IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 21, 2019, accepted December 10, 2019, date of publication December 19, 2019,
date of current version January 6, 2020.

Digital Object Identifier 10.1109/ACCESS.2019.2960853

Lightweight Certificate-Based Public/Private Auditing Scheme Based on Bilinear Pairing for Cloud Storage

FENG WANG[1,2], LI XU[1], (Member, IEEE),
KIM-KWANG RAYMOND CHOO[3,4], (Senior Member, IEEE),
YUEXIN ZHANG[5], HUAQUN WANG[6], AND JIGUO LI[1]

[1]College of Mathematics and Informatics, Fujian Provincial Key Laboratory of Network Security and Cryptology, Fujian Normal University, Fuzhou 350117, China
[2]College of Mathematics and Physics, Fujian Provincial Key Laboratory of Big Data Mining and Applications, Fujian University of Technology, Fuzhou 350118, China
[3]Department of Information Systems and Cyber Security, The University of Texas at San Antonio, San Antonio, TX 78249-0631, USA
[4]Department of Electrical and Computer Engineering, The University of Texas at San Antonio, San Antonio, TX 78249-0631, USA
[5]Department of Computer Science and Software Engineering, Swinburne University of Technology, Hawthorn, VIC 3122, Australia
[6]Jiangsu Key Laboratory of Big Data Security and Intelligent Processing, School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210023, China

Corresponding author: Li Xu (xuli@fjnu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant U1905211, Grant 61771140, Grant U1405255, and Grant 61702100, in part by the Fujian Province University Industry Cooperation of Major Science and Technology Project under Grant 2017H6005, and in part by the China Postdoctoral Science Foundation under Grant 2017M612107 and Grant 2018T110636. The work of K.-K. R. Choo was supported by the Cloud Technology Endowed Professorship.

ABSTRACT Ensuring the integrity of outsourced data is one of several functional requirements in cloud storage services, although designing an effective and secure public auditing scheme remains an ongoing research challenge. For example, a number of existing schemes are designed based on symmetric bilinear pairing, which are computationally expensive. While certificate-based auditing schemes can simplify certificate management and mitigate key escrow concern, such schemes are not popular in the literature. Therefore, we propose a lightweight certificate-based public/private auditing scheme based on asymmetric bilinear pairing for cloud storage. Specifically, we describe a new method of designing lightweight auditing schemes. Additionally, our proposed scheme is proven to be secure in the random oracle model. We then implement our scheme by using type D curve of the pairing-based library. The findings demonstrate that our auditing scheme significantly reduces client's computation cost in the tag-generation (TagGen) phase, particularly in comparison to several other competing schemes. For example, when the block size is 10 KB, our on-line computation cost in the TagGen phase takes only 0.45 seconds, while it requires at least 62.83 seconds in other schemes. Furthermore, the findings show that when the number of blocks is fixed, the on-line computation cost in our TagGen phase is constant despite varying file sizes.

INDEX TERMS Certificate-based auditing, public/private auditing, lightweight, provable data possession, cloud storage.

I. INTRODUCTION

Cloud storage is now a deeply entrenched practice in our society. However, there remains a number of issues and challenges that have not been resolved. For example, how do users ensure the integrity of the outsourced data, particularly in cases where the users may not have the complete copy of the data on their local systems and devices? Potential solutions that have been presented in the literature include provable data possession (PDP) model [1] and proof of retrievability (POR) model [2], [3]. In the PDP model, if the outsourced data successfully passes an audit without the user having to retrieve the data, then the user is assured of the data's integrity. For the POR model, the user can extract the data only when the outsourced data successfully passes an audit.

The associate editor coordinating the review of this manuscript and approving it for publication was Gautam Srivastava.

Generally speaking, there are two main phase in data integrity auditing, namely: TagGen and Proof phases. In the TagGen phase, the user generates the tags associated with the data, and uploads both data and tags to the cloud server. In the Proof phase, the auditor sponsors a challenge, the cloud server returns a response, and the auditor determines integrity of the outsourced data by verifying the validity of the response.

Furthermore, data integrity auditing can be broadly categorized into public auditing and private auditing, depending on whether public verifiability is supported. Private auditing schemes, such as those presented in [2], [4], are more efficient. However, unlike public auditing schemes, in private auditing schemes only the owner can audit the data, and the judge cannot intervene when a dispute occurs. Therefore, public auditing schemes are generally more practical.

Many public auditing schemes [2], [5]–[8] are designed based on conventional public key infrastructure (PKI)-based cryptography. However, a key limitation of PKI-based auditing schemes is expensive certificate management, since the schemes require the use of certificates generated by some trusted third-party to bind the user's identity and public key. There have been attempts to design schemes that do not require such expensive certificate management, such as the identity-based PDP scheme of Wang *et al.* [9]. There have also been a large number of ID-based auditing schemes in the literature, such as those presented in [10]–[15] for different application scenarios.

Schemes based on certificateless cryptography [17] and certificate-based cryptography [18], [19] have also been presented in the literature, designed to solve the key escrow problem [16] inherent in ID-based cryptography. Examples of certificateless auditing schemes include those of [20]–[24]. Certificate-based auditing schemes, on the other hand, appear to be significantly less popular. We are only aware of one such scheme [25], which focuses only on private auditing. However, this particular scheme's efficiency is low.

While public auditing schemes allow third-party intervention (i.e., by a judge) in the case of a dispute, public auditing schemes are generally inefficient. This significantly limits its deployment in a real-world application and hence, interest in, such schemes. In other words, ''How do we minimize computation, communication, and storage overheads in public auditing schemes?'' is an open research issue [26], [27]. In order to minimize the overheads associated with storage and/or communication, Shacham and Waters [2] proposed compact proofs of retrievability scheme to minimize storage space for tags. The approach of Shacham and Waters [2] also inspired the design of several other schemes in the literature [7], [12]–[14], [20]. In addition, to reduce the computation cost, most existing schemes support batch auditing and third-party auditing. A number of schemes achieve constant verification time in the proof phase [11], [20].

However, there are only a small number of schemes that had reduced computation cost in the tag-generation (TagGen) phase. According to prior research [1], [14], if 1% of the total blocks are tampered with, then the auditor can detect the misconduct with a 99% possibility by challenging 4.6% of the total blocks. Thus, the computation cost of the Proof phase is smaller than that of the TagGen phase. Furthermore, most user devices are resource-constrained mobile devices (e.g., Android and iOS devices) and improving efficiency [28] is an important research direction for cryptography researchers. Therefore, it is more intuitive to minimize the computation overhead during the TagGen phase. One could attempt to reduce the computation overhead in the TagGen phase by using a trusted third-party. For example, Li *et al.* [29] used a cloud audit server to generate tags of the data, on behalf of the user, before uploading to the cloud server. However, in this scheme, the cloud audit server can potentially reveal the user's data and secret key. Wang *et al.* [10] used a proxy to facilitate the user to generate tags, and the user only need to encrypt the data prior to sending it to the proxy to ensure data privacy. Shen *et al.* [30] proposed a lightweight auditing scheme by using a third-party medium to help the user to generate tags. In this scheme, the user blinds the data before sending to the third-party medium to ensure data privacy. Han *et al.* [31] proposed a lightweight auditing scheme without bilinear pairings for smart city. Their scheme allows a third-party auditor to generate tags for the data on behalf of the user, and preserve data privacy from the third-party auditor. Furthermore, all trusted third parties (i.e., cloud audit server, proxy, or some third-party entity) are assumed to be honest, in the sense that they will not corrupt the data, in the schemes of [10], [29]–[31]. In other words, in these schemes the user will not be able to detect the third-party's misbehavior. Therefore, we need to be able to minimize computation cost of the TagGen phase without involving a third-party.

We also observe that in the literature, most pairing-based data integrity auditing schemes, such as those of [2], [11], [12], [14], [22], [23], are based on symmetric bilinear pairing, despite the acknowledgement that asymmetric bilinear pairing is more suited for data integrity auditing schemes [14].

The above observations motivate our focus in this paper. Specifically, we design a certificate-based auditing scheme that has a reduced computation cost in the TagGen phase without involving a third party. We are also inspired by Zhang et al.'s certificate-based signature scheme [32], and hence our proposed scheme is also certificate-based. We leverage the fact that a user generally has more information than the auditor, and therefore divide the Proof phase into PublicVerify phase and PrivateVerify phase. Therefore, our scheme is a public/private auditing scheme. We also acknowledge that most user devices are resource-constrained, and thus ensure that the user side's requirement is relatively lightweight. Using specific information of the user, we manage to reduce users' computation cost in the TagGen phase. Furthermore, the public/private auditing scheme also reduces the computation cost of the user in the Proof phase. In other words, our contribution in this paper is a lightweight certificate-based public/private auditing scheme, based on asymmetric bilinear pairing.

To facilitate understanding of our proposed scheme, we will introduce relevant background materials (e.g., bilinear pairing, co-computational Diffie-Hellman problem, and certificate-based signature) in Section II. We then introduce the public/private auditing model and certificate-based auditing model in Section III, and present our proposed certificate-based public/private auditing scheme in Section IV. We prove the security of the proposed scheme in Section V, and implement our scheme using type D curve of the pairing-based library in Section VI. A comparative summary of the proposed schemes and several others are presented in Section VII. It is shown in this section that our auditing scheme outperforms other schemes, in terms of achieving reduced computation cost for users during TagGen and without involving a third-party. The analysis also shows that when the block size is 10 KB, our on-line computation cost during TagGen is only 0.45 seconds, unlike the other schemes (between 62.83 and 250.29 seconds). Furthermore, if we fix the block numbers, the on-line computation cost during TagGen in our scheme is constant even when the file sizes change. In the last section, we present our discussion and conclusion.

II. PRELIMINARIES

We will now review bilinear pairing [8], [11], [33], [34], co-computational Diffie-Hellman problem [35], pseudo-random function, pseudo-random permutation, certificate-based signature [18], [19], [32], forking lemma [36] in Sections II-A to II-E, respectively.

A. BILINEAR PAIRING

Let q be a prime, G_1, G_2, and G_T be three cyclic groups with order q. Let G_1 and G_2 be written additively with respective generators P_1 and P_2, and G_T be written multiplicatively. The map $e : G_1 \times G_2 \rightarrow G_T$ is a bilinear pairing, only when the following three properties are satisfied.

(1). Bilinearity: For any $a, b \in Z_q^*$, $e(a \cdot P_1, b \cdot P_2) = e(P_1, P_2)^{ab}$ holds.

(2). Non-degeneracy: $e(P_1, P_2) \neq 1$.

(3). Computablity: For any $\bar{P}_1 \in G_1, \bar{P}_2 \in G_2$, there exists a polynomial time algorithm to calculate $e(\bar{P}_1, \bar{P}_2)$ efficiently.

If $G_1 = G_2$, then it is said to be symmetric bilinear pairing; otherwise, it is said to be asymmetric bilinear pairing. Furthermore, asymmetric bilinear pairings can be categorized into whether there is an efficiently computable homomorphism $\Phi : G_2 \rightarrow G_1$, or not.

B. CO-COMPUTATIONAL DIFFIE-HELLMAN PROBLEM

Let G_1, G_2, P_1, P_2, q be the parameters described in Subsection II-A and $Q_1 = a \cdot P_1$, $Q_2 = b \cdot P_2$ be two random elements with $a, b \in Z_q^*$ unknown. The co-computational Diffie-Hellman (co-CDH) problem [35] is to calculate $Q_3 = a \cdot b \cdot P_1$.

At the time of this research, there is no efficient polynomial time algorithm to solve the co-CDH problem. That is to say, let the probability of a polynomial time algorithm solving the co-CDH problem, ε, is negligible.

C. PSEUDO-RANDOM FUNCTION AND PSEUDO-RANDOM PERMUTATION

1) PSEUDO-RANDOM FUNCTION

This algorithm takes $\alpha \in Z_q^*, \beta \in N^*$ as input and returns β different pseudo-random numbers from Z_q^*. We denote this algorithm as $\varphi(\alpha, \beta) : Z_q^* \times N^* \rightarrow (Z_q^*)^\beta$.

2) PSEUDO-RANDOM PERMUTATION

This algorithm takes $\alpha \in Z_q^*, \beta \in N^*, \gamma \in N^*$ as input and returns a sequence with β different pseudo-random numbers from Z_γ^*. We denote this algorithm as $\pi(\alpha, \beta, \gamma) : Z_q^* \times N^* \times N^* \rightarrow (Z_\gamma^*)^\beta$.

D. CERTIFICATE-BASED SIGNATURE

There are three entities and five algorithms in a typical certificate-based signature scheme [18], [19], [32]. The three entities are the signer, the verifier, and the certifier, and the five algorithms are Setup, SignerKeyGen, Certify, CBSign, CBVeri. The certifier executes the Setup algorithm, and outputs the the global parameters and the global secret key. The signer executes SignerKeyGen algorithm, and outputs his/her secret key sk and public key PK. In the Certify algorithm, the certifier generates the certificate $Cert$ using the signer's public key and identity ID, prior to sending it to signer via a secret channel. In the CBSign algorithm, given a message m, the signer computes the signature $\sigma = CBSign(sk, Cert, m)$. In the CBVeri algorithm, given PK, ID, m and σ, the verifier computes the value of $CBVeri(ID, PK, m, \sigma)$. It outputs 1/0 based on whether or not the signature is valid.

E. FORKING LEMMA

The forking lemma [36]. Given only the public data as input, if a probabilistic polynomial time Turing machine can find a valid signature (σ_1, h, σ_2) on message m with a non-negligible probability, then this machine can output another valid signature $(\sigma_1, h', \sigma_2')$ on the same m by replaying itself with the same random tape and a different oracle with a non-negligible probability. Here, h is the hash value of (m, σ_1) and σ_2 is dependent only on σ_1. Note that the forking lemma is appropriate for the security proof of both the signature and auditing schemes, as the auditing scheme's framework is based on signature.

III. MODELS

In this section, we introduce the public/private auditing model, the system model and the security model of the certificate-based auditing.

A. Public/Private Auditing Model

Data integrity auditing is either public auditing or private auditing. In the latter, it is the user's duty to audit the integrity of the outsourced data. Private auditing schemes are more efficient, but the judge cannot intervene when a dispute occurs. This is clearly not desirable in e-commerce applications. In public auditing schemes, both the user and the auditor can audit the integrity of the outsourced data. When a dispute occurs, anyone can intervene as a judge. Therefore, public auditing schemes are more geared for

practical deployment. However, such schemes are generally much less efficient.

In existing public auditing schemes, both users and auditors have the same audit stages. However, the user clearly has more information than the auditor. So, we divide the audit phase into the PrivateVerify and PublicVerify phases. The client executes PrivateVerify, which is more efficient due to the fact that the user has more information. The auditor executes PublicVerify when a dispute occurs or when the user is not available to perform the audit. The PublicVerify phase is executed as the audit phase in existing public auditing schemes. Thus, we refer to such auditing scheme as public/private auditing schemes.

B. SYSTEM MODEL OF CERTIFICATE-BASED PDP

There are four entities and eight algorithms in the certificate-based PDP model. The four entities are the certifier, the cloud server, the user (also commonly referred to client in the literature), and the auditor. The certifier is an independent trusted third-party, whose duty is to produce its own system parameters, and generate client's certificate from his/her identity and public key. The cloud server is an entity with significant storage space and computation resources, and provides storage services for the client. The client is generally resource-constrained, and hence the need to outsource data storage to the cloud server. The auditor is an independent trusted third-party. With the client's authorization, the auditor checks the integrity of the outsourced data and returns the result to the client.

Compared with the other PDP models, our proposed scheme is designed to achieve the benefits in both public and private auditing by dividing the ProofVerify phase into PrivateVerify phase and PublicVerify phase. In order to improve audit efficiency, the client executes the PrivateVerify phase to audit the outsourced data. When a dispute happens or when the client is not available to perform the audit, the auditor is authorized to audit the outsourced data by using PublicVerify. The eight algorithms are Init, ClientKeyGen, Certify, TagGen, Chall, ProofGen, PrivateVerify, and PublicVerify in the certificate-based PDP model. Fig. 1 presents the system model of certificate-based PDP, and the algorithms are also explained in detail as follows.

The Init algorithm is performed by the certifier (see step 1 of Fig. 1). Given the security parameters, it generates the global parameters gp and global secret key gsk. The certifier keeps gsk secret, and publishes gp.

The ClientKeyGen algorithm is performed by the client (see steps 2 and 3 of Fig. 1). Given gp, the client picks his/her secret key sk randomly, and computes the corresponding public key PK. The client keeps his/her sk secret, and sends his/her PK and identity ID to the certifier.

The Certify algorithm is executed by the certifier (see steps 4 and 5 of Fig. 1). Given the client's ID and PK, the certifier generates the certificate $Cert$, and sends $Cert$ to the client via a private channel. Upon receiving $Cert$, the client checks the validity of $Cert$.

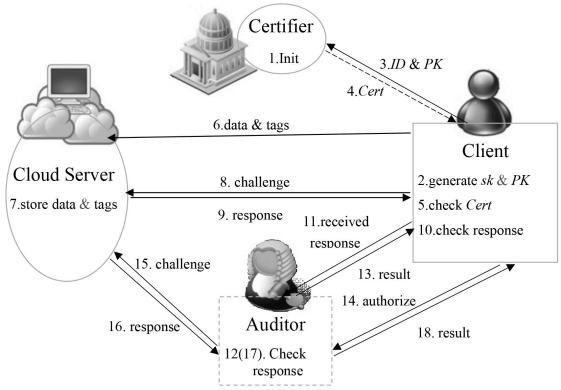

FIGURE 1. System model of certificate-based PDP.

The TagGen algorithm is performed by the client and the cloud server (see steps 6 and 7 of Fig. 1). Given a file F, the client generates the tags by using his/her secret key sk and certificate $Cert$, and sends the data and tags to cloud server. The cloud server stores the data and tags.

The Chall algorithm is performed by the client (see step 8 of Fig. 1). The client sponsors a challenge, and sends it to the cloud server.

The ProofGen algorithm is performed by the cloud server (see step 9 of Fig. 1). Given the challenge, the cloud server responds it.

The PrivateVerify algorithm is performed by the client (see step 10 of Fig. 1). Given the response, the client checks its validity. Upon receiving the response from the cloud sever, the client checks the validity of the response.

The PublicVerify algorithm is performed by the auditor (see steps 11 to 18 of Fig. 1). There are only two cases where this algorithm will be activated. The auditor performs case 1 when a dispute occurs, and case 2 when the client is not available for audit.

Case 1 (see steps 11 to 13). Given the forwarded response from the client, the auditor solves the dispute and returns the result.

Case 2 (see steps 14 to 18). Upon receiving the authorization from the client, the auditor sponsors a challenge, and the cloud server responds it. Then the client checks its validity, and sends the result to client.

C. SECURITY MODEL OF CERTIFICATE-BASED PDP

Similar to certificate-based signature model, there are two adversaries in certificate-based PDP, i.e., the public key replacement adversary and the malicious certifier adversary.

If an adversary has the ability to select clients' identities and replace the public keys at his/her will, but it cannot have access to the global secret key, then we say this adversary is the public key replacement adversary Θ_1.

We say that a certificate-based PDP scheme is (q, ε_1) secure against public key replacement adversary Θ_1, if the probability ε_1 of the adversary Θ_1 winning the following game by making at most q queries is negligible.

The challenger Ψ performs the Setup phase to generate gp, gsk and send gp to Θ_1. Then, Θ_1 adaptively submits

ClientKeyGen queries, Client secret key query, Public key replace queries, Certify queries, and TagGen queries to Ψ and obtains the corresponding value. Here, the queries are described in the proof of Theorem 1 in section 5. Finally, Θ_1 wins the game if he/she successfully forges a valid response $resp^* = \{F_0^*, F_1^*, F_2^*, \ldots, F_{s-1}^*, T^*\}$ for client ID^* on the challenge $chal^*$ under the public key PK^* in the Forgery phase. Here, ID^* has not been submitted to the Certify query, and (ID^*, l, F_l^*) has not been submitted to the TagGen query, where $l \in L^*$, and L^* is generated by $chal^*$.

If an adversary can successfully obtain the global secret key and attempt to impersonate the user, but it cannot replace the public key, then we say this adversary is the malicious certifier adversary.

We say that a certificate-based PDP scheme is (q, ε_2) secure against a malicious certifier adversary Θ_2 if the probability ε_2 of the adversary winning the following game by making at most q queries is negligible.

The challenger Ψ performs the Setup phase to generate gp, gsk and send them to Θ_2. Then, Θ_2 adaptively submits ClientKeyGen queries, Client secret key query, Certify queries, and TagGen queries to Ψ and obtains the corresponding value. Here, the queries are described in the proof of Theorem 2 in Section 5. Finally, Θ_2 wins the game if he/she successfully forges a valid response $resp^{\#} = \{F_0^{\#}, F_1^{\#}, F_2^{\#}, \ldots, F_{s-1}^{\#}, T^{\#}\}$ for client $ID^{\#}$ on the challenge $chal^{\#}$ under the public key $PK^{\#}$ in the Forgery phase. Here, $PK^{\#}$ is the output of ClientKeyGen query, and $(ID^{\#}, l, F_l^{\#})$ has not been submitted to the TagGen query, where $l \in L^{\#}$, and $L^{\#}$ is generated by $chal^{\#}$.

If a certificate-based PDP scheme can resist both the public key replacement adversary and the malicious certifier adversary, then we say the scheme is secure.

IV. OUR PROPOSED CERTIFICATE-BASED PDP SCHEME

Our proposed scheme comprises Init, ClientKeyGen, Certify, TagGen, Chall, ProofGen, PrivateVerify, and PublicVerify. These eight algorithms are also described below.

A. INIT

This algorithm is performed by the certifier. Let parameters $G_1, G_2, G_T, \Phi, P_1, P_2, q, e$ with $P_1 = \Phi(P_2)$, be described in Subsections II-A. Let $h(\cdot) : \{0,1\}^* \to Z_q^*$, $h'(\cdot) : \{0,1\}^* \to Z_q^*$, $H(\cdot) : \{0,1\}^* \to G_1$ be three hash functions, and the first two be ordinary hash functions, and the last one be the map to point hash function. Let $\varphi(\alpha, \beta) : Z_q^* \times N^* \to (Z_q^*)^\beta$ be a pseudo-random function, and $\pi(\alpha, \beta, \gamma) : Z_q^* \times N^* \times N^* \to (Z_\gamma^*)^\beta$ be a pseudo-random permutation. The certifier chooses a random number $gsk \in Z_q^*$ as the global secret key, and calculates $P_{pub} = gsk \cdot P_2$. Let $CBSign(\cdot,\cdot,\cdot)$ and $CBVeri(\cdot,\cdot,\cdot,\cdot)$ be the certificate-based sign and verify algorithms described in Section II-D. For example, the certificate-based sign and verify algorithms can be Zhang et al.'s certificate-based signature [32]. Then, the certifier publishes the global parameters $gp = \{G_1, G_2, G_T, P_1, P_2, q, e, h(\cdot), h'(\cdot), H(\cdot), P_{pub}, \varphi(\cdot,\cdot), \pi(\cdot,\cdot,\cdot), CBSign(\cdot,\cdot,\cdot), CBVeri(\cdot,\cdot,\cdot,\cdot)\}$, and keeps the global secret key gsk secret.

B. CLIENTKEYGEN

This algorithm is performed by the client. The client chooses a random number $sk \in Z_q^*$ as his/her secret key, calculates $PK = sk \cdot P_2$ as his/her public key, and sends his/her identity ID and public key PK to the certifier.

C. CERTIFY

This algorithm is performed by the certifier. Upon receiving ID and PK, the certifier chooses a random number $r \in Z_q^*$, computes $CERT = r \cdot P_2$, $cert = r + gsk \cdot h(ID||PK||CERT) \bmod q$. Then, the certifier sends $Cert = (CERT, cert)$ to the client via a private channel. Upon receiving $Cert$, the client checks whether $cert \cdot P_2 = CERT + h(ID||PK||CERT) \cdot P_{pub}$ holds. If the equation holds, then the client accepts $Cert$ otherwise, it rejects.

D. TAGGEN

There are 7 steps in this algorithm. Steps 1-6 are executed by the client, and step 7 is executed by the cloud server. Note that step 1 can be precomputed off-line.

1) STEP 1

Picks random element $\rho \in Z_q^*$, generates $s-1$ random elements $X = \varphi(\rho, s-1) = (x_1, x_2, \ldots, x_{s-1}) \in (Z_q^*)^{s-1}$, and keeps ρ secret. Then, computes $U_j = x_j \cdot P_1, j = 1, 2, \ldots, s-1$, and denoted as $U = (U_1, U_2, \ldots, U_{s-1})$.

2) STEP 2

Let $|q|$ be the bit-length of q, and $\zeta = |q| - 1$. Given a file with file size fz, appends some bits "0" followed by a ζ-bits integer $(fz \pmod{s \cdot \zeta})$ to the end of the given file (The method is similar to that of secure hash function [38]), so that file size of the padded file F is the multiple of $(s \cdot \zeta)$. Then, we pick a file name $FN \in Z_q$ for the given file F randomly.

3) STEP 3

Divides the padded file F into n equal blocks with block size $(s \cdot \zeta)$-bits, and divides each blocks into s equal sections. i.e., $F = (F_1, F_2, \ldots, F_n), F_i = (F_{i0}, F_{i1}, F_{i2}, \ldots, F_{i,s-1})(1 \le i \le n)$. Therefore, $F_{ij} \in Z_q$ since the bit-length of F_{ij} is ζ, where $1 \le i \le n$, and $0 \le j \le s-1$.

4) STEP 4

Computes the signature $\sigma_C = CBSign(sk, Cert, FN||U||n) = (\sigma_C^*, CERT)$, where $\sigma_C^* = (cert + sk \cdot h'((FN||U||n)||ID||PK||CERT||P_{pub})) \cdot H((FN||U||n)||ID||PK||P_{pub})$. Note that any secure signature algorithm can be used in this step.

5) Step 5

Computes $\omega = h'(ID||PK||P_{pub})$, and for $i = 1$ to n, $\Omega_i = H(FN||i||U)$, $T_i = (sk \cdot \omega + cert) \cdot (\Omega_i + F_{i0} \cdot H(P_{pub}) + (\sum_{j=1}^{s-1} F_{ij} \cdot x_j) \cdot P_1)$.

6) STEP 6

Uploads the data and tags $\{F_i, T_i\}(1 \le i \le n)$, FN, ID, PK, U, and σ_C to the cloud server. Stores FN, n, s, ρ, and deletes $\{F_i, T_i\}(1 \le i \le n)$, U, σ_C from the local storage.

7) STEP 7

Upon receiving $\{F_i, T_i\}(1 \le i \le n)$, and FN, ID, PK, U, σ_C, the cloud server stores them.

E. CHALL

The client picks two random elements $\rho', \rho'' \in Z_q^*$, a random integer $\xi \in N^*$, sponsors the challenge $chal = (\rho', \rho'', \xi)$ to the cloud server.

F. PROOFGEN

Received the challenge $chal$, the cloud server computes $L = \{\delta_i\}_{1\le i\le\xi} = \pi(\rho', \xi, n+1) \in (Z_{n+1}^*)^\xi$, $\{a_l\}_{l\in L} = \varphi(\rho'', \xi) \in (Z_q^*)^\xi$, $F_j' = \sum_{l\in L} a_l \cdot F_{lj}$, $0 \le j \le s-1$, $T' = \sum_{l\in L} a_l \cdot T_l$ and sends the response $resp = \{F_0', F_1', F_2', \ldots, F_{s-1}', T'\}$ to the client.

G. PRIVATEVERIFY

Upon receiving $resp$, the client computes $(x_1, x_2, \ldots, x_{s-1}) = \varphi(\rho, s-1)$, $L = \pi(\rho', \xi, n+1)$, and $\{a_l\}_{l\in L} = \varphi(\rho'', \xi) \in (Z_q^*)^\xi$, checks the validity of T' using equation (1).

$$
\begin{aligned}
&e(T', P_2)\\
&= e(\sum_{l\in L} a_l \cdot \Omega_l + F_0' \cdot H(P_{pub}) + (\sum_{j=1}^{s-1} F_j' \cdot x_j)\\
&\quad \cdot P_1, \omega \cdot PK + CERT + h(ID||PK||CERT) \cdot P_{pub}) \qquad (1)
\end{aligned}
$$

If the equation holds, then the file is not corrupted, otherwise, the file is corrupted.

According to Equation (2), we know that Equation (1) is correct.

$$
\begin{aligned}
&e(T', P_2)\\
&= e(\sum_{l\in L} a_l \cdot T_l, P_2)\\
&= e(\sum_{l\in L} a_l \cdot ((sk \cdot \omega + cert) \cdot (\Omega_l + F_{l0} \cdot H(P_{pub})\\
&\quad + (\sum_{j=1}^{s-1} F_{lj} \cdot x_j) \cdot P_1), P_2)\\
&= e(\sum_{l\in L} a_l \cdot \Omega_l + \sum_{l\in L} a_l \cdot F_{l0} \cdot H(P_{pub}) + (\sum_{l\in L} a_l\\
&\quad \cdot \sum_{j=1}^{s-1} F_{lj} \cdot x_j) \cdot P_1), (sk \cdot \omega + cert) \cdot P_2)\\
&= e(\sum_{l\in L} a_l \cdot \Omega_l + F_0' \cdot H(P_{pub}) + (\sum_{j=1}^{s-1} F_j' \cdot x_j)\\
&\quad \cdot P_1, \omega \cdot PK + CERT + h(ID||PK||CERT) \cdot P_{pub}) \qquad (2)
\end{aligned}
$$

H. PUBLICVERIFY

There are only two cases where this algorithm is used. The auditor performs case 1 when a dispute occurs, and case 2 when the client is not available for audit.

Case 1. Given the forwarded response $resp$ from the client, the auditor then computes $L = \pi(\rho', \xi, n+1)$, and $\{a_l\}_{l\in L} = \varphi(\rho'', \xi) \in (Z_q^*)^\xi$, and checks the validity of T' using Equation (3).

$$
\begin{aligned}
&e(T', P_2)\\
&= e(\sum_{l\in L} a_l \cdot \Omega_l + F_0' \cdot H(P_{pub}) + \sum_{j=1}^{s-1} F_j' \cdot U_j, \omega \cdot PK\\
&\quad + CERT + h(ID||PK||CERT) \cdot P_{pub}) \qquad (3)
\end{aligned}
$$

If the equation holds, then the file is not corrupted, otherwise, the file is corrupted.

According to Equation (1), and $U_j = x_j \cdot P_1$ we know that Equation (3) is correct.

Case 2. Upon receiving the authorization from the client, the auditor sponsors a challenge $chal$ to the cloud server, and the cloud server responds with $resp$ to the auditor by using the method described in the ProofGen phase (see Section IV), The auditor checks the validity of $resp$ using the method described in case 1 of the PublicVerify phase (see Section IV). The results will be sent by the auditor to the client.

Furthermore, our scheme supports batch auditing for different files. For example, if there are two files $F = \{F_i\}_{1\le i\le n} = \{(F_{i0}, F_{i1}, F_{i2}, \ldots, F_{i,s-1})\}_{1\le i\le n}$ and $\tilde{F} = \{\tilde{F}_i\}_{1\le i\le\tilde{n}} = \{(\tilde{F}_{i0}, \tilde{F}_{i1}, \tilde{F}_{i2}, \ldots, \tilde{F}_{i,s-1})\}_{1\le i\le\tilde{n}}$ with file name FN and $\widetilde{FN}$ respectively. Therefore, their tags are $\{T_i\}_{1\le i\le n} = \{(sk \cdot \omega + cert) \cdot (\Omega_i + \tilde{F}_{i0} \cdot H(P_{pub}) + (\sum_{j=1}^{s-1} F_{ij} \cdot x_j) \cdot P_1)\}_{1\le i\le n}$ and $\{\tilde{T}_i\}_{1\le i\le\tilde{n}} = \{(sk \cdot \omega + cert) \cdot (\tilde{\Omega}_i + F_{i0} \cdot H(P_{pub}) + (\sum_{j=1}^{s-1} \tilde{F}_{ij} \cdot x_j) \cdot P_1)\}_{1\le i\le\tilde{n}}$ respectively, where $\omega = h'(ID||PK||P_{pub})$, $\{\Omega_i\}_{1\le i\le n} = \{H(FN||i||U)\}_{1\le i\le n}$, $\{\tilde{\Omega}_i\}_{1\le i\le\tilde{n}} = \{H(\widetilde{FN}||i||U)\}_{1\le i\le\tilde{n}}$.

In the ProofGen phase, the cloud server computes $\{\bar{F}_j'\}_{0\le j\le s-1} = \{\sum_{l\in L} a_l \cdot F_{lj} + \sum_{l\in\tilde{L}} \tilde{a}_l \cdot \tilde{F}_{lj}\}_{0\le j\le s-1}$, $0 \le j \le s-1$, $\bar{T}' = \sum_{l\in L} a_l \cdot T_l + \sum_{l\in\tilde{L}} \tilde{a}_l \cdot \tilde{T}_l$, and sends the response $resp = \{\{\bar{F}_j'\}_{0\le j\le s-1}, \bar{T}'\}$ to the client.

In the PrivateVerify phase, the client checks the validity of $\bar{T}'$ by using the following equation.

$$
\begin{aligned}
&e(\bar{T}', P_2)\\
&= e(\sum_{l\in L} a_l \cdot \Omega_l + \sum_{l\in\tilde{L}} \tilde{a}_l \cdot \tilde{\Omega}_l + \bar{F}_0' \cdot H(P_{pub}) + (\sum_{j=1}^{s-1} \bar{F}_j'\\
&\quad \cdot x_j) \cdot P_1, \omega \cdot PK + CERT + h(ID||PK||CERT) \cdot P_{pub})
\end{aligned}
$$

where $\{\Omega_l\}_{l\in L} = \{H(FN||l||U)\}_{l\in L}$, and $\{\tilde{\Omega}_l\}_{l\in\tilde{L}} = \{H(\widetilde{FN}||l||U)\}_{l\in\tilde{L}}$.

In Case 1 of the PublicVerify phase, the auditor checks the validity of $\bar{T}'$ by using the following equation.

$$
\begin{aligned}
&e(\bar{T}', P_2)\\
&= e(\sum_{l\in L} a_l \cdot \Omega_l + \sum_{l\in\tilde{L}} \tilde{a}_l \cdot \tilde{\Omega}_l + \bar{F}_0' \cdot H(P_{pub}) + \sum_{j=1}^{s-1} \bar{F}_j'\\
&\quad \cdot U_j, \omega \cdot PK + CERT + h(ID||PK||CERT) \cdot P_{pub})
\end{aligned}
$$

where $\{\Omega_l\}_{l\in L} = \{H(FN||l||U)\}_{l\in L}$, and $\{\tilde{\Omega}_l\}_{l\in\tilde{L}} = \{H(\widetilde{FN}||l||U)\}_{l\in\tilde{L}}$.

V. SECURITY PROOF

In this section, inspired by [32], [37] we demonstrate that our scheme is secure by using Theorems 1 and 2.

Theorem 1: Our scheme can resist the public key replacement adversary if the co-CDH problem [35] of the group G_1 and G_2 is hard.

Proof: Let $(P_1, P_2, a\cdot P_1, b\cdot P_2)$ be a random instance of co-CDH problem, Θ_1 be a public key replacement adversary, and Ψ be a challenger we constructed. If Θ_1 can forge a valid response with a nonnegligible probability ε_1, then Ψ can try to output $a\cdot b\cdot P_1$, by using the following methods with a nonnegligible probability.

First, we give the framework of the proof. Ψ sets $P_{pub} = b\cdot P_2$ and ID_O as the challenge identity, then sends gp to Θ_1. Θ_1 adaptively submits queries to Ψ, to be answered by Ψ. Note that Ψ sets $a\cdot P_1$ as a part of the output of $H(\cdot)$ query with probability $1-\lambda$. Then, Θ_1 successfully forges a valid response with probability ε_1. Ψ uses the Forking lemma [36] on $h(\cdot)$ query and Θ_1 outputs another valid response. Ψ solves the co-CDH problem by using the two responses forged by Θ_1. In other words, Ψ outputs the value of $a\cdot b\cdot P_1$ with probability ε_1'. The value of ε_1' is computed at the end of the proof. The detailed proof is as follows.

Setup: Ψ sets $P_{pub} = b\cdot P_2$, and sends $gp = \{G_1, G_2, G_T, P_1, P_2, q, e, h(\cdot), h'(\cdot), H(\cdot), P_{pub}, \varphi(\cdot,\cdot), \pi(\cdot,\cdot,\cdot), CBSign(\cdot,\cdot), CBVeri(\cdot,\cdot,\cdot,\cdot)\}$ to Θ_1. Ψ picks a challenge identity ID_O and answers the following queries.

$h(\cdot)$ Query: With the query message $\{ID_\eta, PK_\eta, CERT_\eta\}$, Ψ checks whether there is a tuple $\{ID_\eta, PK_\eta, CERT_\eta, h_\eta\}$ in L_h. If so, returns h_η to Θ_1; otherwise, picks a random number $h_\eta \in Z_q^*$, adds $\{ID_\eta, PK_\eta, CERT_\eta, h_\eta\}$ into the list L_h, returns h_η to Θ_1.

$h'(\cdot)$ Query: With the query message $\{ID_\eta, PK_\eta, P_{pub}\}$, Ψ checks whether there is a tuple $\{ID_\eta, PK_\eta, P_{pub}, \omega_\eta\}$ in $L_{h'}$. If so, returns ω_η to Θ_1; otherwise, picks a random number $\omega_\eta \in Z_q^*$, adds $\{ID_\eta, PK_\eta, P_{pub}, \omega_\eta\}$ into the list $L_{h'}$, returns ω_η to Θ_1.

$H(\cdot)$ Query: With the query message $\{FN, \theta, U\}$, Ψ checks whether there is a tuple $\{\theta, c_\theta, z_\theta, Z_\theta\}$ in L_H. If so, returns $\Omega_\theta = Z_\theta - F_{\theta 0}\cdot\Omega - \sum_{j=1}^{s-1} F_{\theta j}\cdot U_j$ to Θ_1; otherwise, picks a random bit $c_\theta \in \{0,1\}$ such that $\Pr[c_\theta = 0] = \lambda$, $(0<\lambda<1$, and note that the value of λ can be regarded as the probability of "coin tossing", i.e., $\lambda = 1/2)$, picks a random number $Z_\theta \in Z_q^*$, if $c_\theta = 0$, $Z_\theta = z_\theta\cdot P_1$, otherwise, $Z_\theta = z_\theta\cdot a\cdot P_1$, adds $\{\theta, c_\theta, z_\theta, Z_\theta\}$ into the list L_H, returns $\Omega_\theta = Z_\theta - F_{\theta 0}\cdot\Omega - \sum_{j=1}^{s-1} F_{\theta j}\cdot U_j)$ to Θ_1.

ClientKeyGen Query: With the query message $\{ID_\eta\}$, Ψ checks whether $\{ID_\eta, sk_\eta, PK_\eta\}$ exists in L_k. If so, returns PK_η to Θ_1; otherwise, selects a random number $sk_\eta \in Z_q^*$, computes $PK_\eta = sk_\eta\cdot P_2$, adds $\{ID_\eta, sk_\eta, PK_\eta\}$ into L_k, returns PK_η to Θ_1.

Client Secret Key Query: With the query message $\{ID_\eta\}$, Ψ checks the list L_k and returns sk_η to Θ_1.

Public Key Replace Query: With the query message $\{ID_\eta, PK_\eta'\}$, Ψ checks whether there is a tuple $\{ID_\eta, sk_\eta, PK_\eta\}$ in the list L_k. If so, sets $PK_\eta = PK_\eta'$, $sk_\eta = \perp$; otherwise, adds $\{ID_\eta, \perp, PK_\eta'\}$ into L_k.

Certify Query: With the query message $\{ID_\eta, PK_\eta\}$, Ψ checks whether $ID_\eta = ID_O$. If so, Ψ aborts; otherwise, Ψ checks whether $\{ID_\eta, PK_\eta\}$ exists in L_c. If not, Ψ chooses two random numbers $h_\eta, cert_\eta \in Z_q^*$, computes $CERT_\eta = cert_\eta\cdot P_2 - h_\eta\cdot b\cdot P_2$, checks whether $\{ID_\eta, PK_\eta, CERT_\eta\}$ exists in L_h. If the latter is true, then Ψ picks another $cert_\eta \in Z_q^*$ until there is no collision. Ψ will also add $\{ID_\eta, PK_\eta, CERT_\eta, h_\eta\}$ into L_h, add $\{ID_\eta, PK_\eta, CERT_\eta, cert_\eta\}$ into L_c, and return $Cert_\eta = (CERT_\eta, cert_\eta)$ to Θ_1.

TagGen Query: With the query message $\{ID_\eta, F_\theta\}$, Ψ makes the $H(\cdot)$ query and obtains $\{\theta, c_\theta, z_\theta, Z_\theta\}$ from L_H. If $c_\theta = 1$, then aborts; otherwise, computes $T_\theta = z_\theta\cdot\Phi(\omega_\eta\cdot PK_\eta + CERT_\eta + h_\eta\cdot P_{pub})$, and sends $\{\theta, F_\theta, T_\theta\}$ to Θ_1.

Forgery: Finally, Θ_1 outputs a valid forgery response $resp^* = \{F_0^*, F_1^*, F_2^*, \ldots, F_{s-1}^*, T^*\}$ of a client ID^* on subset L^* with probability ε_1, where L^* is generated by challenge $chal^*$. Here, ID^* has not been submitted to the Certify query, and (ID^*, l, F_l^*) has not been submitted to the TagGen query, where $l\in L^*$, and L^* is generated by $chal^*$.

Then, Ψ uses the Forking lemma [36] on $h(\cdot)$ query, and Θ_1 outputs another valid forgery response $resp^{**} = \{F_0^*, F_1^*, F_2^*, \ldots, F_{s-1}^*, T^{**}\}$ of the same client on the same subset L^*.

Solving Co-CDH Problem: If the client's identity is actually ID_O, and at least one of $c_l \neq 0 (l\in L^*)$, then Ψ looks up L_h, L_h', L_H and obtains

$$e(T^*, P_2) = e(\sum_{l\in L^*}\Omega_l + F_0^*\cdot H(P_{pub}) + \sum_{j=1}^{s-1} F_j^*\cdot U_j, \omega_O\cdot PK_O + CERT_O + h_O^*\cdot P_{pub})$$

$$e(T^{**}, P_2) = e(\sum_{l\in L^*}\Omega_l + F_0^*\cdot H(P_{pub}) + \sum_{j=1}^{s-1} F_j^*\cdot U_j, \omega_O\cdot PK_O + CERT_O + h_O^{**}\cdot P_{pub})$$

Let $\tilde{h} = h_O^{**} - h_O^*$, and we have

$$\begin{aligned}
&e(T^{**} - T^*, P_2)\\
&= e(\sum_{l\in L^*}\Omega_l + F_0^*\cdot H(P_{pub}) + \sum_{j=1}^{s-1} F_j^*\cdot U_j, \tilde{h}\cdot b\cdot P_2)\\
&= e(\sum_{l\in L^*} Z_l, \tilde{h}\cdot b\cdot P_2)\\
&= e(\sum_{l\in L^*, c_l=0} z_l\cdot P_1 + \sum_{l\in L^*, c_l=1} z_l\cdot a\cdot P_1, \tilde{h}\cdot b\cdot P_2)\\
&= e(\sum_{l\in L^*, c_l=0} z_l\cdot P_1, \tilde{h}\cdot b\cdot P_2)\cdot e(\sum_{l\in L^*, c_l=1} z_l\cdot a\cdot P_1, \tilde{h}\cdot b\cdot P_2)\\
&= e(P_1, \tilde{h}\cdot\sum_{l\in L^*, c_l=0} z_l\cdot b\cdot P_2)\cdot e(P_1, \tilde{h}\cdot\sum_{l\in L^*, c_l=1} z_l\cdot a\cdot b\cdot P_2)
\end{aligned}$$

$$= e(\tilde{h} \cdot \sum_{l \in L^*, c_l = 0} z_l \cdot \Phi(b \cdot P_2), P_2) \cdot e(\tilde{h} \cdot \sum_{l \in L^*, c_l = 1} z_l \cdot \Phi(a \cdot b \cdot P_2), P_2)$$
$$= e(\tilde{h} \cdot \sum_{l \in L^*, c_l = 0} z_l \cdot \Phi(b \cdot P_2), P_2) \cdot e(\tilde{h} \cdot \sum_{l \in L^*, c_l = 1} z_l \cdot a \cdot b \cdot P_1, P_2)$$

i.e.,

$$e(T^{**} - T^* - \tilde{h} \cdot \sum_{l \in L^*, c_l = 0} z_l \cdot \Phi(b \cdot P_2), P_2) = e(\tilde{h} \cdot \sum_{l \in L^*, c_l = 1} z_l \cdot a \cdot b \cdot P_1, P_2)$$

Then, Ψ can compute $a \cdot b \cdot P_1 = (\tilde{h} \cdot \sum_{l \in L^*, c_l = 0} z_l)^{-1} \cdot (T^{**} - T^* - \tilde{h} \cdot \sum_{l \in L^*, c_l = 0} z_l \cdot \Phi(b \cdot P_2))$.

Probability: According to the game, Ψ can compute $a \cdot b \cdot P_1$, iff, the following three events happen.

$\hat{A}$: Ψ does not abort in the Certify query.

$\hat{B}$: Θ_1 outputs a valid forgery response with at least one of $c_l \neq 0 (l \in L^*)$.

$\hat{C}$: When $\hat{B}$ happens, Θ_1 outputs the valid forgery response with the identity ID_O.

We have $pr[\hat{A}] \geq (1 - \frac{1}{q_h})^{q_C}$, $pr[\hat{B} \mid \hat{A}] \geq (1 - \lambda^{|L^*|}) \cdot \varepsilon_1$, $pr[\hat{C} \mid (\hat{A} \wedge \hat{B})] \geq \frac{1}{q_h}$, where q_h and q_C are the number of h query and Certify query respectively, $|L^*|$ is the number of elements in set L^*. so, the probability of Ψ solving the co-CDH problem is $\varepsilon_1' = pr[\hat{A} \wedge \hat{B} \wedge \hat{C}] \geq \frac{1}{q_h} \cdot (1 - \frac{1}{q_h})^{q_C} \cdot (1 - \lambda^{|L^*|}) \cdot \varepsilon_1$.

■

Theorem 2: Our proposed certificate-based PDP scheme can resist the malicious certifier adversary if the co-CDH problem [35] of the group G_1 and G_2 is hard.

Proof: Let $(P_1, P_2, a \cdot P_1, b \cdot P_2)$ be a random instance of co-CDH problem, Θ_2 be a malicious certifier adversary, and Ψ be a challenger we constructed. If Θ_2 can forge a valid response with a nonnegligible probability ε_2, then Ψ can try to output $a \cdot b \cdot P_1$ by using the following methods with a nonnegligible probability.

First, we give the framework of the proof. The Ψ sets ID_O be the challenge identity, then sends gp, gsk to Θ_2. The Θ_2 adaptively submits queries to Ψ, and Ψ answers them. Note that Ψ sets $a \cdot P_1$ as a part of the output of $H(\cdot)$ query with probability $1-\lambda$, and $PK_\eta = b \cdot P_2$ as the output of ClientKeyGen query when $ID_\eta = ID_O$. Then, Θ_2 successfully forges a valid response with probability ε_2. Ψ solves the co-CDH problem by using the response forged by Θ_2. that is to say, Ψ outputs the value of $a \cdot b \cdot P_1$ with probability ε_2'. The value of ε_2' are computed in the end of the proof. The detail proof can be seen as follows.

Setup: Ψ picks a random number $gsk \in Z_q^*$ as the global secret key, computes $P_{pub} = gsk \cdot P_2$, and returns the global public parameters $gp = \{G_1, G_2, G_T, P_1, P_2, q, e, h(\cdot), h'(\cdot), H(\cdot), P_{pub}, \varphi(\cdot,\cdot), \pi(\cdot,\cdot,\cdot), CBSign(\cdot,\cdot), CBVeri(\cdot,\cdot,\cdot,\cdot)\}$ and the global secret key gsk to Θ_2. Ψ picks a challenge identity ID_O and answers the following queries. Note that $h'(\cdot)$ and $H(\cdot)$ queries are executed as those in the proof of Theorem 1.

ClientKeyGen Query: With the query message $\{ID_\eta\}$, Ψ checks whether $\{ID_\eta, sk_\eta, PK_\eta\}$ exists in L_k. If so, returns PK_η to Θ_2; otherwise, if $ID_\eta \neq ID_O$, selects a random number $sk_\eta \in Z_q^*$, computes $PK_\eta = sk_\eta \cdot P_2$, adds $\{ID_\eta, sk_\eta, PK_\eta\}$ into L_k, returns PK_η to Θ_2; otherwise, sets $PK_\eta = b \cdot P_2$, adds $\{ID_\eta, \perp, PK_\eta\}$ into L_k, returns PK_η to Θ_2.

Client Secret Key Query: With the query message $\{ID_\eta\}$, Ψ checks whether $ID_\eta = ID_O$. If so, Ψ aborts; otherwise, Ψ checks the list L_k and returns sk_η to Θ_2.

Certify Query: Ψ holds two list L_c and L_h, which are initialized to be empty. With the query message $\{ID_\eta, PK_\eta\}$, Ψ checks whether $\{ID_\eta, PK_\eta\}$ exists in L_c. If so, $\{ID_\eta, PK_\eta\}$ is in L_h too; Ψ returns $Cert_\eta = (CERT_\eta, cert_\eta)$ and the hash value h_η of $\{ID_\eta, PK_\eta, CERT_\eta\}$ to Θ_2. Otherwise Ψ chooses a random number $r_\eta \in Z_q^*$, computes $CERT_\eta = r_\eta \cdot P_2$, picks a random number $h_\eta \in Z_q^*$, adds $\{ID_\eta, PK_\eta, r_\eta, CERT_\eta, h_\eta\}$ into L_h. Then, Ψ computes $cert_\eta = r_\eta + gsk \cdot h_\eta$, adds $\{ID_\eta, PK_\eta, r_\eta, CERT_\eta, cert_\eta\}$ into L_c, and returns $Cert_\eta = (CERT_\eta, cert_\eta)$ and the hash value h_η of $\{ID_\eta, PK_\eta, CERT_\eta\}$ to Θ_2.

TagGen Query: With the query message $\{ID_\eta, F_\theta\}$, Ψ makes the $H(\cdot)$ query and obtains $\{\theta, c_\theta, z_\theta, Z_\theta\}$ from L_H. If $ID_\eta \neq ID_O$, computes $T_\theta = (sk \cdot \omega_\eta + cert_\eta) \cdot Z_\theta$, else if $c_\theta = 1$, then aborts; otherwise, computes $T_\theta = z_\theta \cdot \omega_\eta \cdot \Phi(PK_\eta) + cert_\eta \cdot Z_\theta$, and sends $\{\theta, F_\theta, T_\theta\}$ to Θ_2.

Forgery: Finally, Θ_2 outputs a valid forgery response $\{F_0^\#, F_1^\#, F_2^\#, \ldots, F_{s-1}^\#, T^\#\}$ of a client on subset $L^\#$ with probability ε_2, where $L^\#$ is generated by challenge $chal^\#$. Here, $PK^\#$ is the output of ClientKeyGen query, and $(ID^\#, l, F_l^\#)$ has not been submitted to the TagGen query, where $l \in L^\#$, and $L^\#$ is generated by $chal^\#$.

Solving Co-CDH Problem: If the client's identity is ID_O, and at least one of $c_l \neq 0 (l \in L^\#)$, then Ψ looks up L_h, L_h', L_H and obtains

$$e(T^\#, P_2)$$
$$= e(\sum_{l \in L^\#} \Omega_l + F_0^\# \cdot H(P_{pub}) + \sum_{j=1}^{s-1} F_j^\# \cdot U_j, \omega_O \cdot PK_O + CERT_O + h_O \cdot P_{pub})$$
$$= e(\sum_{l \in L^\#} Z_l, \omega_O \cdot b \cdot P_2 + r_O \cdot P_2 + h_O \cdot gsk \cdot P_2)$$
$$= e(\sum_{l \in L^\#} Z_l, \omega_O \cdot b \cdot P_2) \cdot e(\sum_{l \in L^\#} Z_l, (r_O + h_O \cdot gsk) \cdot P_2)$$
$$= e(\sum_{l \in L^\#, c_l = 0} z_l \cdot P_1 + \sum_{l \in L^\#, c_l = 1} z_l \cdot a \cdot P_1, \omega_O \cdot b \cdot P_2) \cdot e((r_O + h_O \cdot gsk) \cdot \sum_{l \in L^\#} Z_l, P_2)$$
$$= e(\sum_{l \in L^\#, c_l = 0} z_l \cdot P_1, \omega_O \cdot b \cdot P_2) \cdot e(\sum_{l \in L^\#, c_l = 1} z_l \cdot a$$

TABLE 1. Field size of G_1 in each type of curves (bits) [40].

Type of curves	A	B	C	D	E	F	G
Fields size of G_1	512	512	≈ 305	171	1024	160	160

$$\cdot P_1, \omega_O \cdot b \cdot P_2) \cdot e((r_O + h_O \cdot gsk) \cdot \sum_{l \in L^\#} Z_l, P_2)$$
$$= e(\omega_O \cdot \sum_{l \in L^\#, c_l = 0} z_l \cdot \Phi(b \cdot P_2), P_2) \cdot e(\sum_{l \in L^\#, c_l = 1} z_l$$
$$\cdot \omega_O \cdot a \cdot b \cdot P_1, P_2) \cdot e((r_O + h_O \cdot gsk) \cdot \sum_{l \in L^\#} Z_l, P_2)$$

In other words, we have the following:

$$e(T^\# - \omega_O \cdot \sum_{l \in L^\#, c_l = 0} z_l \cdot \Phi(b \cdot P_2) - (r_O + h_O \cdot gsk)$$
$$\cdot \sum_{l \in L^\#} Z_l, P_2) = e(\omega_O \cdot \sum_{l \in L^\#, c_l = 1} z_l \cdot a \cdot b \cdot P_1, P_2)$$

Then, Ψ can compute $a \cdot b \cdot P_1 = (\omega_O \cdot \sum_{l \in L^\#, c_l = 1} z_l)^{-1} \cdot (T^\# - \omega_O \cdot \sum_{l \in L^\#, c_l = 0} z_l \cdot \Phi(b \cdot P_2) - (r_O + h_O \cdot gsk) \cdot \sum_{l \in L^\#} Z_l)$.

Probability: According to the process, Ψ can compute $a \cdot b \cdot P_1$, iff, the following three events happen.

$\hat{E}$: Ψ does not abort in Client secret key query.

$\hat{F}$: Θ_2 outputs a valid forgery response with at least one of $c_l \neq 0 (l \in L^\#)$.

$\hat{G}$: When $\hat{F}$ happens, Θ_2 outputs the valid forgery response with the identity ID_O.

We have $pr[\hat{E}] \geq (1 - \frac{1}{q_{ckg}})^{q_{csk}}, pr[\hat{F} \mid \hat{E}] \geq (1 - \lambda^{|L^\#|}) \cdot \varepsilon_2$, $pr[\hat{G} \mid (\hat{E} \wedge \hat{F})] \geq \frac{1}{q_C}$, where q_{ckg}, q_{csk} and q_C are the number of ClientKeyGen query, Client secret key query and Certify query respectively, $|L^\#|$ is the number of elements in set $L^\#$. so, the probability of Ψ solves the co-CDH problem is $\varepsilon_2' = pr[\hat{E} \wedge \hat{F} \wedge \hat{G}] \geq \frac{1}{q_C} \cdot (1 - \frac{1}{q_{ckg}})^{q_{csk}} \cdot (1 - \lambda^{|L^\#|}) \cdot \varepsilon_2$.

■

VI. IMPLEMENTATION AND EVALUATION

In this section, we implemented our auditing scheme on a notebook with the following specifications: Lenovo with an I7-6500U 2.59GHz processor, 8G bytes memory and Window 10 operating system), a Workstation 8.0 virtual machine (with the Ubuntu operating system), and using pairing-based library (version 0.5.12) [39].

A. CURVE SELECTION

There are seven types of curves in pairing-based library (version 0.5.12) [39]. Types A, B, C, and E curves are for symmetric pairings, and Types D, F and G curves are for asymmetric pairings. When the order q of groups is around 160 bits, Lynn [40] suggested a field size of G_1 in each type of curves – see also TABLE 1. According to TABLE 5 in Section VII-C, the size of tags mainly depends on the size of G_1. Therefore, the asymmetric pairing is a good choice for minimizing storage space and communication cost.

We will omit operations with minimal computation costs, and there remains eight operations in our scheme to be considered. These eight operations are $SM_1, PA_1, H_1, SM_2, PA_2, SM_T, PA_T, Pair$. $SM_i (i = 1, 2, T)$ refers to scalar multiplication operation in group G_i, $PA_i (i = 1, 2, T)$ refers to point addition operation in group G_i, H_1 denotes map-to-point hash function in group G_1, and *Pair* denotes bilinear paring operation. Furthermore, these eight operations are the main operations of the schemes reviewed in Section VII-B (as shown in TABLE 4). Note that there are seven type curves in pairing-based library (version 0.5.12) [39], and Types B and C are not implementable [39]. So, we implement those operations in each type of curves, and evaluate their runtime – see TABLE 2.

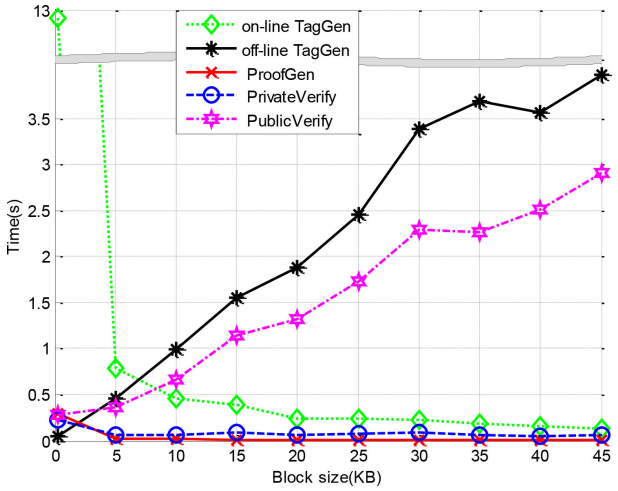

FIGURE 2. Time cost of 1 MB file with different block sizes in our scheme.

From TABLE 4, we observe that the computation costs in the TagGen phase mainly depends on those of operations in G_1. Furthermore, computation costs introduced in the TagGen phase are the main costs for the data integrity auditing schemes. According to the runtime summarized in TABLE 2, it is clear that the asymmetric pairing is a good choice for minimizing computation cost. So, the data integrity auditing schemes based on asymmetric pairing are more practical. Therefore, we will next describe the implementation our scheme using Type D curve (d159.param).

B. IMPLEMENTATION

We fixed a 1 MB file, and increased the block size from 0.3KB to 45KB. As shown in Fig. 2, the on-line time cost in the TagGen phase is only 0.45 seconds with a block size of 10 KB, and the time decreases as the block size increases. Compared with the time cost of 157 seconds for all block sizes in [14], our results are significantly more promising. According to prior studies [1], [14], we selected 4.6% of the entire blocks to sponsor a challenge. Therefore, the time cost of private/public verify phases is less than that of the TagGen phase – see Fig. 2, particularly when the block size is less than 10 KB. When the block size is larger than 10 KB, the time cost of private/public verify phases becomes a little bigger than the actual value, which is because the number of challenging blocks becomes too small and we round it up. Fig. 6 depicts the findings for the large file of 6 MB in size.

TABLE 2. Runtime of key operations in each curve curve (milliseconds).

Operation	$Pair$	SM_1	PA_1	H_1	SM_2	PA_2	SM_T	PA_T
Type A (a.param)	4.425	4.098	0.021	9.620	4.461	0.024	0.716	0.004
Type D (d159.param)	6.661	1.191	0.005	0.063	8.143	0.039	2.118	0.010
Type E (e.param)	12.531	9.097	0.041	48.807	9.034	0.041	0.494	0.003
Type F (f.param)	36.507	1.251	0.005	0.031	2.506	0.009	8.653	0.038
Type G (g149.param)	20.170	1.236	0.005	0.046	17.984	0.083	6.535	0.033

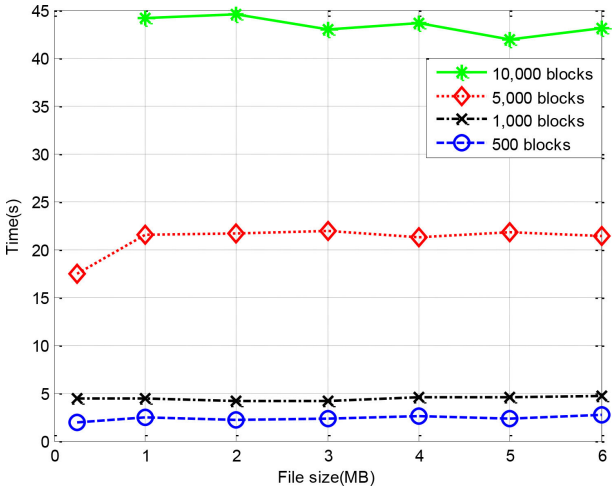

FIGURE 3. On-line time cost of fixed block numbers in TagGen with different file sizes in our scheme.

We fixed the block numbers as "10,000, 5,000, 1,000, and 500", increased the file size from 0.2MB to 6MB, and investigated the on-line time cost introduced in the TagGen phase (as shown in Fig. 3). As shown in Fig. 3, we can observe that the on-line time cost in the TagGen phase is determined by the number of blocks, rather than the file size.

VII. COMPARATIVE SUMMARY

Now, we will compare our scheme with seven existing PDP schemes.

A. PROPERTIES

We can observe from TABLE 3 that our proposed certificated-based PDP scheme not only removes the certificated verify phase in PKI-based PDP scheme, but also addresses the key escrow problem in ID-based PDP scheme. In order to reduce the computation overhead in TagGen, we proposed a certificate-based public/private auditing scheme. In our scheme, the client has more information; thus, it can privately verify the response *resp* with a lower computation cost. When a dispute occurs, the client authorizes the auditor to publicly verify the response *resp* to resolve the dispute. Furthermore, the client can authorize the auditor to publicly verify the response *resp* when he/she is not available (e.g., off-line). Clearly, PublicVerify introduces additional computation cost. Furthermore, we split TagGen into on-line and off-line phases. The client can pre-compute in the TagGen phase for improved performance.

TABLE 3. A comparative summary: Properties.

Schemes	Cryptographic tools	Certificate verify	Key e-scrow	On-line /off-line TagGen	Private /public verify
[2]	PKI-based	Y	N	N	N
[11]	ID-based	N	Y	N	N
[12]	ID-based	N	Y	N	N
[14]	Fuzzy ID-based	N	Y	Y	N
[22]	Certificateless	N	N	N	N
[23]	Certificateless	N	N	N	N
[25]	Certificate-based	N	N	N	N
Proposed	Certificate-based	N	N	Y	Y

B. COMPUTATION COSTS

A summary of the computation costs of our scheme with that of seven other PDP schemes [2], [11], [12], [14], [22], [23], [25] is presented in TABLE 4. Although the seven other schemes are based on symmetric pairing, we analyzed their operations with asymmetric pairing of Type D curve (d159.param). From TABLE 2, one can observe that these seven schemes are more ineffective if the analysis is based on symmetric pairing. In order to avoid the hash value stored attack, we omit the hash function for data of reference [12] in TABLEs 4 and 5. In TABLE 4, n, s, c is the number of the block, the section, and the challenged block respectively, n' and c' are the special case with $s = 1$, and k is the attribute number described in [14].

FIGUREs 4, 5, and 6 also describe the comparison reported in TABLE 4. As discussed earlier, we implemented our scheme using Type D curve. We calculated the computational costs in the seven other schemes using TABLEs 2 and 4 without taking the *Sign* operation into account for simplicity. The findings are presented in FIGUREs 4, 5, and 6. As the block size in [11], [22], [23], [25] are constant (i.e., 160 bits), we do not describe them in the figures. However, we do discuss them below. Note that the computation costs of the scheme in [14] (see FIGUREs 4, 5, and 6) are less than those in [14]. Thus, we set $k = 1$ instead of 3.

In FIGURE 4, we fixed the file size to 1 MB and compared the computation cost in the TagGen phase with different block sizes. The computation costs of the schemes in [11], [22], [23], [25] are constant at 250.296 seconds, 128.448 seconds, 128.455 seconds, 128.448 seconds respectively. From FIGURE 4, we can see that our computation costs in the TagGen phase is much lower than those of other schemes. For example, when the block size is 10 KB, our computation cost in the on-line TagGen phase is only 0.45 seconds, compared with that about 62.83 seconds in the schemes in [2], [12], [14].

TABLE 4. A comparative summary: Computation costs.

Schemes	TagGen	ProofGen	ProofVerify
[2]	$Sign + n \cdot ((s+1) \cdot SM_1 + s \cdot PA_1 + H_1)$	$c \cdot SM_1 + (c-1) \cdot PA_1$	$2 \cdot Pair + (c+s) \cdot SM_1 + (c+s-1) \cdot PA_1 + c \cdot H_1$
[11]	$Sign + n' \cdot (4 \cdot SM_1 + 2 \cdot PA_1)$	$2 \cdot c' \cdot SM_1 + (c'-1) \cdot PA_1$	$3 \cdot Pair + 2 \cdot SM_1 + PA_1 + PA_T$
[12]	$n \cdot ((s+1) \cdot SM_1 + s \cdot PA_1 + H_1)$	$c \cdot SM_1 + (c-1) \cdot PA_1$	$2 \cdot Pair + (c+s+1) \cdot SM_1 + (c+s) \cdot PA_1$
[14]	On-line: $Sign + n \cdot k \cdot ((s+1) \cdot SM_1 + (s+2) \cdot PA_1 + H_1)$ Off-line: $2 \cdot n \cdot k \cdot SM_1$	$k \cdot (c \cdot SM_1 + (c-1) \cdot PA_1)$	$k \cdot (c+2) \cdot Pair + k \cdot (c+s+1) \cdot SM_1 + k \cdot s \cdot PA_1 + (k+c) \cdot SM_T + (k \cdot c + 2 \cdot k + c - 2) \cdot PA_T$
[22]	$n' \cdot (2 \cdot SM_1 + PA_1 + H_1)$	$(c'+1) \cdot SM_1 + (c'-1) \cdot PA_1$	$2 \cdot Pair + (c'+3) \cdot SM_1 + (c'+3) \cdot PA_1 + (c'+1) \cdot H_1$
[23]	$(n'+1) \cdot (2 \cdot SM_1 + PA_1 + H_1)$	$(c'+1) \cdot SM_1 + (c'-1) \cdot PA_1$	$2 \cdot Pair + (c'+6) \cdot SM_1 + (c'+4) \cdot PA_1 + (c'+1) \cdot H_1$
[25]	$n' \cdot (2 \cdot SM_1 + PA_1 + H_1)$	$c' \cdot SM_1 + (c'-1) \cdot PA_1$	$2 \cdot Pair + (c'+2) \cdot (SM_1 + PA_1) + c' \cdot H_1$
Proposed	On-line: $CBSign + n \cdot (3 \cdot SM_1 + 2 \cdot PA_1 + H_1)$ Off-line: $(s-1) \cdot SM_1$	$c \cdot SM_1 + (c-1) \cdot PA_1$	Private: $2 \cdot (Pair + SM_2 + PA_2) + (c+2) \cdot SM_1 + (c+1) \cdot (PA_1 + H_1)$ Public: $2 \cdot (Pair + SM_2 + PA_2) + (c+s) \cdot SM_1 + (c+s-1) \cdot PA_1 + (c+1) \cdot H_1$

$Sign(CBSign)$: The computation cost of signature (certificate-based signature) operation.
$SM_1(SM_2, SM_T)$:The computation cost of scalar multiplication operation in group G_1 (G_2, G_T).
$PA_1(PA_2, PA_T)$:The computation cost of point addition operation in group G_1 (G_2, G_T).
H_1:The computation cost of map-to-point hash function operation in group G_1.
$Pair$: Computation cost of bilinear pairing operation.
s: Number of sections in one block.
$n(n')$: Number of block with s sections (special case with $s = 1$) in one file.
$c(c')$: Number of challenged block with s section (special case with $s = 1$).
k: Number of attribute in reference [14].

In FIGURE 5, we fixed the block number to 5,000 and compared the on-line computation cost in the TagGen phase with different file sizes. Again, since the block sizes in the schemes of [11], [22], [23], [25] are constant at 160 bits, the block number of those schemes cannot be fixed at 5,000 blocks. Clearly, the computation costs of those schemes increase as file size increases. As shown in FIGURE 5, the computation costs of the schemes in [2], [12], [14] increase as the file size increases, which is almost 400 seconds when the file size is 6 MB. However, in the TagGen phase of our scheme, the on-line computation cost is almost constant to 21.8 seconds, and the total computation cost linearly increases, but very slowly. The total computation cost in the TagGen phase of our scheme is estimated to be 34.6 seconds when the file size is 1 GB.

In FIGURE 6, we set $c = 0.046n$ and file size is 6MB. Then, we compared the computation cost in the ProofVerify phase with different block sizes. The computation costs of the schemes in [11], [22], [23], [25] are constant at 0.022 seconds, 18.236 seconds, 18.239 seconds, 18.235 seconds, respectively. We also remark that the computation cost of the scheme in [11] is not related to the file size. From FIGURE 6, we can see that our computation costs in the PrivateVerify phase is much lower than those of the other schemes with the exception of [11], and the computation costs of PublicVerify are the same for the schemes in [2], [12], [14]. However, the computation cost in the TagGen phase of the scheme in [11] is the highest.

TABLE 5. A comparative summary: Storage and communication costs.

Schemes	TagGen	Chall	ProofGen
[2]	$\|F\| + n \cdot \|G_1\| + (s \cdot \|G_1\| + \|n\| + \|FN\| + \|Sign\|)$	$c \cdot (\|n\| + \|Z_q^*\|)$	$\|G_1\| + s \cdot \|Z_q^*\|$
[11]	$\|F\| + 2 \cdot n' \cdot \|G_1\| + (\|n'\| + \|FN\| + \|Sign\|)$	$c' \cdot \|n'\| + \|Z_q^*\|$	$2 \cdot \|G_1\| + \|Z_q^*\|$
[12]	$\|F\| + n \cdot \|G_1\|$	$\|n\| + 2 \cdot \|Z_q^*\|)$	$\|G_1\| + s \cdot \|Z_q^*\|$
[14]	$\|F\| + 3 \cdot n \cdot k \cdot \|G_1\| + (s \cdot \|G_1\| + \|n\| + \|FN\| + \|Sign\|)$	$c \cdot (\|n\| + \|Z_q^*\|)$	$3 \cdot k \cdot \|G_1\| + s \cdot \|Z_q^*\|$
[22]	$\|F\| + n' \cdot (\|G_1\| + \|id_i\|)$	$c' \cdot (\|n'\| + \|Z_q^*\|)$	$2 \cdot \|G_1\| + \|Z_q^*\|$
[23]	$\|F\| + (n'+2) \cdot \|G_1\| + \|Z_q^*\|$	$c' \cdot (\|n'\| + \|Z_q^*\|)$	$4 \cdot \|G_1\| + \|Z_q^*\|$
[25]	$\|F\| + n' \cdot \|G_1\|$	$\|n\| + 2 \cdot \|Z_q^*\|)$	$\|G_1\| + \|Z_q^*\|$
Proposed	$\|F\| + n \cdot \|G_1\| + (s \cdot \|G_1\| + \|n\| + \|FN\| + \|CBSign\|)$	$\|n\| + 2 \cdot \|Z_q^*\|$	$\|G_1\| + s \cdot \|Z_q^*\|$

$|F|$: Size of file F.
$|G_1|$: Size of group G_1.
$|Z_q^*|$: Size of integer set Z_q^*.
$|FN|$: Size of file name FN.
$|id_i|$: Size of unique identity id_i of file blocks.
$|Sign|(|CBSign|)$: Size of signature (certificate-based signature).
s: Number of the section in one block.
$n(n')$: Number of block with s sections (special case with $s = 1$) in one file.
$|n|(|n'|)$: Size of number n (n') of file blocks.
$c(c')$: Number of the challenged block with s sections (special case with $s = 1$).
k: Number of the attribute in reference [14].

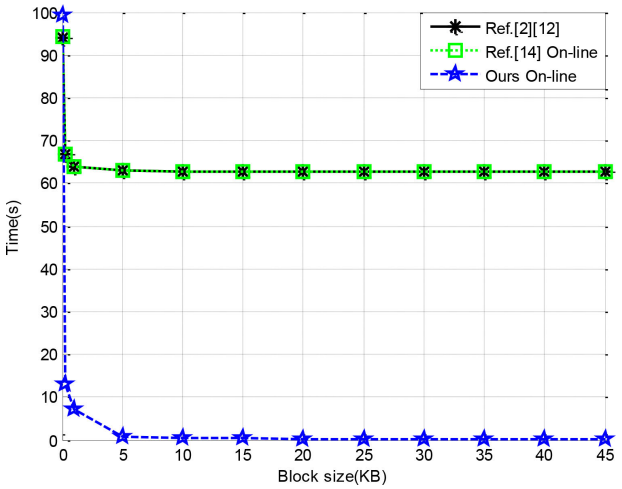

FIGURE 4. A comparative summary: Computation cost in the TagGen phase for 1 MB file with different blocks.

C. STORAGE AND COMMUNICATION COSTS

A comparative summary of the storage space and communication costs for all eight schemes is presented in TABLE 5, and we can see that the communication cost of our scheme

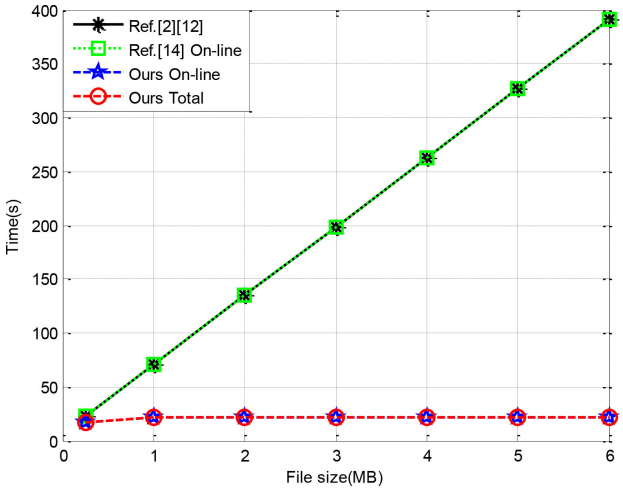

FIGURE 5. A comparative summary: Time cost of 5,000 blocks in the TagGen phase with different file sizes.

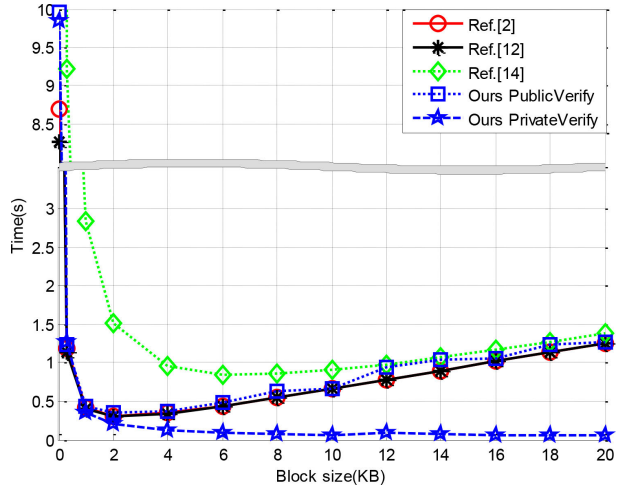

FIGURE 6. A comparative summary: Computation cost in the ProofVerify phase of 6 MB file with different block sizes.

is lower than those of the seven other schemes in the Chall phase.

FIGURE 7 also shows the storage costs of TABLE 5 in the TagGen phase. As the block sizes of the schemes in [11], [22], [23], [25] are constant at 160 bits, we do not describe them in FIGURE 7. However, we can observe that the storage costs of the schemes in [11], [22], [23], [25] are 2.0002MB, 2.4MB, 2.0005MB, 2MB, respectively. For simplicity, we omit the size of *sign* in TABLE 5 for the schemes in [2], [11], [14], and set $k = 1$ for the scheme in [14]. Let size of the file be 1MB, and file section, G_1 and Z_q^* be all 160 bits. We used Type D curve (d159.param) and point compression technology. The findings reported in FIGURE 7 show that our storage costs in the TagGen phase is as same as those of the scheme in [2], and are much lower than those of the remaining schemes except [12]. However, computation costs in the TagGen phase of the schemes in [2], [12] are much higher than those of our scheme.

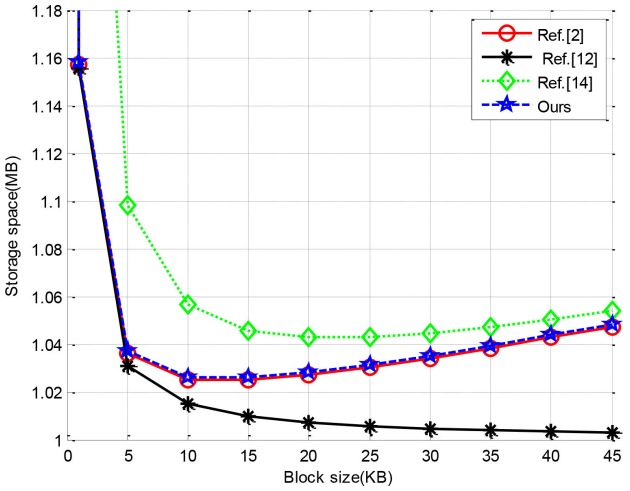

FIGURE 7. The comparisons of storage cost of 1 MB file with different block sizes.

VIII. CONCLUSIONS AND OPEN ISSUES

Data integrity auditing is one important approach to ensure the integrity of outsourced data in cloud storage. Such approach will be increasingly important as more data are being outsourced and stored at the cloud or offshore locations. However, the TagGen phase in data integrity auditing schemes is usually the cause of inefficiency. In addition, most existing schemes are based on symmetric bilinear pairing, and there is a lack of certificate-based auditing schemes.

Therefore, in this paper we presented our lightweight certificate-based public/private auditing scheme based on asymmetric bilinear pairing. Such a scheme can be used to ensure the integrity of data outsourced to the cloud for storage. We then evaluated both the security and the performance of our scheme. In comparison to seven other existing auditing schemes, our auditing scheme has a significant reduced computation cost for the client at TagGen, without relying on a third-party. When the block size is 10 KB, our on-line computation cost in the TagGen phase is only 0.45 seconds, while it needs about 62.83 to 250.29 seconds in other schemes. Furthermore, if we fix the blocks numbers, the on-line computation cost at Taggen is constant even when file size changes.

In this paper, we also introduced the notion of public/private auditing model. Specifically, the client executes the private auditing phase, which is more efficient since the client has more information. The auditor executes the public auditing phase in the event of a dispute or when the client is not available to undertake an audit. Such a notion is key to the efficiency of our scheme. However, how to design a public/private auditing scheme as efficient as private auditing remains an open issue.

ACKNOWLEDGMENT

The authors wish to thank the anonymous reviewers for their valuable comments and suggestions. The authors would like to thank the National Natural Science Foundation of

China (No. U1905211, No. 61771140, No. U1405255 and No. 61702100), Fujian Province University Industry Cooperation of Major Science and Technology Project (No. 2017H6005), China Postdoctoral Science Foundation (No. 2017M612107 and No.2018T110636). K.-K.R. Choo is funded by the Cloud Technology Endowed Professorship.

REFERENCES

[1] 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.

[2] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Proc. 14th Int. Conf. Theory Appl. Cryptol. Inf. Secur. (ASIACRYPT)*, 2008, pp. 90–107.

[3] D. Cash, A. Küpçü, and D. Wichs, "Dynamic proofs of retrievability via oblivious RAM," in *Proc. Adv. Cryptol. (EUROCRYPT)*, 2013, pp. 279–295.

[4] H. Yan, J. Li, J. Han, and Y. Zhang, "A novel efficient remote data possession checking protocol in cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 1, pp. 78–88, Jan. 2017.

[5] H. Yan, J. Li, and Y. Zhang, "Remote data checking with a designated verifier in cloud storage," *IEEE Syst. J.*, to be published, doi: 10.1109/JSYST.2019.2918022.

[6] J. Yu, K. Ren, C. Wang, and V. Varadharajan, "Enabling cloud storage auditing with key–exposure resistance," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 6, pp. 1167–1179, Jun. 2015.

[7] K. Yang and X. Jia, "An efficient and secure dynamic auditing protocol for data storage in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 9, pp. 1717–1726, Sep. 2013.

[8] J. Shen, J. Shen, X. Chen, X. Huang, and W. Susilo, "An efficient public auditing protocol with novel dynamic structure for cloud data," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 10, pp. 2402–2415, Oct. 2017.

[9] 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.

[10] H. Wang, D. He, and S. Tang, "Identity–based proxy–oriented data uploading and remote data integrity checking in public cloud," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 6, pp. 1165–1176, Jun. 2016.

[11] J. Zhang and Q. Dong, "Efficient ID-based public auditing for the outsourced data in cloud storage," *Inf. Sci.*, vols. 343–344, pp. 1–14, May 2016.

[12] H. Wang, "Identity–based distributed provable data possession in multicloud storage," *IEEE Trans. Services Comput.*, vol. 8, no. 2, pp. 328–340, Mar. 2015.

[13] F. Wang, L. Xu, H. Wang, and Z. Chen, "Identity-based non-repudiable dynamic provable data possession in cloud storage," *Comput. Electr. Eng.*, vol. 69, pp. 521–533, Jul. 2018.

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

[15] J. Li, H. Yan, and Y. Zhang, "Efficient identity-based provable multi–copy data possession in multi–cloud storage," *IEEE Trans. Cloud Comput.*, to be published, doi: 10.1109/tcc.2019.2929045.

[16] H. Abelson, R. Anderson, S. M. Bellovin, J. Benaloh, M. Blazes, W. Diffie, J. Gilmore, P. G. Neumann, R. L. Rivest, J. I. Schilier, and B. Schneier, "The risks of key recovery, key escrow, and trusted third-party encryption," *World Wide Web J.*, vol. 2, pp. 241–257, Apr. 1997.

[17] S. S. Al-Riyami and K. G. Paterson, "Certificateless public key cryptography," in *Proc. ASIACRYPT*, 2003, pp. 452–473.

[18] C. Gentry, "Certificate-based encryption and the certificate revocation problem," in *Proc. 22nd Int. Conf. Theory Appl. Cryptograph. Techn. (EUROCRYPT)*, 2003, pp. 272–293.

[19] F. Wang, C.-C. Chang, and L. Harn, "Simulatable and secure certificate-based threshold signature without pairings," *Secur. Commun. Netw.*, vol. 7, no. 11, pp. 2094–2103, Nov. 2014.

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

[21] 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.

[22] 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.

[23] D. He, N. Kumar, S. Zeadally, and H. Wang, "Certificateless provable data possession scheme for cloud–based smart grid data management systems," *IEEE Trans. Ind. Informat.*, vol. 14, no. 3, pp. 1232–1241, Mar. 2018.

[24] J. Li, H. Yan, and Y. Zhang, "Certificateless public integrity checking of group shared data on cloud storage," *IEEE Trans. Services Comput.*, to be published, doi: 10.1109/tsc.2018.2789893.

[25] H. Wang and J. Li, "Private certificate-based remote data integrity checking in public clouds," in *Proc. COCOON*, 2015, pp. 575–586.

[26] M. Sookhak, A. Gani, H. Talebian, A. Akhunzada, S. U. Khan, R. Buyya, and A. Y. Zomaya, "Remote data auditing in cloud computing environments: A survey, taxonomy, and open issues," *ACM Comput. Surv.*, vol. 47, no. 4, p. 65, 2015.

[27] M. Sookhak, H. Talebian, E. Ahmed, A. Gani, and M. K. Khan, "A review on remote data auditing in single cloud server: Taxonomy and open issues," *J. Netw. Comput. Appl.*, vol. 43, pp. 121–141, Aug. 2014.

[28] J.-S. Pan, C.-Y. Lee, A. Sghaier, M. Zeghid, and J. Xie, "Novel systolization of subquadratic space complexity multipliers based on toeplitz matrix–vector product approach," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 7, pp. 1614–1622, Jul. 2019, doi: 10.1109/TVLSI.2019.2903289.

[29] J. Li, X. Tan, X. Chen, D. S. Wong, and F. Xhafa, "OPoR: Enabling proof of retrievability in cloud computing with resource–constrained devices," *IEEE Trans. Cloud Comput.*, vol. 3, no. 2, pp. 195–205, Apr. 2015.

[30] W. Shen, J. Yu, H. Xia, H. Zhang, X. Lu, and R. Hao, "Light-weight and privacy-preserving secure cloud auditing scheme for group users via the third party medium," *J. Netw. Comput. Appl.*, vol. 82, pp. 56–64, Mar. 2017.

[31] J. Han, Y. Li, and W. Chen, "A Lightweight And privacy-preserving public cloud auditing scheme without bilinear pairings in smart cities," *Comput. Standards Interfaces*, vol. 62, pp. 84–97, Feb. 2019.

[32] Y. Zhang, J. Li, Z. Wang, and W. Yao, "A new efficient certificate-based signature scheme," *Chin. J. Electron.*, vol. 24, no. 4, pp. 776–782, 2015.

[33] M. Barbos, *Identity Based Cryptography From Bilinear Pairings*. Braga, Portugal: Centro de Ciências e Tecnologias da Computação do Departamento de Informática da Universidade do Minho, 2005.

[34] S. D. Galbraith, K. G. Paterson, and N. P. Smart, "Pairings for cryptographers," *Discrete Appl. Math.*, vol. 156, no. 16, pp. 3113–3121, Sep. 2008.

[35] D. Fiore and R. Gennaro, "Publicly verifiable delegation of large polynomials and matrix computations, with applications," in *Proc. ACM Conf. Comput. Commun. Secur.-(CCS)*, 2012, pp. 501–512.

[36] D. Pointcheval and J. Stern, "Security proofs for signature schemes," in *Proc. Eurocrypt*, 1996, pp. 387–398.

[37] J. C. Choon and J. H. Cheon, "An identity-based signature from gap Diffie-Hellman groups," in *Proc. PKC*, 2003, pp. 18–30.

[38] *Announcing the Standard for Secure Hash Standard*, Standards 180-1, Federal Information Processing, 1995.

[39] B. Lynn. *PBC Library-the Pairing-Based Cryptography Library*. Accessed: Mar. 1, 2018. [Online]. Available: https://crypto.stanford.edu/pbc/

[40] B. Lynn, "On the implementation of pairing-based cryptosystems," Ph.D. dissertation, Dept. Comput. Sci., Stanford Univ., Stanford, CA, USA, 2007.

FENG WANG received the M.S. degree in applied mathematics from Guangzhou University, in 2006. He is currently pursuing the Ph.D. degree in applied mathematics with Fujian Normal University. He is currently an Associate Professor with the College of Mathematics and Physics, Fujian University of Technology. His research interests include computer cryptography, networks, and information security.

LI XU (Member, IEEE) received the B.S and M.S. degrees from Fujian Normal University, Fuzhou, China, in 1992 and 2001, respectively, and the Ph.D. degree from the Nanjing University of Posts and Telecommunications, Nanjing, China, in 2004.

He is currently a Professor and a Ph.D. Supervisor with the College of Mathematics and Informatics, Fujian Normal University. He is also the Director Central of Network and Data, Fujian Normal University, and also the Director of the Key Lab of Network Security and Cryptography, Fujian Normal University. He has authored or coauthored more than 150 articles in international journals and conferences, including the IEEE TRANSACTIONS ON COMPUTER, *ACM Transactions on Sensor Network*, the IEEE TRANSACTIONS ON RELIABILITY, the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, *Information Science*, and *Computer Network*. His current research interests include network and information security, wireless network and communication, complex network and systems, and intelligent information in communication networks.

Dr. Xu is a member of ACM, and a Senior Member of CCF and CIE in China. He has been invited to act as a PC Chair or a member for more than 30 international conferences.

KIM-KWANG RAYMOND CHOO (Senior Member, IEEE) received the Ph.D. degree in information security from the Queensland University of Technology, Australia, in 2006. He currently holds the Cloud Technology Endowed Professorship at The University of Texas at San Antonio (UTSA). In 2016, he was named the Cybersecurity Educator of the Year—APAC, and in 2015, he and his team received the Digital Forensics Research Challenge organized by Germany's University of Erlangen-Nuremberg. He is also a Fellow of the Australian Computer Society. He was a recipient of the 2019 IEEE Technical Committee on Scalable Computing (TCSC) Award for Excellence in Scalable Computing (Middle Career Researcher), 2018 UTSA College of Business Col. Jean Piccione and Lt. Col. Philip Piccione Endowed Research Award for Tenured Faculty, an Outstanding Associate Editor of 2018 for IEEE ACCESS, British Computer Society's 2019 Wilkes Award Runner-up, 2019 EURASIP JWCN Best Paper Award, Korea Information Processing Society's JIPS Survey Paper Award (Gold) 2019, IEEE Blockchain 2019 Outstanding Paper Award, Inscrypt 2019 Best Student Paper Award, IEEE TrustCom 2018 Best Paper Award, ESORICS 2015 Best Research Paper Award, 2014 Highly Commended Award by the Australia New Zealand Policing Advisory Agency, Fulbright Scholarship in 2009, 2008 Australia Day Achievement Medallion, and British Computer Society's Wilkes Award, in 2008. He is a Co-Chair of IEEE Multimedia Communications Technical Committee's Digital Rights Management for Multimedia Interest Group.

YUEXIN ZHANG received the Ph.D. degree from the School of Information Technology, Deakin University, Australia, in 2017. He currently serves as a Postdoctoral Research Fellow with the Swinburne University of Technology, Australia. He has published more than 20 research articles in refereed international conferences and journals. His research interests include wireless network security, mobile security, and autonomous cyber security.

HUAQUN WANG received the B.S. degree in mathematics education from Shandong Normal University, China, in 1997, the M.S. degree in applied mathematics from East China Normal University, China, in 2000, and the Ph.D. degree in information security from the Nanjing University of Posts and Telecommunications, China, in 2006. He is currently a Professor with the Nanjing University of Posts and Telecommunications. His research interests include applied cryptography, blockchain, network security, and cloud computing security.

JIGUO LI received the B.S. degree in mathematics from Heilongjiang University, Harbin, China, in 1996, and the M.S. degree in mathematics and the Ph.D. degree in computer science from the Harbin Institute of Technology, Harbin, China in 2000 and 2003, respectively. From 2006 to 2007, he was a Visiting Scholar with the Centre for Computer and Information Security Research, School of Computer Science and Software Engineering, University of Wollongong, Australia. From 2013 to 2014, he was a Visiting Scholar with the Institute for Cyber Security, The University of Texas at San Antonio. He is currently a Professor with the College of Mathematics and Informatics, Fujian Normal University, Fuzhou, China. He had been a Professor with the College of Computer and Information, Hohai University, Nanjing, China, from 2003 to 2018. His research interests include cryptography and information security, cloud computing, wireless security, and trusted computing. He has published more than 150 research articles in refereed international conferences and journals. His work has been cited more than 3000 times at Google Scholar. He has served as a Program Committee Member in more than 30 international conferences and served as the Reviewers in more than 90 international journals and conferences.