Research Article

A Certificate-Based Provable Data Possession Scheme in the Standard Model

Caixue Zhou

School of Computer and Big Data Science, Jiujiang University, Jiujiang 332005, China

Correspondence should be addressed to Caixue Zhou; charlesjjjx@126.com

Received 31 March 2021; Revised 8 July 2021; Accepted 17 August 2021; Published 2 September 2021

Academic Editor: AnMin Fu

Copyright © 2021 Caixue Zhou. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Certificate-based cryptosystem can not only resolve the private key escrow problem inherent in the identity-based cryptosystem but also reduce the cost of public key certificate management in the public key infrastructure-(PKI-) based cryptosystem. Provable data possession (PDP) can ensure the integrity of users' data stored in the cloud at a very high probability. By combining these two concepts, we propose a certificate-based PDP scheme. We prove that our scheme is secure in the standard model assuming that the Squ-CDH problem is hard. Based on the index logic table, our scheme can be extended to support dynamic operations easily. Efficiency analysis shows that our scheme has high efficiency.

1. Introduction

With the rapid development of the cloud storage technology, more and more users and companies store their data in the cloud. However, a new problem has emerged, that is, how users can ensure the integrity of their data if they no longer physically hold them. Luckily, provable data possession (PDP) [1] can resolve this problem. PDP is a lightweight probabilistic integrity checking model of cloud data. It can ensure the integrity of data stored in the cloud at a high probability even if users save no data or only a small amount of data locally. It is one of the core technologies to support cloud storage security.

The certificate-based cryptosystem [2] was proposed to reduce the high cost of public key certificate management in the public key infrastructure- (PKI-) based cryptosystem [3] and to eliminate the private key escrow problem inherent in the identity-based cryptosystem [4]. Just like in the PKI-based cryptosystem, a user's public key in the certificate-based cryptosystem needs a certificate generated by the certificate authority (CA), except that in the latter case, the certificate should participate in the decryption or signing process along with the private key. Therefore, the certificate is verified implicitly. In this way, in the process of encryption or signature verification, the user need not care about

whether the certificate has been revoked or out of date, and as a result, the cost of certificate management is reduced. In addition, the private key is produced by the user himself, so the private key escrow problem is also eliminated.

By combining the certificate-based cryptosystem and the concept of PDP, we propose a certificate-based PDP scheme. Based on the assumption that the Squ-CDH problem is hard, we prove the security of our scheme in the standard model. Based on the index logical table [5], our scheme can be extended to support dynamic operations easily. At last, we evaluate the efficiency of our scheme, which shows that it is efficient.

1.1. Related Work. Ateniese et al. [1] first introduced the notion of PDP in 2007. Afterwards, Erway et al. [6] and Wang et al. [7] proposed dynamic PDP schemes, where users can modify, insert, or delete their data stored in the cloud. In addition, Wang et al.'s scheme [7] also supports public verification, that is, users themselves need not check whether their data are stored intact in the cloud; instead, they can delegate a third party auditor (TPA) to do it. Therefore, the users' burden is reduced, and furthermore, TPA or users can afford evidence when a dispute occurs. Afterwards, Wang et al. [8] pointed out that TPAs should not obtain users' data

to protect users' privacy when they audit data in public auditing schemes and proposed a privacy-preserving public PDP scheme. Meanwhile, in their paper, Wang et al. [8] also considered batch auditing so that even if there are multiple tasks from different users to be carried out simultaneously, TPA just needs to do one auditing task, which improves efficiency.

All the above schemes were proposed in the PKI-based cryptosystem. The main drawback in the PKI-based cryptosystem is the high cost of certificate management, which hinders its application on a large-scale. To reduce the cost of certificate management, identity-based PDP schemes [9, 10] were proposed. However, what is inherent in it is the private key escrow problem.

The certificate-based cryptosystem can not only resolve the private key escrow problem but also reduce the cost of public key certificate management, making it much superior. In 2015, Wang and Li [11] first proposed a certificate-based PDP scheme, but it supports neither public verification nor dynamic operations. In 2020, Wang et al. [12] proposed a lightweight certificate-based PDP scheme, which supports both private and public audits, but it does not support dynamic operations. Both of the above certificate-based PDP schemes rely on the random oracle model. Canetti et al. [13] showed that the random oracle is just an ideal model and that those schemes that are secure under the random oracle model will not be still secure when the random oracle is replaced by some concrete hash functions. Therefore, it is necessary to design certificate-based PDP schemes in the standard model. To the best of our knowledge, so far there have only been two certificate-based PDP schemes proposed in the literature, as mentioned above.

In terms of PDP schemes in the standard model, Zhang et al. [14] proposed an identity-based public PDP scheme, but Shen et al. [15] pointed out that the scheme [14] is insecure. A malicious cloud server can modify or delete users' data arbitrarily, while still being able to produce a proof passing the verification equation as long as it keeps just one data block and its corresponding tag valid. Wang et al. [16] proposed a public and dynamic PDP scheme, but it is based on the PKI technology, which thus requires high cost of certificate management. Zhu et al. [17] proposed a shared data PDP scheme, where a group of users share the data stored in the cloud. It also supports malicious member revocation. Thokchom and Saikia [18] proposed an efficient privacy preserving public dynamic PDP scheme and also extended their scheme to support batch auditing in multi-user and multicloud scenarios, but they did not give security proof of their scheme. Yang et al. [19] proposed a quantum resistant lattice-based PDP scheme, which supports privacy preserving public auditing, dynamic operations, and batch auditing.

The rest of this work is organized as follows. In Section 2, we introduce the bilinear pairing and some complexity assumptions. In Section 3, we introduce the system model, the formal definition, and the security model of certificate-based PDP. In Section 4, we propose a concrete certificate-based PDP scheme in the standard model. In Section 5, we extend our scheme to support dynamic operations by using

the index logic table. In Section 6, we prove the security of the proposed scheme. In Section 7, we evaluate the efficiency of the proposed scheme. We conclude the work in Section 8.

2. Preliminaries

2.1. Bilinear Pairing. Let G_1 and G_2 be two multiplicative cyclic groups of prime order q and g be a generator of G_1 . If the following conditions hold, then it is deemed that the map $e: G_1 \times G_1 \rightarrow G_2$ is a bilinear pairing.

- (1) Bilinearity: for any a and $b \in \mathbb{Z}_q$ and P and $Q \in G_1$, the equation $e(P^a, Q^b) = e(P, Q)^{ab}$ holds
- (2) Nondegeneracy: $e(g, g) \neq 1_{G_2}$
- (3) Computability: for any P and $Q \in G_1$, $e(P, Q)$ can be calculated efficiently

2.2. Complexity Assumption. Square Computational Diffie-Hellman (Squ-CDH) problem: given g and $g^a \in G_1$, where $a \in \mathbb{Z}_q$ is selected randomly, one does not know the value of a and needs to calculate g^{a^2} .

Let A be a probabilistic polynomial time (PPT) algorithm, and the advantage of A in solving the Squ-CDH problem on G_1 is defined to be $\text{Adv}_A^{\text{Squ-CDH}} = \Pr[A(g, g^a) = g^{a^2}, a \in \mathbb{Z}_q]$.

Squ-CDH assumption: for every A , $\text{Adv}_A^{\text{Squ-CDH}}$ is negligible.

3. System Model, Definition, and Security Model of the Certificate-Based PDP

3.1. System Model. There are four entities involved in a certificate-based PDP scheme as illustrated in Figure 1.

CA: it is a trusted third party who initializes the system and issues certificates for users.

Users: they have a large amount of data to be stored in the cloud. When they decide to store their data in the cloud, they divide their data into file blocks and produce a tag for each file block. Then, they store their data along with these tags in the cloud and delete them locally.

TPA: TPA is delegated by users to audit the data in the cloud. It periodically audits the data. When TPA plans to audit the data, it will launch a challenge to the cloud by randomly selecting a subset of the index numbers of file blocks to the cloud. After getting a proof from the cloud server, it checks if the proof can pass an equation in advance. If it can, then it is highly probable that the data stored in the cloud are intact.

Cloud server: it provides cloud storage service. When it receives a challenge from TPA, it generates a proof by using the data blocks selected by TPA and their corresponding tags and sends the proof to TPA.

3.2. Definition of the Certificate-Based PDP. A certificate-based PDP scheme consists of the following seven algorithms.

FIGURE 1: System model.

- (1) Setup: given a security parameter 1^k , it outputs the system's public parameters params and a master private key s . It is run by CA.
- (2) User-Key-Generation: given the system's public parameters params , it outputs a public/private key pair (pk, sk) . It is run by the user.
- (3) Certificate-Gen: given a user's public key pk , the system's public parameters params , and the master private key s , it outputs a certificate cert for the user. It is run by CA.
- (4) Tag-Gen: Given n blocks $m_i \in Z_q^*$ ($i = 1, \dots, n$) of a file F , the user's private key sk , and the system's public parameters params , it outputs a tag tag_i for each file block m_i . It is run by the user. Then, the user stores all m_i and tag_i to the cloud and deletes them locally.
- (5) Proof-Gen: given a subset of the index numbers of file blocks selected by TPA, the corresponding tags tag_i , and the system's public parameters Params , it outputs a proof of data. It is run by CSP and CSP sends the proof to TPA.
- (6) Proof-Verify: given the system's public parameters params and the proof, it outputs TRUE or FALSE. It is run by TPA.
- (7) Update-Op: It is run by CSP.

Inserting: given the system's public parameters Params , the index number i , and a file block m_i , it does the updating and outputs TRUE or FALSE.

Modifying: given the system's public parameters Params , the index number i , and a new file block m'_i , it does the updating and outputs TRUE or FALSE.

Deleting: given the system's public parameters Params and the index number i , it does the updating and outputs TRUE or FALSE.

3.3. Security Model of the Certificate-Based PDP. There are two types of adversaries in the certificate-based cryptosystem [2]. The first type of adversary A_I does not know the master private key, but he can replace anyone's public key, as it models any adversary except CA. On the contrary, the second type of adversary A_{II} knows the master private key, but he cannot replace anyone's public key, as it models the honest-but-curious CA. The honest-but-curious CA means that CA will honestly execute the system protocol, but then he can attack the system. In 2009, Wu et al. [20] introduced the malicious-but-passive CA A_{II} adversary in the certificate-based cryptosystem. This type of A_{II} adversary may be malicious in the system protocol execution. In our security model, this type of adversary A_{II} is considered.

Definition 1 (type I adversary). A certificate-based PDP scheme is secure if for any PPT adversary A_I , the probability that A_I wins the following game is negligible.

Setup. Given a security parameter k , challenge C runs the setup algorithm to generate a master private key s and a common parameter Params . Params are given to A_I and C keeps s private.

Queries. A_I can make the following queries adaptively.

- (1) *User-Creation queries:* A_I supplies an identity ID. If the key pair of identity ID has not been created, C

runs the User-Key-Generation algorithm to create a private/public key pair (SK_{ID}, PK_{ID}) of that user and returns PK_{ID} to A_I ; otherwise, C returns PK_{ID} to A_I directly.

- (2) *User-PrivateKey queries*: A_I supplies an already created identity ID . C returns ID 's private key SK_{ID} to A_I .
- (3) *PublicKey-Replacement queries*: A_I supplies an already created identity ID and a new public/private key pair (PK'_{ID}, SK'_{ID}) . C replaces the current public/private key pair (PK_{ID}, SK_{ID}) with the new pair (PK'_{ID}, SK'_{ID}) .
- (4) *Certificate queries*: A_I supplies an already created identity ID and the corresponding public key PK_{ID} . C runs the Certificate-Gen algorithm to produce ID 's certificate $Cert_{ID}$ and sends it to A_I .
- (5) *Tag-Gen queries*: A_I supplies an already created identity ID and a file block m_i . C runs the Tag-Gen algorithm to generate a tag of (ID, m_i) and sends it back to A_I .
- (6) *Update-Op queries*: A_I supplies an already created identity ID and a request for a dynamic operation. C makes the corresponding dynamic operation.

Challenge. C generates a challenge (ID^*, I^*, W^*) and sends it to A_I , where I^* is a subset of the index numbers of file blocks and W^* is a set of random numbers.

Forge. A_I generates a proof P^* for (ID^*, I^*, W^*) and sends it to C .

A_I wins the game if the following conditions hold.

- (1) $True = ProofVerify(ID^*, P^*)$
- (2) A_I has not made a certificate query on ID^*
- (3) At least one query of Tag-Gen on (ID^*, i^*, m_i^*) does not happen, where $i^* \in I^*$

Definition 2 (type II adversary). A certificate-based PDP scheme is secure if for any PPT adversary A_{II} , the probability that A_{II} wins the following game is negligible.

Setup. Given a security parameter k , adversary A_{II} runs the setup algorithm to generate a master private key s and a common parameter $Params$. $Params$ and s are given to C .

Queries. A_{II} can make User-Creation, User-PrivateKey, Tag-Gen, and Update-Op queries adaptively, and these queries are the same as in Definition 1. Other queries are not needed for A_{II} .

Challenge. Same as in Definition 1.

Forge. A_{II} generates a proof P^* for (ID^*, I^*, W^*) and sends it to C .

A_{II} wins the game if the following conditions hold.

- (1) $True = ProofVerify(ID^*, P^*)$
- (2) A_{II} has not made a User-PrivateKey query on ID^*

- (3) At least one query of Tag-Gen on (ID^*, i^*, m_i^*) does not happen, where $i^* \in I^*$

Note. In order to resist the malicious-but-passive CA A_{II} adversary, it must let adversary A_{II} run the setup algorithm rather than challenger C .

4. A Concrete Certificate-Based PDP Scheme

4.1. Concrete Scheme. $H_u: \{0, 1\}^* \rightarrow \{0, 1\}^{n_u}$ denotes a collision-resistant cryptographic hash function for some $n_u \in \mathbb{Z}_q^*$, which is used to create identities of the desired length.

- (1) **Setup**: given a security parameter 1^k , CA chooses two cyclic groups G_1 and G_2 of prime order q , a random generator g of G_1 , a bilinear map $e: G_1 \times G_1 \rightarrow G_2$, three hash functions $H: \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$, $H_1: \{0, 1\}^* \rightarrow \{0, 1\}^{n_u}$, and $H_2: \{0, 1\}^* \rightarrow G_1$, a pseudorandom function $f: \mathbb{Z}_q^* \times \{1, 2, \dots, n\} \rightarrow \mathbb{Z}_q^*$, and a pseudorandom permutation $\pi: \mathbb{Z}_q^* \times \{1, 2, \dots, n\} \rightarrow \{1, 2, \dots, n\}$. He randomly selects $s, v', v_1, v_2, \dots, v_{n_u} \in \mathbb{Z}_q^*$ and sets $g_1 = g^s$, $u' = g^{v'}$, $u_i = g^{v_i}$ ($i = 1, 2, \dots, n_u$), and $U = (u_i) \in G_1^{n_u}$. CA defines a function $F_u(I, D) = u' \prod_{j=1}^{n_u} u_j^{i_j}$, where $ID = i_1 i_2 \dots i_{n_u}$ is a bit string. The public parameters are $Params = \{G_1, G_2, e, g, g_1, u', U, F_u, H, H_1, H_2, f, \pi\}$, and the master secret key is $msk = (s, v', v_1, v_2, \dots, v_{n_u})$.
- (2) **User-Key-Generation**: the user ID randomly selects x_{ID} and $y_{ID} \in \mathbb{Z}_q^*$ as his secret keys and computes his public key as $UPK_{ID} = (UPK_{ID,1}, UPK_{ID,2}, UPK_{ID,3}) = (g^{x_{ID}}, g^{y_{ID}}, g^{\sigma_{ID}})$, where $\sigma_{ID} = y_{ID} + x_{ID}H(UPK_{ID,1}, UPK_{ID,2}, Params) \bmod q$ is a Schnorr one-time signature. The signature can be generated using the technique of Fiat-Shamir transform without random oracles as described in [21].
- (3) **Certificate-Gen**: given a user's identity ID and his public key UPK_{ID} , CA randomly selects $r_{ID} \in \mathbb{Z}_q^*$ and computes the user's certificate as $Cert_{ID} = (Cert_{ID,1}, Cert_{ID,2}) = (g^{s^2} F_u(id)^{r_{ID}}, g^{r_{ID}})$, where $id = H_1(ID, UPK_{ID,1}, UPK_{ID,2}, UPK_{ID,3})$.
- (4) **Tag-Gen**: let a file F be divided into n blocks $m_i \in \mathbb{Z}_q^*$, $i = 1, 2, \dots, n$. User ID computes $t_{ID,1} = Cert_{ID,1}^{x_{ID}}$, $W_{ID,i} = H_2(ID, UPK_{ID,1}, UPK_{ID,2}, UPK_{ID,3}, t_{ID,1}, i)$, and $t_{ID,i,2} = W_{ID,i}(Cert_{ID,2}^{x_{ID}})^{m_i}$. Finally, the tags are $\sigma = (t_{ID,1}, t_{ID,i,2})$ ($i = 1, 2, \dots, n$).
- (5) **Proof-Gen**: TPA selects a random integer c and two random elements k_1 and $k_2 \in \mathbb{Z}_q^*$. TPA sends the challenge (ID, c, k_1, k_2) to CSP. CSP computes $i_j = \pi_{k_1}(j)$, and $w_j = f_{k_2}(j)$, where $j = 1, 2, \dots, c$. Let $I = \{i_1, i_2, \dots, i_c\}$, and CSP computes $S = \prod_{i \in I} t_{ID,i,2}^{w_i}$ and $\delta = \sum_{i \in I} w_i m_i$ and sends the proof $(t_{ID,1}, S, \delta)$ to TPA.
- (6) **Proof-Verify**: TPA computes $i_j = \pi_{k_1}(j)$, and $w_j = f_{k_2}(j)$, where $j = 1, 2, \dots, c$. Let $I = \{i_1, i_2, \dots, i_c\}$,

and TPA computes $c_1 = H(\text{UPK}_{\text{ID},1}, \text{UPK}_{\text{ID},2}, \text{Params}) \bmod q$, $\text{id} = H_1(\text{ID}, \text{UPK}_{\text{ID},1}, \text{UPK}_{\text{ID},2}, \text{UPK}_{\text{ID},3})$, and $W_{\text{ID},i} = H_2(\text{ID}, \text{UPK}_{\text{ID},1}, \text{UPK}_{\text{ID},2}, \text{UPK}_{\text{ID},3}, t_{\text{ID},1}, i)$. TPA verifies whether $e(\text{UPK}_{\text{ID},3}, g) = e(\text{UPK}_{\text{ID},2}, g)e(\text{UPK}_{\text{ID},1}, g)^{c_1}$ and $e(S, g) = e(\prod_{i \in I} W_{\text{ID},i}^{w_i}, g)e(\text{UPK}_{\text{ID},1}, \text{UPK}_{\text{ID},2}, 1)^\delta e(F_u(\text{id}), t_{\text{ID},1})^\delta$ hold. If they do, he accepts it.

4.2. Correctness.

$$\begin{aligned}
 e(S, g) &= e\left(\prod_{i \in I} t_{\text{ID},i,2}^{w_i}, g\right) = e\left(\prod_{i \in I} W_{\text{ID},i}^{w_i} \text{Cert}_{\text{ID},1}^{x_{\text{ID}}^2 m_i w_i}, g\right) \\
 &= e\left(\prod_{i \in I} W_{\text{ID},i}^{w_i}, g\right) e\left(\text{Cert}_{\text{ID},1}^{x_{\text{ID}}^2 \delta}, g\right) \\
 &= e\left(\prod_{i \in I} W_{\text{ID},i}^{w_i}, g\right) e\left(g^{x_{\text{ID}}^2 \delta} F_u(\text{id})^{r_{\text{ID}} x_{\text{ID}}^2 \delta}, g\right) \\
 &= e\left(\prod_{i \in I} W_{\text{ID},i}^{w_i}, g\right) e(\text{UPK}_{\text{ID},1}, \text{UPK}_{\text{ID},1})^\delta e(F_u(\text{id})^{r_{\text{ID}} x_{\text{ID}}^2 \delta}, g) \\
 &= e\left(\prod_{i \in I} W_{\text{ID},i}^{w_i}, g\right) e(\text{UPK}_{\text{ID},1}, \text{UPK}_{\text{ID},1})^\delta e(F_u(\text{id}), t_{\text{ID},1})^\delta.
 \end{aligned} \tag{1}$$

5. Data Dynamic

5.1. Index Logic Table. Wang et al. [5] introduced the index logic table (ILT) to support dynamic operations in PDP schemes, as ILT can overcome the delete-insert attack [5] in the map version table [22] structure. An ILT includes two columns: the index number (the first column) and the logic number (the second column). In addition, one additional column (the third column) is used to explain the ILT and not appear in the ILT actually. Let “iin” denote the first column, “lIn” denote the second one, “blocks” denote the third one, “inEF” denote the last line of column 1, and “InEF” denote the last line of column 2.

The ILT is illustrated in Table 1. Please note that the last row of blocks does not include any file block in Table 1.

There are four operations in ILT, namely, initializing, modifying, inserting, and deleting. We assume a file F is divided into n blocks. The initializing operation is described in algorithm 1(a) of Table 2, with an example of $n=6$ given in Table 1(a). The modifying operation is described in algorithm 1(b) of Table 2, with an example of modifying file block m_4 to m_4' at $i=4$ given in Table 1(b). The inserting operation is described in algorithm 1(c) of Table 2, with an example of inserting a file block m'' after $i=4$ given in Table 1(c). The deleting operation is described in algorithm 1(d) of Table 2, with an example of deleting $i=2$ given in Table 1(d).

5.2. Dynamic Operations. To make our scheme support dynamic operations, we must replace $W_{\text{ID},i} = H_2(\text{ID}, \text{UPK}_{\text{ID},1}, \text{UPK}_{\text{ID},2}, \text{UPK}_{\text{ID},3}, t_{\text{ID},1}, i)$ with $W_{\text{ID},i} = H_2(\text{ID}, \text{UPK}_{\text{ID},1}, \text{UPK}_{\text{ID},2}, \text{UPK}_{\text{ID},3}, t_{\text{ID},1}, \text{lIn}(i))$ in

TABLE 1: Index logic table.

iin	lIn	blocks
<i>(a) Initial</i>		
1	1	M1
2	2	M2
3	3	M3
4	4	M4
5	5	M5
6	6	M6
7	7	null
<i>(b) Modify at 4</i>		
1	1	M1
2	2	M2
3	3	M3
4	7	M4'
5	5	M5
6	6	M6
7	8	null
<i>(c) Insert after 4</i>		
1	1	M1
2	2	M2
3	3	M3
4	7	M4'
5	8	M''
6	5	M5
7	6	M6
8	9	Null
<i>(d) Delete at 2</i>		
1	1	M1
2	3	M3
3	7	M4'
4	8	M''
5	5	M5
6	6	M6
7	9	null

the Tag-Gen and the Proof-Verify algorithms; in other words, we must replace i with $\text{lIn}(i)$. The ILT must be stored in users and TPA locally.

- (1) Data modifying: when a user wants to modify a file block m_i to m_i' at $\text{iin} = i$, he runs algorithm 1(b) first. Then, he computes tag' for m_i' . He uploads m_i' , tag' , iin , and ILT to the cloud and deletes m_i and tag locally. CSP makes the corresponding modification.
- (2) Data inserting: when a user wants to insert a file block m'' after $\text{iin} = i$, he runs algorithm 1(c) first. Then, he computes tag'' for m'' . He uploads m'' , tag'' , iin , and ILT to the cloud and deletes m'' and tag'' locally. CSP makes the corresponding insertion.
- (3) Data deleting: when a user wants to delete a file block m_i at $\text{iin} = i$, he runs algorithm 1(d) first. Then, he uploads iin and ILT to the cloud. CSP makes the corresponding deletion.

6. Unforgeability

Theorem 1 (type I unforgeability). *In the standard model, if a PPT attacker A_I has a nonnegligible advantage ε in winning the game of Definition 1, running for time t and performing at*

TABLE 2: Algorithm 1.

Algorithm 1			
(a) Initial	(b) Modify	(c) Insert	(d) Delete
For (j = 1; j ≤ n; j++)	lln[i] = lnEF;	lnEF = lnEF + 1;	For (j = i; j < lnEF; j++)
{ iin[j] = j;	blocks[i] = mi';	inEF = inEF + 1;	{ lln[j] = lln[j+1];
lln[j] = j;	lnEF = lnEF + 1;	blocks[inEF] = null;	blocks[j] = blocks[j+1];
blocks[j] = mj; }	lln[inEF] = lnEF;	lln[inEF] = lnEF;	inEF = inEF - 1;
inEF = j;		iin[inEF] = inEF;	
lnEF = j;		For (j = inEF - 1; j > i; j--)	
blocks[lnEF] = null;		{ lln[j] = lln[j-1]; }	
iin[inEF] = inEF;		blocks[j] = blocks[j-1];	
lln[inEF] = lnEF;		}	
		lln[i] = lnEF - 1;	
		blocks[i] = m'';	

most $q_{\text{UCreation}}$ user creation queries, q_{Cert} certificate queries, and q_{tag} Tag-Gen queries, then there is an algorithm C that solves the Squ-CDH problem with an advantage $\text{Adv}_C^{\text{Squ-CDH}} \geq (\varepsilon/4)(q_{\text{Cert}} + q_{\text{tag}})(n_u + 1)$ for time $t + O(q_{\text{Cert}} \cdot n_u \cdot t_m + (q_{\text{UCreation}} + q_{\text{Cert}} + q_{\text{tag}})t_e)$, where t_m and t_e denote the time for a multiplication and an exponentiation on G_1 , respectively.

Proof. Suppose C is given $(g, B = g^b) \in G_1$ for randomly chosen $b \in Z_q$. C does not know the value of b and is asked to compute g^{b^2} . To utilize adversary A_I , challenger C will simulate all the oracles defined in Definition 1. C maintains a table $tb = \{\text{ID}, \text{Cert}_{\text{ID}}, x_{\text{ID}}, y_{\text{ID}}, \text{UPK}_{\text{ID}}, \text{tag} = 0\}$, which is initially empty. $\text{tag} = 0$ means that the public key is not replaced.

Setup Let $l_u = 2(q_{\text{Cert}} + q_{\text{tag}})$. C randomly chooses the following elements:

- (1) An integer k_u ($0 \leq k_u \leq n_u$). We assume that $l_u(n_u + 1) < q$ for the given values of q_{Cert} , q_{tag} , and n_u
- (2) An integer $x'_u \in Z_{l_u}$, and a vector $X_u = (x_{u,i})_{n_u}$ ($x_{u,i} \in Z_{l_u}$)
- (3) An integer $y'_u \in Z_q$, and a vector $Y_u = (y_{u,i})_{n_u}$ ($y_{u,i} \in Z_q$)

For convenience, we define the following functions:

$$J_u(i \text{ d}) = x'_u - l_u k_u + \sum_{j=1}^{n_u} i_j x_{u,j},$$

$$K_u(i \text{ d}) = y'_u + \sum_{j=1}^{n_u} i_j y_{u,j},$$
(2)

where $id = i_1 i_2 \dots i_{n_u}$ is a bit string.

Then, C randomly chooses two cyclic groups G_1 and G_2 of prime order q , a random generator g of G_1 , a bilinear map $e: G_1 \times G_1 \rightarrow G_2$, and three hash functions $H: \{0, 1\}^* \rightarrow Z_q^*$, $H_1: \{0, 1\}^* \rightarrow \{0, 1\}^{n_u}$, and $H_2: \{0, 1\}^* \rightarrow G_1$. He sets $g_1 = B$, $u' = B^{x'_u - l_u k_u} g^{y'_u}$, $u_i = B^{x_{u,i}} g^{y_{u,i}}$ ($0 \leq i \leq n_u$), and $U = (u_i) \in G_1^{n_u}$. This assignment means $F_u(id) = B^{J_u(id)} g^{K_u(id)}$ and that the master secret key $s = b$, which is not known to C . C chooses a

pseudorandom function $f: Z_q^* \times \{1, 2, \dots, n\} \rightarrow Z_q^*$ and a pseudorandom permutation $\pi: Z_q^* \times \{1, 2, \dots, n\} \rightarrow \{1, 2, \dots, n\}$. C outputs the public parameters $\text{Params} = \{G_1, G_2, e, g, g_1, u', U, F_u, H, H_1, H_2, f, \pi\}$.

Queries. A_I can adaptively make a polynomial bounded number of queries as follows.

- (1) User-Creation queries: A_I supplies an identity ID. C first checks the table tb to see whether it contains the item or not. If it does, C returns ID's public key UPK_{ID} to A_I ; otherwise, C executes the User-Key-Generation algorithm to produce ID's private/public key pair $(x_{\text{ID}}, y_{\text{ID}}, \text{UPK}_{\text{ID}})$. C puts $(\text{ID}, -, x_{\text{ID}}, y_{\text{ID}}, \text{UPK}_{\text{ID}}, \text{tag} = 0)$ into the table tb and returns UPK_{ID} to A_I .
 - (2) User-PrivateKey queries: A_I supplies an already created identity ID. C searches the table tb to find out the private keys x_{ID} and y_{ID} and returns them to A_I .
 - (3) PublicKey-Replacement queries: A_I supplies an already created identity ID and a new public/private key pair $(\text{UPK}'_{\text{ID}}, x'_{\text{ID}}, y'_{\text{ID}})$. C replaces the current public/private key pair $(\text{UPK}_{\text{ID}}, x_{\text{ID}}, y_{\text{ID}})$ with the new key pair $(\text{UPK}'_{\text{ID}}, x'_{\text{ID}}, y'_{\text{ID}})$ in the table tb and sets $\text{tag} = 1$.
 - (4) Certificate queries: A_I supplies an already created identity ID and ID's public key UPK_{ID} . C first checks table tb to see whether the certificate Cert_{ID} is produced or not. If it does, C returns the certificate Cert_{ID} to A_I ; otherwise, C computes $id = H_1(\text{ID}, \text{UPK}_{\text{ID},1}, \text{UPK}_{\text{ID},2}, \text{UPK}_{\text{ID},3})$ and produces the certificate as follows (C does not know the master private key). Then, C returns the certificate Cert_{ID} to A_I and updates “-” with the Cert_{ID} in table tb .
 - (a) $J_u(id) \neq 0 \text{ mod } q$. C randomly selects $r_{\text{ID}} \in Z_q^*$ and computes the user's certificate as $\text{Cert}_{\text{ID}} = (\text{Cert}_{\text{ID},1}, \text{Cert}_{\text{ID},2}) = (B^{-K_u(id)/J_u(id)} \cdot F_u(id)^{r_{\text{ID}}}, B^{-1/J_u(id)} g^{r_{\text{ID}}}) = (g^{b^2} \cdot F_u(id)^{r_{\text{ID}}}, g^{r_{\text{ID}}})$, where $\overline{r_{\text{ID}}} = r_{\text{ID}} - b/J_u(id)$.
 - (b) $J_u(id) = 0 \text{ mod } q$. C aborts.
- To make the analysis of the simulation easier, we will force C to abort whenever $J_u(id) = 0 \text{ mod } l_u$.

$(J_u(\text{id}) \neq 0 \bmod l_u \text{ implies } J_u(\text{id}) \neq 0 \bmod q, \text{ given the assumption } l_u(n_u + 1) < q).$

- (5) Tag-Gen queries: A_I supplies an already created identity ID and a file block m_i . C computes $\text{id} = H_1(\text{ID}, \text{UPK}_{\text{ID},1}, \text{UPK}_{\text{ID},2}, \text{UPK}_{\text{ID},3})$.
 - (a) $J_u(\text{id}) \neq 0 \bmod q$. C produces the tag as normal because C can get the certificate and private key of ID .
 - (b) $J_u(\text{id}) = 0 \bmod q$. C aborts.
- (6) Update-Op queries: A_I supplies an already created identity ID . For inserting operation, A_I also supplies the “iin” number in the ILT table and a new file block m_i ; for modifying operation, A_I also supplies the “iin” number in the ILT table, the old file block m_i and the new file block m_i' ; for deleting operation, A_I also supplies the “iin” number in the ILT table. CSP makes the corresponding dynamic operations and returns ILT to the user.

Challenge: C generates a challenge $(\text{ID}^*, c, k_1, k_2)$ to A_I . A_I computes $i_j^* = \pi_{k_1}(j)$ and $w_j^* = f_{k_2}(j)$, where $j = 1, 2, \dots, c$. Let $I^* = \{i_1^*, i_2^*, \dots, i_c^*\}$.

Forge: A_I generates a forged proof $(t_{\text{ID}^*,1}, S^*, \delta^*)$ for $(\text{ID}^*, I^*, (w_i^*)_{i \in I^*})$. According to Definition 1, at least one query of Tag-Gen on $(\text{ID}^*, i^*, m_{i^*})$, where $i^* \in I^*$, must not happen, that is, A_I has generated a forged tag $\sigma^* = (t_{\text{ID}^*,1}, t_{\text{ID}^*,i^*,2})$, $i^* \in I^*$, for file block m_{i^*} . C computes $\text{id}^* = H_1(\text{ID}^*, \text{UPK}_{\text{ID}^*,1}, \text{UPK}_{\text{ID}^*,2}, \text{UPK}_{\text{ID}^*,3})$. If $J_u(\text{id}^*) \neq 0 \bmod q$, then C aborts; otherwise, we have $F_u(\text{id}^*) = g^{K_u(\text{id}^*)}$. If $(t_{\text{ID}^*,1}, S^*, \delta^*)$ can pass the Proof-Verify algorithm and A_I does not violate the restrictions of Definition 1, then C retrieves the private key x_{ID^*} of ID^* and computes

$W_{\text{ID}^*,i^*} = H_2(\text{ID}^*, \text{UPK}_{\text{ID}^*,1}, \text{UPK}_{\text{ID}^*,2}, \text{UPK}_{\text{ID}^*,3}, t_{\text{ID}^*,1}, i^*)$ and $t_{\text{ID}^*,i^*,2}/W_{i^*}^{(t_1^{K_u(\text{id}^*)})m_{i^*}} = W_{\text{ID}^*,i^*}^{(t_1^{K_u(\text{id}^*)})m_{i^*}} (\text{Cert}_{\text{ID}^*,1}^{x_{\text{ID}^*}})^{m_{i^*}} / W_{\text{ID}^*,i^*}^{\text{Cert}_{\text{ID}^*,1}^{x_{\text{ID}^*}}}, 2^{x_{\text{ID}^*}^2 K_u(\text{id}^*) m_{i^*}} = g^{b^2 \cdot x_{\text{ID}^*}^2 m_{i^*}} \cdot F_u(\text{id}^*)^{x_{\text{ID}^*}^2} \cdot r_{\text{ID}^*} m_{i^*} / (g^{r_{\text{ID}^*}})^{x_{\text{ID}^*}^2 K_u(\text{id}^*) m_{i^*}} = g^{b^2 \cdot x_{\text{ID}^*}^2 m_{i^*}} \cdot g^{K_u(\text{id}^*) x_{\text{ID}^*}^2 r_{\text{ID}^*} m_{i^*}} / (g^{r_{\text{ID}^*}})^{x_{\text{ID}^*}^2 K_u(\text{id}^*) m_{i^*}} = g^{b^2 \cdot x_{\text{ID}^*}^2 m_{i^*}}$. Because C knows x_{ID^*} and m_{i^*} , he can compute g^{b^2} from $g^{b^2 \cdot x_{\text{ID}^*}^2 m_{i^*}}$. C solves the Squ-CDH problem.

Now, we assess the probability of success. If the simulation is not aborted, the following conditions must be met.

- (1) In all certificate and signature queries, $J_u(\text{id}) \neq 0 \bmod q$
- (2) In the forgery stage, $J_u(\text{id}^*) = 0 \bmod q$

Let $\text{ID}_1, \text{ID}_2, \dots, \text{ID}_{q_I}$ be the identities appearing in these queries, but they do not involve any of the challenge identities. Clearly, we have $q_I \leq q_{\text{Cert}} + q_{\text{tag}}$. Define the events A_1^*, A_2^* , and A_i^* as $A_i^*: J_u(\text{id}^*) = 0 \bmod q$ and $A_i: J_u(i \cdot d) \neq 0 \bmod q, i = 1, 2, \dots, q_I$.

Therefore, the probability of C not aborting the simulation is $\Pr[\text{abort}] \geq \Pr[\bigwedge_{i=1}^{q_I} A_i \wedge A_1^*]$. We have $\Pr[A_1^*] = 1/l_u(n_u + 1)$ and $\Pr[\bigwedge_{i=1}^{q_I} A_i \wedge A_1^*] \geq 1/l_u(n_u + 1)(1$

$-(q_{\text{Cert}} + q_{\text{tag}})/l_u)$. Therefore, $\text{Adv}_C^{\text{Squ-CDH}} \geq \varepsilon/4(q_{\text{Cert}} + q_{\text{tag}})(n_u + 1)$.

The time complexity of challenger C depends on the exponentiations, multiplications, and pairing operations needed in all above queries. The user creation queries need $O(1)$ exponentiations. The certificate queries need $O(n_u)$ multiplications and $O(1)$ exponentiations. The Tag-Gen queries need $O(1)$ exponentiations. Therefore, the time complexity of C is $t + O(q_{\text{Cert}} \cdot n_u \cdot t_m + (q_{\text{UCreation}} + q_{\text{Cert}} + q_{\text{tag}})t_e)$. $\square$

Theorem 2 (type II unforgeability). *In the standard model, if a PPT attacker A_{II} has a nonnegligible advantage ε in winning the game of Definition 2 running for time t and performing at most $q_{\text{UCreation}}$ user creation queries, q_p private key queries, and q_{tag} Tag-Gen queries, then there is an algorithm C that solves the Squ-CDH problem with an advantage $\text{Adv}_C^{\text{Squ-CDH}} \geq \varepsilon/q_{\text{UCreation}}(1 - 1/q_p)(1 - 1/q_{\text{tag}})$ for time $t + O((q_{\text{UCreation}} + q_{\text{tag}})t_e)$, where t_e denotes the time for an exponentiation on G_1 .*

Proof. Let us suppose C is given $(g, B = g^b) \in G_1$ for randomly chosen $b \in \mathbb{Z}_q$. C does not know the value of b and is asked to compute g^{b^2} . To utilize the adversary A_{II} , challenger C will simulate all the oracles defined in Definition 2. C maintains a table $tb = \{\text{ID}, x_{\text{ID}}, y_{\text{ID}}, \text{UPK}_{\text{ID}}\}$, which is initially empty.

Setup. A_{II} randomly chooses two cyclic groups G_1 and G_2 of prime order q , a random generator h of G_1 , a bilinear map $e: G_1 \times G_1 \rightarrow G_2$, and three hash functions $H: \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$, $H_1: \{0, 1\}^* \rightarrow \{0, 1\}^{n_u}$, and $H_2: \{0, 1\}^* \rightarrow G_1$, and $\alpha, y_u', y_{u,1}, y_{u,2}, \dots, y_{u,n_u} \in \mathbb{Z}_q^*$. He/she sets $g_1 = h^\alpha$, $u' = h^{y_u'}$, $u_i = h^{y_{u,i}}$ ($0 \leq i \leq n_u$), and $U = (u_i) \in G_1^{n_u}$, where $\text{id} = i_1 i_2 \dots i_{n_u}$ is a bit string. Let $K_u(\text{id}) = y_u' + \sum_{j=1}^{n_u} (i_j \cdot y_{u,j})$, and this assignment means $F_u(\text{id}) = h^{K_u(\text{id})}$. A_{II} chooses a pseudorandom function $f: \mathbb{Z}_q^* \times \{1, 2, \dots, n\} \rightarrow \mathbb{Z}_q^*$ and a pseudorandom permutation $\pi: \mathbb{Z}_q^* \times \{1, 2, \dots, n\} \rightarrow \{1, 2, \dots, n\}$. The master secret key is $(\alpha, y_u', y_{u,1}, y_{u,2}, \dots, y_{u,n_u})$. A_{II} outputs the public parameters

$\text{Params} = \{G_1, G_2, e, h, g_1, u', U, F_u, H, H_1, H_2, f, \pi\}$ and $(\alpha, y_u', y_{u,1}, y_{u,2}, \dots, y_{u,n_u})$ to C .

As h is a generator of G_1 , there must be a $b' \in \mathbb{Z}_q^*$ to make the equation $B = h^{b'}$ hold. Knowing $(h, B = h^{b'}) \in G_1$, now C 's goal becomes computing $h^{(b')^2}$. C randomly selects an index π from $\{1, \dots, q_{\text{UCreation}}\}$.

Queries. A_{II} can make the following queries adaptively.

- (1) User-Creation queries: A_{II} supplies an identity ID_i . C first checks table tb to see whether it contains the item or not. If it does, C returns ID_i 's public key UPK_{ID_i} to A_{II} . Otherwise, C produces the public/private key pair as follows and returns UPK_{ID_i} to A_{II} .
 - (a) $\text{ID}_i \neq \text{ID}_\pi$. C executes the User-Key-Generation algorithm as normal to produce ID_i 's private/

- public key pair $(x_{ID_i}, y_{ID_i}, \text{UPK}_{ID_i})$. C puts $(ID_i, x_{ID_i}, y_{ID_i}, \text{UPK}_{ID_i})$ into table tb .
- (b) $ID_i = ID_\pi$. C randomly selects $y_{ID_i} \in Z_q^*$ and computes ID_i 's public key as $\text{UPK}_{ID_i} = (B, h^{y_{ID_i}}, h^{y_{ID_i} \cdot B^c})$, where $c = H(K, \text{UPK}_{ID_{i,1}}, \text{UPK}_{ID_{i,2}}, \text{Params})$. C puts $(ID_i, -, y_{ID_i}, \text{UPK}_{ID_i})$ into table tb (the implicitly defined private key $x_{ID_i} = b'$, which is not known to C).
- (2) Private Key queries: A_{II} supplies an already created identity ID_i .
- (a) $ID_i \neq ID_\pi$. C searches table tb to find out the private keys x_{ID_i} and y_{ID_i} and returns them to A_{II} .
- (b) $ID_i = ID_\pi$. C aborts.
- (3) Tag-Gen queries: A_{II} supplies an already created identity ID_i and a file block m_i . C computes $\text{id} = H_1(ID, \text{UPK}_{ID,1}, \text{UPK}_{ID,2}, \text{UPK}_{ID,3})$.
- (a) $ID_i \neq ID_\pi$. C produces the tag as normal because C can get the certificate and private key of ID_i .
- (b) $ID_i = ID_\pi$. C aborts.
- (4) Update-Op queries: same as in Theorem 1.

Challenge: C generates a challenge (ID^*, c, k_1, k_2) to A_{II} . A_{II} computes $i_j^* = \pi_{k_1}(j)$ and $w_j^* = f_{k_2}(j)$, where $j = 1, 2, \dots, c$. Let $I^* = \{i_1^*, i_2^*, \dots, i_c^*\}$.

Forge: A_{II} generates a forged proof $(t_{ID^*,1}, S^*, \delta^*)$ for $(ID^*, I^*, (w_i^*)_{i \in I^*})$. According to Definition 2, at least one query of Tag-Gen on (ID^*, i^*, m_{i^*}) , where $i^* \in I^*$, must not happen, that is, A_{II} has generated a forged tag $\sigma^* = (t_{ID^*,1}, t_{ID^*,i^*,2})$, $i^* \in I^*$ for file block m_{i^*} . If $ID^* \neq ID_\pi$, C aborts; otherwise, if $(t_{ID^*,1}, S^*, \delta^*)$ can pass the Proof-Verify algorithm and A_{II} does not violate the restrictions of Definition 2, then C can compute $\text{id}^* = H_1(ID^*, \text{UPK}_{ID^*,1}, \text{UPK}_{ID^*,2}, \text{UPK}_{ID^*,3})$, $K_u(\text{id}^*) = y_u' + \sum_{j=1}^{n_u} (i_j^* \cdot y_{u,j})$ ($\text{id}^* = i_1^* i_2^* \dots i_{n_u}^*$), $W_{ID^*,i^*} = H_2(ID^*, \text{UPK}_{ID^*,1}, \text{UPK}_{ID^*,2}, \text{UPK}_{ID^*,3}, t_{ID^*,1}, i^*)$, and $t_{ID^*,i^*,2}/W_{ID^*,i^*}(t_{ID^*,1})^{K_u(\text{id}^*)m_{i^*}} = W_{ID^*,i^*}(\text{Cert}_{ID^*,1}^{(b')^2})^{m_{i^*}}/W_{ID^*,i^*} \text{Cert}_{ID^*,2}^{(b')^2 K_u(\text{id}^*)m_{i^*}} = h^{\alpha^2 \cdot (b')^2 m_{i^*} \cdot F_u(\text{id}^*) (b')^2 r_{ID^*} m_{i^*}} / (h^{r_{ID^*}} (b')^2 K_u(i^*))^{m_{i^*}} = h^{\alpha^2 \cdot (b')^2 m_{i^*} \cdot h^{K_u(\text{id}^*) (b')^2 r_{ID^*} m_{i^*}}} / (h^{r_{ID^*}} (b')^2 K_u(\text{id}^*) m_{i^*}) = h^{\alpha^2 \cdot (b')^2 m_{i^*}}$, from which C can compute $h^{(b')^2}$.

Now, we assess the probability of success. If the simulation is not aborted, the following conditions must be met.

- (1) In all private key queries, $ID_i \neq ID_\pi$
- (2) In all Tag-Gen queries, $ID_i \neq ID_\pi$
- (3) In the forgery stage, $ID^* = ID_\pi$

At a probability $1/q_{\text{UCreation}}$, the adversary will output an identity ID_π with index π in the forgery stage. The probability of $ID_i \neq ID_\pi$ in the private key queries is $(1 - 1/q_p)$. The probability of $ID_i \neq ID_\pi$ in the Tag-Gen queries is $(1 - 1/q_{\text{tag}})$. Thus, we can conclude that the advantage of C is $\text{Adv}_C^{\text{Squ-CBHF}} \geq \varepsilon/q_{\text{UCreation}} (1 - 1/q_p)(1 - 1/q_{\text{tag}})$.

The time complexity of challenger C depends on the exponentiations, multiplications, and pairings' operations needed in all above queries. The user creation queries need $O(1)$ exponentiations. The Tag-Gen queries need $O(1)$ exponentiations. Therefore, the time complexity of C is $t + O((q_{\text{UCreation}} + q_{\text{tag}})t_e)$.

Note. The goal of C is to compute g^{b^2} . However, the initialization of system is run by A_{II} , and therefore, there is a high probability that A_{II} will not choose g as one of the system parameters. In this scenario, we change the goal of C from computing g^{b^2} to computing $h^{(b')^2}$. $\square$

7. Analysis of Efficiency

We analyse the efficiency of our scheme in terms of computational time, communication overhead, and storage cost. Let h , $e1$, $e2$, and p denote a map-to-point hash computation, an exponentiation computation on G_1 , an exponentiation computation on G_2 , and a bilinear pairing computation, respectively. We compare our scheme with those proposed also in the standard model, which include schemes [14, 16, 18]. To show a more direct comparison with these schemes, we also adopt the experiment results of the scheme [23]. The computation time of each operation is shown in Table 3 and that of each scheme is shown in Table 4. By combining Tables 3 and 4, we get Table 5. Since the scheme [14, 16] also divided a file block into s sectors, for the sake of fairness, we set $s=1$ in Table 5. From Table 5, we can get Figure 2 for CSP's running time and Figure 3 for TPA's running time. From Table 5, we can see that the scheme [18] is the most efficient one in terms of the Tag-Gen algorithm. From Figure 2, we can see that our scheme is the most efficient one in terms of CSP's running time (the Proof-Gen algorithm). From Figure 3, we can see that scheme [14] is the most efficient one in terms of TPA's running time (the Proof-Verify algorithm). The communication overhead of each scheme is shown in Table 6. Based on the experiment results of the scheme [23], the size of q is 512 bits, and if the technique of point compression is used, the size of an element in $G1$ or $G2$ is 512 bits. An integer is represented by 64 bits. From this, we get Table 7. From Table 7 and by taking $s=1$, we obtain Figure 4. From Table 7, we can see that our scheme is the most efficient one in terms of the Tag-Gen algorithm, and in summary, our scheme is the most efficient one in terms of both storage and communication of tags. From Figure 4, we can see that our scheme is the most efficient one in the communication of Proof-Gen.

We also compare our scheme with those certificate-based PDP schemes, which include schemes [11, 12]. The computation time of each scheme is shown in Table 8. By combining Tables 3 and 8, we get Table 9. From Table 9, we can get Figure 5 for CSP's running time and Figure 6 for TPA's running time. From Table 9, we can see that the scheme [11] is the most efficient one in terms of the Tag-Gen algorithm. From Figure 5, we can see that all schemes are the same in terms of the CSP's running time (the Proof-Gen algorithm). From Figure 6, we can see that scheme [11] is the most efficient one in terms of TPA's running time (the

TABLE 3: Computational time (ms).

H	$E1$	$E2$	p
18.673	7.736	0.160	15.500

TABLE 4: Computational time.

Schemes	Tag-Gen	Proof-Gen	Proof-Verify
[14]	$(2n + s) e1$	$3ce1$	$3p + (s + 1) e1$
[16]	$(s + 3) ne1 + 6e1$	$3p + 2(c + 1) e1 + e1$	$6p + (c + s + 4) e1$
[18]	$ne1$	$c(p + 3e1)$	$c(p + e2)$
Ours	$(n + 1) e1 + nh$	$ce1$	$7p + ce1 + 3e2 + h$

n denotes the total number of file blocks, s means that a file block is divided into s sectors, and c denotes the total number of challenge blocks.

TABLE 5: Computational time.

Schemes	Tag-Gen	Proof-Gen	Proof-Verify
[14]	$15.472n + 7.736$	$23.208c$	61.972
[16]	$30.944n + 46.416$	$15.472c + 69.708$	$7.736c + 131.68$
[18]	$7.736n$	$38.708c$	$15.66c$
Ours	$26.409n + 7.736$	$7.736c$	$7.736c + 127.653$

FIGURE 2: Comparisons of CSP's running time.

Proof-Verify algorithm). The communication overhead of each scheme is shown in Table 10. By using the concrete parameters, we get Table 11. From Table 11, we can see that the scheme [11] is the most efficient one in terms of the Tag-Gen algorithm, and thus, the scheme [11] is the most efficient one in terms of both storage and communication of tags. Also from Table 11, we can see that all schemes are almost the same in the communication of Proof-Gen. Therefore, it can be concluded that our scheme is an efficient scheme.

In our system, the storage cost of ILT is linear to the size of the outsourced data and the file block is directly used as an exponent in the tag generation algorithm. Now, let us analyse how the file block size affects the efficiency. If the file

FIGURE 3: Comparisons of TPA's running time.

TABLE 6: Communication overhead (bit).

Schemes	Tag-Gen	Proof-Gen
[14]	$2n G1 $	$ N c + q + 2 G1 + s q $
[16]	$2n G1 + n q + 4 G1 $	$(N + q)c + (s + 2) q + (c + 4) G1 $
[18]	$n(G1 + q)$	$(N + q)c + G2 $
Ours	$(n + 1) G1 $	$ N + 3 q + G1 $

TABLE 7: Communication overhead (bit).

Schemes	Tag-Gen	Proof-Gen
[14]	$1024n$	$64c + 160s + 2284$
[16]	$1184n + 2048$	$736c + 160s + 2368$
[18]	$672n$	$224c + 512$
Ours	$512n + 512$	1056

FIGURE 4: Comparisons of CSP's communication overhead.

TABLE 8: Computational time.

Schemes	Tag-Gen	Proof-Gen	Proof-Verify
[11]	$(n+1) e1$	$ce1$	$2p + (c+3) e1$
[12]	$(3n+s) e1 + (n+2)h$	$ce1$	$2p + (c+4) e1 + h$
Ours	$(n+1) e1 + nh$	$ce1$	$7p + ce1 + 3e2 + h$

TABLE 9: Computational time.

Schemes	Tag-Gen	Proof-Gen	Proof-Verify
[11]	$7.736n + 7.736$	$7.736c$	$7.736c + 54.208$
[12]	$41.881n + 7.736s + 37.346$	$7.736c$	$7.736c + 80.617$
Ours	$26.409n + 7.736$	$7.736c$	$7.736c + 127.653$

FIGURE 5: Comparisons of CSP's running time.

FIGURE 6: Comparisons of TPA's running time.

TABLE 10: Communication overhead (bit).

Schemes	Tag-Gen	Proof-Gen
[11]	$n G1 $	$ N + 3 q + G1 $
[12]	$(n + s + 2) G1 + q $	$ N + (s + 2) q + G1 $
Ours	$(n + 1) G1 $	$ N + 3 q + G1 $

TABLE 11: Communication overhead (bit).

Schemes	Tag-Gen	Proof-Gen
[11]	$512n$	1056
[12]	$512(n + s) + 672$	$160s + 896$
Ours	$512n + 512$	1056

block size is small, there will be a large number of blocks, and then, the storage cost of ILT will be high. Now, let us take a look at the calculation efficiency of tag. It is related to the number of tags and the computation cost of each tag. Since the block is directly used as an exponent in the tag generation algorithm and $m_i \in Z_q^*$, the size of block m_i has no effect on the computation cost of each tag, that is, no matter what the block size is, the computation cost of each tag is one exponent computation. However, the smaller the file block size is, the more blocks there will be and the more tags there will be, leading to a higher computation cost of tags. Therefore, to reduce the storage cost of ILT and to improve the computation efficiency of tags, the file block size should be as large as possible. But, when the user is performing dynamic operations, the file block size should be as small as possible because in this case, the communication cost of each modification or insertion will be lower. Since how many blocks the file is divided into is decided by the users themselves, the users must make a tradeoff among storage efficiency, computation cost of tags, and efficiency of dynamic operations. The system can give a reference value. Take a 1 GB file for example. If a user divides it into blocks with a size of 16 KB each, then there will be a total of 62500 file blocks. Each entry of ILT contains just two integers iis and lln (total 8 bytes). Therefore, the total storage cost of ILT is 512 KB ($<0.05\%$ of the file size). In dynamic operations, the communication cost of inserting or modifying one block is the communication cost of regular tags plus the transmission cost of 16 KB file blocks. The users can refer to these values when they divide the file.

In addition, the purpose of Ateniese et al. [1] proposing PDP is to realize probabilistic proofs of possession by sampling a small amount of data from the server. They concluded that if there are 1% of the file blocks deleted by the cloud, then the user or TPA just needs to randomly choose 460 blocks as a challenge to the cloud to detect the cloud's misbehaviour with a probability of at least 99%. If there are 10000 file blocks, we just need to challenge $460/10000 = 4.6\%$ of the blocks. In this view, it would be better if the number of file blocks is more than 10000.

8. Conclusions

To exploit the advantages of certificate-based cryptosystems, a certificate-based PDP scheme is proposed. Based on the

index logic table, our scheme can be extended to support dynamic operations easily. We prove that our scheme is secure in the standard model under the Squ-CDH assumption. Efficiency evaluation shows that our scheme is efficient.

Data Availability

Previously reported simulation results of PBC library were used to support this study and are available at DOI: 10.1109/TII.2017.2761806. These prior studies are cited at relevant places within the text as references [23].

Conflicts of Interest

The author declares that there are no conflicts of interest.

Acknowledgments

The author thanks Ms Yan Di for checking the manuscript. This work was supported by the Scientific and Technological Research Project of Jiangxi Provincial Education Department of China under Grant GJJ201807, the National Natural Science Foundation of China under Grant 62041603, and the Key Project of Jiangxi Provincial Natural Science Foundation of China under Grant 20202ACBL202005.

References

- [1] G. Ateniese, R. Burns, R. Curtmola et al., "Provable data possession at untrusted stores," in *Proceedings of the 2007 ACM Conference on Computer and Communications Security*, pp. 598–609, CCSP, Alexandria, VA, USA, November 2007.
- [2] C. Gentry, "Certificate-based encryption and the certificate revocation problem," *Advances in Cryptology-EUROCRYPT, LNCS*, vol. 2656, pp. 272–293, 2003.
- [3] S. Park, H. Kim, and J. Ryou, "Utilizing a lightweight PKI mechanism to guarantee a secure service in a cloud environment," *The Journal of Supercomputing*, vol. 74, no. 12, pp. 6988–7002, 2018.
- [4] Q. Feng, D. B. He, H. Q. Wang, N. Kumar, and K. K. R. Choo, "White-box implementation of shamir's identity-based signature scheme," *IEEE Systems Journal*, vol. 14, no. 2, pp. 1820–1829, 2020.
- [5] F. Wang, L. Xu, H. Q. Wang, and Z. D. Chen, "Identity-based non-repudiable dynamic provable data possession in cloud storage," *Computers & Electrical Engineering*, vol. 69, pp. 521–533, 2018.
- [6] C. C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *Proceedings of the 2009 ACM Conference on Computer and Communications Security*, pp. 213–222, CCS 2009, Chicago, IL, USA, November 2009.
- [7] Q. Wang, C. Wang, J. Li, K. Ren, and W. J. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in *Proceedings of 14th European Symposium on Research in Computer Security (ESORICS 2009)*, pp. 355–370, Saint-Malo, France, September 2009.
- [8] C. Wang, Q. Wang, K. Ren, and W. J. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *Proceedings of 29th IEEE International Conference on Computer Communications (Infocom 2010)*, pp. 525–533, San Diego, CA, USA, March 2010.

- [9] R. Rabaninejad, M. A. Attari, M. R. Asaar, and M. R. Aref, "A lightweight identity-based provable data possession supporting users' identity privacy and traceability," *Journal of Information Security and Applications*, vol. 51, Article ID 102454, 2020.
- [10] H. Q. Wang, D. B. He, J. Yu, and Z. W. Wang, "Incentive and unconditionally anonymous identity-based public provable data possession," *IEEE Transactions on Services Computing*, vol. 12, no. 5, pp. 824–835, 2020.
- [11] H. Q. Wang and J. G. Li, "Private certificate-based remote data integrity checking in public clouds," in *Proceedings of 21st International Conference of Computing and Combinatorics(COCOON) 2015*, pp. 575–586, Beijing, China, August 2015.
- [12] F. Wang, L. Xu, K. K. R. Choo, Y. X. Zhang, H. Q. Wang, and J. G. Li, "Lightweight certificate-based public/private auditing scheme based on bilinear pairing for cloud storage," *IEEE Access*, vol. 8, pp. 2258–2271, 2020.
- [13] R. Canetti, O. Goldreich, and S. Halevi, "The random oracle methodology, revisited," *Journal of the ACM*, vol. 51, no. 4, pp. 557–594, 2004.
- [14] J. H. Zhang, P. Y. Li, and J. Mao, "IPad: ID-based public auditing for the outsourced data in the standard model," *Cluster Computing-the Journal of Networks Software Tools and Applications*, vol. 19, no. 1, pp. 127–138, 2016.
- [15] W. T. Shen, J. Yu, H. Xia, and R. Hao, "A note on "IPad: id-based public auditing for the outsourced data in the standard model"" in *Proceedings of Second International Conference of Cloud Computing and Security (ICCCS 2016)*, pp. 335–342, Nanjing, China, July 2016.
- [16] T. Wang, B. Yang, H. Y. Liu, Y. Yu, G. Y. Qiu, and Z. Xia, "An alternative approach to public cloud data auditing supporting data dynamics," *Soft Computing*, vol. 23, no. 13, pp. 4939–4953, 2019.
- [17] L. H. Zhu, H. Y. Wang, C. Xu, K. Sharif, and R. X. Lu, "Efficient group proof of storage with malicious-member distinction and revocation," *IEEE Access*, vol. 7, pp. 75476–75489, 2019.
- [18] S. Thokchom and D. K. Saikia, "Privacy preserving and public auditable integrity checking on dynamic cloud data," *International Journal on Network Security*, vol. 21, no. 2, pp. 221–229, 2019.
- [19] Y. Yang, Q. Huang, and F. Chen, "Secure cloud storage based on RLWE problem," *IEEE Access*, vol. 7, pp. 27604–27614, 2019.
- [20] W. Wu, Y. Mu, W. Susilo, and X. Y. Huang, "Certificate-based signatures revisited," *Journal of Universal Computer Science*, vol. 15, no. 8, pp. 1659–1684, 2009.
- [21] M. Bellare and S. Shoup, "Two-tier signatures, strongly unforgeable signatures, and Fiat-Shamir without random oracles," in *Proceedings of 10th International Conference on Practice and Theory in Public-Key Cryptography (PKC 2007)*, pp. 201–216, Beijing, China, April 2007.
- [22] A. F. Barsoum and M. A. Hasan, "Provable multicopy dynamic data possession in cloud computing systems," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 3, pp. 485–497, 2015.
- [23] D. B. He, N. Kumar, S. Zeadally, and H. Q. Wang, "Certificateless provable data possession scheme for cloud-based smart grid data management systems," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 3, pp. 1232–1241, 2018.