Certificateless Provable Data Possession Protocol for the Multiple Copies and Clouds Case

GENQING BIAN\textsuperscript{1,2} AND JINYONG CHANG\textsuperscript{1}
\textsuperscript{1}School of Management, Xi’an University of Architecture and Technology, Xi’an 710055, China
\textsuperscript{2}School of Information and Control Engineering, Xi’an University of Architecture and Technology, Xi’an 710055, China
Corresponding author: Jinyong Chang (changjinyong@xauat.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61672416, Grant 61672059, and Grant 61872284, and in part by the Project of Natural Science Research in Shaanxi under Grant 2018JM6105 and Grant 2019JM118.

ABSTRACT For user’s extremely important data, storing multiple copies on cloud(s) may be a good option because even if the integrity of one or more copies is broken, it can still recover data from other intact ones, which increases the availability and durability of the outsourced data. Some provable data possession (PDP) protocols guaranteeing the integrity of multi-copies had been proposed in the past years. But almost all of them considered storing multi-copies to single cloud, and the necessary management of certificates as well as the dependence on PKI greatly decrease their efficiencies. Therefore, in recent work, Li \textit{et al.} proposed an identity-based PDP protocol, which not only avoids the tedious certificates and PKI, but also supports multi-copies stored on multi-clouds. However, it is well-known that identity-based protocols suffer from the key-escrow attack. In this paper, we consider the certificateless multi-copy-multi-cloud protocol. Specifically, we first present its security model and then construct a concrete protocol, whose security can be proven under the classical CDH assumption. Finally, the performance analysis demonstrates that our protocol yields better efficiency and hence is practical.

INDEX TERMS Data storage, certificateless PDP, multi-copy-multi-cloud, data integrity.

I. INTRODUCTION
Outsourcing data to a remote cloud service provider (CSP) instead of private computers allows individuals or organizations to save their space and concentrate on innovations or other aspects, which relieves the burden of constant server updates as well as other computing issues. Meanwhile, some authorized users can access those remotely stored data from different geographic locations, which provides convenience for them. Hence, cloud storage has become a popular trend for individuals or organizations in recent years, and many famous corporations, such as Google, Microsoft, Amazon, Huawei, IBM etc. provide this service to people in the whole world [4].

When uploading their data to CSP, data owners (DOs) will delete them from their local computers for saving space. Thus, they also lose the direct control of their own data. If the remote CSP is not trustworthy, then user’s data is just in a dangerous situation. For example, the CSP may secretly delete user’s data that is not commonly used for saving its own space and hence attract more users to store data on it. How to guarantee the integrity of user’s data is an extremely interesting problem and in recent years, many provable data possession (PDP) schemes are designed [2], [11], [15], [18]–[20], [25].

Generally speaking, in a PDP scheme, the verifier sends a challenge message to CSP, who should return an integrity proof based on DO’s stored data and this challenge message. If this proof can pass the verification of the verifier, then DO believes that its data is intact. Otherwise, it is broken and DO can claim for compensations from CSP. For some resource-constrained DO, it prefers to authorize third-party auditor (TPA), who has more professional knowledge and powerful resources, to perform the verification task. Hence, TPA-based PDP scheme is a useful tool to help DO check data’s integrity in cloud storage.

However, for DO, once CSP loses its data, then it may lose it forever, although it knows this truth by TPA’s verifying. If the data is extremely important, such as bank’s account or daily’s transaction information, then DO would recover its data rather than obtain CSP’s compensations. To improve the availability and durability of data, a good choice for DO may be generate multiple copies and store all of them on CSP.
In this situation, when one copy is tampered, the DO can still recover its data from other copies. To further decrease the risk, the DO can also store its multiple copies on different CSPs. In this situation, it can still recover its data from other CSPs even all the copies in some CSP are lost or tampered. Therefore, the traditional PDP schemes should also be twisted or extended so that the integrity of DO’s data can also be checked in this case. We call this PDP scheme as multi-copy-multi-cloud (MCMC) PDP protocol.

When defining or designing MCMC-PDP protocol, we must note the following facts. (a). Data’s copies should be different from each other. The reason lies in that, if some two copies on one CSP are the same, then the CSP only needs to store one and delete another copy. Meanwhile, it claims that the two same copies are integrally stored. In this sense, we somewhat abuse this word “copy” since in general, copies should be completely same. (b). All copies should be simultaneously checked by only one challenge-response interaction. Or it will be meaningless to define and design MCMC-PDP protocol. (c). From the view of security, TPA should not gain any useful information on DO’s data except for honestly conducting the verification.

In the recent work, Li et al. proposed an efficient MCMC-PDP protocol, which satisfies all those facts [12]. As they said, their protocol is also identity-based (IB), which avoids the tedious certificates management and the dependence on public key infrastructure (PKI) because traditional PDP schemes with public verifiability often encounter the problem that how to recognize the relationship between someone’s true identity and his/her public key. In their IB-MCMC-PDP protocol, an additional entity: key-generation center (KGC) is proposed, who is responsible for issuing keys to different users (or DOs).

It is also well-known that, although identity-based primitives avoid the management of certificates and usage of PKI, they suffer from the key-escrow attack, which means that KGC owns all the private keys of its users and hence can do everything, such as signing or decrypting, on behalf of any user [9], [13], [19]. Therefore, it is extremely important and urgent to design a MCMC-PDP protocol that not only avoids the certificates as well as PKI but also does not have the key-escrow problem.

A. OUR CONTRIBUTIONS

In this paper, we apply the certificateless-technique [3] to MCMC-PDP protocol and thus propose a CL-MCMC-PDP protocol. More precisely, our contributions includes the following aspects.

- For the first time, we define the security model for CL-MCMC-PDP protocol. In fact, in our security model, the adversaries are divided into three types, which respectively model a malicious KGC, a general adversary allowed to replace user’s public key, and a dishonest CSP.
- We propose a concrete CL-MCMC-PDP protocol and, in the random oracle model, prove its security based on the computational Diffie-Hellman (CDH) assumption.
- Analyze the performances of our proposed protocol in terms of communication and computational costs. By comparing it with Li et al.’s IB-MCMC-PDP and another traditional multi-copies PDP protocol [26], we illustrate that this protocol is very efficient and hence practical.

B. RELATED WORKS

In 2007, Ateniese et al. presented the new notion of PDP, which enables DOs to check the integrity of their data stored on an untrusted CSP without downloading the whole file, and realized the so-called “spot-checking” technique [2]. This scheme is also the first one supporting public auditing. In the same period, Juels et al. considered a stronger model: Proof of retrievability (PoR), and proposed a concrete implementation [11]. In the next year, Schamcham and Waters improved Juels et al.’s model and designed a new scheme, which can be proven secure under CDH assumption and compact [15].

To achieve better efficiency and properties, several PDP protocols are proposed, such as [8], [17], [21], [22]. Those previously suggested ones used the traditional public key cryptography technique and hence a certificate, issued by an trusted certificate-authority center to bind some DO’s identity and its public key, is needed. The main problem for this cases lies in that as the number of users grows, the certificate management becomes extremely difficult. Therefore, the certificate-based PDP protocols become very inefficient when they are used in practical scenarios.

In fact, back to 1984, Shamir proposed the notion of identity-based cryptography to resolve this problem [16]. Hence, many researchers considered to apply identity-based cryptographic technique to PDP protocols and hence many IB-PDP schemes have been proposed [18], [23], [24], [27].

However, in later applications, researchers also found that the IB-PDP protocol had become extremely vulnerable under the key-escrow attack since the KGC has too powerful ability. To help the PDP protocol defend from this attack, many certificateless PDP protocols [9], [10], [13], [19] were proposed, which combined the certificateless cryptography [3] with PDP schemes.

C. ORGANIZATIONS

The later parts are organized as follows. In Section 2, we introduce some basic notations and notions. In Section 3, we present our proposed CL-MCMC-PDP protocol and its security analysis. In Section 4, discuss other properties of this protocol. Performance analysis can be found in Section 5. Finally, we conclude this paper in Section 6.

II. PRELIMINARIES

Basic Notations: In this paper, we always use \( \lambda \) as the security parameter and \( 1^\lambda \) is its unary form. For a prime \( q \), the symbol \( \mathbb{Z}_q^* \) defines the finite field \( \{0, 1, \cdots, q-1\} \) and \([q]\)
denotes the same set. \(|Z_q^*|\) means the bit-length of one element in \(Z_q^*\). \(\text{PPT}\) is an abbreviation of the term "probabilistic polynomial time". We call a function \(f(\lambda)\) is negligible, if for any \(c > 0\), there exists some \(\lambda_0 \in \mathbb{Z}\) such that \(f(\lambda) < \lambda^{-c}\) for any \(\lambda > \lambda_0\). Other symbols and their definitions appeared in our paper is listed in Table 1.

**TABLE 1. Symbols and their definitions.**

| Symbols | Definitions |
|---------|-------------|
| CSP     | Cloud Service Provider |
| TPA     | Third-Party Auditor |
| KGC     | Key Generation Center |
| DO      | Data Owner |
| PDP     | Provable Data Possession |
| MCMC    | Multi-Copy Multi-Cloud |
| CL      | Certificateless |
| CDH     | Computational Diffie-Hellman |
| \(G_1, G_2\) | Two Groups |
| \(|G_1|\) | Bit-Length of One Element in \(G_1\) |
| \(e\) | Bilinear Map over \(G_1 \times G_1\) |
| \(P\) | A Generator of \(G_1\) |
| \(ID\) | DO’s identity |
| \(Fid\) | Data File’s Filename |
| \(F\) | The Initial Data File |
| \(F_{i}\) | The \(i\)-th Copy |
| \(n\) | The Number of Copies |
| \(t\) | The Number of Blocks for Each Copy |
| \(\mathcal{C}\) | DO’s Storing Policy |
| \(m_{i,j}\) | The \(j\)-th Block of \(F_i\) |
| \(T_{i,j}\) | The Tag of \(m_{i,j}\) |
| \(Cid\) | The Identity of the Cloud Storing \(F_i\) |
| \(chal\) | The Challenge Message |
| \(\Gamma = (\sigma, M)\) | The Generated Proof |

**A. BILINEAR MAP**

Assume that \(G_1, G_2\) are two multiplicative groups with the same order \(q\). The map \(e\) from \(G_1 \times G_1\) to \(G_2\) is called bilinear if:

- **Non-Degeneracy.** For any generator \(P \in G_1\), it holds that \(e(P, P) \neq 1_{G_2}\).
- **Computability.** It is efficient to compute the values of \(e\).
- **Bilinearity.** For any \(P, Q \in G_1\) and \(a, b \in Z_q\), it holds that \(e(P^a, Q^b) = e(P, Q)^{ab}\).

**B. CDH ASSUMPTION**

For a group \(G\) with the prime order \(q\) and generator \(P\), the CDH assumption means that, for any PPT adversary \(\mathcal{A}\), it is computational infeasible to output the group element \(P^{ab}\) when given the tuple \((P, P^a, P^b)\). In other words, the probability to output \(P^{ab}\) is negligible. Here, the probability is taken over the randomness of \(a, b\) and the inner coins of \(\mathcal{A}\).

**C. SYSTEM MODEL**

In the CL-MCMC-PDP system model, there are five kinds of entities: KGC, DO, CO, CSPs and TPA, which work as follows.

- The KGC issues partial-private keys for different DOs after getting their identities.
- A DO obtains partial-private key from KGC and generates its full private key. Using this key, it can generate authenticated tags for many copies of the outsourced file. Then store those files as well as their tags to different CSPs.
- CSP provides storage services to DOs and rises to the challenge from the TPAs, who may check the integrity of DO’s data.
- The CO is a manager of CSPs. When storing the (authenticated) file copies, a DO first sends them to CO. Then CO assigns different copies to the target CSPs according to DO’s request. When checking the integrity, TPA first gives the challenge message to CO, who will distribute this message to the corresponding CSPs. After receiving all the proofs from those CSPs, this CO aggregates them into the final proof and returns it to TPA.
- TPA verifies the integrity of all outsourced copies on behalf of DO.

A graphic description of those entities as well as their working ways can be seen in Fig. 1.

**FIGURE 1. A graphic description of the system model.**

Generally speaking, a concrete CL-MCMC-PDP protocol includes the following PPT algorithms: Setup, PPGen, Set-Secret-Value, Set-Private-Key, Set-Public-Key, RepGen, TagGen, TagVer, Chal, ProofGen, ProofAggr and Verify, which work across the above entities.

- **Setup(\(\lambda\)) \rightarrow (\text{params}, \text{msk})**. This algorithm is run by KGC and will output the system parameters \text{params} and the KGC’s master secret key \text{msk}. Note that this \text{params} can be publicly obtained by all other entities in the system but \text{msk} is private.
- **PPGen(\text{msk}, \text{ID}) \rightarrow \text{PP}_{\text{ID}}**. This algorithm is run by KGC and used to generate DO’s partial private key \text{PP}_{\text{ID}} based on the queried identity \text{ID}.
- **Set-Secret-Value(\text{ID}) \rightarrow \text{x}_{\text{ID}}**. This algorithm is run by DO and will generate DO’s secret value \text{x}_{\text{ID}}, which is used to construct DO’s full private key.
- **Set-Private-Key(\text{PP}_{\text{ID}}, \text{x}_{\text{ID}}) \rightarrow \text{sk}_{\text{ID}}**. This algorithm is still run by DO and will output DO’s full private key.
key skID based on its partial private key PPID and secret value \( x_{ID} \).

- **Set-Public-Key**(*skID* → *pkID*). The DO uses this algorithm to generate its public key *pkID* based on *skID*.
- **RepGen**(*F*, *N*/*n*) → \( \mathbb{F} \). The DO uses this algorithm to generate *N* copies (denoted by \( \mathbb{F} \)) of the data file *F* to be stored on CSPs. Denote the *i*-th copy by \( F_i \) (\( 1 \leq i \leq N \)). Then this file \( F_i \) is split into *n* blocks: \( m_{i,1}, m_{i,2}, \ldots, m_{i,n} \). The output is \( \mathbb{F} = \{F_i\}_{i=1}^N = \{(m_{i,1}, m_{i,2}, \ldots, m_{i,n})\}_{i=1}^N \).
- **TagGen**(*ID*, *skID*, *pkID*, *F*, *C*): This algorithm is run by DO to generate authenticated tags (denoted by \( T_i \)) with respect to \((ID, pkID)\) for the initial file \( F \) according to some policy \( C \). Here, the policy denotes how the DO will store its copy files and their tags to the CSPs. For example, for the copies \( F_1, F_2, \ldots, F_N \), in the first CSP, he would like to store the tuples \((F_1, T_1), (F_3, T_3)\) and \((F_2, T_2)\), while, in the second one, he would like to store \((F_2, T_2)\) and \((F_4, T_4)\) and so on. We remark that if multiple copies \((F_i, T_i)\) will be stored on a same cloud, then their cloud identities *Cid* are identical.
- **TagVer**(*ID*, *skID*, *pkID*, *Fid*, *i*, *j*, *Cid*, *m_{i,j}*, *T_{i,j}*): This algorithm is run by CO or CSP to check the validity of \((m_{i,j}, T_{i,j})\) w.r.t. the tuple \((ID, pkID, Fid, i, j, Cid)\), where *Fid* is the unique filename of the initial file \( F \), *Cid* is the identity of target CSP that will store the *i*-th copy.
- **Chal(\( \ell \)) → chal(\( \ell \)).** This algorithm is run by TPA to generate challenge message chal. Here, the input \( \ell \) denotes the number of blocks to be challenged (for single copy file).
- **ProofGen**(*Fid*, \((F_i, T_i), \text{chal}) → \( \Gamma_1 \). This algorithm is run by CSP with identity *Cid* that stores the authenticated copy file \((F_i, T_i)\). Based on the challenge message chal(\( \ell \)), this algorithm returns a proof \( \Gamma_1 \).
- **ProofAggr**(*\( \Gamma_1, \Gamma_2, \ldots, \Gamma_{\ell} \)) → \( \Gamma \). This algorithm is run by CSP or CO to generate an aggregated proof \( \Gamma \). If multiple copies are stored on one cloud, then this cloud will aggregate those generated proofs (from the above algorithm ProofGen) into one by using this algorithm. Alternately, if this algorithm is run by CO, then it will generate an aggregated proof from different CSPs.
- **Verify**(*ID*, *pkID*, chal, \( \Gamma, \text{Fid}, \{\text{Cid}\} \)) → 1/0. This algorithm is run by TPA to check the validity of returned proof \( \Gamma \) from CO. Note that this proof should be an aggregation of all the proofs from different CSPs, and hence it needs to take all those clouds’ identity as inputs.

### D. AN EXAMPLE

For clear readability, in this subsection, we take an example to describe how the CL-MCMC-PDP protocol works when combing the algorithms with the entities.

Concretely, a KGC first runs **Setup**(*\( \ell \)) to get *params* and *msk*. Broadcast *params* and keep *msk* secret. When a user DO with identity *ID* requesting its partial private key by submitting *ID*, KGC runs **PPGen**(*msk*, *ID*) and returns it to this user. Now, DO runs **Set-Secret-Value**(*ID*) to get \( x_{ID} \) and **Set-Private-Key**(*PPID*, \( x_{ID} \)) to get the full private key \( skID \). Then obtain its public key \( pkID \) by running **Set-Public-Key**(*skID*). Broadcast \( pkID \) to other entities. This also finishes the setup process of DO’s key.

If DO wants to store its initial file \( F \) to different CSPs, he first randomly chooses \( Fid \) from some set, such as \((0, 1)^*\), and runs \( \mathbb{F} \leftarrow \text{RepGen}(F, N, n) \) to get the *N*-copies of \( F \), each with *n*-blocks. For example, \( N = 5, n = 4 \). Then \( \mathbb{F} = \{F_1, F_2, F_3, F_4, F_5\} \). Now, he chooses a policy \( C = \{(1, 3, 4), (2, 5)\} \). That is, he intends to store \( F_1, F_2, F_3, F_4 \) to the first CSP and \( F_2, F_5 \) to the second CSP. Thus, \( \text{Cid}_1 = \text{Cid}_3 = \text{Cid}_4 \), and \( \text{Cid}_2 = \text{Cid}_5 \). Then run **TagGen**(*ID*, *skID*, *pkID*, *F*, *C*) to generate the corresponding tags \( T_1, T_2, T_3, T_4, T_5 \), where \( T_i = \{T_{i,1}, T_{i,2}, T_{i,3}, T_{i,4}\} \) (\( 1 \leq i \leq 5 \)). Finally, transmit the authenticated data \( \{(F_1, T_1), (F_2, T_2), (F_3, T_3), (F_4, T_4), (F_5, T_5)\} \) and \((\text{Fid}, C)\) to CO.

After receiving DO’s data, CO first runs **TagVer**(*ID*, *pkID*, *Fid*, *i*, *j*, *Cid*, *m_{i,j}*, *T_{i,j}*) to check the validity of each pair \((m_{i,j}, T_{i,j})\) w.r.t. the tuple \((ID, pkID, Fid, i, j, Cid)\). If all the verifications are correct, then transmit \((F_1, T_1), (F_3, T_3), (F_4, T_4), (F_5, T_5)\) to the first CSP and \((F_2, T_2), (F_5, T_5)\) to the second CSP.

If TPA needs to check the integrity of DO’s data stored in the two CSPs after receiving DO’s information \((\text{Fid}, C)\), he runs \( \text{chal} \leftarrow \text{Chal}(\ell) \) and gives it to CO, who will transfer it to the two CSPs. Now, the first CSP runs \( \Gamma_1 \leftarrow \text{ProofGen}(\text{Fid}, (F_1, T_1), \text{chal}) \) for \( i = 1, 3, 5 \) and then aggregate them into a new proof \( \Gamma' \) by running **ProofAggr**(*\( \Gamma_1, \Gamma_3, \Gamma_5 \)) on CSP and \((F_2, T_2)\) and \((F_5, T_5)\).

Obtaining the returned two proofs \( \Gamma' \) and \( \Gamma'' \), the CO aggregates them into a whole proof \( \Gamma \) by running **ProofAggr**(*\( \Gamma', \Gamma'' \)) again. Then return the proof \( \Gamma \) to TPA, who will check the validity of \( \Gamma \) by running **Verify**(*ID*, *pkID*, \( \text{chal}, \Gamma, \text{Fid}, \{\text{Cid}\} \)) \( i \leq 5 \). If the result is 1, then accept this \( \Gamma \). Otherwise, inform DO that the integrity of his data is broken.

### E. REQUIREMENTS

The following fundamental requirements are needed for a CL-MCMC-PDP protocol.

- **Correctness.** If the CSPs honestly generate their proof based on TPA’s challenge message, then the aggregated proof can pass the verification of TPA.
- **Security.** If some CSP loses one or more blocks of DO’s data file, then the TPA can detect it with very high probability. Moreover, for certificateless PDP, it also should be secure against malicious KGC and general adversaries, who may replace DO’s public key and try to forge valid signatures.
- **Blockless.** For the verifier, it can correctly verify data’s integrity without downloading the original data file.
• **Public Verifiability.** The auditing process should be public and hence is suitable to outsourced to the curious TPA, which means that although he honestly performs the auditing task, it may also be interest in the contents of DO’s data.

• **Privacy Preserving.** The TPA should not get any useful information on contents of DO’s stored data during the process of auditing.

• **Certificateless.** In the whole process of auditing, it is not needed to use any certificates to identify the relationship between DO’s identity and it’s public key.

### F. SECURITY MODEL

In this subsection, we define the security model of CL-MCMC-PDP protocol. Concretely, we consider three types of adversaries: Type I, II and III, which model the misbehaving of malicious KGC, general adversary against the signature of single block, and CSP, respectively.

**Type I Adversary.** For this kind of adversary, it has access to the master secret key $\text{msk}$ but is not allowed to replace user’s public key (or it will certainly succeed in the following security game).

**Game I.** This is a game played between a challenger $\text{CH}_1$ and an adversary $\mathcal{A}_1$.

- **Phase-I-1.** First, the challenger runs the algorithm $\text{Setup}(\lambda)$ to obtain system parameter $\text{params}$ and a master secret key $\text{msk}$. Give them to $\mathcal{A}_1$. Note that this adversary can generate any user’s partial private key because it has $\text{msk}$.

- **Phase-I-2.** $\mathcal{A}_1$ can adaptively make the following queries, which are recorded in a list $L$.
  - **Private-Key-Query.** $\mathcal{A}_1$ submits $ID$ together with the corresponding partial private key $PP_{ID}$ (generated by itself) to the challenger, then runs $x_{ID} \leftarrow \text{Set-Secret-Value}(ID)$ and $sk_{ID} \leftarrow \text{Set-Private-Key}(PP_{ID}, x_{ID})$. Finally, record $(ID, PP_{ID}, x_{ID}, sk_{ID})$ to the list $L$ and return $sk_{ID}$ to $\mathcal{A}_1$.

  - **Public-Key-Query.** The adversary $\mathcal{A}_1$ submits $(ID, PP_{ID})$ to the challenger, who checks if $sk_{ID}$ exists in $L$. If it is, recover it. Otherwise, generate it as the above step. Then run $pk_{ID} \leftarrow \text{Set-Public-Key}(sk_{ID})$. Give $pk_{ID}$ to $\mathcal{A}_1$ and update $ID$’s public key as $pk_{ID}$.

  - **Tag-Generation-Query.** $\mathcal{A}_1$ submits the identity $ID$, its partial private key $PP_{ID}$, the data file $F$ and the policy $C$ to $\text{CH}_1$, who will first check if $sk_{ID}, pk_{ID}$ exist in $L$. If it is, recover them. Otherwise, generate them as in the above two steps. Then randomly choose $Fid$ from a proper set, run $T \leftarrow \text{TagGen}(ID, sk_{ID}, pk_{ID}, F, C)$, and return $T$ to $\mathcal{A}_1$.

  - **Transcript-Query.** This query returns the verification transcript among the TPA, CO, and CSPs for any data file $F$, policy $C$, and DO’s identity $ID$ chosen by $\mathcal{A}_1$. Concretely, $\mathcal{A}_1$ submits an identity $ID$, its partial private key $PP_{ID}$, $C$, and a data file $F$ to the challenger, who will first randomly choose the filename $Fid$, generate $N$ copies $F_1, F_2, \ldots, F_N$ by running the algorithm $\text{RepGen}(F, N, n)$ and compute all the tags $T_1, T_2, \ldots, T_N$ for all the copies by running $\text{TagGen}$. Then the challenger stores the files and their corresponding tags according to $C$. Now, $\mathcal{A}_1$ continues to submit $\text{chal}$ to $\text{CH}_1$ by running $\text{Chal}(\ell)$. The challenger will compute the proof $\Gamma_1$ and the aggregated $\Gamma$ by running $\text{ProofGen}$ and $\text{ProofAggr}$. Return all the generated proofs to $\mathcal{A}_1$.

- **Phase-I-3.** Finally, $\mathcal{A}_1$ outputs the tuple $(ID^*, Fid^*, i^*, j^*, Cid^*, m^*, T^*)$ as a forgery.

If it holds that
\[ 1 \leftarrow \text{TagVer}(ID^*, pk_{ID^*}, Fid^*, i^*, j^*, Cid^*, m^*, T^*), \]
and the following conditions hold:

1) $ID^*$ has not been issued as a private-key-query.

2) $T^*$ should not be the generated tag of $m^*$ w.r.t. the tuple $(ID^*, pk_{ID^*}, Fid^*, i^*, j^*, Cid^*)$ by making the tag-generation-query.

then we call $\mathcal{A}_1$ wins the above game.

**Type II Adversary:** For a general adversary against the signature of single block, it is allowed to replace user’s public key but not able to have the master secret key. Next, we consider the following security game.

**Game II:** This is a game played between a challenger $\text{CH}_2$ and an adversary $\mathcal{A}_2$.

- **Phase-II-1.** First, the challenger runs the algorithm $\text{Setup}(\lambda)$ to obtain system parameter $\text{params}$ and a master secret key $\text{msk}$. Give $\text{params}$ to $\mathcal{A}_2$.

- **Phase-II-2.** $\mathcal{A}_2$ is allowed to adaptively make the following queries, which are recorded in a list $L$.

  - **Partial-Private-Key-Query.** The adversary submits $ID$ to the challenger for its partial private key. $\text{CH}_2$ runs

    $PP_{ID} \leftarrow \text{PPGen}(\text{msk}, ID),$

    and return $PP_{ID}$ to $\mathcal{A}_2$. Add $(ID, PP_{ID})$ to $L$.

  - **Private-Key-Query.** $\mathcal{A}_2$ submits $ID$ to the challenger, who will first check if $PP_{ID}$ exists in $L$. If it is, recover it. Otherwise, make a partial-private-key-query. Then run $x_{ID} \leftarrow \text{Set-Secret-Value}(ID)$ and $sk_{ID} \leftarrow \text{Set-Private-Key}(PP_{ID}, x_{ID})$. Finally, update the item $(ID, PP_{ID}, x_{ID}, sk_{ID})$ in the list $L$ and return $sk_{ID}$ to $\mathcal{A}_2$.

1 Note that if the key pair $(sk_{ID}, pk_{ID})$ of $ID$ does not exist in $L$, then generate it by making private-key-query and public-key-query.
– **Public-Key-Query.** The adversary $A_{II}$ submits $ID$ to the challenger, who checks if $sk_{ID}$ exists in $L$. If it is, recover it. Otherwise, generate it as the above step. Then run $pk_{ID} ← \text{Set-Public-Key}(sk_{ID})$. Give $pk_{ID}$ to $A_{II}$ and update $ID$’s public key (in $L$) as $pk_{ID}$.

– **Replace-Public-Key-Query.** $A_{II}$ submits $(ID, pk_{ID}, pk'_{ID})$ to the challenger in order to replace $pk_{ID}$ with $pk'_{ID}$. Then $CH_{II}$ updates $ID$’s public key as $pk'_{ID}$.

– **Tag-Generation-Query.** $A_{II}$ submits the identity $ID$, the data file $F$ and the policy $C$ to $CH_{I}$, who will first check if $sk_{ID}, pk_{ID}$ exist in $L$. If it is, recover them. Otherwise, generate them as in the previous two steps. Then randomly choose $Fid$ from a proper set, run

$$T ← \text{TagGen}(ID, sk_{ID}, pk_{ID}, F, C),$$

and return $T$ to $A_{II}$.

– **Transcript-Query.** This query is the same as the transcript-query of Game I except that $A_{II}$ does not need to submit the partial private key $PP_{ID}$ when making this kind of queries.

- **Phase-II-3.** Finally, $A_{II}$ outputs the tuple $(ID^*, pk_{ID^*}, Fid^*, i^*, j^*, Cid_{F^*}, m^*, T^*)$ as a forgery.

If (1) holds and the following conditions hold:  
1) $ID^*$ has not been issued as a private-key-query. 
2) $ID^*$ cannot be an identity for which both the public key has been replaced and the partial private key been extracted. (Or the adversary can definitely win the game.)

3) $T^*$ should not be the generated tag of $m^*$ w.r.t. the tuple $(ID^*, pk_{ID^*}, Fid^*, i^*, j^*, Cid_{F^*})$ by making the tag-generation-query,

then we call $A_{II}$ wins this game.

**Type III Adversary:** This kind adversary models a malicious CSP who intends to pass the verification of TPA when the integrity of DO’s stored data file is broken. Now, we consider the following security game.

**Game III:** This is a game played between a challenger $CH_{III}$ and an adversary $A_{III}$.

- **Phase-III-1.** Same as Phase-II-1.

- **Phase-III-2.** This phase is the same as Phase-II-2 in the above Game II except that in the transcript-query, the challenger $CH_{III}$ runs $chal ← \text{Chal}(\ell)$ and submits it $A_{III}$, who will compute $\Gamma_1$ and $\Gamma$ by running $\text{ProofGen}$ and $\text{ProofAggr}$, respectively. Then the challenger checks the validity of the returned proof and gives the result to $A_{III}$.

- **Phase-III-3.** Finally, $A_{III}$ outputs the tuple $(ID^*, pk_{ID^*}, Fid^*, \Gamma^*)$ as a forgery.

Denote by $\tilde{\Gamma}$ the returned proofs (based on the same challenge message as that of $A_{III}$ to compute $\Gamma^*$) generated by other honest CSPs. Compute $\tilde{\Gamma}^* ← \text{ProofAggr}(\Gamma^*, \tilde{\Gamma})$. If it holds that

$$1 ← \text{Verify}(ID^*, pk_{ID^*}, chal, \Gamma^*, Fid^*, \{Cid_i\}),$$

and the returned $\Gamma^*$ does not equal to the one that it should be for the challenge message, then we call $A_{III}$ wins this game.

If any PPT adversaries $A_1, A_{II}$ and $A_{III}$ cannot win Game I, II, and III, respectively, then the CL-MCMC-PDP protocol is secure.

### III. THE PROPOSED PROTOCOL AND SECURITY ANALYSIS

#### A. THE PROPOSED PROTOCOL

In this subsection, we propose a concrete CL-MCMC-PDP protocol, which consists of the following algorithms.

- **Setup.** For input of security parameter $\lambda$, this algorithm chooses two cyclic groups $G_1, G_2$ with the same prime order $q$ ($|q| ≥ \lambda$) and $e : G_1 × G_1 → G_2$ the bilinear map. Choose $P$ as a generator of $G_1$ and let $h : [0, 1]^* → Z_q^\ast, H, \bar{H} : [0, 1]^* → G_1$ be three hash functions. Then randomly choose $s \in Z_q^\ast$ and compute $P_{pub} = P^s$. Finally, set $msk = s$ as the master secret key and

$$\text{params} = (q, G_1, G_2, e, h, H, \bar{H}, P, P_{pub})$$

as the system parameter that can be publicly known by other algorithms. Output ($msk, \text{params}$).

- **PPGen:** For the inputs $msk$ and $ID$, this algorithm randomly chooses $t_{ID}$ from $Z_q^\ast$ and computes $T_{ID} = P^{t_{ID}}$. 

$$h_{ID} = h(ID, T_{ID}), s_{ID} = t_{ID} + s \cdot h_{ID} \mod q.$$ 

Output the partial private key $PP_{ID} = (T_{ID}, s_{ID})(for \ ID)$, where $T_{ID}$ is public and $s_{ID}$ is private.

- **Set-Secret-Value:** For the input $ID$, this algorithm randomly chooses $x_{ID}$ from $Z_q^\ast$ and outputs it.

- **Set-Private-Key:** For the inputs $PP_{ID}$ and $x_{ID}$, this algorithm sets $sk_{ID} = (PP_{ID}, x_{ID})$ and outputs it.

- **Set-Public-Key:** For the input $sk_{ID}$, parse it as $(PP_{ID}, x_{ID}) = (T_{ID}, s_{ID}, x_{ID})$ and compute $X_{ID} = P^{s_{ID}}$. Output the public key $pk_{ID} = (T_{ID}, X_{ID})$ (for $ID$).

- **RepGen:** For the inputs of file $F$ and the numbers $N, n$, this algorithm first splits $F$ into $n$ blocks $m_1, m_2, \cdots, m_n$ satisfying each block is smaller than $q$, and for all $i \in [N]$ and $j \in [n]$, encrypt each block $m_{i,j} = E_k(i||m_j)$, where $E$ is a symmetry encryption algorithm (e.g. AES) and $K$ is the corresponding encryption key. Here, we remark that the using of symmetric encryption algorithm is to guarantee the differences of these copies. Define

$$F_i = \{m_{i,1}, m_{i,2}, \cdots, m_{i,n}\}$$

as the $i$-th copy of $F$. Finally, output

$$F = \{F_1, F_2, \cdots, F_N\}.$$
ProofAggr: For the inputs \( ID, \) its key pair \((sk_{ID}, pk_{ID}), \) a data file \( F \) and a policy \( C \), this algorithm first randomly chooses \( Fid \) from \( \{0, 1\}^t \) and runs \( F \leftarrow \) RepGen\((F, N, n) \), where \( F = \{F_1, F_2, \ldots, F_N\} \), and each \( F_i = \{m_{i,1}, m_{i,2}, \ldots, m_{i,n}\} \) for \( 1 \leq i \leq n \). Then compute
\[
Q_{ID} = H(ID, pk_{ID}) = H(ID, T_{ID}, X_{ID}).
\]
For \( 1 \leq i \leq N, 1 \leq j \leq n, \) and \( \text{Cid}_i \in C, \) compute
\[
T_{ij} = (\hat{H}(Fid, i, j, \text{Cid}_i) \cdot (Q_{ID})^{m_{ij}})^{sD+pID}.
\]
Output
\[
T = \{T_1, T_2, \ldots, T_N\} = \{(T_{i,1}, T_{i,2}, \ldots, T_{i,n})|_{1 \leq i \leq N}\}.
\]
ProofTag: For the input of the tuple
\[
(ID, pk_{ID}, Fid, i, j, \text{Cid}_i, m_{ij}, T_{ij}),
\]
this algorithm checks if
\[
e(T_{ij}, P) = e(\hat{H}(Fid, i, j, \text{Cid}_i) \cdot (Q_{ID})^{m_{ij}}, X_{ID} \cdot T_{ID} \cdot (P_{pub})^{ID}) \equiv e(\hat{H}(Fid, i, j, \text{Cid}_i) \cdot (Q_{ID})^{m_{ij}}, \tilde{P}_{ID}) ,
\]
in which we denote by \( \tilde{P}_{ID} \) the element \( X_{ID} \cdot T_{ID} \cdot (P_{pub})^{ID} \). If it is output 1; otherwise output 0.

Chal: For the input \( \ell \), this algorithm chooses \( 1 \leq v_1 < v_2 < \ldots < v_\ell \leq n \) and then randomly chooses \( a_1, a_2, \ldots, a_\ell \) from \( Z_2^n \). Output chal = \( \{(v_r, \tau_r)\}_{r=1}^{\ell} \).

ProofGen: For the inputs \( Fid, (F_i, T_i) \), and chal, this algorithm first parses \( F_i \) as \( \{m_{i,1}, m_{i,2}, \ldots, m_{i,n}\} \), \( T_i \) as \( \{T_{i,1}, T_{i,2}, \ldots, T_{i,n}\} \), and chal as \( \{(v_r, \tau_r)\}_{r=1}^{\ell} \). Compute
\[
\sigma_i = \prod_{r=1}^{\ell} T_{i,v_r}^{a_r m_{i,v_r}} \mod q.
\]
Finally, return \( \Gamma_i = (\sigma_i, M_i) \) as the proof of the stored file \((F_i, T_i)\).

ProofAggr: For the inputs \( \Gamma_{i_1}, \Gamma_{i_2}, \ldots, \Gamma_{i_\xi} \), this algorithm parses each \( \Gamma_{ij} \) as \((\sigma_{ij}, M_{ij})\) for \( 1 \leq j \leq \xi \), and computes
\[
\sigma = \prod_{j=1}^{\xi} \sigma_{ij}, M = \sum_{j=1}^{\xi} M_{ij} \mod q.
\]
Output \( \Gamma = (\sigma, M) \).

Note that this algorithm can be run by any CSP or the CO. Here, consider a concrete example. Assume that the \( N \)-copies will be stored on \( r \) different CSPs and, without loss of generality, the copies \( \{(F_{j_0,1}, T_{j_0,1}), \ldots, (F_{j_r,1}, T_{j_r,1})\} \) are stored on the \( j \)-th (\( 1 \leq j \leq r \)) CSP, where \( i_0 = 1 \) and \( i_r = N \). Then we have \( \text{Cid}_{i_{j-1}} = \cdots = \text{Cid}_{i_j} \).

If it is run by the \( j \)-th CSP, then the returned proof is
\[
\Gamma_j = (\sigma^{(j)}, M^{(j)}) = \left( \prod_{k=j-1}^{j} \sigma_k \cdot \sum_{k=j-1}^{j} M_k \mod q \right).
\]
If it is run by the CO, then the returned proof is
\[
\Gamma = (\sigma, M) = \left( \prod_{j=1}^{r} \sigma^{(j)} \cdot \sum_{j=1}^{r} M^{(j)} \mod q \right) = \left( \prod_{i=1}^{N} \sigma_i \cdot \sum_{i=1}^{N} M_i \mod q \right).
\]

Verify: For the inputs \( ID, pk_{ID} \),
\[
chal = \{(v_r, \tau_r)\}_{r=1}^{\ell}, \Gamma = (\sigma, M),
\]
\( Fid \), all the cloud’s identities \( \{\text{Cid}_{j} \}_{j=1}^{N} \), this algorithm checks the following equation
\[
e(\sigma, P) = e(\prod_{i=1}^{N} \prod_{r=1}^{\ell} \hat{H}_{\tau_r}(Fid, i, v_r, Cid_i)) \cdot (Q_{ID})^{M} \cdot \tilde{P}_{ID} \). \hspace{1cm} (2)
\]
If it is output 1, otherwise output 0.

The correctness of (2) can be verified as follows.
\[
e(\sigma, P) = e(\prod_{i=1}^{N} \prod_{r=1}^{\ell} \hat{H}_{\tau_r}(Fid, i, v_r, Cid_i)) \cdot (Q_{ID})^{M} \cdot \tilde{P}_{ID} \)
\]
\[
= e(\prod_{i=1}^{N} \prod_{r=1}^{\ell} \hat{H}_{\tau_r}(Fid, i, v_r, Cid_i)) \cdot (Q_{ID})^{M} \cdot \tilde{P}_{ID} \)
\]
\[
= e\left( \prod_{i=1}^{N} \prod_{r=1}^{\ell} \hat{H}_{\tau_r}(Fid, i, v_r, Cid_i) \cdot (Q_{ID})^{M} \cdot \tilde{P}_{ID} \right) \)
\]
\[
B. SECURITY ANALYSIS
\
In this subsection, we analyze the security of the protocol proposed above and present the detailed proof. Concretely, we have the following

Theorem: If the CDH assumption holds in the group \( G_1 \) and \( h, H, \hat{H} \) are modeled as random oracles, then our proposed CL-MCMC-PDP protocol is secure.
In order to prove this main theorem, we parse it into the following three lemmas.

**Lemma 1:** If the CDH assumption holds in the group $G_1$ and $h, \bar{h}$ are modeled as random oracles, then, for any PPT Type I adversary $A_1$, the probability that it can win the Game I applied to CL-MCMC-PDP protocol is negligible.

**Proof:** Consider an adversary $B_1$ who attacks on the CDH assumption and uses $A_1$ as a subroutine. Here, assume that $A_1$ queries $p$ different identities in the whole process. When given the tuple $(P, \Theta, \gamma)$, $B_1$ needs to compute and output the group element $P_{pub}$. Now, randomly choose $\gamma^* \in [p]$ as the guess of user’s identity outputted by $A$ in the final phase, and simulate the environment for $A_1$ as follows.

- **Phase-I-1.** Choose another group $G_2$, which has the same order $q$ as $G_1$, and define $e$ as the bilinear map from $G_1 \times G_1$ to $G_2$. Then randomly choose $s \leftarrow Z_q^*$ and compute $P_{pub} = P^s$. Set $msk = s$ and give the public parameters $(q, P, G_1, G_2, e, P_{pub})$ as well as $s$ to $A_1$. Next, choose a symmetric encryption key $K$. The hash functions are simulated by $B_1$. Finally, initialize an empty list $L$.

- **Phase-I-2.** $B_1$ answers $A_1$’s queries as follows.

  - **Hash Queries.** The simulation of $h$ is just lazy-sampling. For the query to $H$, $B_1$ also adopts the same protocol except that, in the response to generate tags of blocks, he uses a special way, which will be described there. For the query $(ID, T_{ID}, X_{ID})$ to $H$-oracle, $B_1$ first randomly chooses $\beta, \gamma \in Z_q^*$ and then checks if the identity $ID$ is the $j^*$-th new identity appeared in $L$. If it is, and return $H(ID, T_{ID}, X_{ID}) = P_{pub} \cdot \gamma^*$. Else, return $H(ID, T_{ID}, X_{ID}) = P_{pub}$ to $A_1$. Store $(ID, T_{ID}, X_{ID}, H(ID, T_{ID}, X_{ID}), \beta, \gamma, \bot)$ to $L$.

  - **Private-Key-Query.** For the query $(ID, PP_{ID})$, it randomly chooses $X_{ID}$ from $Z_q^*$ and returns it to $A_1$. Add $(ID, sk_{ID}) = (ID, (PP_{ID}, X_{ID}))$ to $L$.

  - **Public-Key-Query.** For the query $(ID, PP_{ID})$, $B_1$ first checks if $ID$ is the $j^*$-th new identity appeared in $L$. If it isn’t, pick $sk_{ID} = (PP_{ID}, X_{ID}) = (T_{ID}, s_{ID})$ from $L$ (if it does not exist, make a private-key-query itself) and computes $X_{ID} = P_{pub} s_{ID}$. Otherwise, set $X_{ID}$ as $\gamma (= P_{pub})$. Here, $B_1$ potentially sets the unknown $b$ as the secret value $x_{ID}$, for $ID_{\gamma}$. Return $pk_{ID} = (T_{ID}, X_{ID})$ to $A_1$ and store $pk_{ID}$ as $ID$’s public key into $L$.

  - **Tag-Generation-Query.** When $A_1$ submits a query $(ID, pk_{ID}, F, C)$ to this oracle, $B_1$ runs $F \leftarrow \text{RepGen}(F, N, n)$, where $F = \{F_1, F_2, \ldots, F_N\}$ and for $1 \leq i \leq N, F_i = \{m_{i,1}, m_{i,2}, \ldots, m_{i,n}\}$. Then randomly choose $Fid \in \{0, 1\}$ and check if $ID$ is the $j^*$-th new identity appeared in $L$. If it isn’t, normally compute the tags for all the messages $m_{ij}$ since he knows the whole private key $sk_{ID}$. Otherwise, randomly choose $r_{mi,j}$ from $Z_q^*$ and define

$$\bar{H}(Fid, i, j, Cid) = \frac{P^{r_{mi,j}}}{H(ID, T_{ID}, X_{ID})^{m_{i,j}}} = \frac{P^{r_{mi,j}}}{(P^b \cdot \gamma)^{m_{i,j}}}.$$  

Then the tag of $m_{ij}$ is

$$T_{i,j} = (\bar{H}(Fid, i, j, Cid) \cdot (Q_{ID}^{m_{i,j}}))^{\gamma + q \cdot \mu} = (\bar{H}(Fid, i, j, Cid) \cdot H(ID, T_{ID}, X_{ID})^{m_{i,j}})^{\gamma + q \cdot \mu} = (P^{r_{mi,j}})^{\gamma + q \cdot \mu} \cdot (P_{pub}^{b})^{m_{i,j}} \cdot P_{pub}^{\gamma} \cdot T_{ID}^{m_{i,j}}.$$  

Denote by $T$ all the tags of $(T_{i,j})_{1 \leq i \leq N, 1 \leq j \leq N}$. Return $(Fid, F, T)$ to $A_1$.

- **Transcript-Query.** The answer to this oracle is normal since $B_1$ can simulate the tags for any data file submitted by $A_1$.

**Phase-I-3.** Finally, the adversary outputs a forgery $(m^*, T^*)$ with respect to $(ID^*, Fid^*, i^*, j^*, Cid^*)$.

From

$$X_{ID^*}T_{ID^*}(P_{pub}^{b \cdot \gamma^*}).$$

we know that

$$e(T^*, P) = e(\bar{H}(Fid^*, i^*, j^*, Cid^*) \cdot (Q_{ID^*}^{m^*}), X_{ID^*}T_{ID^*}(P_{pub}^{b \cdot \gamma^*})).$$

(3)

If $ID^*$ is just the $j^*$-th identity appeared in $L$, then the analysis can be divided into two cases:

1. The tuple $(Fid^*, i^*, j^*, Cid^*)$ appears in the tag-generation for some message $m_{i^*, j^*}$.
2. The tuple $(Fid^*, i^*, j^*, Cid^*)$ does not appear in the tag-generation for any queried messages.

For the case 1), according to the simulation process, we know that

$$\bar{H}(Fid^*, i^*, j^*, Cid^*) = \frac{P^{r_{mi^*, j^*}}}{(P^b \cdot \gamma)^{m_{i^*, j^*}}}.$$  

Therefore,

$$\bar{H}(Fid^*, i^*, j^*, Cid^*) \cdot (Q_{ID^*})^{m^*} = \frac{P^{r_{mi^*, j^*}}}{(P^b \cdot \gamma)^{m_{i^*, j^*}}} \cdot (P^b \cdot \gamma)^{m^*} = P^{r_{mi^*, j^*}} \cdot (P^b \cdot \gamma)^{m^* - m_{i^*, j^*}}.$$  

Note that, $m^* \neq m_{i^*, j^*}$. This holds for the reason that, otherwise, it will result in $T^* = T_{i^*, j^*}$. Hence, combining with (3), we know that

$$e(T^*, P) = e(P^{r_{mi^*, j^*}} \cdot (P^b \cdot \gamma)^{m^* - m_{i^*, j^*}} \cdot X_{ID^*}T_{ID^*}(P_{pub}^{b \cdot \gamma^*}), \gamma \cdot T_{ID^*}(P_{pub}^{b \cdot \gamma^*})).$$
from which $B_1$ can easily obtain the solution of the CDH problem.

For the second case, we know that the simulation of $\widetilde{H}$ is by lazy-sampling and hence $\widetilde{H}(\text{Fid}^*, i^*, j^*, \text{Cid}_r) = P^*$ for some randomly chosen $r^* \in \mathbb{Z}_q$. As a result, combining with (3), we know that

$$e(T^*, P) = e(P^* \cdot (\text{Fid}^* \cdot \text{Cid}_r^*)^{m^*}, X_{ID^*} \cdot T_{ID^*} \cdot (P_{pub}^{b_{ID^*}})^\gamma) = e(P^* \cdot (\text{Fid}^* \cdot \text{Cid}_r^*)^{m^*}, \gamma \cdot T_{ID^*} \cdot (P_{pub}^{b_{ID^*}})^\gamma),$$

from which $B_1$ can also easily obtain the solution of the CDH problem. This ends the proof of Lemma 1.

**Lemma 2:** If the CDH assumption holds in the group $G_1$ and $h, H, \widetilde{H}$ are modeled as random oracles, then, for any PPT Type II adversary $A_{II}$, the probability that it can win the Game III applied to CL-MCMC-PDP protocol is negligible.

**Proof:** Consider an adversary $B_{II}$ who attacks on the CDH assumption and uses $A_{II}$ as a subroutine. Here, assume that $A_{II}$ queries $p$ different identities in the whole process. When given the tuple $(P, \Theta, \gamma) = (P, \theta^a, \theta^b) \in G_1 \times G_1$, $B_{II}$ needs to compute and output the group element $P_{pub}^\gamma$. Now, randomly choose $j^* \in [p]$ as the guess of user’s identity outputted by $A$ in the final phase, and simulate the environment for $A_{II}$ as follows.

- **Phase-II-1.** Choose another group $G_2$, which has the same order $q$ as $G_1$, and define $e$ as the bilinear map from $G_1 \times G_1$ to $G_2$. Then set $P_{pub} = \gamma$ and give the public parameters $(q, P, G_1, G_2, e, P_{pub})$ to $A_{II}$. Next, choose a symmetric encryption key $K$. The hash functions are simulated by $B_{II}$. Finally, initialize an empty list $L$.

- **Phase-II-2.** $B_{II}$ answers $A_{II}$’s queries as follows.

  - **Hash Queries.** The simulations of $h, H, \widetilde{H}$ are same as those in the construction of $B_{II}$ except that, in the generation of user’s partial private key, the answer to $h$-oracle will be different, which will be described in the next step.

  - **Partial-Private-Key-Query.** $A_{II}$ submits $ID$ to $B_{II}$, who randomly chooses $s_{ID}, h_{ID}$ from $\mathbb{Z}_q$ and computes $T_{ID} := P_{ID} / \gamma_{ID}$. Here, $B_{II}$ potentially defines $h(ID, T_{ID})$ as $h_{ID}$. Return $PP_{ID} = (T_{ID}, s_{ID})$ to $A_{II}$ and store $(ID, PP_{ID})$ to $L$.

  - **Private-Key-Query.** $A_{II}$ submits $ID$ to $B_{II}$, who will first check if $PP_{ID}$ exists in $L$. If it is, recover it. Otherwise, make a partial-private-key-query. Then randomly choose $x_{ID}$ from $\mathbb{Z}_q$ and set $sk_{ID}$ as $(PP_{ID}, x_{ID})$. Finally, update the item $(ID, PP_{ID}, x_{ID}, sk_{ID})$ in the list $L$ and return $sk_{ID}$ to $A_{II}$.

  - **Public-Key-Query.** The adversary $A_{II}$ submits $ID$ to $B_{II}$, who checks if $sk_{ID}$ exists in $L$. If it is, recover it. Otherwise, generate it as the above step. Then compute $X_{ID} = P_{ID}^{x_{ID}}$ and set $pk_{ID}$ as $(T_{ID}, X_{ID})$. Give $pk_{ID}$ to $A_{II}$ and update $ID$’s public key as $pk_{ID}$ in $L$.

- **Replace-Public-Key-Query.** $A_{II}$ submits $(ID, pk_{ID}, pk'_{ID})$ to $B_{II}$ in order to replace $pk_{ID}$ with $pk'_{ID}$. Then $B_{II}$ updates $ID$’s public key as $pk'_{ID}$.

- **Tag-Generation-Query.** The simulation of this oracle is same as the process in $B_{II}$ except for the computation of $T_{i,j}$. More precisely, The tag of $m_{i,j}$ is

$$T_{i,j} = (P_{ID}^{\gamma_{m_{i,j}}}, (P_{ID}^{\gamma_{m_{i,j}}}, (T_{ID} \cdot \gamma_{ID}))^{\gamma_{m_{i,j}}}$$

- **Transcript-Query.** The answer to this oracle is also normal since $B_{II}$ can simulate the tags for any data file submitted by $A_{II}$.

- **Phase-II-3.** Finally, $B_{II}$ outputs a forgery $(m^*, T^*)$ with respect to $(ID^*, pk_{ID'}, \text{Fid}^*, i^*, j^*, \text{Cid}_r)$.

From

$$1 \leftarrow \text{TagVer}(ID^*, pk_{ID'}, \text{Fid}^*, i^*, j^*, \text{Cid}_r, m^*, T^*),$$

we know that

$$e(T^*, P) = e(h(ID^*, i^*, j^*, \text{Cid}_r), (Q_{ID^*})^{m^*}, X_{ID^*} \cdot T_{ID^*} \cdot (P_{pub}^{b_{ID^*}})^\gamma) = e(h(ID^*, i^*, j^*, \text{Cid}_r), (Q_{ID^*})^{m^*}, X_{ID^*} \cdot T_{ID^*} \cdot \gamma_{ID^*})$$

Performing a similar analysis as Lemma 1, we know that $B_{II}$ can obtain the solution of the CDH problem. This ends the proof of Lemma 2.

**Lemma 3:** If the CDH assumption holds in the group $G_1$ and $h, H, \widetilde{H}$ are modeled as random oracles, then, for any PPT Type III adversary $A_{III}$, the probability that it can win the Game III applied to CL-MCMC-PDP protocol is negligible.

**Proof:** Consider an adversary $B_{III}$ who attacks on the CDH assumption and uses $A_{III}$ as a subroutine. The construction of $B_{III}$ is same as $B_{II}$ except for the Transcript-Query. In particular, $B_{III}$ runs $chal \leftarrow \text{Chal}(\ell)$ and gives it to $A_{III}$, who will compute and aggregate the the proof according to chal.

In the final phase, $A_{III}$ outputs a tuple $(ID^*, pk^*, \text{Fid}^*, \Gamma^*)$ based on $B_{III}$’s challenge $chal = \{(v_i, a_i)\}_{i=1}^\ell$, where $\text{Fid}^*$ is a filename of a previously queried data file $F$ to the tag-generation oracle. Without loss of generality, assume that the first $\mu$ copies $F_1, F_2, \ldots, F_\mu$ is stored on $A_{III}$. Thus, $\text{Cid}_1 = \text{Cid}_2 = \cdots = \text{Cid}_\mu := \text{Cid}$. Parse $\Gamma^*$ as $(\sigma^*, M^*)$, which satisfies

$$e(\sigma^*, P) = e(\prod_{i=1}^\ell \tilde{H}^{\sigma^*}(\text{Fid}^*, i, v_i, \text{Cid}^*))$$

$$\mu$$

$$(Q_{ID^*})^{M^*}, \tilde{P}_{ID^*}$$

(4)

Let $\Gamma^* = (\sigma^*, M^*)$ be the expected response (i.e. the one that would have been obtained from an honest prover), which

$\mu$

Here, we remark that other CSPs honestly generate their proofs.
should also satisfy
\[ e(\sigma', P) = e\left(\prod_{i=1}^{\mu} \prod_{\tau=1}^{\ell} H^{s_{\tau}}(F_{i, \tau}, \tilde{v}_{\tau}, \tilde{C}_{i}) \cdot (Q_{ID'}^{M'}) \cdot P_{ID'}^{M'} \right). \]

Split \( M' \) and \( M^* \) as
\[ \sum_{i=1}^{\mu} \sum_{\tau=1}^{\ell} a_{i, \tau} m_{i, \tau} \mod q, \quad \sum_{i=1}^{\mu} \sum_{\tau=1}^{\ell} a_{i, \tau} m^*_{i, \tau} \mod q, \]
respectively. If for all \( \tau, m_{i, \tau} = m^*_{i, \tau}, \) then \( M' = M^*. \) Thus, combining (4) with (5), it holds that
\[ e(\sigma'/\sigma, P) = e\left( (Q_{ID'}^{M'-M^*}) \cdot P_{ID'} \right). \]

Hence, we have
\[ e(\sigma'/\sigma, P) = e\left( (Q_{ID'}^{M'-M^*}) \cdot X_{ID'} \cdot T_{ID'} \cdot (P_{pub})^{h_{ID'}{\sigma'}} \right) = e\left( (P^{\beta} \cdot \gamma^{\theta})^{M'-M^*} \cdot X_{ID'} \cdot T_{ID'} \cdot Y^{h_{ID'}{\sigma'}} \right), \]
from which \( \mathcal{A}_{II} \) can easily obtain the solution of the CDH problem. This ends the proof of Lemma 3.

Putting all the above facts together, we know that the main theorem holds.

IV. DISCUSSIONS ON OTHER PROPERTIES

The correctness and security of our proposed CL-MCMC-PDP protocol have been proved in Section III. Now, in this section, we discuss other properties that our protocol satisfies.

- **Blockless.** From the description of our CL-MCMC-PDP protocol, we can see that, after sending the challenge message, TPA receives the returned proof \( \Gamma, \) which is an aggregation of the proofs \( \Gamma_i's \) generated by all the CSPs. Each \( \Gamma_i \) only consists of the tuple of \( (\sigma_i, M_i) \in (\mathcal{G}_1, Z_q^*), \) and the aggregated \( \Gamma \) also has the same form. Hence, it is not needed to return all the original data blocks to check their integrity.

- **Public Verifiability.** From the description of the algorithm \texttt{Verify}, we know that it does not need the DO’s private key as input. Instead, only DO’s identity, public key, challenge message, filename, and the identities of CSPs are needed. All of them can be publicly known and hence our protocol satisfies the public verifiability.

- **Privacy Preserving.** For a curious TPA, it will obtain the returned proof \( \Gamma = (\sigma, M), \) in which \( M \) is a linear combination of the messages \( m_{i,j}’s. \) However, recalling that the generation process of these messages, we know that \( m_{i,j} = E_k(i||m_j). \) Here, \( E \) is a symmetric encryption scheme, \( K \) is the corresponding key, and \( m_j \) is the \( j \)-th block of original data file. Since we use a secure encryption technique to “mask” the information of \( m_j’s, \) the returned part \( M \) does not leak any useful information of DO’s original file.

- **Certificateless.** In our definition of security model and the analysis of our proposed protocol, the Type II adversary models a general adversary who is allowed to replace DO’s public key by any other values it chooses. Our scheme is secure against this kind of adversary. Thus, the certificate to describe the relationship of DO’s identity and its public key in our proposed protocol is not needed.

V. PERFORMANCE ANALYSIS

In this section, we analyze the performances of our proposed CL-MCMC-PDP protocol. Concretely, we compare it with other two previous PDP protocols in [12], [26] in terms of communication costs, computational costs and securities. Note that, Li et al.’s protocol uses a traditional signature scheme as a building block, which result in the whole protocol becoming a non-identity-based one. A solution is to change this traditional signature scheme into an IB-signature one such as Galindo’s lightweight scheme in [7]. For presenting a fair comparison, we use Galindo’s IB-signature to replace their standard signature scheme and describe the comparison process in the following parts.

**Communication Costs:** In our protocol, the communication process mainly includes those between DO and KGC, DO to CO/CSPs, CO/CSPs to TPA. Hence, we compute the communication costs for all of them as follows. The total comparisons of communication costs are listed in Table 2.

- **KGCtoDO:** This process considers the communication cost from KGC to DO. More precisely, when getting DO’s identity \( ID, \) the KGC will compute the partial private key \( P_{ID} \) for \( ID. \) Recall that \( P_{ID} = (T_{ID}, s_{ID}) \in (\mathcal{G}_1, Z_q^*). \) Therefore, the required communication bandwidth equals to \( |P_{ID}| = |T_{ID}| + |s_{ID}| = |\mathcal{G}_1| + |Z_q^*|. \) In [12], Li et al.’s protocol is an IB-MCMC-PDP. From their construction, we know that the communication cost from KGC to DO is the length of user’s private key \( |sk_{ID}| = 2|\mathcal{G}_1| + |Z_q^*|. \)

Since Zhu et al.’s protocol is a traditional certificate-based multi-cloud protocol, there does not exist the entity of KGC. Thus, the communication cost from KGC to DO in [26] also does not exist, which is denoted by “--” in Table 2.

- **DOToCO:** This process consider the communication cost from DO to CO. Here, we consider the data file \( F_i \) that is split into \( n \) blocks \( m_{i,1}, m_{i,2}, \ldots, m_{i,n}. \) Each block has one sector. For each block, the generated tag \( T_{i,j} \) in our protocol is only one element in \( \mathcal{G}_1. \) Thus, the communication overhead for our protocol equals to \( n \cdot |\mathcal{G}_1|. \) Similarly, we can compute the communication overheads for the protocols in [12] and [26], which are \( n \cdot |\mathcal{G}_1| + |Sig| \) and \((2n+1) \cdot |\mathcal{G}_1|\), respectively, where \( |Sig| \) is the used signature in [12]. If it is replaced by Galindo’s IB-signature, then \( |Sig| = 2|\mathcal{G}_1| + |Z_q^*|. \)
TABLE 2. The total comparisons of communication costs.

| Protocols     | $[26]$                  | DOCoCO                  | CotoTPA                  |
|---------------|-------------------------|-------------------------|-------------------------|
|               | $[26]$                  | $2 | G_1 | + | Z_q^p$ | $2 | G_1 | + | Z_q^p$ | $2 | G_1 | + | Z_q^p$ |
| [12]          | $(2 + n) \cdot | G_1 |$              | $(2 + n) \cdot | G_1 |$              | $(2 + n) \cdot | G_1 |$              |
| Our Protocol  | $| G_1 | + | Z_q^p$ | $| G_1 |$              | $| G_1 | + | Z_q^p$ | $| G_1 |$              |

TABLE 3. The total comparisons of computational costs.

| Protocols     | [26]                  | [12]                     | Our Protocol             |
|---------------|------------------------|--------------------------|--------------------------|
| Key-Generation| $2 \cdot T_{exp}$      | $T_{exp}$                | $2 \cdot T_{exp}$        |
| Single-Tag-Generation| $(\ell + 1) \cdot (T_{exp} + T_{mul}) + T_p$ | $(\ell + 1) \cdot (T_{exp} + T_{mul}) + T_p$ | $(\ell + 1) \cdot (T_{exp} + T_{mul}) + T_p$ |
| Aggregated Proof Generation| $r \cdot T_{exp} + r \cdot T_{mul} + T_p$ | $r \cdot T_{exp} + r \cdot T_{mul} + T_p$ | $r \cdot T_{exp} + r \cdot T_{mul} + 2 \cdot T_p$ |
| Verification  | $(N \cdot \ell + 1) \cdot T_{exp} + 3 \cdot T_p$ | $(N \cdot \ell + 4) \cdot T_{exp} + (N \cdot \ell + 1) \cdot T_{mul} + 3 \cdot T_p$ | $(N \cdot \ell + 1) \cdot (T_{exp} + T_{mul}) + 2 \cdot T_p$ |

- **CotoTPA:** This process considers the communication cost from the CO to verifier. In our protocol, when TPA submitting the challenge message *chal*, CO will return the aggregated proof $\Gamma = (\sigma, M) \in (G_1, Z_q^p)$. Hence, the cost is $|G_1| + |Z_q^p|$. For the protocol [12], the returned proof is $\Gamma = (\sigma, M, R, u, Sig)$. Thus, the cost equals to $|\Gamma| = |\sigma| + |M| + |R| + |u| + |Sig| = 3|G_1| + |Z_q^p| + |Sig|$. If the signature is Galindo’s IB-signature, then the total cost is $5|G_1| + 2|Z_q^p|$. We can also compute the communication cost from CO to TPA for the protocol in [26], which equals to $2|G_1| + |Z_q^p|$. The total comparisons of communication costs are listed in Table 3.

**Computational Costs:** Denote by $T_p$, $T_{exp}$, and $T_{mul}$ the operations of pairing, exponentiation and multiplication in $G_1$, respectively. Compared with these three operations, other ones, such as hash, addition and multiplication in $Z_q^p$ can be omitted because their time-consumptions are nearly negligible. Here, we consider $N$ copies for the initial data file that will be stored on CSPs. For each CSP, the number of stored copies is $|CT_i|$ ($1 \leq i \leq r$). For TPA, it will challenge $\ell$ blocks.

In our protocol, to generate the partial private key for a user, the KGC will cost one $T_{exp}$. The user needs one $T_{exp}$ to generate its public key. Hence, the computational cost for user generating its full private key and public key equals to $2 \cdot T_{exp}$. Similarly, in [12] and [26], the costs for generating user’s private keys are one $T_{exp}$ and $2 \cdot T_{exp}$, respectively.

To compute all the tags for the $N$ copies, our protocol will totally cost $N \cdot n \cdot (T_{mul} + 2 \cdot T_{exp})$. The same process for Li et al.’s and Zhu et al.’s protocols will cost

$$N \cdot n \cdot (4 \cdot T_{exp} + 2 \cdot T_{mul}),$$

and

$$N \cdot n \cdot (T_{mul} + 3 \cdot T_{exp}),$$

respectively.

For the proof-generation process, the $i$-th CSP runs **ProofGen** to construct a single proof for one copy, which will cost $\ell \cdot (T_{exp} + T_{mul})$. If $|CT_i|$ copies are stored in this cloud, then the cost is

$$|CT_i| \cdot \ell \cdot (T_{exp} + T_{mul}).$$

The CO will aggregate the $r$ returned proof, which will cost $r \cdot T_{mul}$.

Those costs are equal to that of [12]. In [26], single cloud generating its proof costs

$$(\ell + 1) \cdot (T_{exp} + T_{mul}) + T_p.$$

The CO obtaining the aggregated proof costs

$$r \cdot T_{mul} + r \cdot T_{mul} + 2 \cdot T_{exp}.$$  

Finally, the TPA verifies the returned proof in our protocol needs

$$(N \cdot \ell + 1)(T_{exp} + T_{mul}) + 2 \cdot T_p.$$  

The verifications in [12] and [26] cost

$$(N \cdot \ell + 4) T_{exp} + (N \cdot \ell + 1) T_{mul} + 3 \cdot T_p,$$

and

$$(N \cdot \ell + 1) T_{exp} + 3 \cdot T_p,$$

respectively.

The total comparisons of the computational costs are listed in Table 3.

**Experimental Results:** In order to give an intuitive comparison of the three protocols. Here, we implement them in a laptop with Intel Core i5-6200U CPU @2.3GHz and 2GB RAM running Ubuntu 14.04 LTS 64-bit and Python 3.4. The experiments are within the framework of “Charm” [1] and we choose the 512-bit SS elliptic curve from pairing-based cryptography (PBC) library [14] as the basis of these protocols.

We evaluate the computational costs of tag-generation, proof-generation, and verification. In particular, we first choose a 10M data file and generate $N = 20$ copies files. Each copy consists of 600 blocks. We change the block count from 1000 to 6000 with an increment of 1000 in each test. The time-consumption of each instance is computed by repeating 100 times and taken the average time. The time-consumptions for tag-generation are depicted in Fig. 2.
TABLE 4. Comparisons on other aspects.

| Protocols       | Multi-Copy | Multi-Cloud | Certificateless | Key-Escrow Security |
|-----------------|------------|-------------|-----------------|---------------------|
| [26]            | ×          | ✓           | ×               | –                   |
| [12]            | ✓          | ✓           | ✓               | ×                   |
| Our Protocol    | ✓          | ✓           | ✓               | ✓                   |

FIGURE 2. Time-consumptions for tag-generations.

The returned proof $\Gamma$ is run by CO, who aggregates the proofs generated from CSPs based on TPA's challenge message. Here, let the number of challenged blocks changes from $\ell = 30$ to $\ell = 180$ with an increment of 30. Each CSP stores 4 copies and hence $r = 5$. Then the time-consumptions for generating the responses $\Gamma$ are presented in Fig. 3.

FIGURE 3. Time-consumptions for proof-generations.

Finally, we implement the verification algorithm $\text{Verify}$, which is used by TPA to check the validity of the proof returned by CO. For different numbers of challenged blocks, the experiment results are depicted in Fig. 4.

FIGURE 4. Time-consumptions for verifications.

VI. CONCLUSION

In this paper, we consider and design an efficient CL-MCMC-PDP protocol. In particular, we introduce the concrete security model for it and based on the famous CDH assumption construct a proven secure protocol. The performance analysis shows that our protocol is rather practical.

REFERENCES

[1] J. A. Akinyele, C. Garman, I. Miers, M. W. Pagano, M. Rushanan, M. Green, and A. D. Rubin, “Charm: A framework for rapidly prototyping cryptosystems,” J. Cryptograph. Eng., vol. 3, no. 2, pp. 111–128, Jun. 2013.
[2] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, “Provable data possession at untrusted stores,” in Proc. 14th ACM Conf. Comput. Commun. Secur. (CCS), 2007, pp. 598–609.
[3] S. S. Al-Riyami and K. Paterson, “Certificateless public key cryptography,” in Proc. ASIACRYPT, 2003, pp. 452–473.
[4] A. F. Barsoum and M. A. Hasan, “Provable multicopy dynamic data possession in cloud computing systems,” IEEE Trans. Inf. Forensics Security, vol. 10, no. 3, pp. 485–497, Mar. 2015.
[5] J. Chang, Y. Ji, R. Xue, and M. Xu, “General transformations from single-generation to multi-generation for homomorphic message authentication schemes in network coding,” Future Gener. Comput. Syst., vol. 91, pp. 416–425, Feb. 2019.
[6] J. Chang, H. Wang, F. Wang, A. Zhang, and Y. Ji, “RKA security for identity-based signature scheme,” IEEE Access, vol. 8, pp. 17833–17841, 2020.
[7] D. Galindo and F. D. Garcia, “A Schnorr-like lightweight identity-based signature scheme,” in Proc. AFRICACRYPT (Lecture Notes in Computer Science), vol. 5580, 2009, pp. 135–148.
[8] Z. Hao, S. Zhong, and N. Yu, “A privacy-preserving remote data integrity checking protocol with data dynamics and public verifiability,” IEEE Trans. Knowl. Data Eng., vol. 23, no. 9, pp. 1432–1437, Sep. 2011.
[9] D. He, N. Kumar, H. Wang, L. Wang, and K.-K. R. Choo, “Privacy-preserving certificateless provable data possession scheme for big data storage on cloud,” Appl. Math. Comput., vol. 314, pp. 31–43, Dec. 2017.
[10] D. He, S. Zeadally, and L. Wu, “Certificateless public auditing scheme for cloud-assisted wireless body area networks,” IEEE Syst. J., vol. 12, no. 1, pp. 64–73, Mar. 2018, doi: 10.1109/JSYST.2015.2425620.
[11] A. Juels and B. S. Kaliski, “PORs: Proofs of retrievability for large files,” in Proc. 14th ACM Conf. Comput. Commun. Secur. (CCS), 2007, pp. 584–597.

[12] J. Li, H. Yan, and Y. Zhang, “Efficient identity-based provable multi-copy data possession in multi-cloud storage,” IEEE Trans. Cloud Comput., early access, Jul. 16, 2019, doi: 10.1109/TCC.2019.2929045.

[13] D. Kim and I. R. Jeong, “Certificateless public auditing protocol with constant verification time,” in Secur. Commun. Netw., vol. 2017, Jan. 2017, Art. no. 6758618.

[14] B. Lynn. The Standard Pairing Based Crypto Library, PBC-0.5.14. Accessed: Jun. 14, 2013. [Online]. Available: http://crypto.standford.edu/pbc

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

[16] A. Shamir, “Identity-based cryptosystems and signature schemes,” in Proc. CRYPTO (Lecture Notes in Computer Science), vol. 196. Heidelberg, Germany: Springer, 1985, pp. 47–53.

[17] C. Wang, S. Chow, and Q. Wang, “Privacy-preserving public auditing for secure cloud storage,” IEEE Trans. Comput., vol. 62, no. 2, pp. 362–375, Feb. 2013.

[18] H. Wang, D. He, J. Yu, and Z. Wang, “Incentive and unconditionally anonymous identity-based public provable data possession,” IEEE Trans. Services Comput., vol. 12, no. 5, pp. 824–835, Sep. 2019.

[19] B. Wang, B. Li, H. Li, and F. Li, “Certificateless public auditing for data integrity in the cloud,” in Proc. IEEE Conf. Commun. Netw. Secur. (CNS), Oct. 2013, pp. 136–144.

[20] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, “Enabling public verifiability and data dynamics for storage security in cloud computing,” in Proc. ESORICS. Berlin, Germany: Springer-Verlag, 2009, pp. 335–370.

[21] C. Wang, Q. Wang, K. Ren, and W. Lou, “Ensuring data storage security in cloud computing,” in Proc. 17th Int. Workshop Qual. Service (IWQoS), Jul. 2009, pp. 1–9.

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

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

[24] Y. Yu, M. H. Au, G. Atieniese, X. Huang, W. Susilo, Y. Dai, and G. Min, “Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage,” IEEE Trans. Inf. Forensics Security, vol. 12, no. 4, pp. 767–778, Apr. 2017.

[25] R. Zhang, H. Ma, Y. Lu, and Y. Li, “Provably secure cloud storage for mobile networks with less computation and smaller overhead,” Sci. China Inf. Sci., vol. 60, no. 12, Dec. 2017, Art. no. 122104.

[26] Y. Zhu, H. Hu, G.-J. Ahn, and M. Yu, “Cooperative provable data possession for integrity verification in multicloudbased storage,” IEEE Trans. Parallel Distrib. Syst., vol. 23, no. 12, pp. 2231–2244, Dec. 2012.

[27] Y. Zhang, J. Yu, R. Hao, C. Wang, and K. Ren, “Enabling efficient user revocation in identity-based cloud storage auditing for shared big data,” IEEE Trans. Dependable Secure Comput., vol. 17, no. 3, pp. 608–619, May/Jun. 2020, doi: 10.1109/TDSC.2018.2829880.