# Secure and Efficient Certificateless Provable Data Possession for Cloud-Based Data Management Systems

Jing Zhang[1], Jie Cui[1], Hong Zhong[1], Chengjie Gu[2], and Lu Liu[3]

[1] School of Computer Science and Technology, Anhui University, Hefei, China
root_zj@163.com, cuijie@mail.ustc.edu.cn, zhongh@ahu.edu.cn
[2] Security Research Institute, New H3C Group, Hefei, China gu.chengjie@h3c.com
[3] School of Informatics, University of Leicester, UK l.liu@leicester.ac.uk

**Abstract.** Cloud computing provides important data storage, processing and management functions for data owners who share their data with data users through cloud servers. Although cloud computing brings significant advantages to data owners, the data stored in the cloud also faces many internal/external security attacks. Existing certificateless data provider schemes have the following two common shortcomings, i.e., most of which use plaintext to store data and use the complex bilinear pairing operation. To address such shortcomings, this scheme proposes secure and efficient certificateless provable data possession for cloud-based data management systems. In our solution, the data owners and cloud servers need to register with the key generation center only once. To ensure the integrity of encrypted data, we use the public key of the cloud server to participate in signature calculation. Moreover, the third-party verifier can audit the integrity of ciphertext without downloading the whole encrypted data. Security analysis shows that our proposed scheme is provably secure under the random oracle model. An evaluation of performance shows that our proposed scheme is efficient in terms of computation and communication overheads.

**Keywords:** Cloud Data Management · Provable Data Possession (PDP) · Certificateless Cryptography · Security · Efficient.

## 1 Introduction

With the rapid development of cloud computing, more and more people outsource their data to cloud servers[1, 13], which brings three main advantages. Firstly, resource-constrained users no longer need to process and store a large amount of data, so that a lot of computing and storage costs can be saved. Secondly, users can access data anytime and anywhere without requiring high-performance hardware. Thirdly, users can share data conveniently.

Although cloud services bring many benefits to people's lives, many challenges[5, 3] need to be solved properly. Firstly, user loses the direct control of

their outsourced data, i.e., whether the data has been modified or deleted is unknown. Secondly, the leakage of data may damage the privacies of users, such as the time when users are not at home and the routes that users frequently travel. In the worst cases, the property safety of users may be threaten. Therefore, how to ensure the confidentiality and integrity of outsourced data has become a great concern to users.

At present, some researchers have proposed provable data possession (PDP) schemes for the integrity of outsourced data[2, 12, 22, 17, 9, 15, 18, 14, 20, 11, 6, 19, 5, 21, 4]. Although the existing schemes ensure the integrity of cloud storage data, they do not consider the confidentiality of data. Moreover, due to the usage of complex bilinear pairing operation, these schemes also bring heavy computation and communication costs to the third-party verifier (TPV). Therefore, it is urgent to design a secure and efficient provable data possession scheme for cloud data management systems.

### 1.1 Related Work

To ensure the integrity of outsourced data, Ateniese et al.[2] first introduced the concept of PDP in 2007, and further considered public validation. Many PDP schemes[12, 22, 17, 9, 8] that follow Ateniese et al.'s work have been introduced to protect the integrity of outsourced data. Unfortunately, these schemes have a common drawback, i.e., most of which rely on trusted third parties to generate certificates for users, so that users have serious certificate management problems and heavy computing costs.

To solve the certificate management issues, Wang et al.[15] proposed an identity-based PDP scheme and provided a corresponding security model. To improve performance and security, some identity-based PDP schemes have also been proposed[18, 14, 20]. However, these schemes have a common disadvantage, that is, they need the secret key generation center to generate a series of private keys for users, which brings the key escrow problem.

To overcome the key escrow problem, a series of certificateless provable data possession (CL-PDP) schemes have been proposed[11, 6, 19, 5, 21, 4]. Unfortunately, there are still many security issues in these schemes. Zhang et al.[19] pointed out that schemes[11, 6] cannot guarantee the privacy of data. He et al.[5] discovered that scheme[19] had a malicious server attack and proposed an improved scheme. Recently, Zhou et al.[21] discovered that scheme[5] is vulnerable to tag forging and data loss hiding attacks. In addition, these schemes use complex bilinear pairing operations, which bring deficiencies in terms of computation and communication.

### 1.2 Contribution

To achieve the security of outsourced data and further reduce the waste of resources, this paper proposes a secure and efficient certificateless provable data possession scheme for cloud-based data management systems. There are three main contributions of the proposed scheme.

1. We propose to use a symmetric and asymmetric encryption algorithms simultaneously, which cannot only realize the security of data sharing, but also further ensure the confidentiality of outsourced data.
2. The proposed scheme can resist the attack of Type I and Type II adversaries, and can resist the tag forgery attack. The security analysis reveals that our scheme is provably secure under the random oracle model.
3. The detailed comparisons with the existing related schemes in terms of computational and communication overhead on the Tag Generation, Generate-Proof and Verify-Proof Algorithms, demonstrates that our scheme provides better performance.

The outline of the rest study is as follows: In section 2, we introduce the background of this study. In section 3, we put forward the proposed scheme. The security analysis is proved in section 4. The performance evaluation is outlined in section 5. Lastly, we present the conclusion of this study in section 6.

## 2 Background

In this section, we introduce the preliminary knowledge and network model.

### 2.1 Elliptic Curve Cryptosystem (ECC)

Let $E_p$: $y^2 = x^3 + ax + b(\mathrm{mod}p)$ be a non-singular elliptic curve over the finite field $F_p$, where $p > 3$ is a large prime, $a, b \in F_p$, and $4a^3 + 27b^2(\mathrm{mod}\,p) \neq 0$. Let $G$ be a cyclic group on $E_p$ of prime order $q$.

***Discrete Logarithm (DL) Problem:*** Given two random points $P, Q \in G$, where $Q = xP$, $x \in Z_q^*$, and $Z_q^* = \{1, 2, ..., q-1\}$, it is difficult to calculate $x$ from $Q$ in a probabilistic polynomial time (PPT).

### 2.2 Network Model

The system architecture comprises a key generation center KGC, a cloud server CS, a data owner DO and a third-party verifier TPV. As shown in Fig. 1, the details of each component are described as follows:

- **KGC**: It is a trusted third party, which is in charge of generating and publishing system parameters. It also generates a partial key for each DO and delivers these sensitive information to them via secure channels.
- **CS**: It is an honest but curious entity that is assumed to have sufficient computing and storage capabilities.
- **DO**: It is a resource constrained data owner, who outsources their data to CS and entrusts TPV to verify the integrity of cloud storage data.
- **TPV**: It verifies the integrity of cloud storage data when users need it, and is responsible for the verification results.

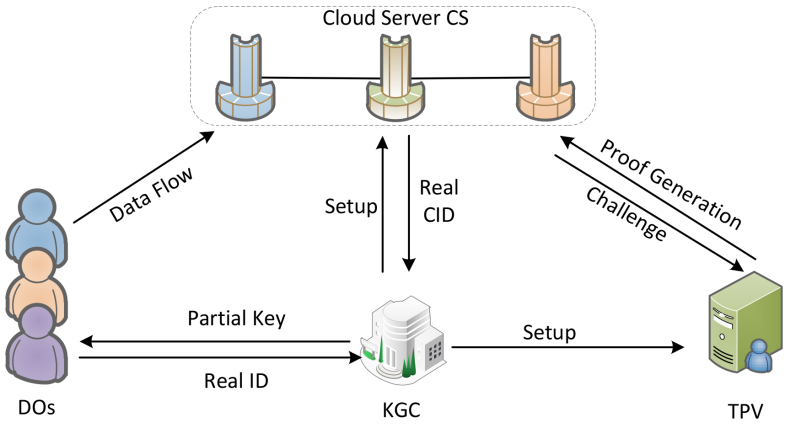


**Fig. 1.** A network model of the CL-PDP protocol

## 3 Proposed Scheme

In this section, we describe the proposed CL-PDP scheme based on ECC to solve the security problem and reduce the time cost.

### 3.1 Setup Algorithm

Given a security parameter $\lambda$, KGC generates a cyclic group $G$ with prime order $q$ and generator $P$. Then, KGC randomly chooses $s \in Z_q^*$ and computes the system public key $P_{pub} = s \cdot P$. KGC selects six one-way hash function $H_0 : \{0,1\}^* \to \{0,1\}^q$, $H_k : \{0,1\}^* \to Z_q^*, k = 1,2,3,4,5$. Finally, KGC publishes system parameters $params = \{q, Z_q^*, P_{pub}, H\}$ and saves the master key $s$ secretly.

### 3.2 Key Generation Algorithm

Given the real identities $ID_i, CID_k \in Z_q^*$ of $DO_i$ and $CS_k$, KGC performs as follows:

- KGC randomly selects $\alpha_k \in Z_q^*$ as $CS_k$'s secret key $sk_k = \alpha_k$, and computes $PK_k = \alpha_k P$ as $CS_k$'s public key. Then, KGC sends the key $\{sk_k, PK_k\}$ to $CS_k$ via a secure channel.
- KGC randomly picks $\alpha_i \in Z_q^*$ and computes $A_i = \alpha_i P$, $h_{i,1} = H_1(ID_i \| A_i)$ and $sk_1 = \alpha_i + h_{i,1} s (\mathrm{mod} q)$. Then, KGC sends the partial key $\{sk_1, A_i\}$ to $DO_i$ via a secure channel.
- DO randomly chooses $\beta \in Z_q^*$ as their secret value $sk_2 = \beta$ and computes $PK_i = \beta P$ as their public key.

### 3.3 Store Algorithm

**Encrypt Data** DO first divides their data $M$ into $n$ blocks: $M = \{M_l\}_{l=1}^{n}$. DO then generates a corresponding signature for each block of data.

- DO randomly picks $x_M \in Z_q^*$, $\delta \in \{0,1\}^q$, computes $X_M = x_M P$, and saves $\{x_M, X_M\}$ as a one-time-use signing key and verification key, respectively.
- DO computes $h_{i,2} = H_2(E_K(M_1)\|...\|E_K(M_n)\|S_1\|...\|S_n\|\delta\|X_M)$, $Z = h_{i,2}PK_k$, $Y = \delta + H_0(h_{i,2}P)$, $h_{i,3} = H_3(ID_i\|A_i\|PK_i)$, and $S_M = h_{i,2}x_M + h_{i,3}sk_2 + sk_1$. Note that authorized users can utilize the secret key $K$ to decrypt data $M_l = D_K(E_K(M_l))$.

**Tag Generation** Through the execution of this algorithm, DO produces a Tag for each block of data and stores the encrypted data into the cloud.

- DO randomly picks $x_l \in Z_q^*$, computes $X_l = x_l P$, $h_{i,4}^l = H_4(ID_i\|name_l\|X_l\|PK_i)$, $h_{i,5}^l = H_5(name_l\|X_l\|A_i)$, and $S_l = E_K(M_l)x_l + h_{i,4}^l sk_2 + h_{i,5}^l sk_1$. Note that $name_l$ denotes the unique name of data $M_l$.
- DO outputs $T_l = \{X_l, S_l, E_K(M_l)\}$ as $M_l$'s tag.
- Finally, DO sends $\{X_M, S_M, Z, Y, \{T_l\}_{l=1}^{n}\}$ to CS.

**Store** After receiving the request from the DO, CS computes $\delta = Y + H_0(Z^{'})$ by decrypting $Z^{'} = h_{i,2}P = Zsk_k^{-1}$.

- CS computes $h_{i,1} = H_1(ID_i\|A_i)$, $h_{i,3} = H_3(ID_i\|A_i\|PK_i)$ and $h_{i,2} = H_2(E_K(M_1)\|...\|E_K(M_n)\|S_1\|...\|S_n\|\delta\|X_M)$. CS then checks whether the following condition is true.

$$S_M P = h_{i,2}X_M + h_{i,3}PK_i + A_i + h_{i,1}P_{pub} \tag{1}$$

- If it is not true, CS immediately stops the session. Otherwise, CS computes $h_{i,4}^l = H_4(ID_i\|name_l\|X_l\|PK_i)$, $h_{i,5}^l = H_5(name_l\|X_l\|A_i)$ and verifies the condition.

$$\sum_{l=1}^{n} S_l P = \sum_{l=1}^{n}[E_K(M_l)X_l] + \sum_{l=1}^{n} h_{i,4}^l PK_i + \sum_{l=1}^{n} h_{i,5}^l(A_i + h_{i,1}P_{pub}) \tag{2}$$

  If the verification holds, CS stores the encrypted data; otherwise, CS rejects the request.

### 3.4 Challenge Algorithm

Through the execution of this algorithm, a TPV produces a challenging message to verify the data integrity of data.

1. TPV chooses a random subset $I \in \{1, 2, ..., n\}$ and a small number $v_j$ for each $j \in I$.
2. TPV outputs $\{j, v_j\}_{j \in I}$ as a challenging message and returns it to CS.

### 3.5 Generate-Proof Algorithm

When CS receives the TPV's auditing challenge $\{j, v_j\}_{j\in I}$, CS produces the following steps to complete the proof.

1. CS calculates $S_{cs} = \sum_{j\in I} v_j S_j P$ and $C_{cs} = \sum_{j\in I}[v_j E_K(M_j)X_j]$.
2. CS outputs the proof $\{S_{cs}, C_{cs}\}$ and returns it to TPV.

### 3.6 Verify-Proof Algorithm

Upon receiving the proof $\{S_{cs}, C_{cs}\}$, TPV executes the following steps to check the correctness.

1. TPV calculates $h_{i,1} = H_1(ID_i\|A_i)$, $h^j_{i,4} = H_4(ID_i\|name_j\|X_l\|PK_i)$ and $h^j_{i,5} = H_5(name_j\|X_l\|A_i)$.
2. TPV checks whether the following equation holds.

$$S_{cs} = C_{cs} + \sum_{j\in I}(v_j h^j_{i,4})PK_i + \sum_{j\in I}(v_j h^j_{i,5})(A_i + h_{i,1}P_{pub}) \tag{3}$$

   If the equation holds, the TPV outputs "Accept"; otherwise, TPV outputs "Reject".

## 4 Security Analysis

In this section, we firstly present a security model for the proposed scheme. Ane then, we analyze and prove the security of the proposed CL-PDP scheme.

### 4.1 Security Model

There are two types of unbounded adversaries namely $\mathcal{A}_1$ and $\mathcal{A}_2$. Type I adversary $\mathcal{A}_1$ can replace the public key of the user but doesn't access the master key. Type II adversary $\mathcal{A}_2$ cannot access replace the public key of the user but has ability to access the master key. The adversary $\mathcal{A}_1$ and $\mathcal{A}_2$ and the challenger $\mathcal{C}$ could make the following queries in the game.

**Setup.** In this query, $\mathcal{C}$ inputs the master key and public parameters. $\mathcal{C}$ keeps the master key secretly and sends the public parameters $\mathcal{A}$. $\mathcal{C}$ also sends the master key to $\mathcal{A}$ if $\mathcal{A}$ is a Type II adversary.

**Query.** In this query, $\mathcal{A}$ can make some queries and $\mathcal{C}$ answers back:

1. *Create Data Owner*: $\mathcal{C}$ executes the key generation algorithm to generate the DO's partial private key and secret value, and returns the DO's public key to $\mathcal{A}$.
2. *Extract Partial Private Key*: $\mathcal{C}$ returns a partial private key of DO to $\mathcal{A}$ as an answer.
3. *Public Key Replacement*: $\mathcal{A}$ can replace the public key of DO with a new value chosen by $\mathcal{A}$.

4. *Extract Secret Value*: $\mathcal{C}$ returns a secret value of ID to $\mathcal{A}$ as an answer.
5. *Generate Tag*: $\mathcal{C}$ generates a Tag of a block and returns it to $\mathcal{A}$.

**Forge.** $\mathcal{A}$ outputs a one-time-use verification key $S^*$ and a Tag $X^*$ corresponding the challenging identity $ID^*$.

$\mathcal{A}$ wins the game if the following requirements are satisfied:

1. $T^*$ is the corresponding valid tag of the challenging identity $ID^*$.
2. $T^*$ is not generated by querying *Generate Tag*.
3. $ID^*$ is independent of algorithm of *Extract Partial Private Key*/*Extract Secret Value* if $\mathcal{A}$ is Type I/Type II adversary.

**Definition 1.** The proposed certificateless provable data possession (CL-PDP) scheme is secure against forging Tag attack, if there is no any adversary $\mathcal{A} \in \{\mathcal{A}_1, \mathcal{A}_2\}$ which wins the above-mentioned game with a non-negligible probability.

### 4.2 Security Theorem

**Theorem 1.** *According to the assumption of the difficulty of the DL problem, the proposed CL-PDP scheme is secure against Type I adversary.*

*Proof.* Assuming given $P, Q = aP$, where $P, Q$ are two points on elliptic curve $E_q$, $\mathcal{A}_1$ can forge a one-time-use verification key $S^*$ and a Tag $X^*$ corresponding the challenging identity $ID^*$. We have built a game between $\mathcal{A}_1$ and a challenger $\mathcal{C}_1$, and $\mathcal{C}_1$ has the ability to run $\mathcal{A}_1$ with a non-negligible probability as a subroutine to solve DL problem.

***Setup***: The master key $s$ is randomly selected by challenger $\mathcal{C}_1$. And $\mathcal{C}_1$ then calculates the corresponding public key $P_{pub} = sP$. Next, $\mathcal{C}_1$ sends the system parameters $params = \{q, Z_q^*, P_{pub}, H\}$ to $\mathcal{A}_1$. $\mathcal{C}_1$ chooses a challenging identity $ID^*$ and answers the following queries from $\mathcal{A}_1$.

***$H_i$ queries***: When $\mathcal{A}_1$ uses the elements $m_i$ for $H_i$ query, $\mathcal{C}_1$ checks whether the elements $(m_i, \tau_{h_i})$ already exists in the hash list $L_{h_i} (i = 0, 1, ..., 5)$. If it is, $\mathcal{C}_1$ sends $\tau_{h_i} = H_1(m_i)$ to $\mathcal{A}_1$. Otherwise, $\mathcal{C}_1$ picks $\tau_{h_i} \in Z_q^*$ randomly and adds the elements $(m_i, \tau_{h_i})$ to the hash list $L_{h_i}$, then $\mathcal{C}_1$ sends $\tau_{h_i} = H_1(m_i)$ to $\mathcal{A}_1$.

***Create Data Owner query***: When $\mathcal{A}_1$ performs a create data owner query on the challenging identity $ID^*$, $\mathcal{C}_1$ checks whether the form $(ID_i, sk_2, sk_1, \alpha_i, PK_i, A_i)$ exists in $L_6$. If exists, $\mathcal{C}_1$ replies $(PK_i, A_i)$ to $\mathcal{A}_1$. Otherwise, $\mathcal{C}_1$ works as following:

- If $ID_i = ID^*$, $\mathcal{C}_1$ picks three elements $sk_1, sk_2, \tau_{h_1} \in Z_q^*$ randomly and computes $PK_i = sk_2 P$ and $A_i = sk_1 P - \tau_{h_1} P_{pub}$. $\mathcal{C}_1$ inserts the tuple $(ID_i, A_i, \tau_{h_1})$ and $(ID_i, sk_2, sk_1, \perp, PK_i, A_i)$ into $L_{h_1}$ and $L_6$, respectively. Note that $\perp$ denotes null.
- Otherwise, $ID_i \neq ID^*$, $\mathcal{C}_1$ picks three elements $\alpha_i, sk_2, \tau_{h_1} \in Z_q^*$ randomly and computes $PK_i = sk_2 P$ and $A_i = \alpha_i P$. $\mathcal{C}_1$ inserts the tuple $(ID_i, A_i, \tau_{h_1})$ and $(ID_i, sk_2, \perp, \alpha_i, PK_i, A_i)$ into $L_{h_1}$ and $L_6$, respectively.

***Extract Partial Private Key query***: Upon receiving $\mathcal{A}_1$'s query, $\mathcal{C}_1$ checks whether $ID_i$ already exists in hash list $L_{h_2}$. If $C_1$ cannot find the corresponding tuple, $C_1$ makes $H_1$ query on $ID_i$ itself to produce $\tau_{h_1}$. Then, $\mathcal{C}_1$ works as following:

- If $ID_i \neq ID^*$, $\mathcal{C}_1$ first checks whether $ID_i$ exists in $L_6$. If exists, $\mathcal{C}_1$ searches the tuple $(ID_i, sk_2, sk_1, \alpha_i, PK_i, A_i)$ and returns $(A_i, sk_1)$ to $\mathcal{A}_1$. Otherwise, $\mathcal{C}_1$ picks two element $sk_1, \tau_{h_1} \in Z_q^*$ and computes $A_i = sk_1 P - \tau_{h_1} P_{pub}$. Then, $\mathcal{C}_1$ returns $(A_i, sk_1)$ to $\mathcal{A}_1$ and stores $(ID_i, sk_2, sk_1, \alpha_i, PK_i, A_i)$ to $L_6$.
- Otherwise, $ID_i = ID^*$, $\mathcal{C}_1$ stops the game.

***Public Key Replacement query***: When $\mathcal{A}_1$ performs a public key replacement query on $(ID_i, A_i^*, PK_i^*)$, $\mathcal{C}_1$ first checks whether $ID_i$ exists in $L_6$. $\mathcal{C}_1$ answers as following:

- If list $L_6$ contains $ID_i$, $\mathcal{C}_1$ replaces the tuple $(ID_i, sk_2, sk_1, \alpha_i, PK_i, A_i)$ with $(ID_i, sk_2, sk_1, \alpha_i, PK_i^*, A_i^*)$.
- Otherwise, $\mathcal{C}_1$ inserts the tuple $(ID_i, \perp, \perp, \perp, PK_i^*, A_i^*)$ to $L_6$.

***Extract Secret Value query***: Upon receiving $\mathcal{A}_1$'s extract secret value query on $ID_i$, $\mathcal{C}_1$ answers as following:

- If list $L_6$ involves $(ID_i, sk_2, sk_1, \alpha_i, PK_i, A_i)$, $\mathcal{C}_1$ checks whether $sk_2 = \perp$ is true. If $sk_2 = \perp$, $\mathcal{C}_1$ sends $sk_2$ to $\mathcal{A}_1$. Otherwise, $\mathcal{C}_1$ performs a create data owner query to generate $PK_i = sk_2 P$. After that, $\mathcal{C}_1$ sends $sk_2$ to $\mathcal{A}_1$ and updates $(sk_i, PK_i)$ to list $L_6$.
- If list $L_6$ does not involve $(ID_i, sk_2, sk_1, \alpha_i, PK_i, A_i)$, $\mathcal{C}_1$ performs a create data owner query and sends $sk_2$ to $\mathcal{A}_1$. After that, $\mathcal{C}_1$ sends $sk_2$ to $\mathcal{A}_1$ and updates $(ID_i, sk_i, PK_i)$ to list $L_6$.

***Generate Tag query***: $\mathcal{A}_1$ performs a generate tag query on $(name_l, M_l)$ under $(ID_i, PK_i, A_i)$. $\mathcal{C}_1$ first checks whether $ID_i$ exists in $L_6$, $L_{h_1}$, $L_{h_4}$ and $L_{h_5}$. $\mathcal{C}_1$ answers as following:

- If $ID_i = ID^*$, $\mathcal{C}_1$ stops the game.
- Otherwise, $\mathcal{C}_1$ picks three elements $S_l, \tau_{h_1}, \tau_{h_4}, \tau_{h_5} \in Z_q^*$ randomly and computes $X_l = E_K(M_l)^{-1}(S_l P - \tau_{h_4} PK_i - \tau_{h_5}(A_i + \tau_{h_1} P_{pub}))$. Then, $\mathcal{C}_1$ returns $(S_l, X_l)$ to $\mathcal{A}_1$. Note that if $\tau_{h_4}$ or $\tau_{h_5}$ already exists in hash list $L_{h_4}$ or $L_{h_5}$, $\mathcal{C}_1$ picks an element $S_l$ and works again.

***Forgery***: At last, $\mathcal{A}_1$ outputs a $M_l$'s Tag $\{X_M, S_M^*, Z, Y, X_l, S_l^*, E_K(M_l)\}$ under $(ID_i, PK_i, A_i)$. If $ID_i \neq ID^*$, $\mathcal{C}_1$ aborts the game. Otherwise, on the basis of the forking lemma [10], $\mathcal{C}_1$ has the ability to get two different valid Tags $T_l = (X_l, S_l)$ and $T_l^* = (X_l, S_l^*)$ in polynomial time through $\mathcal{A}_1$, if $\mathcal{C}_1$ repeat the process with a different choice of $H_1$. We have the following equation:

$$S_l P = E_K(M_l) X_l + h_{i,4}^l PK_i + h_{i,5}^l (A_i + h_{i,1} P_{pub}) \tag{4}$$

$$S_l^* P = E_K(M_l) X_l + h_{i,4}^l PK_i + h_{i,5}^l (A_i + h_{i,1}^* P_{pub}) \tag{5}$$

Hence, we can get that

$$\begin{aligned}(S_l - S_l^*)P =& S_l P - S_l^* P \\ =& E_K(M_l)X_l + h_{i,4}^l PK_i + h_{i,5}^l(A_i + h_{i,1}P_{pub}) \\ & - E_K(M_l)X_l + h_{i,4}^l PK_i + h_{i,5}^l(A_i + h_{i,1}^* P_{pub}) \\ =& (h_{i,1} - h_{i,1}^*)h_{i,5}^l P_{pub} \\ =& a(h_{i,1} - h_{i,1}^*)h_{i,5}^l P\end{aligned} \tag{6}$$

and

$$a = \frac{S_l - S_l^*}{(h_{i,1} - h_{i,1}^*)h_{i,5}^l} \tag{7}$$

Thus, $\mathcal{C}_1$ could solve the DL problem. However, this is in contradiction with the difficulty of DL problem.

Similarly, if $\mathcal{A}_1$ could correctly guess the output of $H_2$, $\mathcal{C}_1$ also has the ability to get two different valid signatures $\{X_M, S_M, Z, Y\}$ and $\{X_M, S_M^*, Z, Y\}$ based on the forking lemma [10]. $\mathcal{C}_1$ also repeat the process with a different choice of $H_1$ and we have the following equation:

$$S_M P = h_{i,2}X_M + h_{i,3}PK_i + A_i + h_{i,1}P_{pub} \tag{8}$$

$$S_M^* P = h_{i,2}^* X_M + h_{i,3}PK_i + A_i + h_{i,1}^* P_{pub} \tag{9}$$

In the same way, if $h_{i,2} = h_{i,2}^*$, we can get $a = \frac{S_M - S_M^*}{h_{i,1} - h_{i,1}^*}$.

Unfortunately, the premise of this equation is not only that $\mathcal{A}_1$ can correctly guess the output of $H_2$, but also that $\mathcal{C}_1$ can solve the DL problem.

***Analysis***: The probability that $\mathcal{A}_1$ can correctly guess the output of $H_2$ is $\frac{1}{2^q}$. Assume $\mathcal{C}_1$ can solve the DL problem with negligible advantage $\varepsilon$. The following three events are used to analyze the probability that $\mathcal{C}_1$ can solve the DL problem.

- Event $E_1$: $\mathcal{A}_1$ can forge a valid Tag $\{X_M^*, S_M^*, Z^*, Y^*, X_l^*, S_l^*, E_K(M_l)^*\}$ under $(ID_i, PK_i, A_i)$.
- Event $E_2$: $\mathcal{C}_1$ does not abort when $\mathcal{A}_1$ performs extract partial private key query and generate tag query.
- Event $E_3$: $ID_i = ID^*$.

Under the random oracle model, a probabilistic polynomial-time adversary $\mathcal{A}_1$ forges a Tag in an attack modeled by the forking lemma after making $q_{H_i}(i = 1, 2, 3, 4, 5)$ times queries, $q_{ppk}$ times extract partial private key queries, and $q_{tag}$ times generate tag queries. We can achieve that $Pr(E_1) = \eta$, $Pr(E_2|E_1) = (1 - \frac{1}{q_{H_1}})^{q_{ppk}+q_{Tag}}$ and $Pr(E_3|E_1 \wedge E_2) = \frac{1}{q_{H_1}}$. The probability that $\mathcal{C}_1$ can solve the DL problem is

$$\begin{aligned}\varepsilon &= Pr(E_1 \wedge E_2 \wedge E_3) \\ &= Pr(E_3|E_1 \wedge E_2)Pr(E_2|E_1)Pr(E_1) \\ &= \frac{1}{q_{H_1}}(1 - \frac{1}{q_{H_1}})^{q_{ppk}+q_{Tag}} \cdot \eta\end{aligned} \tag{10}$$

Thus, the probability that $\mathcal{A}_1$ forges a Tag is $\varepsilon^{'} = \frac{1}{2^q} \cdot \varepsilon$.

Due to $\eta$ is non-negligible, $\varepsilon$ is also non-negligible. Thus, $\mathcal{C}_1$ can solve the DL problem with a non-negligible probability. However, it is difficult to solve the DL problem, namely, the proposed CL-PDP scheme is secure against Type I adversary.

**Theorem 2.** *According to the assumption of the difficulty of the DL problem, the proposed CL-PDP scheme is secure against Type II adversary.*

*Proof.* Assuming given $P, Q = aP$, where $P, Q$ are two points on elliptic curve $E_q$, $\mathcal{A}_2$ can forge a one-time-use verification key $S^*$ and a Tag $X^*$ corresponding the challenging identity $ID^*$. We have built a game between $\mathcal{A}_2$ and a challenger $\mathcal{C}_2$, and $\mathcal{C}_2$ has the ability to run $\mathcal{A}_2$ with a non-negligible probability as a subroutine to solve DL problem.

***Setup***: The master key $s$ is randomly selected by challenger $\mathcal{C}_2$. And $\mathcal{C}_2$ then calculates the corresponding public key $P_{pub} = sP$. Next, $\mathcal{C}_1$ sends the master key $s$ and system parameters $params = \{q, Z_q^*, P_{pub}, H\}$ to $\mathcal{A}_2$. $\mathcal{C}_2$ chooses a challenging identity $ID^*$ and answers the following queries from $\mathcal{A}_2$.

***$H_i$ queries***: Similar to ***$H_i$ queries*** in the Proof of **Theorem 1**.

***Create Data Owner query***: When $\mathcal{A}_2$ performs a create data owner query on the challenging identity $ID^*$, $\mathcal{C}_2$ checks whether the form $(ID_i, sk_2, sk_1, PK_i, A_i)$ exists in $L_6$. If exists, $\mathcal{C}_2$ replies $(PK_i, A_i)$ to $\mathcal{A}_2$. Otherwise, $\mathcal{C}_2$ works as following:

- If $ID_i = ID^*$, $\mathcal{C}_2$ picks three elements $\alpha_i, \tau_{h_1} \in Z_q^*$ randomly and computes $A_i = \alpha_i P$ and $sk_1 = \alpha_i + h_{i,1} s (\mathrm{mod} q)$. $\mathcal{C}_2$ inserts the tuple $(ID_i, A_i, \tau_{h_1})$ and $(ID_i, \perp, sk_1, PK_i, A_i)$ into $L_{h_1}$ and $L_6$, respectively.
- Otherwise, $ID_i \neq ID^*$, $\mathcal{C}_2$ picks three elements $\alpha_i, sk_2 \in Z_q^*$ randomly and computes $PK_i = sk_2 P$, $A_i = \alpha_i P$, $\tau_{h_1} = H_1(ID_i \| A_i)$ and $sk_1 = \alpha_i + \tau_{h_1} s (\mathrm{mod} q)$. $\mathcal{C}_2$ inserts the tuple $(ID_i, A_i, \tau_{h_1})$ and $(ID_i, sk_2, \perp, PK_i, A_i)$ into $L_{h_1}$ and $L_6$, respectively.

***Extract Partial Private Key query***: Upon receiving $\mathcal{A}_2$'s query, $\mathcal{C}_2$ checks whether $ID_i$ already exists in hash list $L_{h_2}$. If $C_2$ cannot find the corresponding tuple, $C_2$ makes $H_1$ query on $ID_i$ itself to produce $\tau_{h_1}$. Then, $\mathcal{C}_2$ works as following:

- If $ID_i \neq ID^*$, $\mathcal{C}_2$ first checks whether $ID_i$ exists in $L_6$. If exists, $\mathcal{C}_2$ searches the tuple $(ID_i, sk_2, sk_1, PK_i, A_i)$ and returns $(A_i, sk_1)$ to $\mathcal{A}_2$. Otherwise, $\mathcal{C}_2$ picks two element $sk_1, \tau_{h_1} \in Z_q^*$ and computes $A_i = sk_1 P - \tau_{h_1} P_{pub}$. Then, $\mathcal{C}_2$ returns $(A_i, sk_1)$ to $\mathcal{A}_2$ and stores $(ID_i, sk_2, sk_1, PK_i, A_i)$ to $L_6$.
- Otherwise, $ID_i = ID^*$, $\mathcal{C}_2$ searches the tuple $(ID_i, sk_2, sk_1, PK_i, A_i)$ and returns $(A_i, sk_1)$ to $\mathcal{A}_2$.

***Extract Secret Value query***: Upon receiving $\mathcal{A}_2$'s extract secret value query on $ID_i$, $\mathcal{C}_2$ answers as following:

- If $ID_i \neq ID^*$, $\mathcal{C}_2$ first checks whether $ID_i$ exists in $L_6$. If exists, $\mathcal{C}_2$ searches the tuple $(ID_i, sk_2, sk_1, PK_i, A_i)$ and returns $sk_2$ to $\mathcal{A}_2$. Otherwise, $\mathcal{C}_1$ picks two element $sk_2 \in Z_q^*$ and computes $pk_i = sk_2 P$. Then, $\mathcal{C}_2$ returns $sk_2$ to $\mathcal{A}_2$ and stores $(ID_i, sk_2, sk_1, PK_i, A_i)$ to $L_6$.
- Otherwise, $ID_i = ID^*$, $\mathcal{C}_2$ stops the game.

***Generate Tag query***: $\mathcal{A}_2$ performs a generate tag query on $(name_l, M_l)$ under $(ID_i, PK_i, A_i)$. $\mathcal{C}_2$ first checks whether $ID_i$ exists in $L_6$, $L_{h_1}$, $L_{h_4}$ and $L_{h_5}$. $\mathcal{C}_2$ answers as following:

- If $ID_i = ID^*$, $\mathcal{C}_2$ stops the game.
- Otherwise, $\mathcal{C}_2$ picks three elements $S_l, \tau_{h_1}, \tau_{h_4}, \tau_{h_5} \in Z_q^*$ randomly and computes $X_l = E_K(M_l)^{-1}(S_l P - \tau_{h_4} PK_i - \tau_{h_5}(A_i + \tau_{h_1} P_{pub}))$. Then, $\mathcal{C}_2$ returns $(S_l, X_l)$ to $\mathcal{A}_2$. Note that if $\tau_{h_4}$ or $\tau_{h_5}$ already exists in hash list $L_{h_4}$ or $L_{h_5}$, $\mathcal{C}_2$ picks an element $S_l$ and works again.

***Forgery***: At last, $\mathcal{A}_2$ outputs a $M_l$'s Tag $\{X_M, S_M^*, Z, Y, X_l, S_l^*, E_K(M_l)\}$ under $(ID_i, PK_i, A_i)$. If $ID_i \neq ID^*$, $\mathcal{C}_2$ aborts the game. Otherwise, on the basis of the forking lemma [10], $\mathcal{C}_2$ has the ability to get two different valid Tags $T_l = (X_l, S_l)$ and $T_l^* = (X_l, S_l^*)$ in polynomial time through $\mathcal{A}_2$, if $\mathcal{C}_2$ repeat the process with a different choice of $H_4$. We have the following equation:

$$S_l P = E_K(M_l) X_l + h_{i,4}^l PK_i + h_{i,5}^l (A_i + h_{i,1} P_{pub}) \tag{11}$$

$$S_l^* P = E_K(M_l) X_l + h_{i,4}^{l*} PK_i + h_{i,5}^l (A_i + h_{i,1} P_{pub}) \tag{12}$$

Hence, we can get that

$$\begin{aligned}(S_l - S_l^*)P =& S_l P - S_l^* P \\ =& E_K(M_l) X_l + h_{i,4}^l PK_i + h_{i,5}^l (A_i + h_{i,1} P_{pub}) \\ & - E_K(M_l) X_l + h_{i,4}^{l*} PK_i + h_{i,5}^l (A_i + h_{i,1} P_{pub}) \\ =& (h_{i,4}^l - h_{i,4}^{l*}) PK_i \\ =& sk_2 (h_{i,4}^l - h_{i,4}^{l*}) P\end{aligned} \tag{13}$$

and

$$sk_2 = \frac{S_l - S_l^*}{(h_{i,4}^l - h_{i,4}^{l*})} \tag{14}$$

Thus, $\mathcal{C}_2$ could solve the DL problem. However, this is in contradiction with the difficulty of DL problem.

Similarly, if $\mathcal{A}_2$ could correctly guess the output of $H_2$, $\mathcal{C}_2$ also has the ability to get two different valid signatures $\{X_M, S_M, Z, Y\}$ and $\{X_M, S_M^*, Z, Y\}$ based on the forking lemma [10]. $\mathcal{C}_2$ also repeat the process with a different choice of $H_1$ and we can get $sk_2 = \frac{S_l - S_l^*}{(h_{i,4}^l - h_{i,4}^{l*})}$.

***Analysis***: The probability that $\mathcal{A}_2$ can correctly guess the output of $H_2$ is $\frac{1}{2^q}$. Assume $\mathcal{C}_2$ can solve the DL problem with negligible advantage $\varepsilon$. The following three events are used to analyze the probability that $\mathcal{C}_2$ can solve the DL problem.

- Event $E_1$: $\mathcal{A}_2$ can forge a valid Tag $\{X_M^*, S_M^*, Z^*, Y^*, X_l^*, S_l^*, E_K(M_l)^*\}$ under $(ID_i, PK_i, A_i)$.
- Event $E_2$: $\mathcal{C}_2$ does not abort when $\mathcal{A}_2$ performs extract secret value query and generate tag query.
- Event $E_3$: $ID_i = ID^*$.

Under the random oracle model, a probabilistic polynomial-time adversary $\mathcal{A}_2$ forges a Tag in an attack modeled by the forking lemma after making $q_{H_i}(i = 1, 2, 3, 4, 5)$ times queries, $q_{sev}$ times extract secret value queries, and $q_{tag}$ times generate tag queries. We can achieve that $Pr(E_1) = \eta$, $Pr(E_2|E_1) = (1 - \frac{1}{q_{H_1}})^{q_{sev}+q_{Tag}}$ and $Pr(E_3|E_1 \wedge E_2) = \frac{1}{q_{H_1}}$. The probability that $\mathcal{C}_2$ can solve the DL problem is

$$\begin{aligned} \varepsilon &= Pr(E_1 \wedge E_2 \wedge E_3) \\ &= Pr(E_3|E_1 \wedge E_2)Pr(E_2|E_1)Pr(E_1) \\ &= \frac{1}{q_{H_1}}(1 - \frac{1}{q_{H_1}})^{q_{sev}+q_{Tag}} \cdot \eta \end{aligned} \tag{15}$$

Thus, the probability that $\mathcal{A}_2$ forges a Tag is $\varepsilon^{'} = \frac{1}{2^q} \cdot \varepsilon$.

Due to $\eta$ is non-negligible, $\varepsilon$ is also non-negligible. Thus, $\mathcal{C}_2$ can solve the DL problem with a non-negligible probability. However, it is difficult to solve the DL problem, namely, the proposed CL-PDP scheme is secure against Type II adversary.

### 4.3 Discussion

Table 1 compares the security and functionality feature analyse of the related schemes[19, 7, 5, 4] and our scheme. The symbol $\surd$ indicates that the scheme is secure or provides that feature. In contrast, the symbol $\times$ indicates that the scheme is insecure or does not provide that feature. This table indicates that only our proposed scheme can provide better security features than those of existing schemes[19, 7, 5, 4].

**Table 1.** Comparison of Security and Functionality Features.

| Security Features | Zhang et al.[19] | Kang et al.[7] | He et al.[5] | Gao et al.[4] | The proposed |
|---|---|---|---|---|---|
| Public verifiability | $\surd$ | $\surd$ | $\surd$ | $\surd$ | $\surd$ |
| Storage correctness | $\surd$ | $\surd$ | $\times$ | $\times$ | $\surd$ |
| Data privacy preserving | $\times$ | $\times$ | $\times$ | $\times$ | $\surd$ |
| Tag cannot be forged | $\surd$ | $\surd$ | $\times$ | $\surd$ | $\surd$ |
| Batch verification | $\surd$ | $\surd$ | $\surd$ | $\surd$ | $\surd$ |

## 5 Performance Evaluation

In this section, we discuss comparisons of computation and communication costs of the proposed CL-PDP scheme and other existing related schemes[19, 7, 5, 4].

Because the analyses of the other existing schemes are similar to the analysis of our proposed scheme, we discuss only our proposed scheme in the following subsection.

To compare fairness, bilinear pairing is constructed as follows: bilinear pairing $\bar{e}$: $G_1 \times G_1 \rightarrow G_2$ are built on the security level of 80-bit. $G_1$ is an additive group whose order is $\bar{q}$ and the generator is $\bar{p}$, which is a point on the super singular elliptic curve $\bar{E} : y^2 = x^3 + x \mod \bar{p}$ with an embedding degree of 2, where $\bar{p}$ is a 512-bit prime number and $\bar{q}$ is a 160-bit prime number. For elliptic curve-based scheme, we construct an additive group $G$ generated by a point $P$ with order $p$ on a non-singular elliptic curve $E$: $y^2 = x^3 + ax + b(\mod q)$ to achieve a security level of 80 bits, where $p, q$ are two 160 bit prime numbers.

### 5.1 Computation Cost

In our experiments, we used a computer that is HP with an Intel(R) Core(TM) i7-6700@ 3.4GHz processor, 8GB main memory, and the Ubuntu 14.04 operation system to derive the average execution time of the running 5000 times based on the MiRACL library[16]. To facilitate the analysis of computational cost, we list some notations about execution time, as shown in Table 2.

**Table 2.** Execution time of different cryptographic operations.

| Notations | Definitions | Execution time |
|---|---|---|
| $T_{bp}$ | Bilinear pairing operation | 5.086 ms |
| $T_{bp.m}$ | The scale multiplication operation based on bilinear pairing | 0.694 ms |
| $T_{bp.a}$ | The point addition operation based on bilinear pairing | 0.0018 ms |
| $T_H$ | The hash-to-point operation based on bilinear pairing | 0.0992 ms |
| $T_{e.m}$ | The scale multiplication operation based on ECC | 0.3218 ms |
| $T_{e.a}$ | Calculating the point addition operation related to ECC | 0.0024 ms |
| $T_h$ | Hash operation | 0.001 ms |

Table 3 shows the computational overhead of Tag Generation, Generate-Proof and Verify-Proof Algorithms. Note that $I$ represents the size of the subset $I \in \{1, 2, ..., n\}$.

For the Tag Generation Algorithm of the proposed scheme, the DO needs to execute one scalar multiplication operation and two hash function operations for each block $M_l$. Thus, the execution time of $n$ blocks is $nT_{e.m} + 2nT_h \approx 0.3266nms$. The computation of Generate-Proof Algorithm requires $(I + 1)$ scalar multiplication operations and $(I - 1)$ point addition operations related to the ECC, thus, the computation time of the phase is $(I + 1)T_{e.m} + (I - 1)T_{e.a} \approx 0.3242I + 0.3194ms$. For the Verify-Proof Algorithm of the proposed scheme, verifier executes three scalar multiplication operations, three point addition operations and $(2I + 1)$ hash function operations, Therefore, the execution time of the phase is $3T_{e.m} + 3T_{e.a} + (2I + 1)T_h \approx 0.002I + 0.9736ms$.

To make a more significant comparison, Fig. 2 and Fig. 3 are used to show that the computation cost of Generate-Proof Algorithm and Verify-Proof Algorithm increases with an increasing number of blocks, respectively. Based on

**Table 3.** Comparison of computation cost.

| Schemes | Tag Generation | Generate-Proof | Verify-Proof |
|---|---|---|---|
| Zhang et al.[19] | $(3n+3)T_{bp.m}+4T_{bp.a}$ $+3T_H+nT_h$ $\approx 2.083n+2.3868ms$ | $(2I)T_{bp.m}+(2I-2)T_{bp.a}$ $\approx 1.3916I-0.0036ms$ | $4T_{bp}+5T_{bp.m}+2T_{bp.a}$ $+5T_H+IT_h$ $\approx 0.001I+24.3136ms$ |
| Kang et al.[7] | $(4n)T_{bp.m}+(2n)T_{bp.a}$ $+(n+1)T_H+nT_h$ $\approx 2.8798n+0.0992ms$ | $(I+1)T_{bp.m}+(I-1)T_{bp.a}$ $+T_h\approx 0.6958I+0.6932ms$ | $4T_p+(2I+3)T_{bp.m}+IT_H$ $+(2I)T_{bp.a}+(I+1)T_h$ $\approx 1.3926I+22.5262ms$ |
| He et al.[5] | $(2n+2)T_{bp.m}+(n)T_{bp.a}$ $+(n+1)T_H+2T_h$ $\approx 1.488n+1.4892ms$ | $(I+1)T_{bp.m}+(I-1)T_{bp.a}$ $+T_h\approx 0.6958I+0.6932ms$ | $2T_p+(I+5)T_{bp.m}+4T_h$ $+(I+4)T_{bp.a}+(I+1)T_H$ $\approx 0.794I+13.7524ms$ |
| Gao et al.[4] | $(2n)T_{bp.m}+(n)T_{bp.a}$ $+T_H+(2n+1)T_h$ $\approx 1.3918n+0.1002ms$ | $(2I+2)T_{bp.m}+T_H+$ $(2I-1)T_{bp.a}+(I+1)T_h$ $\approx 1.3926I+1.4864ms$ | $3T_p+(I+3)T_{bp.m}+$ $IT_{bp.a}+T_H+4T_h$ $\approx 0.6958I+17.4464ms$ |
| The proposed | $nT_{e.m}+2nT_h$ $\approx 0.3266nms$ | $(I+1)T_{e.m}+(I-1)T_{e.a}$ $\approx 0.3242I+0.3194ms$ | $3T_{e.m}+3T_{e.a}+(2I+1)T_h$ $\approx 0.002I+0.9736ms$ |

an analysis and comparison of Table 3, Fig. 2 and Fig. 3, we conclude that the computation cost of the proposed scheme is lower than those of the related schemes[19, 7, 5, 4].

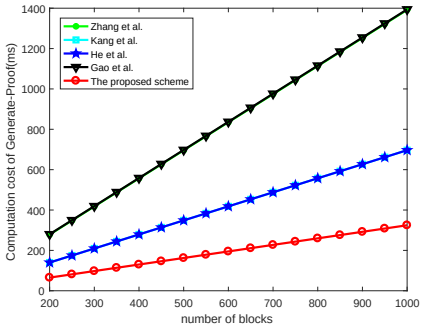


**Fig. 2.** Cost of Generate-Proof

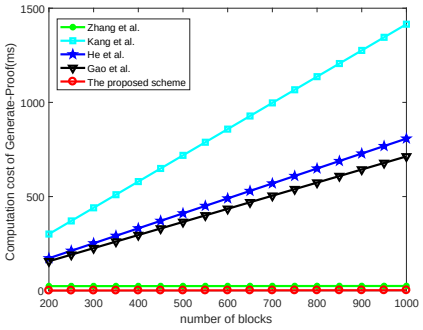


**Fig. 3.** Cost of Verify-Proof

### 5.2 Communication Cost

As $\overline{p}$ and $p$ are 64 and 20 bytes, the sizes of the elements in $G_1$ and $G$ are 64 $\times$ 2 = 128 bytes and 20 $\times$ 2 = 40 bytes, respectively. Set the size of block $l$ be 4 bytes and the length of $Z_q^*$ be 20 bytes. The communication cost of the five scheme are shown in Table 4.

In the proposed scheme, the TPV sends the challenging message $\{j, v_j\}_{j\in I}$ to CS, and the CS generates the response proof $\{S_{cs}, C_{cs}\}$ and returns it to TPV, where $j \in l$, $v_j \in Z_q^*$ and $S_{cs}, C_{cs} \in G$. Therefore, the communication cost of the proposed scheme is $(|l|+|Z_q^*|)I+2|G| = 24I+80$ bytes. According

**Table 4.** Comparison of Communication Cost.

| Schemes | Communication Costs |
|---|---|
| Zhang et al.[19] | $(\|l\| + \|Z_q^*\|)I + 2\|G_1\| + 2\|Z_q^*\| = 24I + 196$ bytes |
| Kang et al.[7] | $(\|l\| + \|Z_q^*\|)I + 2\|G_1\| + 1\|Z_q^*\| = 24I + 176$ bytes |
| He et al.[5] | $(\|l\| + \|Z_q^*\|)I + 4\|G_1\| + 1\|Z_q^*\| = 24I + 532$ bytes |
| Gao et al.[4] | $(\|l\| + \|Z_q^*\|)I + 2\|G_1\| + 1\|Z_q^*\| = 24I + 176$ bytes |
| The proposed | $(\|l\| + \|Z_q^*\|)I + 2\|G\| = 24I + 80$ bytes |

to Table 4, our proposed scheme expends less communication cost than those of other existing schemes[19, 7, 5, 4].

## 6 Conclusion

The proposed scheme can realize the confidentiality of outsourcing data and solve the problem of data privacy leakage in the cloud data management system. In order to ensure the integrity of encrypted data, we used not only a third-party verifier to randomly check and verify, but also the public key of cloud services to encrypt random strings, so that the reliability of the stored data can be further ensured. Moreover, the detailed analysis showed that the proposed scheme is secure against the Type-I and Type-II adversaries under the random oracle model. Additionally, we compared and analyzed existing schemes from the perspective of Tag Generation and Generate-Proof and Verify-Proof Algorithms. The results verified that our scheme can effectively reduce delays and improve authentication efficiency.

## Acknowledgment

The work was supported by the NSFC grant (No. U1936220, No. 61872001, No. 62011530046), and the Special Fund for Key Program of Science and Technology of Anhui Province, China (Grant No. 202003A05020043).

## References

1. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al.: A view of cloud computing. Communications of the ACM **53**(4), 50–58 (2010)
2. Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., Song, D.: Provable data possession at untrusted stores. In: Proceedings of the 14th ACM conference on Computer and communications security. pp. 598–609 (2007)
3. Fernandes, D.A., Soares, L.F., Gomes, J.V., Freire, M.M., Inácio, P.R.: Security issues in cloud environments: a survey. International Journal of Information Security **13**(2), 113–170 (2014)
4. Gao, G., Fei, H., Qin, Z.: An efficient certificateless public auditing scheme in cloud storage. Concurrency and Computation: Practice and Experience p. e5924 (2020)

5. He, D., Kumar, N., Zeadally, S., Wang, H.: Certificateless provable data possession scheme for cloud-based smart grid data management systems. IEEE Transactions on Industrial Informatics **14**(3), 1232–1241 (2018)
6. He, D., Zeadally, S., Wu, L.: Certificateless public auditing scheme for cloud-assisted wireless body area networks. IEEE Systems Journal **12**(1), 64–73 (2015)
7. Kang, B., Wang, J., Shao, D.: Certificateless public auditing with privacy preserving for cloud-assisted wireless body area networks. Mobile Information Systems **2017** (2017)
8. Ming, Y., Shi, W.: Efficient privacy-preserving certificateless provable data possession scheme for cloud storage. IEEE Access **7**, 122091–122105 (2019)
9. Nayak, S.K., Tripathy, S.: Sepdp: Secure and efficient privacy preserving provable data possession in cloud storage. IEEE Transactions on Services Computing (2018)
10. Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. Journal of cryptology **13**(3), 361–396 (2000)
11. Wang, B., Li, B., Li, H., Li, F.: Certificateless public auditing for data integrity in the cloud. In: 2013 IEEE conference on communications and network security (CNS). pp. 136–144. IEEE (2013)
12. Wang, C., Wang, Q., Ren, K., Lou, W.: Privacy-preserving public auditing for data storage security in cloud computing. In: 2010 proceedings ieee infocom. pp. 1–9. Ieee (2010)
13. Wang, F., Xu, L., Gao, W.: Comments on sclpv: Secure certificateless public verification for cloud-based cyber-physical-social systems against malicious auditors. IEEE Transactions on Computational Social Systems **5**(3), 854–857 (2018)
14. Wang, H., He, D., Tang, S.: Identity-based proxy-oriented data uploading and remote data integrity checking in public cloud. IEEE Transactions on Information Forensics and Security **11**(6), 1165–1176 (2016)
15. Wang, H., Wu, Q., Qin, B., Domingo-Ferrer, J.: Identity-based remote data possession checking in public clouds. IET Information Security **8**(2), 114–121 (2013)
16. Wenger, E., Werner, M.: Evaluating 16-bit processors for elliptic curve cryptography. In: International Conference on Smart Card Research and Advanced Applications. pp. 166–181. Springer (2011)
17. Yang, K., Jia, X.: An efficient and secure dynamic auditing protocol for data storage in cloud computing. IEEE transactions on parallel and distributed systems **24**(9), 1717–1726 (2012)
18. Yu, Y., Au, M.H., Ateniese, G., Huang, X., Susilo, W., Dai, Y., Min, G.: Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage. IEEE Transactions on Information Forensics and Security **12**(4), 767–778 (2016)
19. Zhang, Y., Xu, C., Yu, S., Li, H., Zhang, X.: Sclpv: Secure certificateless public verification for cloud-based cyber-physical-social systems against malicious auditors. IEEE Transactions on Computational Social Systems **2**(4), 159–170 (2015)
20. Zhang, Y., Yu, J., Hao, R., Wang, C., Ren, K.: Enabling efficient user revocation in identity-based cloud storage auditing for shared big data. IEEE Transactions on Dependable and Secure computing **17**(3), 608–619 (2020)
21. Zhou, C.: Security analysis of a certificateless public provable data possession scheme with privacy preserving for cloud-based smart grid data management system. International Journal of Network Security **22**(4), 584–588 (2020)
22. Zhu, Y., Hu, H., Ahn, G.J., Yu, M.: Cooperative provable data possession for integrity verification in multicloud storage. IEEE transactions on parallel and distributed systems **23**(12), 2231–2244 (2012)