Ji *et al. Journal of Cloud Computing: Advances, Systems and Applications* (2022) 11:7
https://doi.org/10.1186/s13677-022-00279-5

Journal of Cloud Computing: Advances, Systems and Applications

RESEARCH Open Access

Identity-based remote data checking with a designated verifier

Yanyan Ji[1*], Bilin Shao[1], Jinyong Chang[2,3], Maozhi Xu[4] and Rui Xue[3]

Abstract

Traditional remote data possession auditing mechanisms are usually divided into two types: private auditing and public auditing. Informally, private auditing only allows the original data owner to check the integrity of its outsourced data, whereas anyone is allowed to perform the checking task in public auditing. However, in many practical applications, the data owner expects some designated verifier (instead of others) to audit its data file, which cannot be covered by the existing private or public auditing protocols. Thus, in a recent work, Yan et al. proposed a new auditing technique with a designated verifier [*IEEE Systems Journal, 12(4): 1788-1797, 2020*]. Nevertheless, we note that this protocol suffers from complicated management of certificates and hence heavily relies on public key infrastructure. To overcome this shortcoming, in this paper, we propose an identity-based auditing protocol with a designated verifier, which not only avoids the introduction of certificates, but also has the desired property of only allowing specific verifier to audit. Its security is based on the classical computational Diffie-Hellman and Weil Diffie-Hellman assumptions. Finally, performance analysis shows that our proposed protocol is very efficient and suitable for some real-life applications.

Keywords: Cloud storage, Identity-based cryptography, Designated verifier, Integrity auditing

Introduction

In recent years, cloud storage has become an attractive technique for users or data owners (DOs) to store their data since it may have much lower prices than the cost to maintain them on personal devices. When enjoying the services provided by cloud companies, the DOs can freely access and conveniently share these outsourced data on different devices and locations. However, they also completely lose the control of its data after uploading them to the cloud service provider (CSP). Meanwhile, for kinds of internal or external reasons, it is extremely possible to lose user's data for the CSP [1]. For example, the CSP may deliberately delete user's rarely used data to save its own space and thus attract more clients to gain more economic benefits. In addition, the cloud companies may also be attacked by hackers and hence "passively" lose user's data [2].

Under this situation, the DOs naturally worry about the integrity of their outsourced data and urgently hope that there exists a mechanism to help them to audit or check data's integrity. If it is broken, then they should obtain compensations. As a result, remote data possession checking (RDPC) mechanisms, like provable data possession (PDP) or proof of retrievability (PoR) were developed and broadly studied in the last decades [3, 4].

According to the manners of auditing, RDPC mechanisms are divided into two types: private auditing (see [5]) and public auditing (see [6]). In private auditing, the auditor is just the DO itself. But in public auditing, anyone is allowed to perform the task of auditing. A popular way is to outsource the auditing task to some third-party auditor (TPA), who may have more professional knowledge on auditing and more computing power. Therefore, the public auditing is very popular in practical applications.

However, in some situations, the DO hopes or needs to limit the verifier's identity, and just allows a designated auditor to check its data's integrity. For example, a financial staff in a company outsourced some sensitive data,

*Correspondence: yany_ji@163.com
[1]School of Management, Xi'An University of Architecture and Technology, Xi'An 710055, Shaanxi, People's Republic of China
Full list of author information is available at the end of the article

such as employees' salaries, to some CSP. Then the chief financial officer is just the one that is allowed to check the integrity of these data. Another example is that some TPA company gives its clients a lower discount if the DO only chooses this TPA company (instead of others) as the auditor of its stored data. Therefore, in these practical scenarios, how to design RDPC mechanisms with a designated verifier is an interesting but challenging problem.

In [7], Yan et al. recently designed an efficient designated-verifier PDP (DV-PDP, for short) protocol, where the DO specifies a designated verifier to audit the integrity of its data. Meanwhile, they also analyzed its security in the random oracle model.

However, it is well known that most of the existing public (DV-)PDP protocols are based on the management of certificates and heavily relies on public key infrastructure (PKI) technique [8]. The reason lies in that other entities in the protocol need to securely obtain and use the public keys of some participants, but how to fix the relationship between the received public key and its true identity is hard to handle. In other words, the genuineness of user's public key is hard to guarantee. The traditional method is issuing certificates for these users by a trusted certificate authority (CA) center. However, the complex management procedures on certificates, including generation, storing, delivery, updating, and revocation etc., additionally increase communication/computational costs, which are time-consuming and expensive, and thus greatly reduce the efficiencies of the designed PDP protocols. In addition, PKI's security cannot be completely guaranteed, especially when CA center is controlled by some malicious hacker. For instance, after discovering more than 500 fake certificates, web browser vendors were forced to blacklist all certificates issued by DigiNotar, a Dutch CA, in 2011 [9].

Motivation. We note that the identity-based (IB) cryptography does not suffer this problem [10]. More specifically, in identity-based primitives, user's public key is set as its name, email address, or identity-card number and so on, which does not need the certificates to authenticate its validity. Then a trusted key generation center (KGC) generates secret keys for different users corresponding to these identities. When all users have their secret keys issued by the same KGC, individual public keys become obsolete, which removes the need for explicit certification and all necessary costs. These features make the identity-based paradigm particularly appealing for use in conjunction with PDP protocols. This is also the reason why many proposed PDP protocols were twisted into IB-PDP ones [11, 12]. Hence, it becomes an interesting problem to introduce the identity-based technique to IB-PDP protocol.

Our Contributions. In this paper, we consider the new notion of IB-PDP protocol with a designated verifier (IB-DV-PDP, for short), which can be seen as an improvement on DV-PDP proposed by Yan et al. in [8]. The main contributions of this paper can be summarized as follows.

1) For the first time, we propose the notion of IB-DV-PDP protocol, which provides the integrity checking of user's stored data under the constraint of a designated verifier. In our proposed protocol, anyone knows the cloud user's identity but only the designated verifier is allowed to perform the auditing task on behalf of the data owner, which may find special applications in practical scenarios.
2) Second, we define the security model of IB-DV-PDP protocol, which not only satisfies the security definition of previous IB-PDP protocol, but also describes the security against un-designated verifier. More precisely, we first consider the security against malicious CSP, which is the essential security for PDP protocol. Then in order to guarantee the DV's interests, we discuss the security against any unauthorized verifier.
3) Then, we propose a concrete construction of IB-DV-PDP protocol and prove its security within our model based on the computational Diffie-Hellman (CDH) and Weil Diffie-Hellman (WDH) assumptions. Meanwhile, we emphasize that this security relies on the classical random oracle model.
4) Finally, we also analyze the performances of our proposed protocol, including the communication overheads, computational/storage costs (theoretical analysis), and experimental results, which shows that our protocol is very efficient and suitable for practical applications.

Related Works. In 2007, Juels et al. first proposed the notion of PoR to model the security of outsourced data, where the error-correcting code was combined with the spot-checking of data to verify the integrity [13]. However, this scheme does not support public checking and only allows limited number of checking. At the same time, Ateniese et al. presented a similar technique named PDP, which is based on the RSA-homomorphic authenticators [14]. It allows public auditing and unlimited number of verifications. Later, Dodis, Vadhan and Wichs constructed PoR schemes via hardness amplification [15]. Note that the notion PoR is obviously stronger than PDP. Therefore, constructing a secure PDP protocol is easier than instantiating a PoR scheme. In 2017, Zhang et al. presented a general framework to design secure PDP protocols using homomorphic encryption schemes [2].

In 2016, Chen et al. first revealed an intrinsic relationship between network coding (see [16–18]) and secure cloud storage protocol, although both of them seem to be very different in their nature and were studied independently [19]. But the reverse direction does not hold in

general unless imposing additional constraints. In recent work [20], Chang et al. defined a new notion of admissible PoR and proved that it is also possible to design secure network coding from admissible PoR protocols.

Consider that most of the previous PDP or PoR protocols heavily reply on PKI technique and must deal with complicated management of public key certificates. As a result, many identity-based PDP or PoR protocols were also proposed in past years,which borrow the identity-based idea in public key cryptography [10].More concisely, Wang et al. designed IB-PDP protocol in public clouds [21], in which they did not consider the problem of data's privacy preserving. Then, in 2017, Yu et al. constructed a secure IB-PDP protocol with perfect data privacy preserving to improve Wang et al.'s result [9]. Moreover, Xue et al. designed identity-based public auditing protocol against malicious auditors via the new technique of blockchain [12].

In order to decrease the system complexity, later, Li et al. also proposed a new identity-based data storage protocol, which uses homomorphic verifiable tag to achieve it [8]. In addition, in multi-copy-multi-cloud case, they also considered the efficient construction of IB-PDP protocol [22]. Recently, Chang et al. revisited this protocol and proposed some improvements on its performances [23].

In addition, Zhang et al. constructed an IB-PDP protocol, which is secure against key-exposure attack and based on Lattices [24]. Later, Mary and Rhymend proposed a stronger key-exposure model and gave some suggestions on how to construct this kind of IB-PDP protocol [25].

Note that the previous IB-PDP protocols did not consider the designated-verifier case. Hence, our proposed IB-DV-PDP protocol is the first one and will find applications in practical scenarios.

Organizations. The organizations of this paper are as follows. In "Preliminaries" section, we introduce some basic notations and notions. In "Our proposed construction and its security" section, the proposed protocol and its security analysis are given. In "Performances analysis" section, we analyze the performances and compare them with three other related PDP protocols. Finally, conclusions are presented in "Conclusions" section.

Preliminaries

Basic notations

Now, we first present the basic notations used in this paper. More precisely, we use λ or 1^λ to denote the security parameter. PPT is the abbreviation of "Probabilistic Polynomial Time". For a natural number n, the symbol $[n]$ means the set of $\{1, 2, \cdots, n\}$. For an algorithm A, the symbol "$y \leftarrow A(x_1, x_2, \cdots)$" denotes that it takes $x_1, x_2, \cdots$ as inputs, and will output y. For a group $\mathbb{G}$, $|\mathbb{G}|$ denotes the bit length of each element in it. A function $f(\lambda)$ is negligible if, for any $c > 0$, there exists a $\lambda_0 \in \mathbb{Z}$ such that for any $\lambda > \lambda_0$, it always holds that $f(\lambda) < \lambda^{-c}$. Other symbols and their corresponding definitions are listed in Table 1.

Blinear map

Consider two cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$ with the same prime order q. A map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is called a bilinear map if the following conditions hold.

- **Non-Degeneracy.** For any generators g_1, g_2 of $\mathbb{G}_1$, it holds that $e(g_1, g_2) \neq 1_{\mathbb{G}_2}$, in which $1_{\mathbb{G}_2}$ is the identity element of $\mathbb{G}_2$.
- **Computability.** It is efficient to compute $e(g_1, g_2)$ for any $g_1, g_2 \in \mathbb{G}_1$.
- **Bilinearity.** For any $a, b \in \mathbb{Z}_q$, $g_1, g_2 \in \mathbb{G}_1$, it holds that

$$e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}.$$

CDH assumption

Define $\mathbb{G}_1$ as a cyclic group with prime order q and generator g. Randomly choose a, b from $\mathbb{Z}_q$ and compute

Table 1 Symbols and their corresponding definitions

Symbols	Definitions
DO	Data Owner
CSP	Cloud Service Provider
RDPC	Remote Data Possession Checking
DV	Designated Verifier
PDP	Provable Data Possession
PoR	Proof of Retrievability
PKI	Public Key Infrastructure
CA	Certificate Authority
CDH	Computational Diffie-Hellman
WDH	Weil Diffie-Hellman
KGC	Key-Generation Center
IBS	Identity-Based Signature
$\mathbb{G}, \mathbb{G}_1, \mathbb{G}_2$	cyclic groups
q	the order of $\mathbb{G}_1$ and $\mathbb{G}_2$
$\mathbb{Z}_q$	$\{0, 1 \cdots, q-1\}$
g	the generator of $\mathbb{G}_1$
e	bilinear map from $\mathbb{G}_1 \times \mathbb{G}_1$ to $\mathbb{G}_2$
H, H_1, H_2	hash functions
params	system parameter
msk	master secret key
trd	trapdoor of DV
F	original data file
Fid	filename of F
n	number of blocks for F
T	authenticated data file
T_{Fid}	generated IBS
chal	the challenge message
Γ	the returned proof

g^a, g^b. The CDH assumption means that, given the tuple (g, g^a, g^b), any PPT algorithm cannot compute and output the element $g^{ab} \in \mathbb{G}_1$.

Weil Diffie-Hellman Assumption

Here, we recall the Weil Diffie-Hellman (WDH) assumption in [26]. Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ be a bilinear map and g be a generator of $\mathbb{G}_1$. Randomly choose a, b, c from $\mathbb{Z}_q$ and compute g^a, g^b, g^c. The WDH assumption refers to that, given the tuple (g, g^a, g^b, g^c), there is no efficient algorithm to compute $e(g, g)^{abc} \in \mathbb{G}_2$.

System model

In this subsection, we formally introduce the system model of IB-DV-PDP protocol. A graphical description can be found in Fig. 1. In particular, the whole system includes four entities: KGC, DO, DV, and CSP.

- KGC: Given the identity of DO or DV, this entity will issue secret key for it.
- DO: This entity is just the data owner, who intends to outsource its original data file to some CSP.
- DV: This is an entity which is designated by DO and will check the integrity of DO's stored file.
- CSP: This entity has great storage spaces and provides storing services for lots of DOs.

Then, an IB-DV-PDP protocol consists of the following seven PPT algorithms.

- $(params, msk) \leftarrow \texttt{Setup}(1^\lambda)$. This algorithm initializes the system and generates the public system parameter *params* as well as the master secret key *msk*.
- $sk_{ID} \leftarrow \texttt{Extract}(msk, ID)$. This algorithm generates secret keys for users (including the DV) with different identities *ID*. Now, the DO (resp. DV) can obtain its secret key sk_O (resp. sk_V) by inputting its identity ID_O (resp. ID_V) into this algorithm.
- $trd \leftarrow \texttt{TrapdoorGen}((sk_O, ID_V)$ or $(sk_V, ID_O))$. This is a trapdoor-generation algorithm, which takes the pair (sk_O, ID_V) or (sk_V, ID_O) as input, and computes a common private trapdoor *trd* between the DO and DV.
- $T \leftarrow \texttt{TagGen}(sk_O, F)$. This algorithm is run by DO to generate an authenticated file T for the original data file F. Then T is transmitted to a CSP for storing.
- $chal \leftarrow \texttt{Challenge}$. This is an algorithm run by the DV (in the auditing phase) to generate a challenge message *chal*, which will be sent to the CSP.
- $\Gamma \leftarrow \texttt{ProofGen}(chal, T)$. When receiving the challenge message *chal* from the DV, the CSP computes a proof Γ based on the stored T.

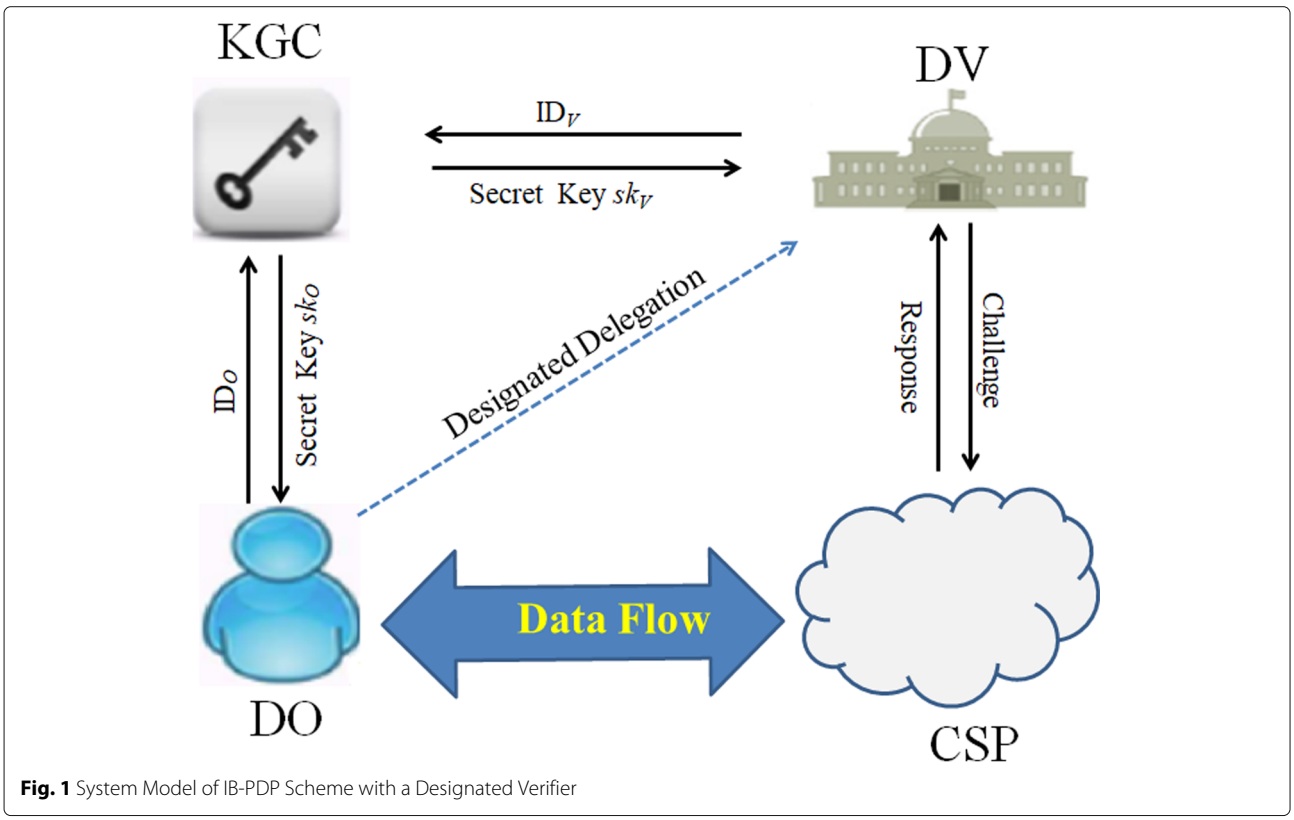

Fig. 1 System Model of IB-PDP Scheme with a Designated Verifier

- $b \leftarrow \texttt{Verify}(trd, chal, \Gamma)$. This is the final checking algorithm of the DV, which will output a bit b based on the challenge message *chal* and the returned proof Γ. If $b = 1$, then the DO's stored data file is intact. Otherwise, its integrity is broken.

When combining the entities with the above algorithms, the system will work as follows. The KGC first runs `Setup` to obtain *params* and *msk*, and broadcasts *params* to other entities. Given the identity *ID*, the KGC runs $sk_{ID} \leftarrow \texttt{Extract}(msk, ID)$ and returns sk_{ID} to the entity with *ID*. After receiving the secret key sk_O and DV's identity ID_V, the DO computes the trapdoor *trd*, which can also be computed by the DV, by running `TrapdoorGen`. Then the DO continues to generate the authenticated file T for the original data file F by performing $\texttt{TagGen}(sk_O, F)$, and transmits T to CSP for storing. In the auditing phase, the DV runs `Challenge` to get *chal* and sends it to CSP, who will compute Γ based on *chal* and the stored T, and then returns it to the DV. Finally, the DV checks the validity of the returned Γ by running `Verify`.

Security model

In this subsection, we define the security model of an IB-DV-PDP protocol. Before describing its definition, we would like to give some underlying intuition. Concretely, here, we need to consider two kinds of attacks. One is to model the misbehavior of malicious CSP. That is, if some stored data blocks are lost by CSP, then the designated verifier can detect it with extremely high probability. Another one is to model the security of DV, which also means that checking whether an un-designated verifier is able to perform the task of auditing or not. For a secure IB-DV-PDP protocol, it should satisfy that any un-designated verifier cannot obtain the corresponding trapdoor and thus cannot perform the task of auditing.

Now, we formally give the definition of the former one, which is modeled by the following security game played between a challenger CH^I and an adversary $\mathcal{CSP}$.

- **Initialization.** The challenger first runs $\texttt{Setup}(1^\lambda)$ to get the public system parameter *params* and the master secret key *msk*. Give *params* to $\mathcal{CSP}$.
- **Queries.** The adversary is allowed to adaptively make the following queries.
  - `Secret-Key-Extract`. $\mathcal{CSP}$ submits an identity *ID* to CH^I, who will run $sk_{ID} \leftarrow \texttt{Extract}(msk, ID)$ and return sk_{ID} to it.
  - `Authenticated-File-Query`. The adversary $\mathcal{CSP}$ submits an identity *ID* and a data file F to CH^I, who runs
    $$T \leftarrow \texttt{TagGen}(sk_{ID}, F)$$
    and returns it to $\mathcal{CSP}$.
  - `Auditing-Query`. This kind of query is based on the queried T in the previous step. More precisely, the challenger first runs $chal \leftarrow \texttt{Challenge}$ and gives *chal* to $\mathcal{CSP}$, who computes and returns a proof Γ by running $\texttt{ProofGen}(chal, T)$. When receiving Γ, CH^I obtains the trapdoor *trd* by running the algorithm $\texttt{TrapdoorGen}(sk_O, ID_V)$, and continues to run
    $$b \leftarrow \texttt{Verify}(trd, chal, \Gamma).$$
    Finally, the bit b is transmitted to $\mathcal{CSP}$.
- **Final Phase.** In this phase, the challenger CH^I submits a challenge message $chal^*$ to $\mathcal{CSP}$ in order to check the integrity of some data file F queried in `Authenticated-File-Query`, who finally returns a proof Γ^*.

If

$$1 \leftarrow \texttt{Verify}(trd, chal^*, \Gamma^*),$$

and the following conditions hold,

1) The identities ID_O and ID_V are not queried to the oracle `Secret-Key-Extract`, and
2) The returned proof Γ^* does not equal to the correct one Γ, which would be honestly computed in $\texttt{ProofGen}(chal^*, T)$,

then we call the IB-DV-PDP protocol is secure against any PPT malicious CSP.

Next, we consider the security on the trapdoor of the DV. In order to clearly introduce the security model on trapdoor, we would like to give some underlying intuition. More concretely, for an adversary, it wants to find the correct trapdoor after seeing the transmissions between the entities, including the initialization, queries and so on. Here, we still introduce the following security game played by a challenger CH^{II} and an adversary $\mathcal{A}$.

- These phases "**Initialization**" and "**Queries**" are the same as the ones appeared in the above game except that CH^I and $\mathcal{CSP}$ are replaced by CH^{II} and $\mathcal{A}$, respectively.
- **Final Phase.** In final, the adversary outputs a trapdoor trd^*. Naturally, the restriction is that $\mathcal{A}$ is not allowed to query ID_O and ID_V to `Secret-Key-Extract` oracle.

If for any PPT adversary $\mathcal{A}$, the probability that its output trd^* equals to the trapdoor *trd* of the DV is negligible, then we call the IB-DV-PDP protocol is secure for the

DV. In other words, this security describes that any undesignated entity is "hard" to obtain the trapdoor and hence is not able to perform the task of auditing.

Our proposed construction and its security

Now, we formally introduce the construction of our proposed IB-DV-PDP protocol as follows.

- $(params, msk) \leftarrow \texttt{Setup}(1^\lambda)$: This is an algorithm that generates the system parameters *params* and KGC's master secret key *msk* when given the security parameter 1^λ. Concretely, it chooses two cyclic groups $\mathbb{G}_1, \mathbb{G}_2$ with the same prime order q ($|q| \geq \lambda$), and e a bilinear map from $\mathbb{G}_1 \times \mathbb{G}_1$ to $\mathbb{G}_2$. Assume that g is a generator of $\mathbb{G}_1$, and H_1, H_2 are two hash functions from $\{0,1\}^*$ to $\mathbb{G}_1$. Moreover, define $\phi : \mathbb{Z}_q^* \times \{1, 2, \cdots, n\} \rightarrow \mathbb{Z}_q^*$, $\pi : \mathbb{Z}_q^* \times \{1, 2, \cdots, n\} \rightarrow \{1, 2, \cdots, n\}$ as pseudorandom function and pseudorandom permutation, respectively. Then randomly choose x from $\mathbb{Z}_q^*$ and compute $P_0 = g^x$. Finally, set

$$params = (\mathbb{G}_1, \mathbb{G}_2, e, q, g, H_1, H_2, P_0, \phi, \pi)$$

and $msk = x$.
- $sk_{ID} \leftarrow \texttt{Extract}(msk, ID)$: This is a secret-key-generation algorithm for a user with identity *ID*. In particular, for the inputs of KGC's master secret key *msk* and user's identity *ID*, it computes $sk_{ID} = H_1(ID)^x$ and outputs it as the generated secret key (for *ID*).
In fact, by running this algorithm, the DO with identity ID_O and the DV with ID_V can obtain their secret keys sk_O and sk_V, respectively.
- $trd \leftarrow \texttt{Trapdoor}((sk_O, ID_V)$ or $(sk_V, ID_O))$: If the input is (sk_O, ID_V), then this algorithm computes the trapdoor *trd* as follows.

$$trd := e\left(sk_O, H_1(ID_V)\right) = e\left(H_1(ID_O), H_1(ID_V)\right)^x.$$

Similarly, if the input is (sk_V, ID_O), then this algorithm computes the trapdoor *trd* as follows.

$$trd = e\left(sk_V, H_1(ID_O)\right) = e\left(H_1(ID_O), H_1(ID_V)\right)^x.$$

- $T \leftarrow \texttt{TagGen}(sk_O, F)$: This is the tag-generation algorithm for the original data file F and run by DO, who has the secret key sk_O. Specifically, first randomly choose a filename *Fid* from $\{0,1\}^\lambda$ (for F). Parse F into n blocks $\{m_1, m_2, \cdots, m_n\}$, and each block m_i has s same-length segments $(m_{i,1}, m_{i,2}, \cdots, m_{i,s}) \in \left(\mathbb{Z}_q\right)^s$. That is,

$$\begin{aligned} F &= \{m_1, \cdots, m_n\} \\ &= \{(m_{1,1}, m_{1,2}, \cdots, m_{1,s}), \cdots, (m_{n,1}, m_{n,2}, \cdots, m_{n,s})\}. \end{aligned}$$

Then randomly choose $u_1, u_2, \cdots, u_s$ from $\mathbb{G}_1$, t from $\mathbb{Z}_q^*$, and compute

$$T_i = sk_O \cdot \left(H_2\left(trd||Fid||i\right) \cdot \prod_{j=1}^{s} u_j^{m_{i,j}}\right)^t, \tag{1}$$

for each $i \in [n]$. Define $R = g^t$ and select an identity-based signature (IBS) algorithm `IBS.Sig` to compute

$$T_{Fid} = \texttt{IBS.Sig}(u_1||u_2||\cdots||u_s||R||Fid).$$

Now, define

$$T = (F, u_1, u_2, \cdots, u_s, R, Fid, T_{Fid}, T_1, T_2, \cdots, T_n)$$

and output it, which will be transmitted to CSP for storing.
- $chal \leftarrow \texttt{Challenge}$: This is the challenge-message-generation algorithm and run by the DV. In particular, in order to check the integrity of user's data, the designated verifier randomly chooses ℓ from $[n]$, which denotes the number of challenged blocks, and k_1, k_2 from $\mathbb{Z}_q^*$. Output the challenge message $chal = (\ell, k_1, k_2)$, which will be sent to CSP.
- $\Gamma \leftarrow \texttt{ProofGen}(chal, T)$: This algorithm will generate the returned proof Γ and hence run by the CSP. Specifically, when receiving the challenge message *chal* from the verifier, it first computes

$$a_i = \phi(k_1, i) \in \mathbb{Z}_q^*,\ c_i = \pi(k_2, i) \in [n] \tag{2}$$

for $1 \leq i \leq \ell$. Then calculate

$$\sigma = \prod_{i=1}^{\ell} T_{c_i}^{a_i}, \text{ and } M = \sum_{i=1}^{\ell} a_i \cdot m_{c_i} \in \left(\mathbb{Z}_q\right)^s.$$

Finally, return

$$\Gamma = (u_1, u_2, \cdots, u_s, R, Fid, T_{Fid}, \sigma, M)$$

to the DV.
- $1/0 \leftarrow \texttt{Verify}(trd, chal, \Gamma)$: After receiving the returned proof Γ from CSP, the DV first checks if T_{Fid} is a valid signature on the message $(u_1||u_2||\cdots||u_s||R||Fid)$. If it isn't, refuse this proof and output 0. Otherwise, compute a_i, c_i from k_1, k_2 as in (2). Then check if

$$\begin{aligned} e\left(\sigma, g\right) = {} & e\left(H_1(ID_O), P_0\right)^{\sum_{i=1}^{\ell} a_i} \cdot \\ & e\left(\prod_{i=1}^{\ell} \left(H_2\left(trd||Fid||c_i\right)^{a_i}, R\right)\right) \cdot e\left(\prod_{j=1}^{s} u_j^{M_j}, R\right), \end{aligned} \tag{3}$$

in which M_j is the jth component of M. If it is, output 1. Otherwise, output 0.

The correctness of (3) can be verified as follows.

$$
\begin{aligned}
e(\sigma, g) &= e\left(\prod_{i=1}^{\ell} T_{c_i}^{a_i}, g\right) \\
&= e\left(\prod_{i=1}^{\ell}\left(sk_O \cdot \left(H_2(trd||Fid||c_i) \cdot \prod_{j=1}^{s} u_j^{m_{c_i j}}\right)^t\right)^{a_i}, g\right) \\
&= e\left(\prod_{i=1}^{\ell}\left(sk_O^{a_i} \cdot \left(H_2(trd||Fid||c_i) \cdot \prod_{j=1}^{s} u_j^{m_{c_i j}}\right)^{ta_i}\right), g\right) \\
&= e\left(\prod_{i=1}^{\ell} sk_O^{a_i}, g\right) \cdot e\left(\prod_{i=1}^{\ell}\left(H_2(trd||Fid||c_i) \cdot \prod_{j=1}^{s} u_j^{m_{c_i j}}\right)^{a_i}, R\right) \\
&= e\left(\prod_{i=1}^{\ell} H_1(ID_O)^{x a_i}, g\right) \cdot \\
&\qquad e\left(\prod_{i=1}^{\ell}\left(H_2(trd||Fid||c_i)^{a_i} \cdot \prod_{j=1}^{s} u_j^{a_i m_{c_i j}}\right), R\right) \\
&= e\left(H_1(ID_O)^{\sum_{i=1}^{\ell} a_i}, P_0\right) \cdot e\left(\prod_{i=1}^{\ell}\left(H_2(trd||Fid||c_i)^{a_i}, R\right)\right) \cdot \\
&\qquad e\left(\prod_{j=1}^{s}\left(u_j^{\sum_{i=1}^{\ell} a_i m_{c_i j}}\right), R\right) \\
&= e\left(H_1(ID_O), P_0\right)^{\sum_{i=1}^{\ell} a_i} \cdot e\left(\prod_{i=1}^{\ell}\left(H_2(trd||Fid||c_i)^{a_i}, R\right)\right) \cdot \\
&\qquad e\left(\prod_{j=1}^{s} u_j^{M_j}, R\right).
\end{aligned}
$$

Brief review of yan et al.'s dV-PDP protocol

In order to clearly express our improvement on Yan et al.'s DV-PDP protocol, here, we briefly review their protocol. More precisely, the public-private key pairs of DO and DV are $(x, X = g^x) \in \mathbb{Z}_q^* \times \mathbb{G}_1$ and $(y, Y = g^y) \in \mathbb{Z}_q^* \times \mathbb{G}_1$, respectively. Then the DO (resp. DV) can calculate the trapdoor $\alpha = H_1(Y^x) \in \{0,1\}^k$ (resp. $\alpha = H_1(X^y)$). For each data block $m_i \in \mathbb{Z}_q^*$, compute its tag

$$T_i = \left(H_2(\alpha||F_{id}||i) \cdot u^{m_i}\right)^x \in \mathbb{G}_1,$$

where F_{id} is the unique identification of the file, and $u \in \mathbb{G}_1$ is public. The computations of challenge message *chal* and returned proof Γ are the same as the ones in our construction. In the final verification algorithm, the DV computes and checks if

$$e(\sigma, g) = e\left(\prod_{i=1}^{\ell} H_2(\alpha||F_{id}||c_i)^{a_i} \cdot u^M, X\right).$$

If it is, then output 1. Otherwise, output 0.

Security analysis

In this subsection, we analyze the security of the IB-DV-PDP protocol presented in the above section.

Theorem 1 *If the CDH assumption holds in $\mathbb{G}_1$, and H_1, H_2 are seen as random oracles, then the above IB-DV-PDP protocol is secure against any malicious CSP in the random oracle.*

Before presenting the detailed proof, we would like to introduce the underlying reasons why the security of IB-DV-PDP can be reduced to the CDH assumption. In particular, given the tuple $(g, g^a, g^b) \in \mathbb{G}_1^3$, the algorithm $\mathcal{B}$, who tries to solve the CDH problem, can set $P_0 = g^a$ as the master public key since the malicious CSP is not allowed to query the master secret key in the whole query process. The extracted secret key for each user can also be simulated by randomly masking the group element P_0. The trickiest step is the simulation of tag-generation for each data block, which can be handled by embedding $\mathcal{B}$'s another group element g^b. Then the auditing query is natural and easy to deal with.

Proof. Let $\mathcal{CSP}$ be a PPT CSP, who attacks on the proposed IB-DV-PDP protocol. Consider $\mathcal{B}$ as an attacker on the CDH assumption. That is, given the tuple $(g, g^a, g^b) \in \mathbb{G}_1^3$, its target is to output the element g^{ab} for the unknown $a, b \in \mathbb{Z}_q$. Now, $\mathcal{B}$ simulates the environment for $\mathcal{CSP}$ and wants to obtain its answer by using $\mathcal{CSP}$ as a subroutine.

- **Initialization.** $\mathcal{B}$ first generates *params* as in `Setup` except for the item P_0. More precisely, it sets $P_0 = g^a$ and implicitly defines the master secret key x as the unknown a. Give *params* to $\mathcal{CSP}$. In addition, $\mathcal{B}$ also initializes an empty list L used to store the query-answer pairs.
- **Queries.** The adversary $\mathcal{CSP}$ is allowed to adaptively make the following queries.
  - `Hash-Queries.` For the query ID to the H_1-oracle, $\mathcal{B}$ checks if $(ID, H_1(ID))$ is in the list L. If it is, return $H_1(ID)$ to $\mathcal{CSP}$. Otherwise randomly choose $\tau \in \mathbb{Z}_q$ and compute $H_1(ID) = g^\tau$. Return $H_1(ID)$ to $\mathcal{CSP}$ and store $(ID, H_1(ID))$ into L. In addition, $\mathcal{B}$ also defines $H_1(ID_O)$ as g^b.
  - `Secret-Key-Extract.` $\mathcal{CSP}$ submits an identity ID to $\mathcal{B}$, who computes

    $$sk_{ID} = (g^a)^\tau = g^{a\tau} = (H_1(ID))^a.$$

    Here, τ is the random chosen element when answering the query to H_1-oracle on ID. Since $\mathcal{CSP}$ is not allowed to query the secret key of ID_O, $\mathcal{B}$ implicitly sets $sk_O = (g^b)^a = g^{ab}$.
  - `Authenticated-File-Query.` The adversary $\mathcal{CSP}$ submits an identity ID and a data file F to $\mathcal{B}$. If $ID \neq ID_O$, then $\mathcal{B}$ normally answers this query because it has the true secret key sk_{ID}. Else if $ID = ID_O$, then $\mathcal{B}$ will

simulate as follows. First, compute the trapdoor trd by running $\texttt{Trapdoor}(sk_V, ID_O)$. Then randomly choose Fid as the filename of F and parse F into n blocks

$$F = \{m_1, \cdots, m_n\} = \{(m_{1,1}, \cdots, m_{1,s}), \cdots, (m_{n,1}, \cdots, m_{n,s})\}.$$

Randomly choose $x_1, \cdots, x_s, r$ from $\mathbb{Z}_q$ and compute

$$u_1 = (g^b)^{x_1}, \cdots, u_s = (g^b)^{x_s}, R = (g^a)^r.$$

That is,

$$\prod_{j=1}^{s} u_j^{m_{i,j}} = \prod_{j=1}^{s} \left(g^b\right)^{x_j m_{i,j}} = \left(g^b\right)^{\sum_{j=1}^{s} x_j m_{i,j}},$$

and implicitly set $t = ar$. Moreover, define

$$H_2(trd||Fid||i) = \left(g^b\right)^{\alpha_i} \cdot g^{\beta_i},$$

where α_i satisfies

$$1 + r\alpha_i + r\sum_{j=1}^{s} x_j m_{i,j} = 0.$$

Here, we know that, for $1 \le i \le n$, it holds that

$$\begin{aligned} T_i &= sk_O \cdot \left(H_2\left(trd||Fid||i\right) \cdot \prod_{j=1}^{s} u_j^{m_{i,j}}\right)^t \\ &= g^{ab} \cdot \left(\left(g^b\right)^{\alpha_i} g^{\beta_i} \cdot \left(g^b\right)^{\sum_{j=1}^{s} x_j m_{i,j}}\right)^{ar} \\ &= g^{ab}\left(g^{(ab)(r\alpha_i + r\sum x_j m_{i,j})} \cdot \left(g^a\right)^{r\beta_i}\right) \\ &= \left(g^{ab}\right)^{1 + r\alpha_i + r\sum x_j m_{i,j}} \cdot \left(g^a\right)^{r\beta_i} \\ &= \left(g^a\right)^{r\beta_i}. \end{aligned}$$

Finally, generate T_{Fid} by signing the item

$$u_1||u_2||\cdots||u_s||R||Fid,$$

and return

$$T = (F, u_1, u_2, \cdots, u_s, R, Fid, T_{Fid}, T_1, T_2, \cdots, T_n)$$

to $\mathcal{CSP}$.

– $\texttt{Auditing-Query}$. In this step, $\mathcal{B}$ simulates the DV and sends a challenge message $chal$ to $\mathcal{CSP}$, who computes and returns a proof Γ. Then $\mathcal{B}$ checks the validity of Γ by running

$$b \leftarrow \texttt{Verify}(trd, chal, \Gamma),$$

and gives the bit b to $\mathcal{CSP}$.

- **Final Phase.** In this phase, the algorithm $\mathcal{B}$ submits a challenge message $chal^* = (\ell, k_1, k_2)$ to $\mathcal{CSP}$, who returns a forged proof

$$\Gamma^* = \left(u_1, u_2, \cdots, u_s, R, Fid, T_{Fid}, \sigma^*, M^*\right).$$

Let

$$\Gamma = (u_1, u_2, \cdots, u_s, R, Fid, T_{Fid}, \sigma, M)$$

be the correct proof, which would be honestly computed in $\texttt{ProofGen}(chal^*, T)$. Hence, $(\sigma^*, M^*) \neq (\sigma, M)$.

Since both of the proofs Γ and Γ^* can pass the verification of (3), it holds that

$$e\left(\sigma, g\right) = e\left(H_1(ID_O), P_0\right)^{\sum_{i=1}^{\ell} a_i} \cdot e\left(\prod_{i=1}^{\ell}\left(H_2\left(trd||Fid||c_i\right)^{a_i}, R\right)\right) \cdot e\left(\prod_{j=1}^{s} u_j^{M_j}, R\right), \tag{4}$$

and

$$e\left(\sigma^*, g\right) = e\left(H_1(ID_O), P_0\right)^{\sum_{i=1}^{\ell} a_i} \cdot e\left(\prod_{i=1}^{\ell}\left(H_2\left(trd||Fid||c_i\right)^{a_i}, R\right)\right) \cdot e\left(\prod_{j=1}^{s} u_j^{M_j^*}, R\right). \tag{5}$$

Divide (4) by (5) and we can get

$$\begin{aligned} e(\sigma/\sigma^*, g) &= e\left(\prod_{j=1}^{s} u_j^{M_j - M_j^*}, R\right) \\ &= e\left(\prod_{j=1}^{s}\left(g^b\right)^{x_j(M_j - M_j^*)}, g^{ar}\right) \\ &= e\left(\left(g^b\right)^{\sum x_j(M_j - M_j^*)}, g^{ar}\right) \\ &= e\left(\left(g^{ab}\right)^{\sum r x_j(M_j - M_j^*)}, g\right). \end{aligned}$$

Because $r, x_1, x_2, \cdots, x_s$ are random and at least one $(M_j - M_j^*)$ does not equal to 0, we know that $\sum_{j=1}^{s} r x_j(M_j - M_j^*) = 0$ occurs with probability $1/q$, which is negligible. Therefore, $\mathcal{B}$ can obtain the answer g^{ab} by computing

$$g^{ab} = \left(\sigma/\sigma^*\right)^{\frac{1}{\sum r x_j(M_j - M_j^*)}}.$$

This ends the proof of Theorem 1.

As for the security of the DV, we have the following:

Theorem 2 *If the WDH assumption holds for the bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$, then our proposed IB-DV-PDP protocol is secure for the designated verifier.*

Now, we first explain the underlying intuition of this theorem's proof. Given the tuple $(g, g^a, g^b, g^c) \in \mathbb{G}_1^4$, the algorithm $\mathcal{B}'$ attacking on the WDH assumption can naturally set g^a as the master public key P_0, and g^b, g^c as the random-oracle queries to H_1 for the identities

ID_O and ID_V, respectively. Then the finding of trapdoor $trd = e(g,g)^{abc}$ for un-designated verifier is just the WDH-solution of $\mathcal{B}'$. This is also the reason why WDH assumption is used in this paper.

Proof. Assume that $\mathcal{A}$ is an adversary who wants to find out the trapdoor of the DV. Then we will construct another algorithm $\mathcal{B}'$ attacking on the WDH assumption. In particular, given the tuple $(g, g^a, g^b, g^c) \in \mathbb{G}_1^4$, $\mathcal{B}'$ intends to compute and output the element $e(g,g)^{abc} \in \mathbb{G}_2$. Hence, it simulates the environment for $\mathcal{A}$ and invokes $\mathcal{A}$ as a subroutine.

The simulation of $\mathcal{B}'$ is similar to that of the algorithm $\mathcal{B}$ constructed in Theorem 1. That is, $\mathcal{B}'$ also sets $P_0 = g^a$, $H_1(ID_O) = g^b$. Moreover, it defines $H_1(ID_V) = g^c$. In this way, $\mathcal{B}'$ can simulate all the queries from $\mathcal{A}$ as in the above theorem. Finally, when $\mathcal{A}$ outputting trd^* as a guess of the trapdoor of DV, $\mathcal{B}'$ also outputs it as its answer to the WDH assumption.

From the simulation of $\mathcal{B}'$, we know that

$$trd = e(sk_O, H_1(ID_V)) = e(g^{ab}, g^c) = e(g,g)^{abc}.$$

Therefore, if $\mathcal{A}$ can correctly find the trapdoor trd, then $\mathcal{B}'$ is also able to solve the WDH assumption. From the hardness of WDH assumption, we know that our proposed IB-DV-PDP protocol is secure for the DV. This also ends the proof of Theorem 2.

Further improvement

Note that, in our proposed protocol, the authenticated file T has the form of

$$T = (F, u_1, u_2, \cdots, u_s, R, Fid, T_{Fid}, T_1, T_2, \cdots, T_n).$$

In order to further reduce the communication cost, we suggest the following improvements. Define a new hash function $H : \{0,1\}^* \to \mathbb{G}_1$. Then generate

$$u_1 = H(Fid||1), u_2 = H(Fid||2), \cdots, u_s = H(Fid||s),$$

and define T_{Fid} as the IBS of $(R||Fid)$. In the security proof, the hash function H is still viewed as a random oracle, which provides randomness for $u_1, u_2, \cdots, u_s$. In this case, the communications of $u_1, u_2, \cdots, u_s$ can be reduced from T. Similarly, they can also be reduced from the returned proof Γ because the DV can compute them on-line from Fid.

Error detection probability

Since our proposed IB-DV-PDP protocol adopts the random sampling method to detect the corruption of user's data, we now discuss its error detection probability for the DV. Specifically, in our protocol, the DV chooses ℓ blocks in each challenge. Assume that d blocks are corrupted by CSP and define X as a random variable, which describes the number of challenged blocks matching the corrupted ones. Moreover, P_X denotes the probability that CSP's misbehavior is detected. Hence, we have

$$\begin{aligned} P_X &= \Pr\{X \geq 1\} = 1 - \Pr\{X = 0\} \\ &= 1 - \frac{n-d}{n} \cdot \frac{n-d-1}{n-1} \cdots \frac{n-d-(\ell-1)}{n-\ell+1} \\ &\geq 1 - \left(\frac{n-d}{n}\right)^{\ell}. \end{aligned}$$

Obviously, the more challenged blocks, the higher error detection probability. If 5,000 blocks out of 1,000,000 ones are tampered, then the error detection probability P_X is greater than 80% when challenging 321 blocks (for the DV). Similarly, if 10,000 blocks are corrupted, then randomly choosing only 300 blocks will realize that the error detection probability P_X is at least 95%.

Performances analysis

In this section, we evaluate the performances of our proposed IB-DV-PDP protocol from communication overhead, storage and computational costs as well as the experimental result. In order to present the practicality of our protocol, we compare it with two other IB-PDP protocols and a certificateless PDP protocol with privacy-preserving property, which were designed in [8] [9], and [4], respectively.

Communication overhead

The communication contents for an IB-PDP protocol include the parts of transmitting the authenticated file T from data owner to CSP, a challenge message $chal$ from the DV to CSP, and the returned proof Γ from CSP to the DV. Now, we respectively denoted by *DOtoCSP*, *DVtoCSP*, and *CSPtoDV* the corresponding communication overheads.

Recall that, in our protocol, the DO will send the authenticated file

$$T = (F, R, Fid, T_{Fid}, T_1, T_2, \cdots, T_n)$$

to CSP. Thus, the communication overhead (i.e. *DOtoCSP*) for our protocol equals to

$$|\mathbb{G}_1| + |Fid| + |T_{Fid}| + n \cdot |\mathbb{G}_1| = (n+1) \cdot |\mathbb{G}_1| + \lambda + |T_{Fid}|.$$

Similarly, we can evaluate the communication overheads from DO to CSP for the three protocols in [8], [9] and [4] as

$$(n+2) \cdot |\mathbb{G}_1| + \lambda + |T_{Fid}|,$$

$$(n+1) \cdot |\mathbb{G}_1| + \lambda + |T_{Fid}|,$$

and

$$n \cdot (\lambda + |n| + |\mathbb{G}_1|),$$

respectively.

The challenge message *chal* in our protocol is

$$chal = (\ell, k_1, k_2) \in [n] \times \mathbb{Z}_q^* \times [n],$$

which has the length of $2 \cdot |n| + |\mathbb{Z}_q^*|$. The challenge-message-generation algorithm in [8] is the same as that of our protocol and thus *DVtoCSP* for this protocol is also $2 \cdot |n| + |\mathbb{Z}_q^*|$. In addition, in [9], the challenge message is

$$chal = (c_1, c_2, \{(i, v_i)\}_{i=1}^{\ell}, pf),$$

in which, c_1, c_2 are in $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, $i \in [n]$, $v_i \in \mathbb{Z}_q^*$, and pf is a proof of knowledge. Therefore, the communication overhead *DVtoCSP* for this protocol equals to

$$|\mathbb{G}_1| + |\mathbb{G}_2| + \ell \cdot (|n| + |\mathbb{Z}_q^*|) + |pf|.$$

We also know that the challenge message of Ji et al.'s protocol has length of

$$\ell \cdot (|n| + |\mathbb{Z}_q|).$$

Finally, in our protocol, the returned proof Γ has the form of

$$\Gamma = (R, Fid, T_{Fid}, \sigma, M),$$

which has the length of

$$|\mathbb{G}_1| + |T_{Fid}| + |\mathbb{G}_1| + |\mathbb{Z}_q| = 2 \cdot |\mathbb{G}_1| + |T_{Fid}| + |\mathbb{Z}_q|.$$

Similarly, we are able to compute the communication overheads from CSP to DV for [8], [9], and [4] as

$$4 \cdot |\mathbb{G}_1| + |T_{Fid}| + |\mathbb{Z}_q|,$$

$$|\mathbb{G}_1| + |T_{Fid}| + |\mathbb{G}_2|,$$

and

$$3|\mathbb{G}_1| + |T_{Fid}|,$$

respectively.

The total comparisons on the communication overheads are listed in Table 2.

Storage cost

Now, we analyze the storage costs of the chosen protocols. First, in our protocol, there are four entities: KGC, DO, DV, and CSP. In the running of this protocol, the KGC will store its own master secret key $msk = x \in \mathbb{Z}_q$, which has length of $|\mathbb{Z}_q|$. For DO, after outsourcing its original authenticated data file to CSP, it only needs to store its private key $sk_{ID} = H_1(ID)^x \in \mathbb{G}_1$, which has length of $|\mathbb{G}_1|$. For CSP, it will store the transmitted data file T (from DO), which has length of

$$(n+1) \cdot |\mathbb{G}_1| + \lambda + |T_{Fid}|.$$

Finally, the DV needs to store its own private key $sk_V \in \mathbb{G}_1$ and the challenged message $chal = (\ell, k_1, k_2) \in [n] \times \mathbb{Z}_q^* \times \mathbb{Z}_q^*$, which have lengths of $|\mathbb{G}_1|$ and $|n| + 2|\mathbb{Z}_q^*|$, respectively. Note that the storage cost of CSP just equals to the communication cost from DO to CSP, which has be discussed in the previous subsection. Hence, we do not consider it in the following parts.

Similarly, we can evaluate the storage costs of the involved entities for other protocols, and present the comparisons in Table 3.

Computational cost (Theoretical analysis)

Now, we analyze the computational cost of our protocol, which mainly consists of the computations of authenticated data file T (for DO), returned proof Γ (for the CSP), and the verification of Γ. The computational cost mainly relies on the expensive operations like pairing, multiplication, and exponentiation, since other operations such as the hash function or addition on group only have negligible costs. For clarity of the theoretical analysis, we denote by T_p, T_{mul}, and T_{exp} the computational costs for pairing, multiplication and exponentiation (on group $\mathbb{G}_1$), respectively.

First, in our protocol, the generation of the authenticated data file T will need to compute n tags

$$T_i = sk_O \cdot \left(H_2\left(trd||Fid||i\right) \cdot \prod_{j=1}^{s} u_j^{m_{i,j}} \right)^t,$$

$u_1, u_2, \cdots, u_s, R$, and $T_{Fid} = \texttt{IBS.Sig}(R||Fid)$. Hence, the computation cost is

$$\begin{aligned} &n \cdot ((s+1) \cdot T_{exp} + (s+2) \cdot T_{mul}) + (s+1) \cdot T_{exp} + T_{IBS} \\ &= (ns + n + s + 1) \cdot T_{exp} + n(s+2) \cdot T_{mul} + T_{IBS}, \end{aligned}$$

where T_{IBS} denotes the time-consumption of generating a signature in an IBS scheme. If we set $s = 1$ as in [8], then the cost is

$$(2n+2) \cdot T_{exp} + (3n) \cdot T_{mul} + T_{IBS}.$$

Table 2 The comparisons on the communication overheads

Protocol	*DOtoCSP*	*DVtoCSP*	*CSPtoDV*
Our Protocol	$(n+1) \cdot \lvert\mathbb{G}_1\rvert + \lambda + \lvert T_{Fid}\rvert$	$2 \cdot \lvert n\rvert + \lvert\mathbb{Z}_q^*\rvert$	$2 \cdot \lvert\mathbb{G}_1\rvert + \lvert T_{Fid}\rvert + \lvert\mathbb{Z}_q\rvert$
Li et al. [8]	$(n+2) \cdot \lvert\mathbb{G}_1\rvert + \lambda + \lvert T_{Fid}\rvert$	$2 \cdot \lvert n\rvert + \lvert\mathbb{Z}_q^*\rvert$	$4 \cdot \lvert\mathbb{G}_1\rvert + \lvert T_{Fid}\rvert + \lvert\mathbb{Z}_q\rvert$
Yu et al. [9]	$(n+1) \cdot \lvert\mathbb{G}_1\rvert + \lambda + \lvert T_{Fid}\rvert$	$\lvert\mathbb{G}_1\rvert + \lvert\mathbb{G}_2\rvert + \ell \cdot (\lvert n\rvert + \lvert\mathbb{Z}_q^*\rvert) + \lvert pf\rvert$	$\lvert\mathbb{G}_1\rvert + \lvert T_{Fid}\rvert + \lvert\mathbb{G}_2\rvert$
Ji et al. [4]	$n \cdot (\lambda + \lvert n\rvert + \lvert\mathbb{G}_1\rvert)$	$\ell \cdot (\lvert n\rvert + \lvert\mathbb{Z}_q^*\rvert)$	$3\lvert\mathbb{G}_1\rvert + \lvert T_{Fid}\rvert$

Table 3 The comparisons on the storage costs

Protocol	KGC	DO	DV or TPA
Our Protocol	$\lvert\mathbb{Z}_q\rvert$	$\lvert\mathbb{G}_1\rvert$	$\lvert n\rvert + 2\lvert\mathbb{Z}_q\rvert + \lvert\mathbb{G}_1\rvert$
Li et al. [8]	$\lvert\mathbb{Z}_q\rvert$	$\lvert\mathbb{G}_1\rvert$	$\lvert n\rvert + 2\lvert\mathbb{Z}_q\rvert + \lvert Fid\rvert$
Yu et al. [9]	$\lvert\mathbb{Z}_q\rvert$	$\lvert\mathbb{G}_1\rvert$	$\lvert n\rvert + \lvert\mathbb{Z}_q\rvert + \lvert\mathbb{G}_1\rvert + \lvert\mathbb{G}_2\rvert + \lvert pf\rvert$
Ji et al. [4]	$\lvert\mathbb{Z}_q\rvert$	$2\lvert\mathbb{Z}_q\rvert$	$\lvert n\rvert + \lvert\mathbb{Z}_q\rvert$

Similarly, we can evaluate the computational costs for the authenticated data file T in [8], [9], and [4] as

$$(2n+1)\cdot T_{exp}+(2n)\cdot T_{mul}+T_S,$$

where T_S is the time-consumption for a signature algorithm (like BLS-signature),

$$(2n+1)\cdot T_{exp}+n\cdot T_{mul}+T_{IBS},$$

and

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

respectively.

Next, we consider the computational cost of generating the proof Γ. In our protocol, Γ is generated by computing

$$\sigma=\prod_{i=1}^{\ell}T_{c_i}^{a_i},\ \text{and}\ M=\sum_{i=1}^{\ell}a_i\cdot m_{c_i},$$

which costs

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

However, the time-consumptions of Γ in [8], [9], and [4] are

$$(\ell+1)\cdot T_{exp}+(\ell-1)\cdot T_{mul},$$

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

and

$$(\ell+3)\cdot T_{exp}+\ell\cdot T_{mul}$$

respectively.

Finally, the computational costs of verifying Γ for the three protocols are analyzed as follows. In our protocol, the DV will first verify the validity of T_{Fid}, whose time-consumption is denoted by T_{Ver}, and then check the equality (3). The total computational cost is

$$T_{Ver}+3\cdot T_p+(2\ell+s)\cdot T_{exp}+(2\ell+s-2)\cdot T_{mul}.$$

If $s=1$, then it is

$$T_{Ver}+3\cdot T_p+(2\ell+1)\cdot T_{exp}+(2\ell-2)\cdot T_{mul}.$$

Similarly, we can evaluate the verifications of other protocols and calculate their computational costs as

$$T'_{Ver}+3\cdot T_p+(2\ell+2)\cdot T_{exp}+(2\ell+1)\cdot T_{mul},$$

where T'_{Ver} denotes the time-consumption for the verification of a signature scheme,

$$T_{Ver}+T_p+(\ell+1)\cdot T_{exp}+(\ell-1)\cdot T_{mul},$$

and

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

respectively.

In addition, we remark that our protocol is the first one for a designated verifier but the other two protocols are not. The total comparisons on the computational costs are listed in Table 4.

Table 4 The comparisons on the computational costs

Protocol	T	Γ	Verify	DV
Our Protocol	$(2n+2)\cdot T_{exp}+(3n)\cdot T_{mul}+T_{IBS}$	$\ell\cdot T_{exp}+(\ell-1)\cdot T_{mul}$	$T_{Ver}+3\cdot T_p+(2\ell+1)\cdot T_{exp}+(2\ell-1)\cdot T_{mul}$	Yes
Li et al. [8]	$(2n+1)\cdot T_{exp}+(2n)\cdot T_{mul}+T_S$	$(\ell+1)\cdot T_{exp}+(\ell-1)\cdot T_{mul}$	$T'_{Ver}+3\cdot T_p+(2\ell+2)\cdot T_{exp}+(2\ell+1)\cdot T_{mul}$	No
Yu et al. [9]	$(2n+1)\cdot T_{exp}+n\cdot T_{mul}+T_{IBS}$	$(\ell+1)\cdot T_{exp}+\ell\cdot T_{mul}+T_p$	$T_{Ver}+T_p+(\ell+1)\cdot T_{exp}+(\ell-1)\cdot T_{mul}$	No
Ji et al. [4]	$3n\cdot T_{exp}+3n\cdot T_{mul}$	$(\ell+3)\cdot T_{exp}+\ell\cdot T_{mul}$	$2\cdot T_p+(\ell+1)\cdot T_{exp}+(\ell+2)\cdot T_{mul}$	No

Experimental results

In order to further evaluate the performance of our proposed protocol, we implement it within the framework of "Charm" [27]. In particular, the 512-bit SS elliptic curve from pairing-based cryptography (PBC) library is set as the basis of our experiments [28]. The DO and DV are simulated by a Huawei MateBook with the configuration of Intel Core i5-6200U CPU @2.3GHz and 16GB RAM. Then the CSP is simulated by a Huawei FusionServer 2288H V5 with the configuration of Xeor Bronze 3106@1.7GHz, 16GB RAM.

Since, in the three protocols, standard signature or IBS scheme is needed, we choose the BLS-signature [29] and the IBS scheme in [30] or [4] as building blocks to implement them. Now, we choose a data file with size of 500 MB, which is parsed as 100, 200, 300, 400, and 500 blocks. Then the time-consumptions for the generations of authenticated files in the three protocols are listed in Fig. 2. For each authenticated file, we set the numbers of challenged blocks as 30, 60, 90, 120, and 150. Then the time-consumptions for the generation of returned proof and the verification for DV are presented in Fig 3 and 4, respectively.

From the comparisons of experimental results, we can see that our proposed protocol is competitive especially for the generation of the returned proof. However, we still explain that our protocol is the first one for the DV but the other ones are not. Hence, our protocol does not obviously reduce IB-PDP's efficiency, and can be naturally used in future real-life applications.

Conclusions

In many practical scenarios, the data owner only hopes some designated verifier to perform the auditing task, which is not covered by private or public auditing model. Yan et al. recently designed an auditing protocol with a designated verifier to resolve this problem. But their protocol naturally suffers from complicated management of certificates and heavy dependence on PKI. In this paper, for the first time, we introduce the identity-based remote data checking with a designated verifier. Compared with

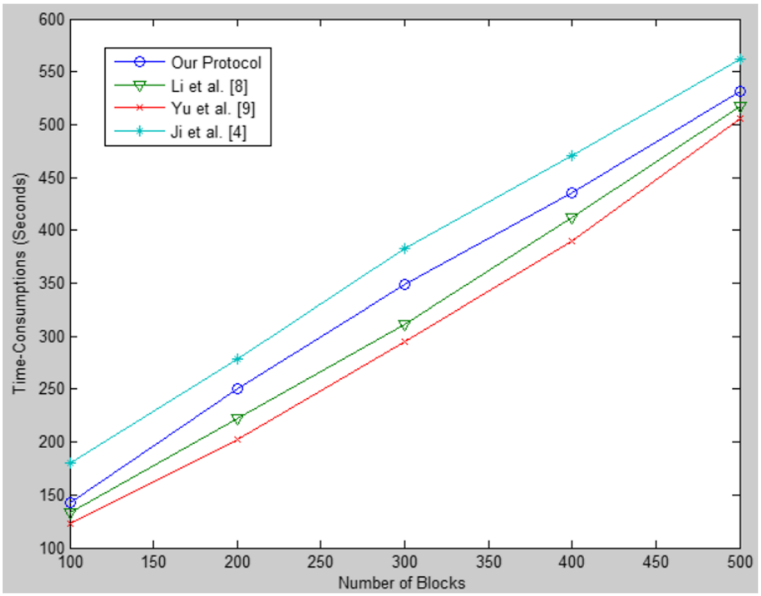

Fig. 2 Time-Consumptions for the Generations of Authentication Files

the existing PDP protocol with a designated verifier, our protocol avoids the introduction of PKI and management of public key certificates. Moreover, we also give the security model and prove that our protocol is provable secure based on the CDH and WDH assumptions in the random oracle model. The final analysis on performance shows that this protocol is also efficient and can be used in future real-life applications.

Future works. The first interesting problem is how to design more efficient IB-DV-PDP protocol so that it can

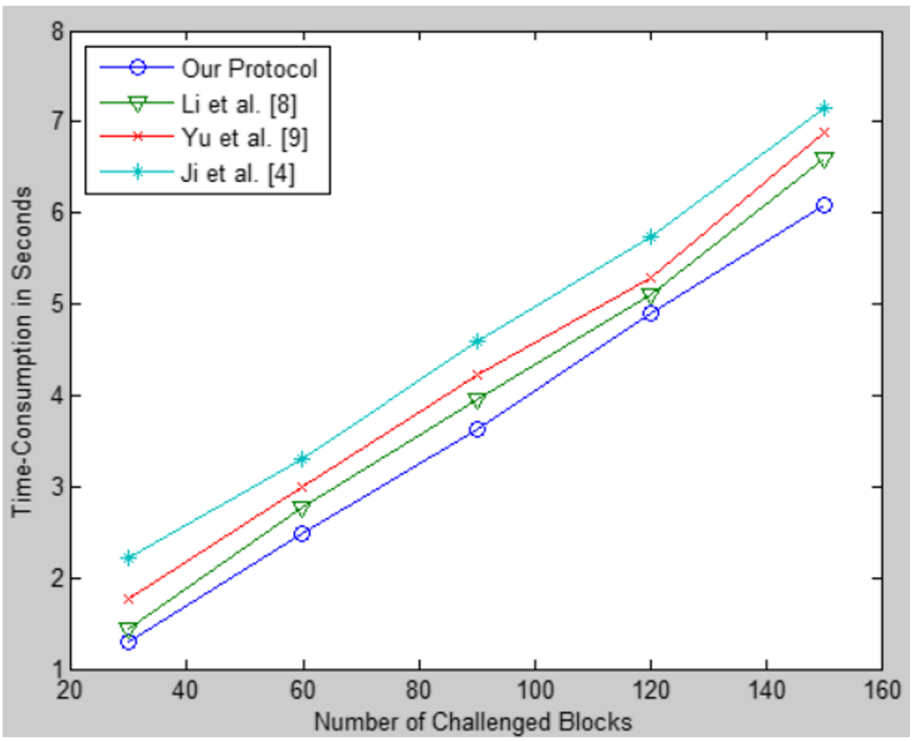

Fig. 3 Time-Consumption for the Generation of Returned Proof

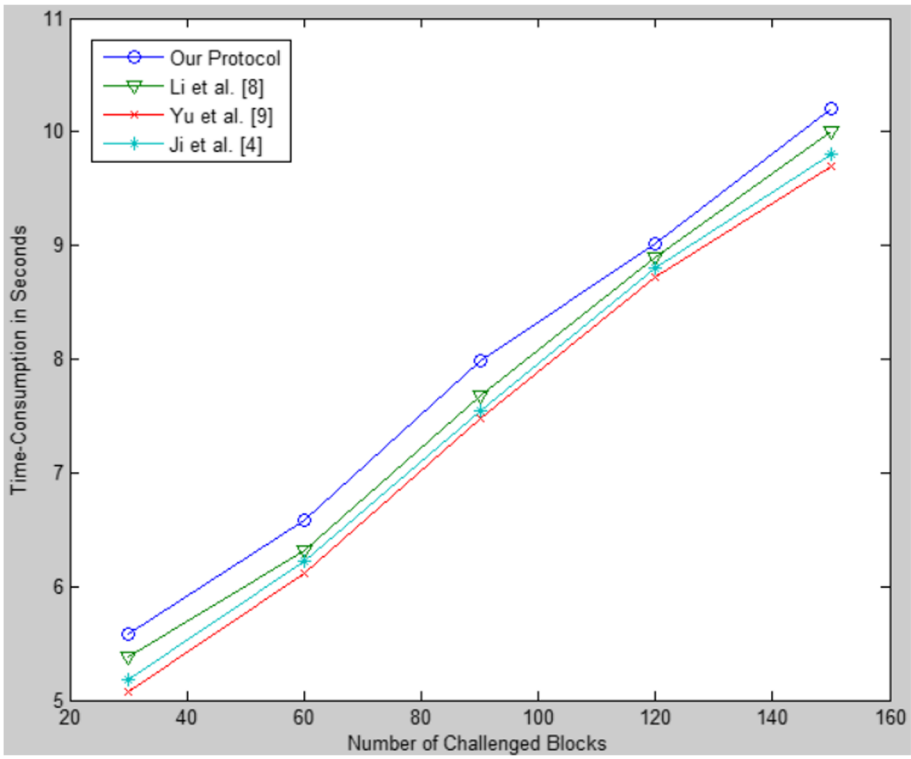

Fig. 4 Time-Consumption for the DV's Verification

find applications in practical scenarios. In addition, note that the security of our proposed IB-DV-PDP protocol relies on the ideal random oracle, and thus the second future work is finding and designing an efficient IB-DV-PDP protocol, which is provable secure in the standard model.

Acknowledgments
The authors would like to thank anonymous referees for their valuable suggestions and comments.

Authors' contributions
Yanyan Ji and Bilin Shao gave the main idea of this paper. The other three authors had worked equally during all this paper's stages. The authors read and approved the final manuscript.

Authors' information
Not applicable.

Funding
This work is supported in part by National Natural Science Foundation of China (No. 61872284), and in part by Foundation of SKLOIS (No. 2021-MS-04).

Availability of data and materials
The data and materials are available from the corresponding author on reasonable request.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
All authors read and approve the final manuscript.

Competing interests
The authors declare that they have no competing interests.

Author details
[1]School of Management, Xi'An University of Architecture and Technology, Xi'An 710055, Shaanxi, People's Republic of China. [2]School of Information and Control Engineering, Xi'An University of Architecture and Technology, Xi'An 710055, Shaanxi, People's Republic of China. [3]State key laboratory of information security (SKLOIS), Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, People's Republic of China. [4]School of mathematics, Peking University, Beijing 100871, People's Republic of China.

Received: 1 December 2021 Accepted: 31 January 2022
Published online: 14 February 2022

References

1. Tian H, Nan F, Chang C, et al. (2019) Privacy-preserving public auditing for secure data storage in fog-to-cloud computing. J Netw Comput Appl 127:59–69
2. Zhang J, Yang Y, Chen Y, et al. (2017) A general framework to design secure cloud storage protocol using homomorphic encryption scheme. Comput Netw 129:37–50
3. Gan Q, Wang X, Fang X (2018) Efficient and secure auditing scheme for outsourced big data with dynamicity in cloud. Sci China Inf Sci 61(12):122104
4. Ji Y, Shao B, Chang J, et al. (2020) Privacy-preserving certificateless provable data possession scheme for big data storage on cloud, revisited. Appl Math Comput 386:125478
5. Zhang R, Ma H, Lu Y (2017) Provably secure cloud storage for mobile networks with less computation and smaller overhead. Sci China Inf Sci 60(12):122104
6. Shacham H, Waters B (2013) Compact proofs of retrievability. J Cryptol 26:442–483
7. Yan H, Li J, Zhang Y (2020) Remote data checking with a designated verifier in cloud storage. IEEE Syst J 14(2):1788–1797

8. Li J, Yan H, Zhang Y (2020) Identity-based privacy preserving remote data integrity checking for cloud storage. IEEE Syst J. https://doi.org/10.1109/JSYST.2020.2978146
9. Yu Y, Au MH, Ateniese G (2017) Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage. IEEE Trans Inf Forensics Secur 12(4):767–778
10. Shamir A (1985) Identity-based cryptosystems and signature schemes. In: CTYPTO'84. IACR, Santa Barbara. pp 47–53
11. Wang H, He D, Yu J, et al. (2019) Incentive and unconditionally anonymous identity-based public provable data possession. IEEE Trans Serv Comput 12(5):824–835
12. Xue J, Xu C, Zhao J, et al. (2019) Identity-based public auditing for cloud storage systems against malicious auditors via blockchain. Sci China Inf Sci 62:32104
13. Juels A, Kaliski BS (2007) Proofs of retrievability for large files. In: CCS'07. ACM, Los Angeles. pp 584–597
14. Ateniese G, Burns R, Curtmola R (2007) Provable data possession at untrusted stores. In: CCS'07. ACM, Los Angeles. pp 598–609
15. Dodis Y, Vadhan S, Wichs D (2009) Proofs of retrievability via hardness amplification. In: TCC' 2009. Springer, Berlin. pp 109–127
16. Chang J, Shao B, Ji Y, et al. (2020) Comment on a tag encoding scheme against pollution attack to linear network coding. IEEE Trans Parallel Distrib Syst 31(11):2618–2619
17. Wu Y, Chang J, Xue R, et al. (2017) Homomorphic MAC from algebraic one-way functions for network coding with small key size. Comput J 60:1785–1800
18. Chang J, Ji Y, Shao B, et al. Certificateless homomorphic signature scheme for network coding. IEEE/ACM Trans Netw. https://doi.org/10.1109/TNET.2020.3013902
19. Chen F, Xiang T, Yang Y, et al. (2016) Secure cloud storage meets with secure network coding. IEEE Trans Comput 65:1936–1948
20. Chang J, Shao B, Ji Y, et al. (2021) Secure network coding from secure PoR. Sci China Inf Sci 64(12):229301
21. Wang H, Wu Q, Qin B (2014) Identity-based remote data possession checking in public clouds. IET Inf Secur 8(2):114–121
22. Li J, Yan H, Zhang Y (2020) Efficient identity-based provable multi-copy data possession in multi-cloud storage. IEEE Trans Cloud Comput. https://doi.org/10.1109/TCC.2019.2929045
23. Chang J, Shao B, Ji Y, et al. Efficient identity-based provable multi-copy data possession in multi-cloud storage, revisited. IEEE Commun Lett. https://doi.org/10.1109/LCOMM.2020.3013280
24. Zhang X, Wang H, Xu C (2019) Identity-based key-exposure resilient cloud storage pubic auditing scheme from lattices. Inf Sci 472:223–234
25. Mary Virgil Nithya S, Rhymend Uthariaraj V (2020) Identity-based public auditing scheme for cloud storage with strong key-exposure resilience. Security and Communication Networks, article ID: 4838497, https://doi.org/10.1155/2020/4838497
26. Boneh D, Franklin M (2003) Identity-based encryption from the weil pairing. SIAM J Comput 32(3):586–615
27. Akinyele J, et al. (2013) Charm: a framework for rapidly prototyping cryptosystems. J Cryptography Eng 3(2):111–128
28. Lynn B The standard pairing based crypto library. http://crypto.standford.edu/pbc. Accessed Oct 2021
29. Boneh D, Lynn B, Shacham H (2004) Short signatures from the pairing. J Cryptol 17(4):297–319
30. Chang J, Wang H, Wang F, et al. (2020) RKA security for identity-based signature scheme. In: IEEE Access, vol. 8. IEEE, USA. pp 17833–17841
31. Ji Y, Shao B, Chang J, et al. (2021) Flexible identity-based remote data integrity checking for cloud storage with privacy preserving property. Clust Comput. https://doi.org/10.1007/s10585-021-03408-y

Publisher's Note