliptic curve cryptography, which has been proved to be unforgeable against adaptive chosen message attacks in the random oracle model. At the same time, the data integrity can be audited by the third party auditor without downloading the whole data. The proposed scheme gives away no information of the stored data to the third party auditor during the audit phase and the data owner's identity privacy is protected. The experiment results show that the proposed scheme is of efficiency and feasibility as far as computation and communication costs are concerned.

INDEX TERMS Cloud storage, privacy preserving, provable data possession, certificateless cryptography.

I. INTRODUCTION

In recent years, scholars as well as industry practitioners have expressed increasing interest in cloud computing [1]. Cloud storage, as an integral part of cloud computing, has been frequently used for people to store their data, producing advantages in abundance. For illustration, the data owner no longer needs to deal with plenty of data. Moreover, it becomes convenient to access data wherever the data user's geographical location is.

Nonetheless, the outsourced data in cloud are exposed to security threats [2]. An untrustworthy cloud service provider might abandon or adjust the data for the sake of space or for personal gains respectively. Accordingly, the data user should be guaranteed that their data will be stored in the cloud server as they originally are. In other words, the cloud service providers are supposed to take the responsibility of keeping the data user's data integrity.

As regards traditional cryptographic methods of auditing data integrity, a local copy of the data (which the data user does not have) is an imperative or the entire data need downloading, which apparently is a waste of communication bandwidth. Therefore, the provable data possession (PDP) scheme is designed to solve the above problem. With auditing, the data user with no local copy of data or direct control of data can remotely audit the integrity of their data stored in the cloud. Aimed at making audit greatly convenient and energy efficient, a third party auditor was introduced, which receives and implements the auditing delegation from the data user. Such an entity at hand, the client side can be relieved of some burden.

Until now, lots of PDP schemes [3]–[27] have been put forward. However, most present PDP schemes [3]–[9] on the basis of public key infrastructure (PKI) are faced with certificate management problem. Against such a background,

The associate editor coordinating the review of this article and approving it for publication was Yassine Maleh.

identity-based PDP schemes [10]–[15] have been proposed. Unfortunately, these schemes [10]–[15] suffer from key escrow problem. Namely, the key generation center (KGC) can produce all private keys for the users. Therefore, the certificateless provable data possession (CLPDP) schemes [16]–[27] have been proposed to address the above-mentioned problems. Despite there are significant advances, the CLPDP schemes [16]–[27] based on the bilinear pairing and map-to-point hash operations are still time consuming.

In addition, some CLPDP schemes [16]–[18] don't protect data privacy considering the third party auditor has access to the data owner's data after solving a series of linear equations [5]. Meanwhile, the real identities of the data owner is exposed in the CLPDP schemes [16]–[27], which brings serious security and privacy threats to users. Therefore, the data owner's data and identity privacy ought to be duly considered in the CLPDP schemes.

To achieve enhanced security and improvement in performance, an efficient privacy-preserving CLPDP scheme is built in this paper. The main contributions are presented below.

- First, based on elliptic curve cryptography (ECC) and certificateless cryptography [28], we propose an efficient privacy-preserving CLPDP scheme. The proposed scheme meets provable data possession that a third party auditor checks the integrity of the data stored in the cloud. In addition, the proposed scheme avoids the certificate management problem found in the PKI-based PDP schemes and the key escrow problem found in the identity-based PDP schemes.
- Second, security analysis shows that the proposed scheme is provably secure in the random oracle model. Meanwhile, the proposed scheme achieves data privacy preservation and leaks no the identify information of the data owner.
- Finally, the proposed scheme is the first CLPDP scheme without employing bilinear pairing and map-to-point hash operations. The computational and communicational performance is estimated via quantitative calculations. Compared with the other CLPDP schemes, the computation cost and communication cost of the proposed scheme are the lowest.

The remainder of the paper is organized as follows. In Section II, we review related work. Some preliminaries are offered in Sections III. In Section IV, the proposed CLPDP scheme is described in details. The security analysis for the proposed scheme is presented in Section V. Section VI provides the performance evaluation. Finally, we come to the conclusions in Section VII.

II. RELATED WORK

Traditional cryptographic methods for integrity checking of data fail to examine the integrity of outsourced data, since it does not make sense to download data from the cloud server for verification (e.g., costly and security considerations). Accordingly, provable data possession (PDP) has been suggested as a solution, which allows remote integrity verification without downloading data from cloud. Specifically speaking, Ateniese *et al.* [3] established a PDP scheme for the first time and applied the random sampling method to audit the data stored in cloud. Soon after Shacham and Waters [4] proposed the first PDP scheme using BLS signature [29], which was provably secure and could audit data integrity. Wang *et al.* [5] supplied a privacy-preserving public verification PDP scheme according to the random masking technique and the homomorphic linear authenticator. To better security and performance, several public auditing schemes have been proposed [6]–[9]. Nonetheless, all PDP schemes based on traditional PKI mechanism are stuck in certificate management.

The ID-based PDP schemes had been proposed to address the problem mentioned above. Based on the ID-based cryptography [30], Wang *et al*. [10] firstly presented an ID-based PDP scheme, where the identity of the user was treated as a public key, and the private key was genereted by a trusted KGC. Besides, Yu *et al*. [11] supplied an ID-based PDP scheme to achieve perfect privacy protection using zero knowledge proof. To realize data integrity auditing in a private preserving, delegated and public remote manner, Wang *et al.* [12] proposed an ID-based proxy-oriented data uploading and PDP scheme. Later on, Wang *et al.* [13] provided an incentive and unconditionally anonymous ID-based PDP scheme, which enabled users to expose bad events with their identity privacy protected. Li *et al*. [14] put forward a fuzzy ID-based PDP scheme that simplified the complex key management for reliable cloud storage systems. Zhang *et al.* [15] proposed an ID-based PDP scheme to realize efficient user revocation. However, these ID-based PDP schemes [10]–[15] existed the key escrow problem, where users' private keys are generated and known by the KGC.

Facing the problems of key escrow, certificateless PDP schemes has been proposed. Incorporating the certificateless cryptography [28], Wang *et al.* [16] invented a CLPDP scheme to tackle the key escrow problem. In their scheme, the KGC would by no means compromise the users' private keys by creating partial keys only. However, He *et al*. [17], who pointed out that Wang *et al.*'s CLPDP scheme [16] was not secure against the type I adversary, suggested a CLPDP scheme instead. Zhang *et al.* [18] presented a CLPDP scheme against malicious verifiers to ensure the integrity of data. But Wang *et al.* [23] found that Zhang *et al.*'s scheme [18] had some security flaws and gave an improved scheme. For the sake of efficiency, Kim and Jeong [22] put forward a CLPDP scheme with constant verification time. However, all the previous schemes [16]–[18] fail to preserve data privacy given the fact that the verifier can access users' data by solving linear equations of the system. Therefore, secure and more efficient CLPDP schemes [19]–[21] which preserve data privacy have been presented. But, Liao *et al.* [24] pointed out that he *et al.*'s CLPDP scheme [20] is unsafe. In addition, in 2018, Li *et al*. [25] introduced a CLPDP scheme for shared group data, but it disclosed users' identities and data to the third party. By virtue of the zero-knowledge proof

TABLE 1. Notations.

Symbol	Definition
KGC	A key generation center
CSP	A cloud service provider
TPA	A third party auditor
p, q	Two large prime numbers
$\mathbb{G}$	A additive group
P	A generator of $\mathbb{G}$
DU	A data user
DO_i	The i-th data owner
ID_i	The DO_i's real identity
PID_i	The DO_i's pseudo identity
M	The shared data
m_l	The l-th block of data M
id_l	The unique identify of m_l
H_1, H_2	Four hash functions
H_3, H_4	$H_i : \{0,1\}^* \rightarrow \mathbb{Z}_q^*,\ i = 1, 2, 3, 4$
λ	The KGC's private key
P_{pub}	The KGC's public key
(D_i, y_i)	The DO_i's partial key
x_i	The DO_i's secret value
(D_i, X_i)	The DO_i's public key
$a\|\|b$	a is concatenated with b
$\oplus$	XOR operation

and randomization method, Yang *et al.* [26] constructed a privacy-preserving certificateless provable group shared data possession protocol. In order to achieve multi-user privacy protection, in 2019, Wu *et al.* [27] proposed a CLPDP under multiple users.

In summary, the PDP schemes [3]–[9] based on PKI face some issues on certificate management, while the identify-based PDP schemes [10]–[15] suffer from the key escrow issue. To solve above challenges, the CLPDP schemes [16]–[27] have been proposed. However, these schemes [16]–[27] require bilinear pairing and map-to-point hash operations which incur a huge burden of calculation. Therefore, we propose an efficient CLPDP scheme without bilinear pairing and map-to-point hash operations. In addition, the proposed scheme can protect the identity of the data owner and leak no stored data to verifiers during verification.

III. PRELIMINARIES

In this section, the preliminaries related to the proposed scheme are introduced, including the system model, elliptic curve, security assumption, security model and security requirements.

A. SYSTEM MODEL

As illustrated in Figure 1, the system model considers five entities, namely a key generation center (KGC), a data owner (DO), a cloud service provider (CSP), a data user (DU) and a third party auditor (TPA). Table 1 exhibits the notations used in this paper.

- KGC: It represents a trusted entity that generates the system parameters and the master key, and uses the DO's identity to generate the partial key.
- DO: It intends to keep his/her data in the CSP. The DO generates his/her secret value and public key. Further, the DO generates the tags of his/her date and uploads them to the CSP.

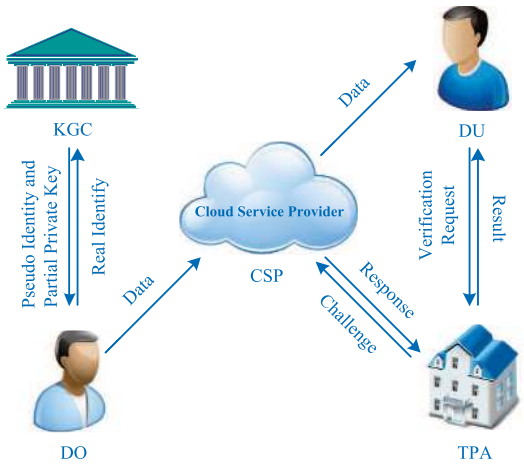

FIGURE 1. System model.

- CSP: It's a semi-trusted entity that is used to store and process the received data. Meanwhile, the CSP might sometimes mislead the DU into believing that the data remain unchanged when delete or modify instead.
- DU: It wants to access and operate data in the CSP. Before any utilization, the DU needs to check the integrity of the data stored in the cloud by sending an audit request to the TPA.
- TPA: It's a semi-trusted entity standing for the DU to audit data integrity. The TPA forwards a challenging message to the CSP, and then the CSP returns proof based on the challenging message. The TPA can audit data integrity by checking the validity of proof.

The proposed scheme consists of three phases: key generation phase, tag generation phase, and audit phase. The details of these phases are described as follows.

Key generation phase. This phase is composed of the *Setup*, *PartialKeyGen*, *SecretValueGen* and *PublicKeyGen* algorithms.

- *Setup*. The KGC executes the algorithm, which inputs the security parameter k, and outputs the system parameters *params* and the master key λ.
- *PartialKeyGen*. The KGC executes the algorithm, which inputs the master key λ and the identity ID_i of the DO_i, and outputs the DO_i's pseudo identity PID_i and the partial keys $\{D_i, y_i\}$.
- *SecretValueGen*. The DO_i executes the algorithm, which randomly selects x_i as the secret value for the DO_i.
- *PublicKeyGen*. The DO_i executes the algorithm, which inputs the DO_i's secret value x_i, and outputs the DO_i's public key $\{D_i, X_i\}$.

Tag generation phase. This phase is composed of the *TagGen* algorithm.

- *TagGen*. The DO_i executes the algorithm, which inputs the DO_i's partial key $\{D_i, y_i\}$, the secret value x_i and the data M, and outputs the tags σ of data M.

Audit phase. This phase consists of the *Challenge*, *ProofGen* and *Verify* algorithms.

- *Challenge*. The TPA executes the algorithm, which inputs the count c of challenging data blocks, and outputs the challenging message *chal*.
- *ProofGen*. The CSP executes the algorithm, which inputs the challenging data blocks, the tags of challenging data blocks and the challenging message *chal*, and outputs the data integrity proof *proof*.
- *Verify*. The TPA executes the algorithm, which inputs the data integrity proof *proof*, the challenging message *chal* and the DO_i's public key $\{D_i, X_i\}$. If *proof* is correct, the algorithm outputs 1; otherwise it outputs 0.

B. ELLIPTIC CURVE

The concept of elliptic curve cryptography (ECC) was proposed by Miller [31] and Koblitz [32]. Given a large prime q, the F_q is a prime finite field. The equation $y^2 = x^3 + ax + b \bmod q$ defines the elliptic curve E over F_q, where $a, b \in F_q$ and $\Delta{=}4a^3 + 27b^2 \bmod q \neq 0$. The infinite point O and all point on E over F_q form an additive group $\mathbb{G}$. Let $kP = P+P+\cdots+P(k\ times)$ express a scalar multiplication operation, where P is a generator of $\mathbb{G}$.

C. SECURITY ASSUMPTION

Elliptic curve discrete logarithm (ECDL) problem: Given two random elements $P, Q \in \mathbb{G}$, the ECDL problem is to compute an integer $x \in \mathbb{Z}_q^*$, such that $Q = x \cdot P$.

Elliptic curve discrete logarithm (ECDL) assumption: There are no polynomial-time algorithms to solve the ECDL problem with non-negligible probability.

D. SECURITY MODEL

In our security model, there are three types of unbounded adversaries namely $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$. Both $\mathcal{A}_1$ and $\mathcal{A}_2$ attempt to forge the data's tags. Type I adversary $\mathcal{A}_1$ doesn't access the master key but can replace the public key of the user with random value. Type II adversary $\mathcal{A}_2$ has access the master key but can't replace the public key of user. Additionally, the type III adversary $\mathcal{A}_3$, who deemed as the untrusted CSP, aims to forge the integrity proof without correct data to pass the verification.

The security of a CLPDP scheme is defined by the following three games between a challenger $\mathcal{C}$ and three adversaries $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$ respectively. The following queries can be made by $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$ adversaries.

- *Hash query*. The adversary makes a hash query, $\mathcal{C}$ returns a random value to the adversary.
- *Partial key query*. The adversary makes a partial key queries on the pseudo identity PID_i, $\mathcal{C}$ performs the *PartialKeyGen* algorithm to compute the partial key (D_i, y_i) for PID_i and returns it to the adversary.
- *Secret value query*. The adversary makes a secret value queries on the pseudo identity PID_i, $\mathcal{C}$ performs the *SecretValueGen* algorithm to obtain the secret value for PID_i and returns it to the adversary.
- *Public key query*. The adversary makes a public key queries on the pseudo identity PID_i, $\mathcal{C}$ performs the *PublicKeyGen* algorithm to generate the public key for PID_i and returns it to the adversary.
- *Public key replacement query*. The adversary chooses a new public key (D_i', X_i') for PID_i and replaces the old public key (D_i, X_i) by (D_i', X_i').
- *Tag query*. The adversary makes a tag queries on the block (m_l, id_l) under (PID_i, D_i, X_i), $\mathcal{C}$ performs the *TagGen* algorithm to obtain the tag for m_l and returns it to the adversary.

Game 1. This game is played by $\mathcal{C}$ and $\mathcal{A}_1$.

Setup: $\mathcal{C}$ runs the *Setup* algorithm to generate the system parameters and the master key. $\mathcal{C}$ sends the system parameters to $\mathcal{A}_1$ and keeps the master key secretly.

Queries: $\mathcal{A}_1$ adaptively makes the hash, partial key, secret value, public key, public key replacement and tag query to $\mathcal{C}$.

Forgery: Finally, $\mathcal{A}_1$ outputs a forged tag (s^*, R^*) on (m^*, id^*) under (PID^*, D^*, X^*).

If the following conditions are satisfied, $\mathcal{A}_1$ wins the game.

- (s^*, R^*) is valid for the block (m^*, id^*) under (PID^*, D^*, X^*).
- $\mathcal{A}_1$ doesn't make the partial key query on PID^*.
- $\mathcal{A}_1$ doesn't make the secret value query on PID^* and substitute the public key of PID^* simultaneously.
- $\mathcal{A}_1$ doesn't make the tag query on (m^*, id^*) under (PID^*, D^*, X^*).

Game 2. This game is played by $\mathcal{C}$ and $\mathcal{A}_2$.

Setup: $\mathcal{C}$ runs the *Setup* algorithm to obtain the system parameters and the master key. $\mathcal{C}$ sends the system parameters and the master key to $\mathcal{A}_2$.

Queries: $\mathcal{A}_2$ adaptively makes the hash, secret value, public key and tag query to $\mathcal{C}$.

Forgery: Finally, $\mathcal{A}_2$ outputs a forged tag (s^*, R^*) on (m^*, id^*) under (PID^*, D^*, X^*).

If the following conditions are satisfied, $\mathcal{A}_1$ wins the game.

- (s^*, R^*) is valid for the block (m^*, id^*) under (PID^*, D^*, X^*).
- $\mathcal{A}_2$ doesn't make the secret value query on PID^*.
- $\mathcal{A}_2$ doesn't make the tag query on (m^*, id^*) under (PID^*, D^*, X^*).

Definition 1: A CLPDP scheme is secure against forging tag attack, if there is no any adversary $\mathcal{A}$ ($\mathcal{A}_1$ or $\mathcal{A}_2$) which wins the Game 1 and Game 2 with a non-negligible probability.

Game 3. This game is played by $\mathcal{C}$ and $\mathcal{A}_3$.

Setup: $\mathcal{C}$ runs the *Setup* algorithm to generate the system parameters and the master key. $\mathcal{C}$ sends the system parameters to $\mathcal{A}_3$ and keeps the master key secretly.

Queries: $\mathcal{A}_3$ adaptively makes the hash, partial key, secret value, public key, public key replacement and tag query to $\mathcal{C}$.

Challenge: $\mathcal{C}$ generates a random challenging message *chal* and sends it to $\mathcal{A}_3$.

Forgery: Finally, $\mathcal{A}_3$ outputs a data integrity proof *proof* for *chal*.

If *proof* without correct data can pass the verification, $\mathcal{A}_3$ wins the game.

Definition 2: A CLPDP scheme is secure against forging proof attack, if there is no any adversary $\mathcal{A}_3$ which wins the Game 3 with a non-negligible probability.

E. SECURITY REQUIREMENTS

The purpose of the proposed scheme is to achieve the following security requirements simultaneously

- *Public verifiability*. It means that the TPA can audit the integrity of the DU's data.
- *Blockless verification*. It means that the TPA can audit the integrity of all desired blocks immediately by checking a block (linear combination of all those blocks).
- *Data privacy preservation*. It means that any adversary or curious verifier can't obtain the data stored in the CSP during the audit integrity process.
- *Identify privacy preservation and traceability*. It means that the DO's real identity isn't disclosed during the auditing procedures. When the DO uploads forged data to the CSP, no entity can obtain the real identity of the DO by analyzing transmitted messages except the KGC.

IV. THE PROPOSED SCHEME

In this section, we present the concrete construction for our CLPDP scheme in detail. Specially, the KGC first registers the DO and generates the corresponding pseudo identity and partial key according to the DO's real identity in key generation phase. After then, the DO uploads the data and the corresponding tags to the CSP in tag generation phase. Finally, in order to audit data integrity, the TPA sends the challenge message to the CSP and the CSP returns a proof message to the TPA in audit phase.

A. KEY GENERATION PHASE

Setup: The KGC produces the systems parameters and the master key as follows:

- [(1)] The KGC chooses a group $\mathbb{G}$ of the prime order q based on an elliptic curve E defined over a finite field F_p, where P is a generator of $\mathbb{G}$.
- [(2)] The KGC chooses four cryptographic secure hash functions: $H_1 : \{0,1\}^* \to \mathbb{Z}_q^*$, $H_2 : \{0,1\}^* \to \mathbb{Z}_q^*$, $H_3 : \{0,1\}^* \to \mathbb{Z}_q^*$ and $H_4 : \{0,1\}^* \to \mathbb{Z}_q^*$.

(3) The KGC randomly selects $\lambda \in \mathbb{Z}_q^*$ and computes $P_{pub} = \lambda \cdot P$.

The KGC publics the system parameters $params = \{p, q, \mathbb{G}, P, P_{pub}, H_1, H_2, H_3, H_4\}$ and keeps the master key λ secretly.

PartialKeyGen: Given the DO_i's real identity $ID_i \in \mathbb{Z}_q^*$, the KGC executes as follows:

1) [(1)] The KGC randomly chooses $u_i \in \mathbb{Z}_q^*$, computes $PID_{i,1} = u_i \cdot P$ and $PID_{i,2} = ID_i \oplus H_1(u_i \cdot P_{pub}||PID_{i,1})$. The DO_i's pseudo identity is $PID_i = \{PID_{i,1}, PID_{i,2}\}$.
2) The KGC randomly chooses $d_i \in \mathbb{Z}_q^*$, computes $D_i = d_i \cdot P$, $\tau_i = H_2(PID_i||D_i)$ and $y_i = d_i + \lambda \cdot \tau_i$.
3) The KGC sends the DO_i's pseudo identity $PID_i = \{PID_{i,1}, PID_{i,2}\}$ and the partial key $\{D_i, y_i\}$ to the DO_i through a secure channel.

SecretValueGen: The DO_i randomly chooses $x_i \in \mathbb{Z}_q^*$ as his/her secret value.

PublicKeyGen: The DO_i computes $X_i = x_i \cdot P$. The public key of the DO_i is $\{D_i, X_i\}$.

B. TAG GENERATION PHASE

TagGen: Given the data M, the DO_i generates the tags as follows:

(1) The DO_i splits the data M into n blocks as $M = \{m_1, m_2, \cdots, m_n\}$, where $m_l \in \mathbb{Z}_q^*$, $l \in \{1, 2, \cdots, n\}$.

(2) For $l \in \{1, 2, \cdots, n\}$, the DO_i randomly chooses $r_l \in \mathbb{Z}_q^*$ and computes $R_l = r_l \cdot P$, $\omega_l = H_3(X_i||R_l||id_l)$, $\phi_l = H_4(D_i||R_l||id_l)$ and $s_l = r_l \cdot m_l + \omega_l \cdot x_i + \phi_l \cdot y_i$, where id_l denotes the unique identify of m_l.

(3) The DO_i outputs the tags $\sigma = \{R_1, R_2, \cdots, R_n, s_1, s_2, \cdots, s_n\}$. Then, the DO_i uploads data M and tags σ to the CSP. At the same time, the DO_i deletes the local data M.

C. AUDIT PHASE

Challenge: After receiving the delegation from the DU, the TPA produces a challenging message *chal* by executing the following steps:

(1) The TPA randomly selects a subset Q (with c elements, $c \le n$) of set $\{1, 2, \cdots, n\}$, $|Q| = c$.

(2) For $j \in Q$, the TPA randomly chooses $v_j \in \mathbb{Z}_q^*$ and sends $chal = \{(j, v_j)\}_{j \in Q}$ to the CSP.

ProofGen: After receiving the challenging message $chal = \{(j, v_j)\}_{j \in Q}$, the CSP produces a data integrity proof *proof* by executing the following steps:

(1) The CSP computes

$$\alpha = \sum_{j \in Q} v_j \cdot s_j \cdot P, \beta = \sum_{j \in Q} v_j \cdot m_j \cdot R_j.$$

(2) The CSP sends the proof $proof = \{\alpha, \beta\}$ to the TPA.

Verify: After receiving the data integrity proof $proof = \{\alpha, \beta\}$, the TPA checks the correctness by executing the following steps:

(1) The TPA computes $\tau_i = H_2(PID_i||D_i)$.

(2) For $j \in Q$, the TPA computes $\omega_j = H_3(X_i||R_j||id_j)$ and $\phi_j = H_4(D_i||R_j||id_j)$.

(3) The TPA verifies whether equation

$$\alpha \stackrel{?}{=} \beta + (\sum_{j \in Q} \omega_j \cdot v_j) \cdot X_i + (\sum_{j \in Q} \phi_j \cdot v_j) \cdot (D_i + \tau_i \cdot P_{pub}).$$

If the equation holds, the TPA outputs 1, otherwise it outputs 0.

The overview of the proposed scheme is depicted in Figure 2.

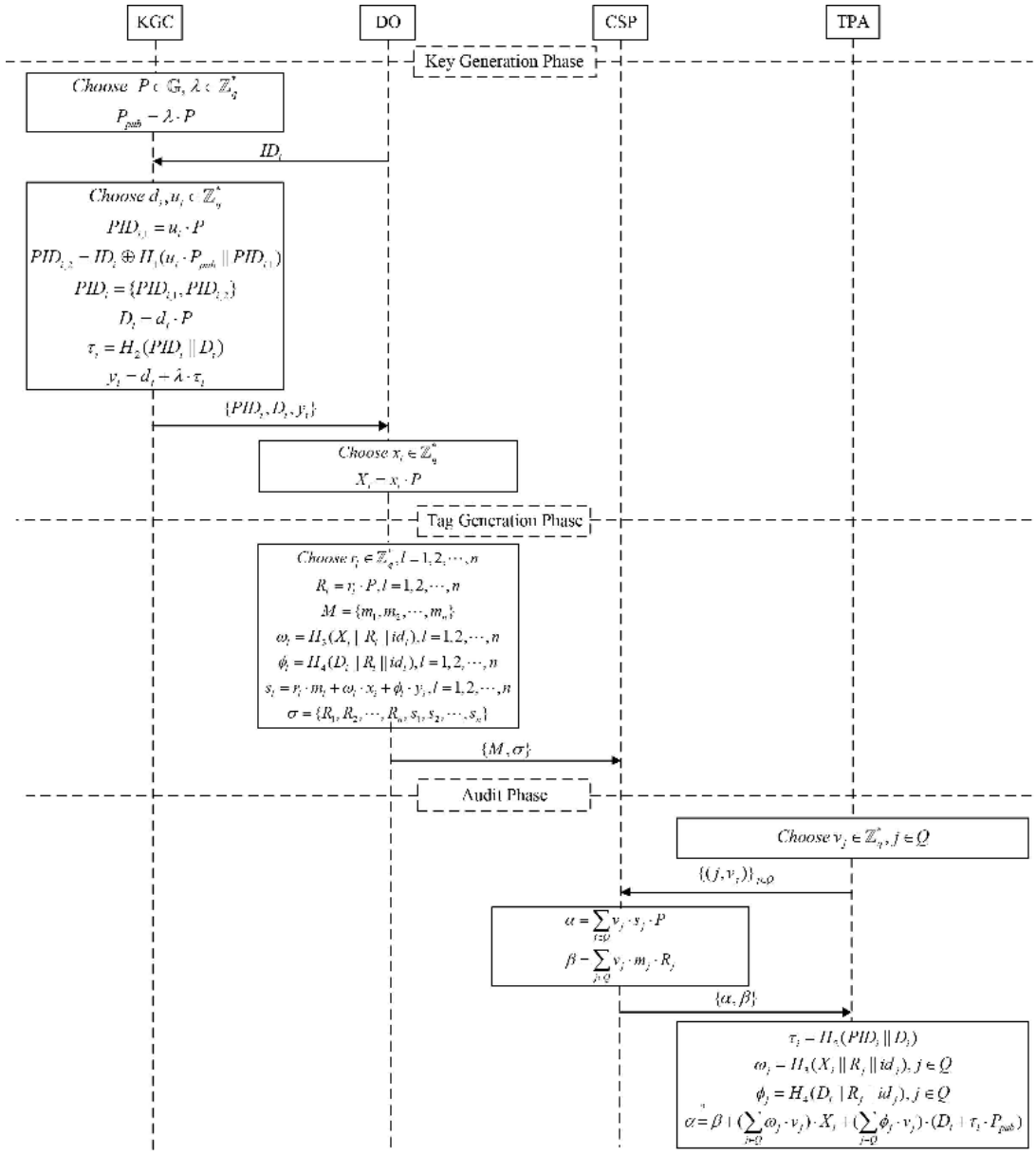

FIGURE 2. Overview of the proposed scheme.

Correctness

$$
\begin{aligned}
\alpha &= \sum_{j \in Q} v_j s_j \cdot P \\
&= \sum_{j \in Q} v_j (r_j \cdot m_j + \omega_j x_i + \phi_j \cdot y_i) \cdot P \\
&= \sum_{j \in Q} v_j r_j m_j \cdot P + \sum_{j \in Q} v_j \omega_j x_i \cdot P + \sum_{j \in Q} v_j \phi_j y_i \cdot P \\
&= \sum_{j \in Q} v_j m_j \cdot R_j + (\sum_{j \in Q} v_j \omega_j) \cdot X_i + (\sum_{j \in Q} v_j \phi_j) \\
&\quad \cdot (D_i + \tau_i \cdot P_{pub}) \\
&= \beta + (\sum_{j \in Q} \omega_j \cdot v_j) \cdot X_i + (\sum_{j \in Q} \phi_j \cdot v_j) \cdot (D_i + \tau_i \cdot P_{pub})
\end{aligned}
$$

V. SECURITY ANALYSIS

In this section, we give a formal and strict security proof of the proposed CLPDP scheme. Furthermore, error detection probability, data privacy preservation, identify privacy preservation and traceability are analyzed.

A. SECURITY PROOF

Theorem 1: Suppose there exists an adversary $\mathcal{A}_1$ could break the proposed scheme with the non-negligible probability ε, we can construct an algorithm $\mathcal{B}$ to solve the ECDL problem with the probability $\varepsilon' \geq \varepsilon \cdot (1 - \frac{q_{H_2} \cdot q_p}{q}) \cdot (1 - \frac{q_p}{q_{H_2}}) \cdot (1 - \frac{(q_{H_3} + q_{H_4}) \cdot q_t}{q}) \cdot (1 - \frac{q_t}{q_{H_2}}) \cdot \frac{1}{q_{H_2}}$, where q_{H_i} is the number of $H_i (i = 2, 3, 4)$ queries, q_p is the number of partial key queries and q_t is the number of tag queries.

Proof: Given an instance $(P, a\cdot P)$ of the ECDL problem, the task of $\mathcal{B}$ is finding an integer $a \in \mathbb{Z}_q^*$. $\mathcal{B}$ chooses an pseudo identity PID_O as a challenging identity and simulates $\mathcal{C}$ to interact with $\mathcal{A}_1$ in the **Game 1**. To response quickly and avoid the inconsistency, $\mathcal{B}$ maintains the following lists.

- H_1 list L_1: The form of L_1 is (Δ_i, μ_i) and it's initially an empty list.
- H_2 list L_2: The form of L_2 is (PID_i, D_i, τ_i) and it's initially an empty list.
- H_3 list L_3: The form of L_3 is $(X_i, R_l, id_l, \omega_l)$ and it's initially an empty list.
- H_4 list L_4: The form of L_4 is (D_i, R_l, id_l, ϕ_l) and it's initially an empty list.
- List L_5: The form of L_5 is $(PID_i, y_i, D_i, x_i, X_i)$ and it's initially an empty list.

Setup: $\mathcal{B}$ sets $P_{pub} = a\cdot P$ and returns the system parameters $params = \{p, q, \mathbb{G}, P, P_{pub}, H_1, H_2, H_3, H_4\}$ to $\mathcal{A}_1$.

H_1 query: $\mathcal{A}_1$ makes a hash H_1 query on Δ_i, $\mathcal{B}$ answers as follows:

- If Δ_i exists in L_1, $\mathcal{B}$ retrieves the tuple (Δ_i, μ_i) and returns μ_i to $\mathcal{A}_1$.
- Otherwise, $\mathcal{B}$ randomly picks a value $\mu_i \in \mathbb{Z}_q^*$, returns it to $\mathcal{A}_1$ and inserts (Δ_i, μ_i) to L_1.

H_2 query: $\mathcal{A}_1$ makes a hash H_2 query on (PID_i, D_i), $\mathcal{B}$ answers as follows:

- If (PID_i, D_i) exists in L_2, $\mathcal{B}$ retrieves the tuple (PID_i, D_i, τ_i) and returns τ_i to $\mathcal{A}_1$.
- Otherwise, $\mathcal{B}$ randomly picks a value $\tau_i \in \mathbb{Z}_q^*$, returns it to $\mathcal{A}_1$ and inserts (PID_i, D_i, τ_i) to L_2.

H_3 query: $\mathcal{A}_1$ makes a hash H_3 query on $(X_i, R_l, id_l, \omega_l)$, $\mathcal{B}$ answers as follows:

- If (X_i, R_l, id_l) exists in L_3, $\mathcal{B}$ retrieves the tuple $(X_i, R_l, id_l, \omega_l)$ and returns ω_l to $\mathcal{A}_1$.
- Otherwise, $\mathcal{B}$ randomly picks a value $\omega_l \in \mathbb{Z}_q^*$, returns it to $\mathcal{A}_1$ and inserts $(X_i, R_l, id_l, \omega_l)$ to L_3.

H_4 query: $\mathcal{A}_1$ makes a hash H_4 query on (D_i, R_l, id_l), $\mathcal{B}$ answers as follows:

- If (D_i, R_l, id_l) exists in L_4, $\mathcal{B}$ retrieves the tuple (D_i, R_l, id_l, ϕ_l) and returns ϕ_l to $\mathcal{A}_1$.
- Otherwise, $\mathcal{B}$ randomly picks a value $\phi_l \in \mathbb{Z}_q^*$, returns it to $\mathcal{A}_1$and inserts (D_i, R_l, id_l, ϕ_l) to L_4.

Partial key query: $\mathcal{A}_1$ makes a partial key query on PID_i, $\mathcal{B}$ first checks whether PID_i exists in L_2. If not, $\mathcal{B}$ makes the hash H_2 query on PID_i itself to generate τ_i. Then $\mathcal{B}$ answers as follows:

- If $PID_i \neq PID_O$, $\mathcal{B}$ checks whether PID_i exists in L_5. If PID_i exists in L_5, $\mathcal{B}$ retrieves the tuple $(PID_i, y_i, D_i, x_i, X_i)$ and returns (D_i, y_i) to $\mathcal{A}_1$. Otherwise, $\mathcal{B}$ randomly picks $y_i \in \mathbb{Z}_q^*$ and computes $D_i = y_i \cdot P - \tau_i \cdot a \cdot P$. If the tuple including τ_i already appears on the list L_2, $\mathcal{B}$ picks another $y_i \in \mathbb{Z}_q^*$, and tries again. Then, $\mathcal{B}$ returns (D_i, y_i) to $\mathcal{A}_1$ and inserts $(PID_i, y_i, D_i, x_i, X_i)$ to L_5.
- If $PID_i = PID_O$, $\mathcal{B}$ aborts.

Secret value query: $\mathcal{A}_1$ makes the secret value query on PID_i, $\mathcal{B}$ answers as follows:

- If PID_i exists in L_5, $\mathcal{B}$ retrieves the tuple $(PID_i, y_i, D_i, x_i, X_i)$ and returns x_i to $\mathcal{A}_1$.
- Otherwise, $\mathcal{B}$ randomly chooses $x_i \in \mathbb{Z}_q^*$ and sets $X_i = x_i \cdot P$. Then, $\mathcal{B}$ returns x_i to $\mathcal{A}_1$ and inserts $(PID_i, y_i, D_i, x_i, X_i)$ to L_5.

Public key query: $\mathcal{A}_1$ makes the public key query on PID_i, $\mathcal{B}$ answers as follows:

- If PID_i exists in L_5, $\mathcal{B}$ retrieves the tuples $(PID_i, y_i, D_i, x_i, X_i)$ and returns (D_i, X_i) to $\mathcal{A}_1$.
- Otherwise, $\mathcal{B}$ makes the partial key query and secret value query on PID_i. Then, $\mathcal{B}$ returns (D_i, X_i) to $\mathcal{A}_1$.

Public key replacement query: $\mathcal{A}_1$ makes the public key replacement query on (PID_i, D_i', X_i'), $\mathcal{B}$ executes as follows:

- If PID_i exists in L_5, $\mathcal{B}$ updates the tuple $(PID_i, y_i, D_i, x_i, X_i)$ to $(PID_i, y_i, D_i', x_i, X_i')$.
- Otherwise, $\mathcal{B}$ inserts $(PID_i, \perp, D_i', \perp, X_i')$ to L_5.

Tag query: $\mathcal{A}_1$ makes the tag query on (id_l, m_l) under (PID_i, D_i, X_i). $\mathcal{B}$ first checks whether PID_i exists in L_2, L_3 and L_4. If not, $\mathcal{B}$ makes the hash H_2, hash H_3 and hash H_4 query itself to generate τ_i, ω_i and ϕ_i. Then $\mathcal{B}$ answers as follows:

- If $PID_i \neq PID_O$, $\mathcal{B}$ randomly chooses $s_l \in \mathbb{Z}_q^*$, computes $R_l = m_l^{-1} \cdot (s_l \cdot P - \omega_l \cdot X_i - \phi_l(D_i + \tau_i))$ and returns (s_l, R_l) to $\mathcal{A}_1$. If ω_l already appears on the list L_3 or ϕ_l already appears on the list L_4, $\mathcal{B}$ picks another $s_l \in \mathbb{Z}_q^*$ and tries again.
- If $PID_i = PID_O$, $\mathcal{B}$ aborts.

Forgery: Finally, $\mathcal{A}_1$ outputs a forgery tag (s^*, R^*) on the block (m^*, id^*) under (PID^*, D^*, X^*). If $PID^* \neq PID_O$, $\mathcal{B}$ aborts and outputs fail. Otherwise, based on the forking lemma [33], $\mathcal{B}$ outputs another valid tag (s'^*, R^*) on the same block (m^*, id^*) by replaying procedure with the same random tape but a different choice of H_2. Then $\mathcal{B}$ has

$$s^* \cdot P = m^* \cdot R^* + \omega^* \cdot X^* + \phi^* \cdot (D^* + \tau^* \cdot a \cdot P),$$
$$s'^* \cdot P = m^* \cdot R^* + \omega^* \cdot X^* + \phi^* \cdot (D^* + \tau'^* \cdot a \cdot P).$$

Thus, $\mathcal{B}$ gets

$$s^* \cdot P - s'^* \cdot P = \phi^* \cdot \tau^* \cdot a \cdot P - \phi^* \cdot \tau'^* \cdot a \cdot P$$
$$s^* - s'^* = \phi^* \cdot \tau^* \cdot a - \phi^* \cdot \tau'^* \cdot a$$
$$a = \frac{s^* - s'^*}{\phi^* \cdot \tau^* - \phi^* \cdot \tau'^*}$$

The solution of ECDL problem is $a = \frac{s^* - s'^*}{\phi^* \cdot \tau^* - \phi^* \cdot \tau'^*}$.

Now, we analyze the probability ε' that $\mathcal{B}$ could solve the ECDL problem, and there are three possible events.

- E_1: $\mathcal{B}$ doesn't abort in the partial key query and tag query.
- E_2: $\mathcal{A}_1$ outputs a forgery tag (s^*, R^*) on (m^*, id^*) under (PID^*, D^*, X^*).
- E_3: $PID^* = PID_O$.

Therefore, the successful probability of solving ECDL problem is defined as

$$\varepsilon' = \Pr\ [E_1 \wedge E_2 \wedge E_3]$$
$$= \Pr\ [E_1] \cdot \Pr\ [E_2|E_1] \cdot \Pr\ [E_3|E_1 \wedge E_2].$$

Assuming that $\mathcal{A}_1$ can make at most q_{H_i} times $H_i(i = 1, 2, 3, 4)$ queries, q_p times partial key queries, q_s times secret value queries, q_{pk} times public key queries, q_{pkr} times public key replacement queries and q_t times tag queries.

We analyze the lower bounds of Pr $[E_1]$, Pr $[E_2|E_1]$, and Pr $[E_3|E_1 \wedge E_2]$ below.

The simulation of the partial key query aborts if the random oracle $H_2(PID_i||D_i)$ is inconsistent. Now, it occurs with the probability at most $\frac{q_{H_2}}{q}$. Therefore, the simulation for q_p times is successful with the probability at least $(1 - \frac{q_{H_2}}{q})^{q_p}$.

Partial key query successfully produces a partial key for $PID_i \neq PID_O$ with the probability $\frac{1}{q_{H_2}}$. Thus, it doesn't abort for q_p times with the probability at least $(1 - \frac{1}{q_{H_2}})^{q_p}$.

The simulation of the tag query fails if the random oracle $H_3(X_i||R_l||id_l)$ or $H_4(D_i||R_l||id_l)$ is inconsistent. Now, it occurs with the probability at most $\frac{q_{H_3}+q_{H_4}}{q}$. Therefore, the simulation for q_t times is successful with the probability at least $(1 - \frac{q_{H_3}+q_{H_4}}{q})^{q_t}$.

Tag query successfully produces a tag for $PID_i \neq PID_O$ with the probability $\frac{1}{q_{H_2}}$. Therefore, it doesn't abort for q_t times with the probability at least $(1 - \frac{1}{q_{H_2}})^{q_t}$.

It is assumed that the probability of $\mathcal{A}_1$ outputs a forgery tag on the block (m^*, id^*) under (PID^*, D^*, X^*) is ε.

$\mathcal{B}$ succeeds to forge a tag for $PID^* = PID_O$ with the probability $\frac{1}{q_{H_2}}$.

Therefore, we have

$$\begin{aligned}\Pr\ [E_1] &\geq (1-\frac{q_{H_2}}{q})^{q_p}\cdot(1-\frac{1}{q_{H_2}})^{q_p}\cdot(1-\frac{q_{H_3}+q_{H_4}}{q})^{q_t}\\ &\quad\cdot(1-\frac{1}{q_{H_2}})^{q_t}\\ &\geq (1-\frac{q_{H_2}\cdot q_p}{q})\cdot(1-\frac{q_p}{q_{H_2}})\\ &\quad\cdot(1-\frac{(q_{H_3}+q_{H_4})\cdot q_t}{q})\cdot(1-\frac{q_t}{q_{H_2}})\\ \Pr\ [E_2|E_1] &\geq \varepsilon\\ \Pr\ [E_3|E_1\wedge E_2] &\geq \frac{1}{q_{H_2}}\end{aligned}$$

Thus, the probability that $\mathcal{B}$ solves the ECDL problem is

$$\begin{aligned}\varepsilon' &= \Pr\ [E_1\wedge E_2\wedge E_3]\\ &\geq \varepsilon\cdot(1-\frac{q_{H_2}\cdot q_p}{q})\cdot(1-\frac{q_p}{q_{H_2}})\cdot(1-\frac{(q_{H_3}+q_{H_4})\cdot q_t}{q})\\ &\quad\cdot(1-\frac{q_t}{q_{H_2}})\cdot\frac{1}{q_{H_2}}\end{aligned}$$

Theorem 2: Suppose there exists an adversary $\mathcal{A}_2$ could break the proposed scheme with the non-negligible probability ε, we can construct an algorithm $\mathcal{B}$ to solve the ECDL problem with the probability $\varepsilon' \geq \varepsilon\cdot(1-\frac{q_s}{q_{H_2}})\cdot(1-\frac{(q_{H_3}+q_{H_4})\cdot q_t}{q})\cdot(1-\frac{q_t}{q_{H_2}})\cdot\frac{1}{q_{H_2}}$, where q_{H_i} is the number of $H_i(i = 2, 3, 4)$ queries, q_s is the number of secret value queries and q_t is the number of tag queries.

Proof: Given an instance $(P, a\cdot P)$ of the ECDL problem, the task of $\mathcal{B}$ is finding an integer $a \in \mathbb{Z}_q^*$. $\mathcal{B}$ chooses an identity PID_O as a challenging identity and simulates $\mathcal{C}$ to interact with $\mathcal{A}_2$ in the **Game 2**. As in **Theorem 1**, the lists L_1, L_2, L_3, L_4 and L_5 are maintained by $\mathcal{A}_2$.

Setup: $\mathcal{B}$ randomly picks a value $\lambda \in \mathbb{Z}_q^*$ as the master key and sets $P_{pub} = \lambda\cdot P$. $\mathcal{B}$ returns the master key λ and the system parameters $params = \{p, q, \mathbb{G}, P, P_{pub}, H_1, H_2, H_3, H_4\}$ to $\mathcal{A}_2$.

Hash H_1, H_2, H_3, H_4 queries: It is same to **Theorem 1**.

Secret value query: $\mathcal{A}_2$ makes the secret value query on the pseudo identify PID_i, $\mathcal{B}$ answers as follows:

- If $PID_i \neq PID_O$, $\mathcal{B}$ checks whether PID_i exists in L_5. If PID_i exists in L_5, $\mathcal{B}$ retrieves the tuple $(PID_i, y_i, D_i, x_i, X_i)$ and returns x_i to $\mathcal{A}_2$. Otherwise, $\mathcal{B}$ randomly picks $x_i \in \mathbb{Z}_q^*$ and computes $X_i = x_i\cdot P$. Then, $\mathcal{B}$ returns x_i to $\mathcal{A}_2$ and inserts $(PID_i, y_i, D_i, x_i, X_i)$ to L_5.
- If $PID_i = PID_O$, $\mathcal{B}$ aborts.

Public key query: $\mathcal{A}_2$ makes the public key query on PID_i, $\mathcal{B}$ answers as follows:

- If $PID_i \neq PID_O$, $\mathcal{B}$ randomly picks $x_i \in \mathbb{Z}_q^*$ and computes $X_i = x_i\cdot P$. Then, $\mathcal{B}$ returns X_i to $\mathcal{A}_2$ and inserts $(PID_i, y_i, D_i, x_i, X_i)$ to L_5.
- If $PID_i = PID_O$, $\mathcal{B}$ randomly picks $x_i \in \mathbb{Z}_q^*$ and computes $X_i = x_i\cdot a\cdot P$. Then, $\mathcal{B}$ returns X_i to $\mathcal{A}_2$ and inserts $(PID_i, y_i, D_i, \perp, X_i)$ to L_5.

Tag query: It is the same to **Theorem 1**.

Forgery: Finally, $\mathcal{A}_2$ outputs a forgery tag (s^*, R^*) on the block (m^*, id^*) under (PID^*, X^*, D^*). If $PID^* \neq PID_O$, $\mathcal{B}$ aborts and outputs fail. Otherwise, based on the forking lemma [33], $\mathcal{B}$ outputs another valid tag (s'^*, R^*) on the same block (m^*, id^*) by replaying procedure with the same random tape but a different choice of H_3. Then $\mathcal{B}$ has

$$\begin{aligned}s^*\cdot P &= m^*\cdot R^* + \omega^*\cdot X^* + \phi^*\cdot(D^* + \tau^*\cdot P_{pub}),\\ s'^*\cdot P &= m^*\cdot R^* + \omega'^*\cdot X^* + \phi^*\cdot(D^* + \tau^*\cdot P_{pub}).\end{aligned}$$

Thus, $\mathcal{B}$ gets

$$\begin{aligned}s^*\cdot P - s'^*\cdot P &= \omega^*\cdot X^* - \omega'^*\cdot X^*\\ (s^* - s'^*)\cdot P &= (\omega^* - \omega'^*)\cdot x^*\cdot a\cdot P\\ s^* - s'^* &= (\omega^* - \omega'^*)\cdot x^*\cdot a\\ a &= \frac{s^* - s'^*}{(\omega^* - \omega'^*)\cdot x^*}\end{aligned}$$

The solution of ECDL problem is $a = \frac{s^*-s'^*}{(\omega^*-\omega'^*)\cdot x^*}$.

Now, we analyze the probability ε' that $\mathcal{B}$ could solve the ECDL problem, and there are three possible events.

- E_1: $\mathcal{B}$ doesn't abort in the secret value query and tag query.
- E_2: $\mathcal{A}_2$ outputs a forgery tag (s^*, R^*) on (m^*, id^*) under (PID^*, D^*, X^*).
- E_3: $PID^* \neq PID_O$.

Therefore, the successful probability of solving ECDL problem is defined as

$$\begin{aligned}\varepsilon' &= \Pr\ [E_1\wedge E_2\wedge E_3]\\ &= \Pr\ [E_1]\cdot\Pr\ [E_2|E_1]\cdot\Pr\ [E_3|E_1\wedge E_2].\end{aligned}$$

Assuming that $\mathcal{A}_2$ can make at most q_{H_i} times $H_i(i = 1, 2, 3, 4)$ queries, q_s times secret value queries, q_{pk} times public key queries and q_t times tag queries.

We analyze the lower bounds of Pr $[E_1]$, Pr $[E_2|E_1]$ and Pr $[E_3|E_1 \wedge E_2]$ below.

Secret value query successfully produces a secret value for $PID_i \neq PID_O$ with the probability $\frac{1}{q_{H_2}}$. Therefore, it doesn't abort for q_s times with the probability at least $(1 - \frac{1}{q_{H_2}})^{q_s}$.

The probability of success for the tag query is the same to **Theorem 1**.

It is assumed that the probability of $\mathcal{A}_2$ outputs a forgery tag on the block (m^*, id^*) under (PID^*, D^*, X^*) is ε.

$\mathcal{B}$ succeeds to forge a tag for $PID^* = PID_O$ with the probability $\frac{1}{q_{H_2}}$.

Therefore, we have

$$\begin{aligned}\Pr\ [E_1] &\geq (1-\frac{1}{q_{H_2}})^{q_s}\cdot(1-\frac{q_{H_3}+q_{H_4}}{q})^{q_t}\cdot(1-\frac{1}{q_{H_2}})^{q_t}\\ &\geq (1-\frac{q_s}{q_{H_2}})\cdot(1-\frac{(q_{H_3}+q_{H_4})\cdot q_t}{q})\cdot(1-\frac{q_t}{q_{H_2}})\\ \Pr\ [E_2|E_1] &\geq \varepsilon\\ \Pr\ [E_3|E_1\wedge E_2] &\geq \frac{1}{q_{H_2}}\end{aligned}$$

Thus, the probability that $\mathcal{B}$ solves the ECDL problem is

$$\begin{aligned}\varepsilon' &= \Pr\ [E_1 \wedge E_2 \wedge E_3]\\ &\geq \varepsilon\cdot(1-\frac{q_s}{q_{H_2}})\cdot(1-\frac{(q_{H_3}+q_{H_4})\cdot q_t}{q})\\ &\quad\cdot(1-\frac{q_t}{q_{H_2}})\cdot\frac{1}{q_{H_2}}\end{aligned}$$

Theorem 3: If the ECDL assumption holds in $\mathbb{G}$, the proposed CLPDP scheme is secure against adversary $\mathcal{A}_3$.

Proof: Assume the challenging message be $chal = \{(j, v_j)\}_{j\in Q}$, where Q is a subset (with c elements) of set $\{1, 2, \cdots, n\}$.

If $\mathcal{A}_3$ generates a data integrity proof $proof^* = (\alpha, \beta^*)$ for $chal = \{(j, v_j)\}_{j\in Q}$ that passes verification with the corrupt block m_j^*. Thus

$$\alpha = \beta^* + (\sum_{j\in Q}\omega_j\cdot v_j)\cdot X_i + (\sum_{j\in Q}\phi_j\cdot v_j)\cdot(D_i + \tau_i\cdot P_{pub}).$$

Meanwhile, according to the proposed CLPDP scheme, the real integrity proof $proof = (\alpha, \beta)$ for $chal = \{(j, v_j)\}_{j\in Q}$ also passes verification. So

$$\alpha = \beta + (\sum_{j\in Q}\omega_j\cdot v_j)\cdot X_i + (\sum_{j\in Q}\phi_j\cdot v_j)\cdot(D_i + \tau_i\cdot P_{pub}).$$

Based on the above two equations, it is clear that $\beta^* = \beta$. Hence

$$\sum_{j\in Q} v_j\cdot m_j^*\cdot R_j = \sum_{j\in Q} v_j\cdot m_j\cdot R_j.$$

Define $\Delta\theta = \sum_{j\in Q} v_j\cdot(m_j^* - m_j)\cdot r_j$, we have

$$\begin{aligned}\beta^* - \beta &= \sum_{j\in Q} v_j\cdot m_j^*\cdot R_j - \sum_{j\in Q} v_j\cdot m_j\cdot R_j\\ &= \sum_{j\in Q} v_j\cdot(m_j^* - m_j)\cdot r_j\cdot P\\ &= \Delta\theta\cdot P\\ &= 0\end{aligned}$$

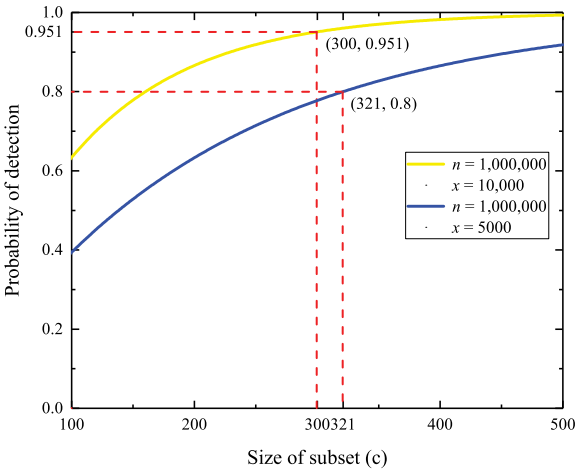

FIGURE 3. Error detection probability.

Given an instance $(W, a\cdot W)$ of the ECDL problem, we will find an integer $a \in \mathbb{Z}_q^*$. Let $P = \delta\cdot a\cdot W + \varphi\cdot W$, in which δ and φ are randomly selected from $\mathbb{Z}_q^*$. We can get the following equality

$$\begin{aligned}\Delta\theta\cdot P &= 0\\ \Delta\theta\cdot(\delta\cdot a\cdot W + \varphi\cdot W) &= 0\\ \delta\cdot\Delta\theta\cdot a\cdot W + \varphi\cdot\Delta\theta\cdot W &= 0\\ a\cdot W &= -\frac{\varphi}{\delta}\cdot W\end{aligned}$$

Then, we can obtain $a = -\frac{\varphi}{\delta}$, unless δ is zero. As δ is a random element in $\mathbb{Z}_q^*$, so δ is zero with the probability $\frac{1}{q}$, where q is a l arge prime. Therefore, the probability of solving the ECDL problem is $1 - \frac{1}{q}$.

B. ERROR DETECTION PROBABILITY

The proposed CLPDP scheme adopts the random sampling method [3] to detect data corruption which reduces the communication and computation costs of the TPA. The DO splits the data M into n blocks. The TPA chooses c $(c < n)$ blocks for challenge. Suppose that the CSP corrupts x blocks of n outsourced blocks. Let X be the number of challenged blocks that will match the corrupted blocks, and P_X be the probability that the misbehavior of the CSP will be detected.

$$\begin{aligned}P_X &= P\{X \geq 1\} = 1 - P\{X = 0\}\\ &= 1 - \frac{n-x}{n}\cdot\frac{n-1-x}{n-1}\cdot\cdots\cdot\frac{n-c+1-x}{n-c+1}\\ &\geq 1 - (\frac{n-x}{n})^c\end{aligned}$$

As a result, we get $P_X \geq 1 - (\frac{n-x}{n})^c$.

The comparisons of detection the probability with different corrupted blocks are denoted in Figure 3. Assume that the number of blocks in data M is $n = 1,000,000$. If the CSP corrupts $x = 5,000$ blocks out of $1,000,000$ blocks, then

TABLE 2. Security comparisons.

Schemes	Public verifiability	Blockless verification	Probabilistic detection	Data privacy preservation	Identify privacy preservation and traceability
Wang *et al.*'s scheme [16]	✓	✓	✓	✗	✗
He *et al.*'s scheme [17]	✓	✓	✓	✗	✗
Zhang *et al.*'s scheme [18]	✓	✓	✓	✗	✗
Kang *et al.*'s scheme [19]	✓	✓	✓	✓	✗
He *et al.*'s scheme [20]	✓	✓	✓	✓	✗
He *et al.*'s scheme [21]	✓	✓	✓	✓	✗
The proposed scheme	✓	✓	✓	✓	✓

the TPA randomly selects $c = 321$ blocks for the CSP to achieve the detection probability of at least 80%. Similarly, for $x = 10,000$, the TPA randomly selects only $c = 300$ blocks out of the $1,000,000$ blocks to realize the detection probability of at least 95%.

C. DATA PRIVACY PRESERVATION

In the audit phase, upon receiving the challenging message from the TPA, the CSP responds with the data integrity proof $proof = \{\alpha, \beta\}$, where $\alpha = \sum_{j\in Q} v_j \cdot s_j \cdot P$ and $\beta = \sum_{j\in Q} v_j \cdot m_j \cdot R_j$.

Case 1: The TPA tries to get data from α. Since $\alpha = \sum_{j\in Q} v_j \cdot s_j \cdot P$, the data is only included in s_j. As $v_j \cdot s_j \cdot P$ is an instance of the ECDL problem, it's computationally infeasible to obtain s_j from α. Therefore, the TPA can't find the DO's data from α.

Case 2: The TPA tries to get data from β. As $\beta = \sum_{j\in Q} v_j \cdot m_j \cdot R_j$, the TPA will try to deal with the equation $\beta = \sum_{j\in Q} v_j \cdot m_j \cdot R_j$ to find m_j, which is equivalent to solving the ECDL problem. Hence, the TPA can't find the DO's data from β.

Thus, the proposed CLPDP scheme satisfies data privacy preservation.

D. IDENTIFY PRIVACY PRESERVATION AND TRACEABILITY

In the proposed scheme, the DO_i uploads data M and tags σ to the CSP under the pseudo identity $PID_{i,1} = u_i \cdot P$ and $PID_{i,2} = ID_i \oplus H_1(u_i \cdot P_{pub}||PID_{i,1})$ for unknown u_i, where the real identity ID_i of the DO_i is perfectly hidden in random pseudo identity PID_i. To extract the DO_i's real identity ID_i, the adversary should compute $PID_{i,2} = ID_i \oplus H_1(u_i \cdot \lambda \cdot P||PID_{i,1})$. However, since $u_i \cdot \lambda \cdot P$ is an instance of ECDL problem, no adversary can use the pseudo identity PID_i to obtain the real identity ID_i. Therefore, the identity privacy is duly preserved.

By virtue of the pseudo identity $PID_i = \{PID_{i,1}, PID_{i,2}\}$ and the master key λ, it is the KGC that can trace the real identity of the DO_i based on $PID_{i,2} \oplus H_1(\lambda \cdot PID_{i,1}||PID_{i,1}) = ID_i$. Therefore, once a block is disputed, the KGC has the ability to trace the DO_i from the tag of the disputed block, where the traceability can be well satisfied.

TABLE 3. Execution time of operation (millisecond).

Notations	Descriptions	Execution time
T_p	Bilinear pairing operation	10.3092
T_H	Map-to-point hash function operation	3.5819
T_h	General hash operation	0.0294
T_A	Scale addition operation in $\mathbb{G}_1$	0.0172
T_M	Scale multiplication operation in $\mathbb{G}_1$	1.4202
T_a	Scale addition operation in $\mathbb{Z}_q^*$	0.0044
T_m	Scale multiplication operation in $\mathbb{Z}_q^*$	0.0044
T_{A-ECC}	Scale addition operation in ECC	0.0029
T_{M-ECC}	Scale multiplication operation in ECC	0.3851

The security comparisons between the proposed CLPDP scheme and several existing schemes [16]–[21] are shown in Table 2.

It is clear from Table 2 that all schemes realized probabilistic detection, public verifiability and blockless verification. However, in schemes [16]–[18], the TPA extracts the data during the auditing process by solving a series of linear equations [6]. Meanwhile, the DO's identity has been exposed in schemes [16]–[21]. In particular, only the proposed scheme simultaneously protects the data privacy and the identity privacy of the data owner.

VI. PERFORMANCE EVALUATION

In this section, the performance of the proposed scheme is compared with that of existing CLPDP schemes [16]–[21] in the perspective of computation and communication costs.

A. COMPUTATION COST

To be fair, we compare the proposed scheme and the existing schemes [16]–[21] under the same security level of 80-bit. For the pairing-based schemes [16]–[21], we choose a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$, where $\mathbb{G}_1$ is an additive group formed by a generator P with the order q on a super singular elliptic curve $E : y^2 = x^3 + x \bmod p$ with embedding degree 2, p is a 512-bit primer number and q is a 160-bit prime number satisfying the equation $q \cdot 12 \cdot r = p + 1$. For the proposed scheme, we employ an additive group $\mathbb{G}$ generated by a point P with the order q on a non-singular elliptic curve $E : y^2 = x^3 + ax + b \bmod p$, where p, q are 160-bit prime numbers, $a = -3$, and b is a random 160-bit prime number.

We rely on the MIRACL Crypto SDK [34] to derive the running time of related cryptographic operations. The experiment is run on 64-bit windows 10 operating system equipped with 2.53 GHz i5 CPU and 4 GB memory. The average execution time of the cryptographic operations running 10000 times is shown in Table 3.

TABLE 4. Comparison of computation cost.

Schemes	$TagGen$	$ProofGen$	$Verify$
Wang *et al.*'s scheme [16]	$nT_H + 2nT_A$ $+2nT_M$ $\approx 6.4567n$	$(c-1)T_A + cT_M$ $+(c-1)T_a + cT_m$ $\approx 1.4462c - 0.0216$	$3T_p + (c+1)T_H + (2c-1)T_A$ $+(2c+1)T_M$ $\approx 6.4567c + 35.9125$
He *et al.*'s scheme [17]	$(n+1)T_H + T_h + nT_A$ $+2nT_M + T_a + T_m$ $\approx 6.4395n + 3.6201$	$(c-1)T_A + cT_M$ $+(c-1)T_a + cT_m$ $\approx 1.4462c - 0.0216$	$2T_p + (c+1)T_H + 2T_h$ $+(c+2)T_A + (c+3)T_M$ $\approx 5.0193c + 28.4953$
Zhang *et al.*'s scheme [18]	$3T_H + nT_h + 4T_A$ $+(3n+3)T_M$ $\approx 4.2900n + 15.0751$	$(2c-2)T_A + 2cT_M$ $+(c-1)T_a + cT_m$ $\approx 2.8748c - 0.0388$	$4T_p + 5T_H + cT_h + 2T_A$ $+5T_M + (2c-2)T_a + cT_m$ $\approx 0.0426c + 66.2729$
Kang *et al.*'s scheme [19]	$(n+1)T_H + nT_h + 2nT_A$ $+4nT_M + nT_a$ $\approx 9.3137n + 3.5819$	$T_h + (c-1)T_A + (c+1)T_M$ $+cT_a + (c+1)T_m$ $\approx 1.4462c + 1.4368$	$4T_p + cT_H + (c+1)T_h + 2cT_A$ $+(2c+3)T_M + (c-1)T_a + cT_m$ $\approx 6.4949c + 45.5224$
He *et al.*'s scheme [20]	$nT_H + nT_A + 2nT_M$ $+T_a$ $\approx 6.4395n + 0.0044$	$2T_h + (c-1)T_A + (c+1)T_M$ $+cT_a + (c+1)T_m$ $\approx 1.4462c + 1.4662$	$2T_p + (c+1)T_H + 2T_h$ $+(c+3)T_A + (c+2)T_M$ $\approx 5.0193c + 27.1511$
He *et al.*'s scheme [21]	$(n+1)T_H + 2T_h + nT_A$ $+(2n+2)T_M + 2T_a + 2T_m$ $\approx 6.4395n + 6.4987$	$T_h + (c-1)T_A + (c+1)T_M$ $+cT_a + (c+1)T_m$ $\approx 1.4462c + 1.4368$	$2T_p + (c+1)T_H + 4T_h$ $+(c+4)T_A + (c+5)T_M$ $\approx 5.0193c + 31.4877$
The proposed scheme	$2nT_h + 2nT_a + 3nT_m$ $+nT_{M-ECC}$ $\approx 0.4659n$	$(c-1)T_a + 2cT_m + (c-1)T_{A-ECC}$ $+(c+1)T_{M-ECC}$ $\approx 0.4012c + 0.3866$	$(2c+1)T_h + (2c-2)T_a + 2cT_m$ $+3T_{A-ECC} + 3T_{M-ECC}$ $\approx 0.0764c + 1.1846$

Table 4 shows a comparison of the computation cost of the proposed CLPDP schemes with that of the existing CLPDP schemes [16]–[21].

In *TagGen* phase, the DO in Wang *et al.*'s scheme [16] has to carry out n map-to-point hash function operations, $2n$ scale addition operations in $\mathbb{G}_1$ and $2n$ scale multiplication operations in $\mathbb{G}_1$. Therefore, the running time is $nT_H + 2nT_A + 2nT_M \approx 6.4567n$ ms. The DO in He *et al.*'s scheme [17] has to carry out $n+1$ map-to-point hash function operations, a general hash operation, n scale addition operations in $\mathbb{G}_1$, $2n$ scale multiplication operations in $\mathbb{G}_1$, a scale addition operation in $\mathbb{Z}_q^*$ and a scale multiplication operation in $\mathbb{Z}_q^*$. Therefore, the running time is $(n+1)T_H + T_h + nT_A + 2nT_M + T_a + T_m \approx 6.4395n + 3.6201$ ms. The DO in Zhang *et al.*'s scheme [18] has to carry out three map-to-point hash function operations, n general hash operations, four scale addition operations in $\mathbb{G}_1$ and $3n+3$ scale multiplication operations in $\mathbb{G}_1$. Therefore, the running time is $3T_H + nT_h + 4T_A + (3n+3)T_M \approx 4.2900n + 15.0751$ ms. The DO in Kang *et al.*'s scheme [19] has to carry out $n+1$ map-to-point hash function operations, n general hash operations, $2n$ scale addition operations in $\mathbb{G}_1$, $4n$ scale multiplication operations in $\mathbb{G}_1$ and n scale addition operations in $\mathbb{Z}_q^*$. Therefore, the running time is $(n+1)T_H + nT_h + 2nT_A + 4nT_M + nT_a \approx 9.3137n + 3.5819$ ms. The DO in He *et al.*'s scheme [20] has to carry out n map-to-point hash function operations, n scale addition operations in $\mathbb{G}_1$, $2n$ scale multiplication operations in $\mathbb{G}_1$ and an addition point operation in $\mathbb{Z}_q^*$. Therefore, the running time is $nT_H + nT_A + 2nT_M + T_a \approx 6.4395n + 0.0044$ ms. The DO in He *et al.*'s scheme [21] has to carry out $n+1$ map-to-point hash function operations, two general hash operations, n scale addition operations in $\mathbb{G}_1$, $2n+2$ scale multiplication operations in $\mathbb{G}_1$, two scale addition operations in $\mathbb{Z}_q^*$ and two scale multiplication operations in $\mathbb{Z}_q^*$. Therefore, the running time is $(n+1)T_H + 2T_h + nT_A + (2n+2)T_M + 2T_a + 2T_m \approx 6.4395n + 6.4987$ ms. The DO in the proposed scheme has to carry out $2n$ general hash operations, $2n$ scale addition operations in $\mathbb{Z}_q^*$, $3n$ scale multiplication operations in $\mathbb{Z}_q^*$ and n scale multiplication operations in $\mathbb{G}$. Therefore, the running time is $2nT_h + 2nT_a + 3nT_m + 2T_{M-ECC} \approx 0.4659n$ ms.

In *ProofGen* phase, the CSP in Wang *et al.*'s scheme [16] requires to execute $c-1$ scale addition operations in $\mathbb{G}_1$, c scale multiplication operations in $\mathbb{G}_1$, $c-1$ scale addition operations in $\mathbb{Z}_q^*$ and c scale multiplication operations in $\mathbb{Z}_q^*$. Thus, the running time is $(c-1)T_A + cT_M + (c-1)T_a + cT_m \approx 1.4462c - 0.0216$ ms. The CSP in He *et al.*'s scheme [17] requires to execute $c-1$ scale addition operations in $\mathbb{G}_1$, c scale multiplication operations in $\mathbb{G}_1$, $c-1$ scale addition operations in $\mathbb{Z}_q^*$ and c scale multiplication operations in $\mathbb{Z}_q^*$. Thus, the running time is $(c-1)T_A + cT_M + (c-1)T_a + cT_m \approx 1.4462c - 0.0216$ ms. The CSP in Zhang *et al.*'s scheme [18] requires to execute $2c-2$ scale addition operations in $\mathbb{G}_1$, $2c$ scale multiplication operations in $\mathbb{G}_1$, $c-1$ scale addition operations in $\mathbb{Z}_q^*$ and c scale multiplication operations in $\mathbb{Z}_q^*$. Thus, the running time is $(2c-2)T_A + 2cT_M + (c-1)T_a + cT_m \approx 2.8748c - 0.0388$ ms. The CSP in Kang *et al.*'s scheme [19] requires to execute a general hash operation, $c-1$ scale addition operations in $\mathbb{G}_1$, $c+1$ scale multiplication operations in $\mathbb{G}_1$, c scale addition operations in $\mathbb{Z}_q^*$ and $c+1$ scale multiplication operations in $\mathbb{Z}_q^*$. Thus, the running time is $T_h + (c-1)T_A + (c+1)T_M + cT_a + (c+1)T_m \approx 1.4462c + 1.4368$ ms. The CSP in He *et al.*'s scheme [20] requires to

execute two general hash operations, $c-1$ scale addition operations in $\mathbb{G}_1$, $c+1$ scale multiplication operations in $\mathbb{G}_1$, c scale addition operations in $\mathbb{Z}_q^*$ and $c+1$ scale multiplication operations in $\mathbb{Z}_q^*$. Thus, the running time is $2T_h+(c-1)T_A+(c+1)T_M+cT_a+(c+1)T_m \approx 1.4462c+1.4662$ ms. The CSP in He *et al.*'s scheme [21] requires to execute a general hash operation, $c-1$ scale addition operations in $\mathbb{G}_1$, $c+1$ scale multiplication operations in $\mathbb{G}_1$, c scale addition operations in $\mathbb{Z}_q^*$ and $c+1$ scale multiplication operations in $\mathbb{Z}_q^*$. Thus, the running time is $T_h+(c-1)T_A+(c+1)T_M+cT_a+(c+1)T_m \approx 1.4462c+1.4368$ ms. The CSP in the proposed scheme requires to execute $c-1$ scale addition operations in $\mathbb{Z}_q^*$, $2c$ scale multiplication operations in $\mathbb{Z}_q^*$, $c-1$ scale addition operations in $\mathbb{Z}_q^*$ and $c+1$ scale multiplication operations in $\mathbb{G}$. Therefore, the running time is $(c-1)T_a+2cT_m+(c-1)T_{A-ECC}+(c+1)T_{M-ECC} \approx 0.4012c+0.3866$ ms.

In terms of *Verify* phase, the TPA in Wang *et al.*'s scheme [16] needs to run three bilinear pairing operations, $c+1$ map-to-point hash function operations, $2c-1$ scale addition operations in $\mathbb{G}_1$ and $2c+1$ scale multiplication operations in $\mathbb{G}_1$. Thus, the running time is $3T_p+(c+1)T_H+(2c-1)T_A+(2c+1)T_M \approx 6.4567c+35.9125$ ms. The TPA in He *et al.*'s scheme [17] needs to run two bilinear pairing operations, $c+1$ map-to-point hash function operations, two general hash operations, $c+2$ scale addition operations in $\mathbb{G}_1$ and $c+3$ scale multiplication operations in $\mathbb{G}_1$. Therefore, the running time is $2T_p+(c+1)T_H+2T_h+(c+2)T_A+(c+3)T_M \approx 5.0193c+28.4953$ ms. The TPA in Zhang *et al.*'s scheme [18] needs to run four bilinear pairing operations, five map-to-point hash function operations, c general hash operations, two scale addition operations in $\mathbb{G}_1$, five scale multiplication operations in $\mathbb{G}_1$, $2c-2$ scale addition operations in $\mathbb{Z}_q^*$ and c scale multiplication operations in $\mathbb{Z}_q^*$. Thus, the running time is $4T_p+5T_H+cT_h+2T_A+5T_M+(2c-2)T_a+cT_m \approx 0.0426c+66.2729$ ms. The TPA in Kang *et al.*'s scheme [19] needs to run four bilinear pairing operations, c map-to-point hash function operations, $c+1$ general hash operations, $2c$ scale addition operations in $\mathbb{G}_1$, $2c+3$ scale multiplication operations in $\mathbb{G}_1$, $c-1$ scale addition operations in $\mathbb{Z}_q^*$ and c scale multiplication operations in $\mathbb{Z}_q^*$. Thus, the running time is $4T_p+cT_H+(c+1)T_h+2cT_A+(2c+3)T_M+(c-1)T_a+cT_m \approx 6.4949c+45.5224$ ms. The TPA in He *et al.*'s scheme [20] needs to run two bilinear pairing operations, $c+1$ map-to-point hash function operations, two general hash operations, $c+3$ scale addition operations in $\mathbb{G}_1$ and $c+2$ scale multiplication operations in $\mathbb{G}_1$. Therefore, the running time is $2T_p+(c+1)T_H+2T_h+(c+3)T_A+(c+2)T_M \approx 5.0193c+27.1511$ ms. The TPA in He *et al.*'s scheme [21] needs to run two bilinear pairing operations, $c+1$ map-to-point hash function operations, four general hash operations, $c+4$ scale addition operations in $\mathbb{G}_1$ and $c+5$ scale multiplication operations in $\mathbb{G}_1$. Therefore, the running time is $2T_p+(c+1)T_H+4T_h+(c+4)T_A+(c+5)T_M \approx 5.0193c+31.4877$ ms. The TPA in the proposed scheme needs to run $2c+1$ general hash operations, $2c-2$ scale addition operations in $\mathbb{Z}_q^*$, c scale multiplication operations in $\mathbb{Z}_q^*$, three scale addition operations in $\mathbb{G}$ and three scale multiplication operations in $\mathbb{G}$. Thus, the running time is $(2c+1)T_h+(2c-2)T_a+2cT_m+3T_{A-ECC}+3T_{M-ECC} \approx 0.0764c+1.1846$ ms.

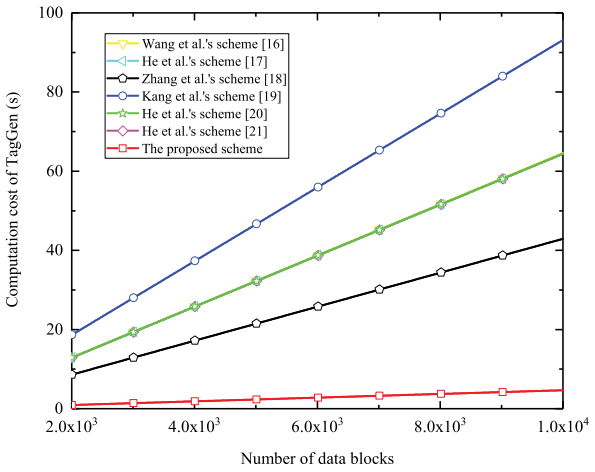

FIGURE 4. Computation cost of TagGen versus the number of data blocks.

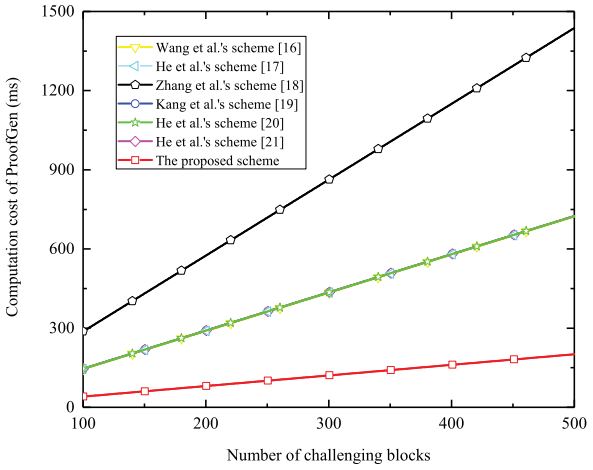

FIGURE 5. Computation cost of ProofGen versus the number of challenging blocks.

The computation cost of *TagGen*, *ProofGen* and *Verify* are illustrated in Figure 4, Figure 5 and Figure 6, respectively. As shown in Figures 4-6, the computation cost of the proposed scheme is always lower than that of other schemes [16]–[21]. It is worth note that the proposed scheme's advantage becomes more significant when the number of blocks increased. We assume $n = 10000$ and $c = 300$. In *TagGen* phase, the proposed scheme requires about 4.6590 s to generate tags while other schemes [16]–[21] require 64.5670 s, 64.3986 s, 42.9151 s, 93.1406 s, 64.3950 s, 64.4015 s, respectively, which are significantly decreased by 92.78%, 92.77%, 89.14%, 95.00%, 92.77%, 92.77%, separately. As for *ProofGen* phase, the computation cost of generating the proof for the compared schemes [16]–[21]

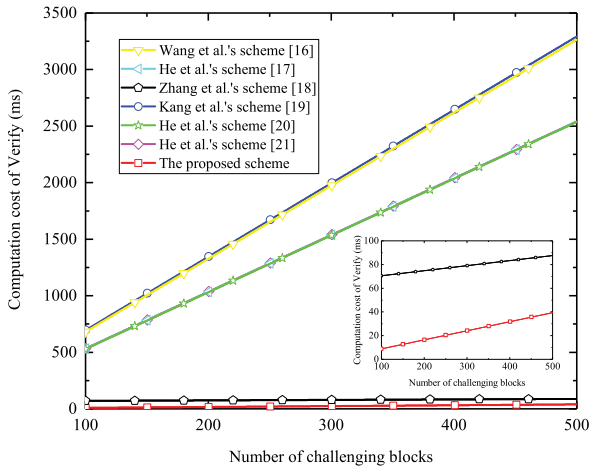

FIGURE 6. Computation cost of Verify versus the number of challenging blocks.

TABLE 5. Comparison of communication cost.

Schemes	Challenging Message (TPA to CSP)	Response Message (CSP to TPA)
Wang *et al.*'s scheme [16]	$c\left(\lvert n\rvert + \lvert\mathbb{Z}_q^*\rvert\right) = 192\times c$	$1\lvert\mathbb{G}_1\rvert + 1\lvert\mathbb{Z}_q^*\rvert = 672$
He *et al.*'s scheme [17]	$c\left(\lvert n\rvert + \lvert\mathbb{Z}_q^*\rvert\right) = 192\times c$	$1\lvert\mathbb{G}_1\rvert + 1\lvert\mathbb{Z}_q^*\rvert = 672$
Zhang *et al.*'s scheme [18]	$c\left(\lvert n\rvert + \lvert\mathbb{Z}_q^*\rvert\right) = 192\times c$	$2\lvert\mathbb{G}_1\rvert + 2\lvert\mathbb{Z}_q^*\rvert = 1344$
Kang *et al.*'s scheme [19]	$c\left(\lvert n\rvert + \lvert\mathbb{Z}_q^*\rvert\right) = 192\times c$	$2\lvert\mathbb{G}_1\rvert + 1\lvert\mathbb{Z}_q^*\rvert = 1184$
He *et al.*'s scheme [20]	$c\left(\lvert n\rvert + \lvert\mathbb{Z}_q^*\rvert\right) = 192\times c$	$2\lvert\mathbb{G}_1\rvert + 1\lvert\mathbb{Z}_q^*\rvert = 1184$
He *et al.*'s scheme [21]	$c\left(\lvert n\rvert + \lvert\mathbb{Z}_q^*\rvert\right) = 192\times c$	$4\lvert\mathbb{G}_1\rvert + 1\lvert\mathbb{Z}_q^*\rvert = 2208$
The proposed scheme	$c\left(\lvert n\rvert + \lvert\mathbb{Z}_q^*\rvert\right) = 192\times c$	$2\lvert\mathbb{G}\rvert = 320$

are 433.8384 ms, 433.8384 ms, 862.4012 ms, 435.2968 ms, 435.3262 ms, 435.2968 ms, respectively, while the proposed scheme is 120.7466 ms, approximately 27.83%, 27.83%, 14.00%, 27.74%, 27.84%, 27.74% separately of the time required for other schemes [16]–[21]. In *Verify* phase, the computation cost of the proposed scheme reaches 24.1046 ms whereas that of other schemes [16]–[21] reach 1972.9225 ms, 1534.2853 ms, 79.0529 ms, 1993.9924 ms, 1532.9411 ms, 1537.2777 ms respectively, nearly 98.78%, 98.43%, 69.51%, 98.79%, 98.43%, 98.43% of saving, respectively. Obviously, the proposed scheme is more efficient than other schemes [16]–[21].

B. COMMUNICATION COST

At the point, we present the comparison of the communication cost of the proposed scheme with the existing CLPDP schemes [16]–[21]. Specifically, we use the length of the transmitted message to represent the communication cost. According to the above, the length of $\mathbb{G}_1$, $\mathbb{Z}_q^*$, $\mathbb{G}$ and n are 512 bits, 160 bits, 160 bits and 32 bits separately, where n denotes the number of blocks in data M.

The communication cost includes the challenge message from TPA to CSP and a response message from CSP to TPA. Table 5 shows a comparison of the communication cost of the proposed CLPDP schemes with that of the existing CLPDP schemes [16]–[21].

1)From TPA to CSP, in the proposed scheme and existing schemes [16]–[21], the TPA generates the challenging message $chal = \{(j, v_j)\}_{j\in Q}$ and sends it to the CSP, where $j \in n$ and $v_j \in \mathbb{Z}_q^*$. Therefore, from TPA to CSP, all schemes have the same communication cost is $192 \times c$ bits as

$$c\left(|j| + |v_j|\right) = c\,(32 + 160) = 192 \times c \text{ bits.}$$

2)From CSP to TPA, in Wang *et al.*'s scheme [16], the CSP generates response message $proof = \{\sigma^*, \mu\}$ and sends it to the TPA, where $\sigma^* \in \mathbb{G}_1$ and $\mu \in \mathbb{Z}_q^*$. Thus, the communication cost is 672 bits as

$$|\sigma^*| + |\mu| = 512 + 160 = 672 \text{ bits.}$$

In He *et al.*'s scheme [17], the CSP generates response message $proof = \{S, m\}$ and sends it to the TPA, where $S \in \mathbb{G}_1$ and $m \in \mathbb{Z}_q^*$. Thus, the communication cost is 672 bits as

$$|S| + |m| = 512 + 160 = 672 \text{ bits.}$$

In Zhang *et al.*'s scheme [18], the CSP generates response message $proof = \{S, R, \mu, \Delta\}$ and sends it to the TPA, where $S, R \in \mathbb{G}_1$ and $\mu, \Delta \in \mathbb{Z}_q^*$. Thus, the communication cost is 1344 bits as

$$|S|+|R|+|\mu|+|\Delta| = 512+512+160+160 = 1344 \text{ bits.}$$

In Kang *et al.*'s scheme [19], the CSP generates response message $proof = \{\sigma, L, \mu\}$ and sends it to the TPA, where $\sigma, L \in \mathbb{G}_1$ and $\mu \in \mathbb{Z}_q^*$. Thus, the communication cost is 1184 bits as

$$|\sigma| + |L| + |\mu| = 512 + 512 + 160 = 1184 \text{ bits.}$$

In He *et al.*'s scheme [20], the CSP generates response message $proof = \{R, S, \omega\}$ and sends it to the TPA, where $R, S \in \mathbb{G}_1$ and $\omega \in \mathbb{Z}_q^*$. Thus, the communication cost is 1184 bits as

$$|R| + |S| + |\omega| = 512 + 512 + 160 = 1184 \text{ bits.}$$

In He *et al.*'s scheme [21], the CSP generates response message $proof = \{X_F, \Phi_F, R_{CS}, \Phi_{CS}, s_{CS}\}$ and sends it to the TPA, where $X_F, \Phi_F, R_{CS}, \Phi_{CS} \in \mathbb{G}_1$ and $s_{CS} \in \mathbb{Z}_q^*$. Thus, the communication cost is 2208 bits as

$$\begin{aligned}|X_F| + |\Phi_F| + |R_{CS}| + |\Phi_{CS}| + |s_{CS}| \\ = 512 + 512 + 512 + 512 + 160 = 2208 \text{ bits.}\end{aligned}$$

In the proposed scheme, the CSP generates response message $proof = \{\alpha, \beta\}$ and sends it to the TPA, where $\alpha, \beta \in \mathbb{G}$. Thus, the communication cost is 320 bits as

$$|\alpha| + |\beta| = 160 + 160 = 320 \text{ bits.}$$

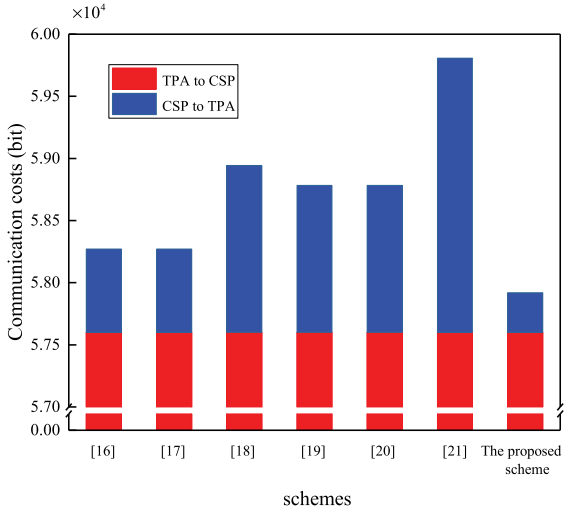

FIGURE 7. Communication cost.

When $c = 300$, Figure 7 depicts a comparison of the communication cost between the existing schemes [16]–[21] and the proposed scheme. As shown in Figure 7, from TPA to CSP, all schemes have the same communication cost of 57600 bit. However, from CSP to TPA, the proposed scheme can save 352 bits, 352 bits, 1024 bits, 864 bits, 864 bits and 1888 bits bandwidth, respectively, significantly decreased by 52.38%, 52.38%, 76.19%, 72.97%, 72.97% and 85.51% respectively compared with other schemes [16]–[21]. Apparently, the proposed scheme has less communication cost than other schemes [16]–[21].

VII. CONCLUSION

In this paper, we have put forward an efficient privacy-preserving CLPDP scheme for cloud storage. The proposed scheme can eliminate the issues of expensive certificate management and key escrow by taking advantage of certificateless cryptography. By means of the random sampling method, the proposed scheme satisfies provable data possession. In addition, the proposed scheme doesn't need any bilinear pairing and map-to-point hash operations. Meanwhile, the proposed scheme settles the issue of data privacy preservation and the DO's identify privacy preservation. Furthermore, we show that the proposed scheme is provably secure in the random oracle model. Lastly, performance analysis has illustrated that the proposed scheme is efficient and practical in the perspective of computation and communication costs.

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, Apr. 2010.

[2] D. Quick, B. Martini, and R. Choo, *Cloud Storage Forensics*. Waltham, MA, USA: Syngress, 2013.

[3] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proc. CCS*, 2007, pp. 598–609.

[4] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Proc. ASIACRYPT*. Berlin, Germany: Springer-Verlag, 2008, pp. 90–107.

[5] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *Proc. INFOCOM*, Mar. 2010, pp. 1–9.

[6] G. Ateniese, R. Di Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in *Proc. SecureCOMM*, 2008, Art. no. 9.

[7] F. Sebe, J. Domingo-Ferrer, A. Martinez-balleste, Y. Deswarte, and J. Quisquater, "Efficient remote data possession checking in critical information infrastructures," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 8, pp. 1034–1038, Aug. 2008.

[8] C. C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *Proc. CCS*, 2009, pp. 213–222.

[9] S. K. Nayak and S. Tripathy, "SEPDP: Secure and efficient privacy preserving provable data possession in cloud storage," *IEEE Trans. Serv. Comput.*, to be published. doi: 10.1109/TSC.2018.2820713.

[10] H. Wang, Q. Wu, B. Qin, and J. Domingo-Ferrer, "Identity-based remote data possession checking in public clouds," *IET Inf. Secur.*, vol. 8, no. 2, pp. 114–121, Mar. 2014.

[11] 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.

[12] H. Wang, D. He, and S. Tang, "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, Jun. 2016.

[13] H. Wang, D. He, J. Yu, and Z. Wang, "Incentive and unconditionally anonymous identity-based public provable data possession," *IEEE Trans. Serv. Comput.*, to be published. doi: 10.1109/TSC.2016.2633260.

[14] Y. Li, Y. Yu, G. Min, W. Susilo, J. Ni, and K.-K. R. Choo, "Fuzzy identity-based data integrity auditing for reliable cloud storage systems," *IEEE Trans. Depend. Sec. Comput.*, vol. 16, no. 1, pp. 72–83, Feb./Feb. 2019.

[15] Y. Zhang, J. Yu, R. Hao, C. Wang, and K. Ren, "Enabling efficient user revocation in identity-based cloud storage auditing for shared big data," *IEEE Trans. Depend. Sec. Comput.*, to be published. doi: 10.1109/TDSC.2018.2829880.

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

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

[18] Y. Zhang, C. Xu, S. Yu, H. Li, and X. Zhang, "SCLPV: Secure certificateless public verification for cloud-based cyber-physical-social systems against malicious auditors," *IEEE Trans. Comput. Social Syst.*, vol. 2, no. 4, pp. 159–170, Dec. 2015.

[19] B. Kang, J. Wang, and D. Shao, "Certificateless public auditing with privacy preserving for cloud-assisted wireless body area networks," *Mobile Inf. Syst.*, vol. 2017, Jul. 2017, Art. no. 2925465.

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

[21] D. He, "Certificateless provable data possession scheme for cloud-based smart grid data management systems," *IEEE Trans. Ind. Informat.*, vol. 14, no. 3, pp. 1232–1241, Mar. 2018.

[22] D. Kim and I. R. Jeong, "Certificateless public auditing protocol with constant verification time," *Secur. Commun. Netw.*, vol. 2017, Jan. 2017, Art. no. 6758618.

[23] F. Wang, L. Xu, and W. Gao, "Comments on 'SCLPV: Secure certificateless public verification for cloud-based cyber-physical-social systems against malicious auditors,'" *IEEE Trans. Comput. Social Syst.*, vol. 5, no. 3, pp. 854–857, Sep. 2018.

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

[25] 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.

[26] H. Yang, S. Jiang, W. Shen, and Z. Lei, "Certificateless provable group shared data possession with comprehensive privacy preservation for cloud storage," *Future Internet*, vol. 18, no. 6, p. 49, Jun. 2018.
[27] G. Wu, Y. Mu, W. Susilo, F. Guo, and F. Zhang, "Privacy-preserving certificateless cloud auditing with multiple users," *Wireless Pers. Commun.*, vol. 106, no. 3, pp. 1161–1182, Jun. 2019.
[28] S. S. Al-Riyami and K. G. Paterson, "Certificateless public key cryptography," in *Proc. ASIACRYPT*. Berlin, Germany: Springer-Verlag, 2003, pp. 452–473.
[29] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," in *Proc. ASIACRYPT*. Berlin, Germany: Springer-Verlag, 2001, pp. 514–532.
[30] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Proc. ASIACRYPT*. Berlin, Germany: Springer-Verlag, 1984, pp. 47–53.
[31] V. S. Miller, "Use of elliptic curves in cryptography," in *Proc. Crypto*. Berlin, Germany: Springer-Verlag, 1985, pp. 416–426.
[32] N. Koblitz, "Elliptic curve cryptosystems," *Math. Comput.*, vol. 48, no. 177, pp. 203–209, Jan. 1987.
[33] D. Pointcheval and J. Stern, "Security arguments for digital signatures and blind signatures," *J. Cryptol.*, vol. 13, no. 3, pp. 361–396, Mar. 2000.
[34] Shamus Software. *Multi Precision Integer and Rational Arithmetic Cryptographic Library (MIRACL)*. Accessed: Jun. 10, 2019. [Online]. Available: http://www.certivox.com/miracl/

YANG MING (M'18) received the B.S. and M.S. degrees in mathematics from the Xi'an University of Technology, in 2002 and 2005, respectively, and the Ph.D. degree in cryptography from Xidian University, in 2008. He is currently a Professor with Chang'an University. His main research interests include cryptography and wireless network security.

WENCHANG SHI received the B.S. degree from Shanxi Agricultural University, in 2016. He is currently pursuing the master's degree with Chang'an University, Xi'an. His main research interests include cryptography and cloud computing security.