A Security-Enhanced Identity-Based Batch Provable Data Possession Scheme for Big Data Storage

Jining Zhao[1*], Chunxiang Xu[1] and Kefei Chen[2]
[1] School of Computer Science and Engineering, University of Electronic Science and Technology of China
Chengdu, 611731, China
[e-mail: kevinchao86@gmail.com;chxxu@uestc.edu.cn]
[2] Hangzhou Key Laboratory of Cryptography and Network Security, Hangzhou Normal University
Hangzhou, 311121, China
[e-mail: kfchen@hznu.edu.cn]
*Corresponding author: Jining Zhao

Received November 2, 2017; revised February 13, 2018; accepted April 24, 2018; published September 30, 2018

Abstract

In big data age, flexible and affordable cloud storage service greatly enhances productivity for enterprises and individuals, but spontaneously has their outsourced data susceptible to integrity breaches. Provable Data Possession (PDP) as a critical technology, could enable data owners to efficiently verify cloud data integrity, without downloading entire copy. To address challenging integrity problem on multiple clouds for multiple owners, an identity-based batch PDP scheme was presented in ProvSec 2016, which attempted to eliminate public key certificate management issue and reduce computation overheads in a secure and batch method. In this paper, we firstly demonstrate this scheme is insecure so that any clouds who have outsourced data deleted or modified, could efficiently pass integrity verification, simply by utilizing two arbitrary block-tag pairs of one data owner. Specifically, malicious clouds are able to fabricate integrity proofs by 1) universally forging valid tags and 2) recovering data owners' private keys. Secondly, to enhance the security, we propose an improved scheme to withstand these attacks, and prove its security with CDH assumption under random oracle model. Finally, based on simulations and overheads analysis, our batch scheme demonstrates better efficiency compared to an identity based multi-cloud PDP with single owner effort.

Keywords: Cloud Storage, Provable Data Possession, Identity-Based Cryptography, Provable Security

1. Introduction

A new white paper by IDC and Seagate [1], forecasts that global datasphere will swell to 163 zettabytes (ZB, i.e., a trillion gigabytes) by 2025, a tenfold rise of the 16.1 ZB of data generated in 2016 worldwide. In the big data age, by outsourcing local data to remote clouds, individuals and enterprises could make the most of cloud platforms' powerful on-demand computation and storage resources, to accelerate their business with customized and scalable applications on the "big data" and "data that is big". In 2017, 122.5 billion worldwide will be spent on public cloud services and infrastructure, 24.4 % shifting from 2016, according to another update report of IDC [2]. Although cloud service providers could deploy much protections for the owners' outsourced data, e.g. access control [3], key word search over encrypted data [4], yet still there is security concern for storage when the hardware or software failure occurs [5], like Amazon EC2 Cloud Crash accident in 2011. Any small percentage of data breach for cloud service provider, could be disaster for some companies. As one of most critical cloud security problems, outsourcing data integrity [6] raised a lot of attentions, in the sense that cloud service providers might hide data loss news to maintain good reputation.

Provable data possession (PDP), which is proposed by Ateniese et al. [7], could enable data owners to efficiently determine whether actual data possession delegated to cloud service provider remains the same as its original data copy. With homomorphic verifiable tags [8] and random sampling technologies, data owners could verify proof of storage from clouds with low overheads, instead of entire cloud data copy. Therefore, any delete or modification could be efficiently detected. Among recent years' works based on Public Key Infrastructure (PKI), Wang et al. designed privacy preserving PDP scheme for public auditing [9], and Zhu et al. proposed cooperative PDP [10] to enable data integrity checking on multiple clouds setting. Some studies continue to make critical progress to allow fully dynamic PDP [8], or support user revocation PDP [16]. However, all of them above have to confront with complicated public key certificate management problem, where additional certificates are to be verified before using public keys and forthcoming relevant distributing issues.

To further enhance the efficiency of PDP, identity-based cryptosystem recently received much attention for desirable feature of eliminating complicated public key certificate management. In 2013, Zhao et al. proposed the first identity-based public auditing [13] scheme to realize the identity-based PDP primitive, which also supported batch verification for multiple owners' data. To enable integrity verification for individual owner's data outsourced on multiple clouds, Wang et al. designed the first identity-based distributed PDP [15] system called ID-DPDP, which might be vulnerable for some security flaw and fortunately was fixed by Peng et al. in [18]. In 2015, Yu et al. considered constructing identity-based PDP by transforming certificate-based solutions [11], and proposed identity-based remote data integrity checking scheme with privacy-preserving [12] in 2016. Meanwhile, excellent architecture designs benefit us a lot from data delivery aspects [22][23].

In ProvSec 2016, Zhou et al. proposed an identity-based batch PDP scheme named ID-BPDP [17] to verify multiple data owners' data on multiple clouds. They aimed to eliminate the resource-consuming issue of certificate management, and meanwhile to reduce the computation cost greatly by batch verification for multiple owners and multiple clouds simultaneously. The authors presented a simulation of this scheme upon the computation and communication cost, and claimed that its security relies on assumption of computational hard problem.

Contributions : In this paper, for big data storage integrity, firstly we will show that ID-BPDP [17] proposed by Zhou et al. is not able to achieve claimed security in the sense that malicious clouds could first universally forge valid tag for any modified data even without the data owner's private key, and thus manage to deceive verifier and data owners when the data is deleted or modified. Besides, it is feasible for malicious clouds to recover any data owner's private key to generate tags. Secondly, to remedy the security flaws of forging of tags and recovering private key for fraudulent proofs, we propose our improved solution of original scheme, and prove its security from hardness assumption of Computational Diffie-Hellman problem under original security model. Thirdly, extrapolated to large scale of big data storage, our improved scheme could be more efficient in computation and require less cost of communication compared with existing identity based PDP scheme for single owner on multiple clouds storage, based on final complexity analysis of overheads and the trend illustrated from simulations.

Paper Organization: The rest of the paper starts with notations and reviews of definition of ID-BPDP along with its system and security model for in Section 2. After revisiting of ID-BPDP scheme in Section 3, two security flaws are demonstrated in Section 4. We present our improved scheme in Section 5, and formally prove its security in Subsection 5.1 under random oracle model. In Section 6, we compare our improved scheme with Peng et al.'s ID-DPDP and and Zhou et al.'s ID-BPDP, in the context of overheads based on complexity analysis and simulations, to study the trend of efficiency for extrapolating to big data storage. Section 7 concludes our paper with open problem.

2. Preliminary

2.1 Notations and computational assumption

- G_1 and G_2 are two cyclic groups of same large prime order q, additive and multiplicative groups respectively. e is a bilinear pairing mapping, where $e: G_1 \times G_1 \to G_2$.
- (*mpk*,*msk*) are the Private Key Generator (PKG)'s master public key and master private key, respectively. sk_i is i-th data owner's corresponding identity-based private key.
- There are n_O number of data owners, outsourcing total N number of blocks, on n_J number of clouds. M_{ijk} is the i -th data owner's k-th block outsourced on j -th cloud with s number of sectors ($s<N$), where sectors $\{F_{ijkl}\}_{l\in[1,s]}$ with common $\{v_l\}_{l\in[1,s]}$ for combination m_{ijk} . σ_{ijk} is the tag of block M_{ijk}.
- *chal* is the challenge token generated by verifier, and $chal_j$ is the specific challenge token for the j-th cloud. c is number of challenged blocks, where $c<N$.
- $n \in [1,c]$ indicates the n -th selected block of total c challenged blocks, which should be further specified as i_n-th data owner's k_n-th block outsourced on j_n-th cloud..
- f is a pseudo random function (PRF) $f: Z_q \times \{1,\cdots,N\} \to Z_q$ for generating challenging co-efficient to combine challenged blocks.
- C is the index set of challenged clouds picked by verifier. O is the index set of data owner's identities upon challenged blocks, and J is the index set of challenged clouds, where $|O|=n_1$, $|J|=n_2$. P_j is the proof of storage generated by j-th challenged cloud.

CDH problem on G_1: Given g, g^a ,g^b $\in G_1$, to compute g^{ab} with a probabilistic polynomial time (PPT) algorithm, without knowing random $a,b \in Z_q$.

DL problem on G_1: For $g \in G_1$, $a \in Z_q$, given g^a, to compute a with a PPT algorithm.

2.2 Definition of ID-BPDP

In this section, we will present the definition of Identity-Based Batch Provable Data Possession (ID-BPDP) from the original paper [17], in the six algorithms below.

- Setup (1^k)→ (*params*, *mpk*, *msk*) is initialized by PKG with security parameter *k* as input. It outputs public parameters *params* , master public/ private key pairs (*params*,*mpk*,*msk*).
- Extract (*params*, *msk*, ID_i) →sk_i is executed by PKG with as input parameters *params* and *i*-th data owner DO_i's identity ID_i, it outputs the private key sk_i for the owner.
- TagGen (*params*, ID_i, sk_i, *mpk*, $\{M_{ijk}\}$)→$\{\sigma_{ijk}\}$ is run by every data owner. It takes as input public parameters *params*, owner DO_i's identity ID_i, the personal private key sk_i, master public key *mpk* and data blocks $\{M_{ijk}\}$ to be outsourced on the corresponding clouds. Then the tags $\{\sigma_{ijk}\}$ of above blocks could be generated.
- Challenge ($\{(i,j,k)\}$) → (*chal*, $\{chal_j\}$) is executed by verifier. It takes as input data index set $\{(i,j,k)\}$ and randomly selects some indexes as the challenge token *chal* for one instance. According to the specified indexes $\{j\}$, challenge token *chal* is further divided into a set of tokens $\{chal_j\}$ and only forward $chal_j$ to the corresponding *j*-th cloud.
- ProofGen (*params*, $chal_j$, $\{ID_i\}$, $\{\sigma_{ijk}\}$, $\{M_{ijk}\}$) →P_j is run by each challenged cloud. It takes as input parameters *params*, challenge token received $chal_j$, the specified set of data owners' identities $\{ID_i\}$, the set of tags $\{\sigma_{ijk}\}$, and the blocks $\{M_{ijk}\}$, and the proof P_j is generated for challenge token $chal_j$. Then P_j is sent back to verifier by cloud CS_j .
- Verify (*params*, *chal*, $\{ID_i\}$, { P_j }, *mpk*) →{0,1} is executed by verifier. It takes as input public parameters *params*, challenge token *chal*, specified set of data owners' identities $\{ID_i\}$, set of proofs { P_j } from all challenged clouds, and the master public key *mpk*. 1 will be output if the proofs are valid, otherwise 0 is output.

2.3 System Model

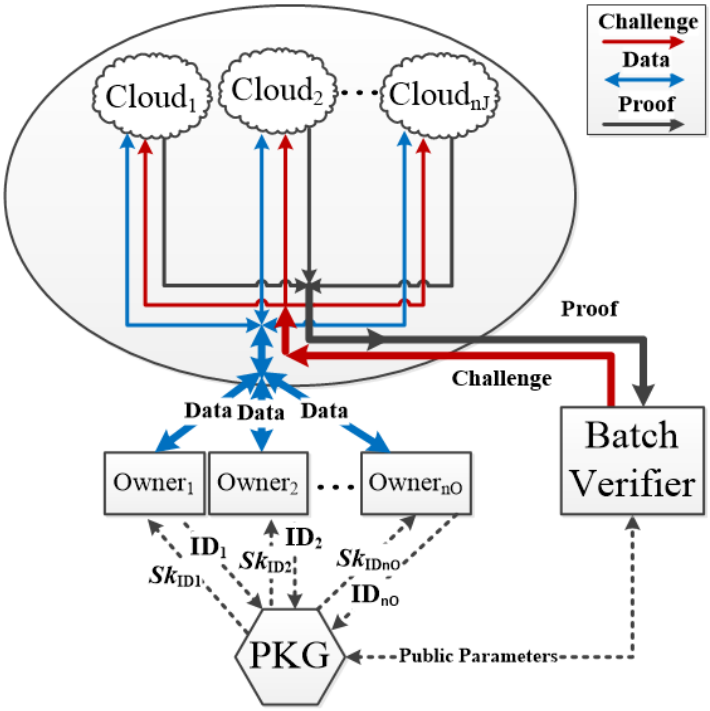


Fig. 1. Architecture of ID-BPDP

As it depicts in **Fig. 1**, there are four kinds of entities in an ID-BPDP scheme, i.e., the *PKG*, data *Owners*, multiple *Clouds*, and a batch *Verifier*. *PKG* initializes the system parameters and extracts private keys for data owners of their own identities. Data *Owners* create their own data and tags, and store them in multiple clouds. Multiple *Clouds* maintain data owners' data in the form of sectors, and provide data access to the data owners. The batch *Verifier* is a trusted third party to offer the batch data integrity verification on multiple clouds for the data owners.

2.4 Security Model

In ID-BPDP [17], both the data owners and verifier are assumed to be trusted to perform the scheme, while the clouds might be malicious to forge tags and proofs to deceive them if data loss incidents occur in terms of data delete and modification, such that clouds' reputations and economic interest will not be affected according to the regulation. We review the two formal security definitions as follows:

- **Unforgeability of Tags**: A tag is unforgeable if any malicious clouds win the *Tag-Forge* game below in probabilistic polynomial time, with negligible probability.
  - Setup: Challenger ***B*** (in the role of PKG and verifier) generates system parameter for PKG, and sends PKG's public key to adversary ***A*** (malicious clouds).
  - Queries: ***A*** could adaptively query private keys $\{sk_i\}$ for set of identities $\{ID_i\}(i \in S_1)$. It could also request tags $\{\sigma_{ijk}\}$ for set of blocks $\{M_{ijk}\}$ $((i, j, k, M_{ijk}) \in I_1)$.
  - Forge: ***A*** wins the game if it forges a valid tag $\sigma_{i^*j^*k^*}$ for data block $M_{i^*j^*k^*}$, for which it has neither extracted private key for the identity nor queried the tag, i.e., where $i^* \notin S_1$ and $(i^*, j^*, k^*, M_{i^*j^*k^*}) \notin I_1$.
- **Unforgeability of Proofs**: A proof is unforgeable if any malicious clouds win the *Proof-Forge* game below in probabilistic polynomial time, with negligible probability.
  - Setup, and Firstphase queries are same as the *Tag-Forge* game's corresponding parts. The set of identities which have extracted private keys are denoted as S_1, and the set of index and blocks which have queried tags are denoted as I_1.
  - Challenge: Challenger ***B*** generates challenge token $chal^*$ for a set of index and blocks $\{(i_n^*, j_n^*, k_n^*, M_{i_n^*j_n^*k_n^*}) \mid n = [1, \cdots, c]\}$, where at least one $ID_{i_n^*}$ $(i_n^* \notin S_1)$ and for the same i_n^*, $(i_n^*, j_n^*, k_n^*, M_{i_n^*j_n^*k_n^*}) \notin I_1$, and sends $chal^*$ to ***A***.
  - Secondphase queries: Similar to Firstphase queries, the set of identities which have extracted private keys are denoted as S_2, and the set of index and blocks which have queried tags are denoted as I_2, where at least one $ID_{i_n^*} \notin (S_1 \cup S_2)$ and for the same i_n^*, $(i_n^*, j_n^*, k_n^*, M_{i_n^*j_n^*k_n^*}) \notin (I_1 \cup I_2)$.
  - Forge: ***A*** wins the game if it forges valid proofs $\{P_{j^*}\}$ for challenge token $chal^*$, where for the above two phases of queries, at least one identity has neither extracted private key and nor queried tag.

3. Revisiting of ID-BPDP

In this section,we will revisit the ID-BPDP scheme of six algorithms in [17].

- Setup: PKG uses this algorithm to generate a bilinear map $e: G_1 \times G_1 \to G_2$ with two groups G_1 and G_2 of the same order $q > 2^k$, where g is the generator of G_1 and k is security parameter. It also selects three cryptographic hash functions $H_1: \{0,1\}^* \to G_1$, $H_2: \{0,1\}^* \to Z_q$, $H_3: G_1 \to G_1$ and a pseudo random function $f: Z_q \times \{1,\cdots,N\} \to Z_q$. It picks random x and $\{v_l\}_{l\in[1,s]}$ from Z_q, and computes g^x, where $s < N$ is the number of sectors for each file block. The PKG's master private key is $msk = x$ and master public key is $mpk = g^x$. The global parameters are $(e, G_1, G_2, g, mpk, H_1, H_2, H_3, f, \{v_l\}_{l\in[1,s]})$
- Extract: Given identity ID_i, PKG extracts the identity-based private key as $sk_i = H_1(ID_i)^x$ and returns to the data owner.
- TagGen: Data owner of ID_i uses this algorithm to generate tag for block M_{ijk} in the form of s sectors $\{F_{ijkl}\}$. It selects a random $u_i \in Z_q$ and computes

$$S_{ijk} = g^{u_i}, T_{ijk} = sk_i^{m_{ijk}+H_2(ID_i)} \cdot H_3(mpk)^{u_i}$$

as tag $\sigma_{ijk} = (S_{ijk}, T_{ijk})$, where $m_{ijk} = \sum_{l=1}^{s} v_l \cdot F_{ijkl}$. Upon receiving block M_{ijk}, the j-th cloud could verify the tag σ_{ijk} as

$$e(T_{ijk}, g) = e(H_1(ID_i)^{\sum_{l=1}^{s} v_l F_{ijkl} + H_2(ID_i)}, mpk) \cdot e(H_3(mpk), S_{ijk}) \quad (1)$$

- Challenge: For verifying c blocks of total N blocks on the multiple clouds storage system, the verifier uses this algorithm to generate challenge token $chal = (I, K)$, in order to specify each challenged block $M_{i_n j_n k_n}$ belonging the i_n-th data owner of ID_{i_n} and as the k_n-th block outsourced on j_n-th cloud, where the index $(i_n, j_n, k_n) \in I$ $(1 \le n \le c \le N)$, and the random temporary key $\kappa_{i_n} \in K$ $(\kappa_{i_n} \in Z_q)$. The challenge token $chal$ is further distributed to each challenged cloud as disjoint challenge tokens $\{chal_j\}$, and thus $chal_j = (I_j, K_j)$ will be sent to the j-th cloud where $I_j = \{(i_n, j, k_n) \mid \forall n_1 \ne n_2, i_{n_1} \ne i_{n_2}\} \subseteq I$ and $K_j = \{\kappa_{i_n} \mid (i_n, j, k_n) \in I_j\} \subseteq K$. All the indexes of challenged clouds $\{j\}$ form a set C, and $\forall j_1, j_2 \in C, j_1 \ne j_2$, $I_{j_1} \cap I_{j_2} = \varnothing$, $\cup_{j\in C} I_j = I$.
- ProofGen: The j-th challenged cloud picks up challenged blocks $\{M_{i_n j k_n}\}$ and computes co-efficient values $\{r_n\} = \{f_{\kappa_{i_n}}(i_n, j, k_n)\}$ with received challenged token $chal_j = \{I_j, K_j\}$ $(I_j = \{(i_n, j, k_n) \mid \forall n_1 \ne n_2, i_{n_1} \ne i_{n_2}\} \subseteq I$ and $K_j = \{\kappa_{i_n} \mid (i_n, j, k_n) \in I_j\} \subseteq K)$. The proof of storage $P_j = (S'_j, T'_j, \{F'_{ijl}\}_{(i,j,k)\in I_j})$ is generated with blocks $\{M_{i_n j k_n}\}$ and tags $\{\sigma_{i_n j k_n} = (S_{i_n j k_n}, T_{i_n j k_n})\}$ as:

$$S'_j=\prod_{chal_j} S^{r_n}_{i_n jk_n}, T'_j=\prod_{chal_j} T^{r_n}_{i_n jk_n}, F'_{ijl}=\sum_{chal_j, i_n=i} F_{i_n jk_n l}\cdot r_n.$$

$P_j=(S'_j, T'_j, \{F'_{ijl}\}_{(i,j,k)\in I_j})$ will be sent to the verifier.

Note: As feasible computation optimization claimed in [17], in ProofGen, $\prod_{chal_j} S^{r_n}_{i_n jk_n}$ could be computed by cloud CS_j as $\prod_{i\in O}(g^{u_i})^{\sum_{i_n=i} r_n}$ to avoid some exponentiation. Simply because when data owner DO_i runs TagGen, it first chooses a random u_i to generate tag for one block, and reuses the same u_i for the rest blocks' tags, i.e., $S_{ijk}=g^{u_i}$. Therefore, the distinct exponent $\{r_n\}=\{f_{\kappa_{i_n}}(i_n, j_n, k_n)\}$, could be summed up for the same base $g^{u_i}=S_{i_n jk_n}(i_n=i)$ specified by same u_i, even if they vary with indexes $\{(i_n, j_n, k_n)\}$. (It is specified in the 4th notes of computation analysis for ProofGen in [17]'s page 120)

· Verify: After receiving all the proofs $\{P_j\}$ from all challenged clouds, the verifier views index set $\{i\}$ as O for all the challenged data owners, and computes $\{r_n\}=\{f_{\kappa_{i_n}}(i_n, j_n, k_n)\,|\,n=1,\cdots,c\}$, with the challenge token $chal=(I,K)$ for $I=\{(i_n, j_n, k_n)\,|\,n=1,\cdots,c\}$ and $K=\{\kappa_{i_n}\,|\,n=1,\cdots,c\}$ (The index and coefficient generation are similar to Section 5). It will outputs 1 (valid) if the following equation holds:

$$e(\prod_{j\in C} T'_j, g)=e(\prod_{i\in O} H_1(ID_i)^{\sum_{j\in C}\sum_{l=1}^{s} v_l F'_{ijl}+H_2(ID_i)\sum_{i_n=i} r_n}, mpk)\cdot e(H_3(mpk), \prod_{j\in C} S'_j) \quad (2)$$

and 0 (valid) otherwise.

4. On the security of ID-BPDP

In [17], by reusing u_i generate tags for all blocks $\{M_{ijk}\}$, data owner DO_i in TagGen, ID-BPDP could elegantly allow j-th cloud to avoid some exponentiation in computing S_j' of the proof P_j in ProofGen, where distinct exponents $\{r_n\}$ could be summed up for the same base g^{u_i}. There is comprehensive analysis for its security in paper, however, ID-BPDP scheme may suffer from this elegance of scheme designing. In the following, we will investigate that ID-BPDP is vulnerable to two security attacks: universal forgery of tags and recovery of data owner's private key. Especially, even if improving tags generation with different u_i as u_{ijk}, universal forgery of tags remains valid. Therefore, the security properties of unforgeability of tags and unforgeability of proofs may not hold.

4.1 First attack: universally forging tag without private key

We assume there is a malicious cloud denoted as j_m-th cloud. For modified block $M^*_{ij_mk}$, the malicious j_m -th cloud chooses another two arbitrary pairs of data tag and block outsourced from the same owner, e.g., $(S_{ij_mk_1}, T_{ij_mk_1}, M_{ij_mk_1})$, $(S_{ij_mk_2}, T_{ij_mk_2}, M_{ij_mk_2})$, and computes $d=\left(m^*_{ij_mk}-m_{ij_mk_2}\right)\Big/\left(m_{ij_mk_1}-m_{ij_mk_2}\right)\in Z_q$ with blocks' sectors. The tag of modified data

$M^*_{ij_mk}$ could be generated by cloud as:

$$T^*_{ij_mk} = T^d_{ij_mk_1} \cdot T^{1-d}_{ij_mk_2} = sk_i^{[(m_{ij_mk_1}+H_2(ID_i))d+(m_{ij_mk_2}+H_2(ID_i))(1-d)]} \cdot H_3(mpk)^{u_i d+u_i(1-d)}$$
$$= sk_i^{d(m_{ij_mk_1}-m_{ij_mk_2})+m_{ij_mk_2}+H_2(ID_i)} \cdot H_3(mpk)^{u_i} = sk_i^{m^*_{ij_mk}+H_2(ID_i)} \cdot H_3(mpk)^{u_i}$$
$$S^*_{ij_mk} = S^d_{ij_mk_1} \cdot S^{1-d}_{ij_mk_2} = (g^{u_i})^d \cdot (g^{u_i})^{(1-d)} = g^{u_i}$$

where $S_{ij_mk_1} = S_{ij_mk_2} = g^{u_i}$ according to the tag generation in TagGen.

Since random number set $\{v_l\}$ are accessible for all clouds to verify the validity of tags and blocks by equation (1), $m_{ij_mk_1} = \sum_{l=1}^{s} v_l F_{ij_mk_1l}$, $m_{ij_mk_2} = \sum_{l=1}^{s} v_l F_{ij_mk_2l}$ and $m^*_{ij_mk} = \sum_{l=1}^{s} v_l F^*_{ij_mkl}$ could be generated with sectors $\{F_{ij_mk_1l}\}, \{F_{ij_mk_2l}\}$ and $\{F^*_{ij_mkl}\}$ of above blocks. It is easy to verify $(T^*_{ij_mk}, S^*_{ij_mk}) = (T^d_{ij_mk_1} \cdot T^{1-d}_{ij_mk_2}, S^d_{ij_mk_1} \cdot S^{1-d}_{ij_mk_2})$ as a valid tag for block $M^*_{ij_mk}$ by equation (1).

For example, for modified block $M^*_{1j1} = \{F^*_{1j1l}\}_{l\in[1,s]}$, where $m^*_{1j1} = \sum_{l=1}^{s} v_l F^*_{1j1l}$,

$$(T^d_{1j2} \cdot T^{1-d}_{1j3}, S^d_{1j2} \cdot S^{1-d}_{1j3}) = (sk_1^{m^*_{1j1}+H_2(ID_1)} \cdot H_3(mpk)^{u_1}, g^{u_1})$$

could be fabricated as valid tag of block M^*_{1j1} with tags (T_{1j2}, S_{1j2}) and (T_{1j3}, S_{1j3}).

Surprisingly, if simply improving the ID-BPDP scheme by modifying $S_{ijk} = g^{u_{ijk}}$ with different randomness u_{ijk},

$$T^*_{ij_mk} = T^d_{ij_mk_1} \cdot T^{1-d}_{ij_mk_2} = sk_i^{[(m_{ij_mk_1}+H_2(ID_i))d+(m_{ij_mk_2}+H_2(ID_i))(1-d)]} \cdot H_3(mpk)^{u_{ij_mk_1}d+u_{ij_mk_2}(1-d)}$$
$$= sk_i^{m^*_{ij_mk}+H_2(ID_i)} \cdot H_3(mpk)^{u_{ij_mk_1}d+u_{ij_mk_2}(1-d)}$$
$$S^*_{ij_mk} = S^d_{ij_mk_1} \cdot S^{1-d}_{ij_mk_2} = (g^{u_{ij_mk_1}})^d \cdot (g^{u_{ij_mk_2}})^{(1-d)} = g^{u_{ij_mk_1}d+u_{ij_mk_2}(1-d)}$$

The forged tag $(T^*_{ij_mk}, S^*_{ij_mk}) = (T^d_{ij_mk_1} \cdot T^{1-d}_{ij_mk_2}, S^d_{ij_mk_1} \cdot S^{1-d}_{ij_mk_2})$ is still able to pass equation (1) as a valid tag for block $M^*_{ij_mk}$.

For example, for modified block $M^*_{1j1} = \{F^*_{1j1l}\}_{l\in[1,s]}$, where $m^*_{1j1} = \sum_{l=1}^{s} v_l F^*_{1j1l}$,

$$(T^d_{1j2} \cdot T^{1-d}_{1j3}, S^d_{1j2} \cdot S^{1-d}_{1j3}) = (sk_1^{m^*_{1j1}+H_2(ID_1)} \cdot H_3(mpk)^{u_{1j2}d+u_{1j3}(1-d)}, g^{u_{1j2}d+u_{1j3}(1-d)})$$

could be fabricated as valid tag of block M^*_{1j1} with tags (T_{1j2}, S_{1j2}) and (T_{1j3}, S_{1j3}).

Therefore, the security flaw of universal forgery does not lie in simply re-use of randomness but incapability of detecting fabrication of original randomness. These flaws make improving security of existing ID-BPDP scheme more challenging.

4.2 Second attack: recovering private key and forging

In the original ID-BPDP scheme, parts of the i-th data owner's tags $\{\sigma_{ijk} = (S_{ijk}, T_{ijk})\}$ are identical from Taggen, i.e., $\exists i \in O$, $\forall S_{ijk} = g^{u_i}$, T_{ijk} shares $H_3(mpk)^{u_i}$. This reusing of

random value u_i for same data owner DO_i's tags, further contributes to clouds sides' computation optimization of ProofGen. Unexpectedly, the elegance of designing may introduce potential vulnerability of exposing remaining private key part by efficient exponentiation operations on any two different tags.

Recovering individual private key: The malicious j_m-th cloud selects two arbitrary pairs of data tag and block outsourced from the same owner, e.g., $(S_{ij_mk_1}, T_{ij_mk_1}, M_{ij_mk_1})$, $(S_{ij_mk_2}, T_{ij_mk_2}, M_{ij_mk_2})$, and computes:

$$\left(\frac{T_{ij_mk_1}}{T_{ij_mk_2}}\right)^{\frac{1}{m_{ij_mk_1}-m_{ij_mk_2}}} = \left(sk_i^{[(m_{ij_mk_1}+H_2(ID_i))-(m_{ij_mk_2}+H_2(ID_i))]} \cdot H_3(mpk)^{u_i-u_i}\right)^{\frac{1}{m_{ij_mk_1}-m_{ij_mk_2}}}$$

$$= sk_i^{\frac{m_{ij_mk_1}-m_{ij_mk_2}}{m_{ij_mk_1}-m_{ij_mk_2}}} = sk_i .$$

where $m_{ij_mk_1}$ and $m_{ij_mk_2}$ could be generated by their blocks' sectors with $\{v_l\}$.

Obviously, the data owner of identity ID_i's individual private key could be revealed as

$$sk_i = \left(\frac{T_{ij_mk_1}}{T_{ij_mk_2}}\right)^{\frac{1}{\sum_{l=1}^{s} v_l (F_{ij_mk_1l}-F_{ij_mk_2l})}}, \text{ e.g., } sk_1 = \left(\frac{T_{1j2}}{T_{1j3}}\right)^{\frac{1}{\sum_{l=1}^{s} v_l (F_{1j2l}-F_{1j3l})}}.$$

Forging tag: With private key sk_i recovered from above, malicious j_m-th cloud could generate a forged tag $\sigma^*_{ij_mk}$ for a modified block $M^*_{ij_mk}$, with regarding to the original M_{ij_mk}.

$$T^*_{ij_mk} = T_{ij_mk} \cdot sk_i^{m^*_{ij_mk}-m_{ij_mk}} = sk_i^{m^*_{ij_mk}+H_2(ID_i)} \cdot H_3(mpk)^{u_i}, S^*_{ij_mk} = S_{ij_mk} = g^{u_i}$$

where $m^*_{ij_mk}$, m_{ij_mk} could be generated with block sectors $\{F^*_{ij_mkl}\}, \{F_{ij_mkl}\}$ as above.

Because $(T^*_{ij_mk}, S^*_{ij_mk}) = (T_{ij_mk} \cdot sk_i^{m^*_{ij_mk}-m_{ij_mk}}, S_{ij_mk})$ is exactly equal to genuine tag for $M^*_{ij_mk}$ under the same original random value u_i coherent with other authentic block's tag. The forged tag could be easily verified with equation (1), e.g., $(T^*_{1j1}, S^*_{1j1}) = (T_{1j1} \cdot sk_1^{m^*_{1j1}-m_{1j1}}, S_{1j1})$ is exactly equal to genuine tag for M^*_{1j1} under the same original random value u_1, with private key recovered as $sk_1 = \left(\frac{T_{1j2}}{T_{1j3}}\right)^{\frac{1}{\sum_{l=1}^{s} v_l (F_{1j2l}-F_{1j3l})}}$.

Note: With the first attack approach, the adversary is able to generate valid tag of modified data with another two correct tags, by utilizing their relationship that could be retrieved. For i th owner, assume that its original data M_{ij_mk} is modified to an arbitrary data $M^*_{ij_mk}$. Adversary could easily calculate relationship $d = \left(m^*_{ij_mk} - m_{ij_mk_2}\right) / \left(m_{ij_mk_1} - m_{ij_mk_2}\right)$, with target modified block $M^*_{ij_mk}$ and two other correct blocks $M_{ij_mk_1}$ and $M_{ij_mk_2}$. Then, the target tag $\sigma^*_{ij_mk}$ = $(T^*_{ij_mk}, S^*_{ij_mk})$ for modified $M^*_{ij_mk}$, could be generated as

$(T_{ij_mk_1}^{d} \cdot T_{ij_mk_2}^{1-d}, S_{ij_mk_1}^{d} \cdot S_{ij_mk_2}^{1-d})$, with d and tags of the above two blocks. The adversary may also launch the second attack by recovering owner's secret key. With the two pairs of correct tags and blocks, it could surprisingly recover the owner's private key as $sk_i = \left(\frac{T_{ij_mk_1}}{T_{ij_mk_2}}\right)^{\frac{1}{m_{ij_mk_1} - m_{ij_mk_2}}}$. Then, according to original tag and data pair $(\sigma_{ij_mk}, M_{ij_mk})$, adversary could compute target tag $\sigma_{ij_mk}^{*} = (T_{ij_mk}^{*}, S_{ij_mk}^{*})$ as $(T_{ij_mk} \cdot sk_i^{m_{ij_mk}^{*} - m_{ij_mk}}, S_{ij_mk})$, with private key sk_i and the modified data $M_{ij_mk}^{*}$. As above two attack approaches, malicious clouds could easily fabricate valid tag for any modified data, which will pass the verification equations (1) and (2).

On the other hand, for the sake of saving storage cost, the clouds might delete some data $\{M_{ij_mk}\}$ copies, only having to keep one piece of data copy $M_{i_1j_mk_1}^{*}$ and replace their tags with corresponding new forged tags $\{\sigma_{ij_mk}^{*}\}$. Since these forged tags are valid to pass equation (1), with the correctness of IB-BPDP, it is feasible to pass the verification equation (2), by constructing proof with these foraged tags and data. Therefore, malicious clouds are able to forge proof of integrity to cheat both verifier and data owners, when the outsourced data are modified or deleted.

5. Improved scheme

As above analysis, both recovering private key and universal forgery of tags should be fixed. If we simply modify tags generation with different u_i as u_{ijk}, the scheme will still suffer from universal forgery of tags. Meanwhile, without unique index and file name, different blocks of different files share the same tag, which might result in vulnerability in substituting valid blocks for passing verification. Therefore, more efforts need to be made to contribute to efficient and rigorous improvement for the deep security flaws. According to the definition of ID-BPDP in subsection 2.2, our improved scheme with following six algorithms, is able to allow verifier to verify proofs from challenged clouds for the challenged data owners and their corresponding data blocks and tags, at one time in a batch way.

- Setup(1^k) $\rightarrow$ ($params, mpk, msk$) PKG uses this algorithm to generate a bilinear map $e: G_1 \times G_1 \rightarrow G_2$ with two groups G_1 and G_2 of the same order $q > 2^k$, where g is the generator of G_1 and k is security parameter. It also selects three cryptographic hash functions $H_1: \{0,1\}^{*} \rightarrow G_1$, $H_2: \{0,1\}^{*} \rightarrow Z_q$, $H_3: G_1 \rightarrow G_1$, and a pseudo random function $f: Z_q \times \{1, \cdots, N\} \rightarrow Z_q$. It picks random $x \in Z_q$ and $\{v_l\}_{l \in [1,s]}$, and computes g^x, where $s < N$ is the number of sectors for each file block. The PKG's master private key is $msk = x$ and master public key is $mpk = g^x$. The global parameters are $(e, G_1, G_2, g, mpk, H_1, H_2, H_3, f, \{v_l\}_{l \in [1,s]})$.
- Extract ($params, msk, ID_i$) $\rightarrow sk_i$: Given identity ID_i, PKG extracts the identity-based private key as $sk_i = H_1(ID_i)^x$ and returns to the data owner.

· TagGen ($params, ID_i, sk_i, M_{ijk}) \to \sigma_{ijk}$: For one file named $name_i$, data owner of identity ID_i runs this algorithm to generate tag for every block M_{ijk} , which are further split into s sectors $\{F_{ijkl}\}$. The data owner first generates random $u_i \in Z_q$, and computes

$$S_{ijk} = g^{u_i}, T_{ijk} = sk_i^{m_{ijk}+H_2(name_i||i||j||k)} \cdot H_3(S_{ijk} \,||\, name_i \,||\, i \,||\, j \,||\, k)^{u_i}$$

as tag $\sigma_{ijk} = (S_{ijk}, T_{ijk})$, where $m_{ijk} = \sum_{l=1}^{s} v_l \cdot F_{ijkl}$. Meanwhile upon receiving block M_{ijk} , the j -th cloud could verify the tag σ_{ijk} as

$$e(T_{ijk}, g) = e(H_1(ID_i)^{\sum_{l=1}^{s} v_l F_{ijkl} + H_2(name_i||i||j||k)}, mpk) \cdot e(H_3(S_{ijk} \,||\, name_i \,||\, i \,||\, j \,||\, k), S_{ijk}) \quad (3)$$

· Challenge $(\{(i, j, k)\}) \to (chal, \{chal_j\})$: For verifying c blocks of total N blocks on the multiple clouds storage system, the verifier uses this algorithm to generate challenge token $chal = (I, K)$, in order to specify each challenged block $M_{i_n j_n k_n}$ belonging the i_n -th data owner of ID_{i_n} 's file named $name_{i_n}$ and as the k_n -th block outsourced on j_n -th cloud, where $(name_{i_n}, i_n, j_n, k_n) \in I$ $(1 \le n \le c \le N)$, and the random temporary key $\kappa_{i_n} \in K$ ($\kappa_{i_n} \in Z_q$). The challenge token $chal$ is further distributed to each challenged cloud as disjoint challenge tokens $\{chal_j\}$, and $chal_j = (I_j, K_j)$ will be sent to the j -th cloud where $I_j = \{(name_{i_n}, i_n, j, k_n) \,|\, \forall n_1 \ne n_2, i_{n_1} \ne i_{n_2}\} \subseteq I$ and $K_j = \{\kappa_{i_n} \,|\, (i_n, j, k_n) \in I_j\} \subseteq K$. All the indexes of challenged clouds $\{j\}$ form a set C , and $\forall j_1, j_2 \in C, j_1 \ne j_2$, $I_{j_1} \cap I_{j_2} = \varnothing$, $\cup_{j \in C} I_j = I$.

· ProofGen ($params, chal_j, \{ID_i\}, \{\sigma_{ijk}\}, \{M_{ijk}\}) \to P_j$: The j -th challenged cloud selects challenged blocks $\{M_{i_n j k_n}\}$ for the owners' files named $\{name_{i_n}\}$ and computes co-efficient values $\{r_n\} = \{f_{\kappa_{i_n}}(i_n, j, k_n)\}$ with received challenged token $chal_j = \{I_j, K_j\}$ ($I_j = \{(name_{i_n}, i_n, j, k_n) \,|\, \forall n_1 \ne n_2, i_{n_1} \ne i_{n_2}\} \subseteq I$ and $K_j = \{\kappa_{i_n} \,|\, (i_n, j, k_n) \in I_j\} \subseteq K$). Since for the same owner of ID_i $\forall i_n = i, S_{i_n j_n k_n} = g^{u_{i_n}} = g^{u_i} = S_{ijk}$, the proof of storage $P_j = (\{S'_{i_n j}\}_{i_n \in chal_j}, T'_j, \{F'_{ijl}\}_{(i,j,k) \in I_j})$ is generated with blocks $\{M_{i_n j k_n}\}$ and tags $\{\sigma_{i_n j k_n} = (S_{i_n j k_n}, T_{i_n j k_n})\}$ as:

$$S'_{i_n j} = S_{i_n j k_n}, T'_j = \prod_{chal_j} T_{i_n j k_n}^{r_n}, F'_{ijl} = \sum_{chal_j, i_n = i} F_{i_n j k_n l} \cdot r_n .$$

and will be sent to the verifier.

· Verify ($params, chal, \{ID_i\}, \{P_j\}, mpk) \to \{0,1\}$: After receiving all the proofs $\{P_j\}$ from all challenged clouds, the verifier first looks up the challenge token $chal = (I, K)$ ($I = \{(name, i_n, j_n, k_n) \,|\, n = 1, \cdots, c\}$ and $K = \{\kappa_{i_n} \,|\, n = 1, \cdots, c\}$) , for the specific index (i_n, j_n, k_n) and corresponding random key κ_{i_n} . With the index and random key of each

block, the co-efficient r_n could be generated by the pseudo random function as $r_n = f_{\kappa_{i_n}}(i_n, j_n, k_n)$. The index set of all challenged owners' identities, could be viewed as $O = \{i \mid i = i_n, i_n \in I\}$ from their challenged blocks' indexes. Meanwhile, as valid tag for data owner of ID_i, $\forall i_n = i$, $S'_{i_n j} = S_{i_n jk_n} = S_{ijk} = g^{u_i}$. Therefore, verifer simply sets S'_i by any one $S'_{i_n j}(i_n = i)$ and views $S'_{i_n jk_n} = S'_{i_n j}$ for $H_3(\cdot)$'s input for every (i_n, j, k_n). It will outputs 1 (valid) if the following equation holds:

$$e(\prod_{j\in C} T'_j, g) = e(\prod_{i\in O} H_1(ID_i)^{\sum_{j\in C}\sum_{l=1}^{s} v_l F'_{ijl} + \sum_{i_n=i} H_2(name_{i_n}\|i_n\|j_n\|k_n) r_n}, mpk) \cdot \prod_{i\in O} e(\prod_{j\in C}\prod_{i_n=i, chal_j} H_3(S'_{i_n jk_n} \| name_{i_n} \| i_n \| j \| k_n)^{r_n}, S'_i) \quad (4)$$

and 0 (valid) otherwise, where $S'_i = S'_{i_n j}(\forall i_n = i), S'_{i_n jk_n} = S'_{i_n j}$.

5.1 Security analysis of improved scheme

Based on the formal definition of ID-BPDP scheme (Subsection 2.2) and corresponding system model (Subsection 2.3) and security model (Subsection 2.4), in this section, we prove the unforgeability of tags and proofs for our improved scheme. Compared with [17]'s security analysis, we also utilize Coron [14]'s random oracle model to define the interactions between adversary ***A*** of our scheme and simulator ***B***, but with refined oracles for hash and tag queries. To prevent their security flaw, corresponding security reduction methods are also re-designed. Our security analysis below shows that CDH problem will be solved with non-negligible probability, if breaking our scheme through forging valid tag for data block and fabricating storage proof without rejection under polynomial time.

A probabilistic polynomial time (PPT) algorithm ***B*** could be constructed to solve CDH problem (given g^a, g^b, computing g^{ab}):

- with non-negligible probability for at least $\varepsilon_1/(\hat{e}(q_E+q_T+1))$ in at most $t_1 + t_{G_1} \cdot (q_H + q_E + 3q_T + 3)$ time, if adversary ***A*** of our improved scheme in Section 5 could win *Tag-Forge* games defined in Subsection 2.4 with probability ε_1 within t_1 time;
- with non-negligible probability for at least $\varepsilon_2/(\hat{e}(q_E+q_T+1))$ in at most $t_2 + t_{G_1} \cdot (q_H + q_E + 3q_T + c^* + 3)$ time, if adversary ***A*** of our improved scheme in Section 5 could win *Proof-Forge* games defined in Subsection 2.4 with probability ε_2 within t_2 time.

Assume polynomial number q_H queries are made for hash function, q_E queries for private key extraction, q_T tag generation. t_{G_1} is the time of performing once exponentiation or inversion computation on group G_1, c^* is the number of challenged blocks and $\hat{e}$ is the natural logarithm.

Proof: We will first discuss the *Forgery of Tag* and then *Forgery of Proof*.
Firstly, **Forgery of Tag** for *Tag-Forge* game:

- Setup: Simulator ***B*** plays in the role of PKG to choose random $a \in Z_q$ as master private key

msk and upon generator $g \in G_1$ master public key $mpk = g^a$. It also picks random $b \in Z_q$, and thus the instance of CDH problem as given g^a, g^b, computing g^{ab}.

· Hash function Oracle:

H_1-oracle. For $\boldsymbol{A}$ ' s H_1 query on ID_i,

- If ID_i is already in the H_1 -list, $\boldsymbol{B}$ retrieves $(ID_i, d_i, y_i, h_{1,i})$ from the list $\{(ID_i, d_i, y_i, h_{1,i})\}$ and responds as $H_1(ID_i) = h_{1,i}$.
- Else $\boldsymbol{B}$ tosses a random coin $d_i \in \{0,1\}$ so that probability $Pr[d_i = 0] = \delta$. If $d_i = 0$, computes $h_{1,i} = g^{y_i}$ by selecting random $y_i \in Z_q$, else $d_i = 1$, computes $h_{1,i} = g^{by_i}$ by selecting random $y_i \in Z_q$. $\boldsymbol{B}$ adds $(ID_i, d_i, y_i, h_{1,i})$ in the list and responds as $H_1(ID_i) = h_{1,i}$.

H_2-oracle. For $\boldsymbol{A}$'s H_2 query on $name_i \| i \| j \| k$,

- If $(name_i, i, j, k)$ is already in the H_2-list, $\boldsymbol{B}$ retrieves $(name_i, i, j, k, h_{2,name_i ijk})$ from the list $\{(name_i, i, j, k, h_{2,name_i ijk})\}$ and responds as $h_{2,name_i ijk}$.
- Else $\boldsymbol{B}$ picks $h_{2,name_i ijk} \in Z_q$ to add $(name_i, i, j, k, h_{2,name_i ijk})$ in the list and responds as $H_2(name_i \| i \| j \| k) = h_{2,name_i ijk}$.

H_3-oracle. For $\boldsymbol{A}$'s H_3 query on $S_{ijk} \| name_i \| i \| j \| k$,

- If $(S_{ijk}, name_i, i, j, k)$ is already in the H_3 -list, $\boldsymbol{B}$ retrieves ($S_{ijk}, name_i, i, j, k,$ $z_{name_i ijk}, h_{3,name_i ijk}$) from the list $\{(S_{ijk}, name_i, i, j, k, z_{name_i ijk}, h_{3,name_i ijk})\}$ and responds as $H_3(S_{ijk} \| name_i \| i \| j \| k) = h_{3,name_i ijk}$.
- Else $\boldsymbol{B}$ computes $h_{3,name_i ijk} = g^{z_{name_i ijk}}$ by picking $z_{name_i ijk} \in Z_q$ to add（$S_{ijk}, name_i, i, j, k,$ $z_{name_i ijk}, h_{3,name_i ijk}$) in its list. It responds as $H_3(S_{ijk} \| name_i \| i \| j \| k) = h_{3,name_i ijk}$.

· Extract-oracle: Given identity ID_i, $\boldsymbol{B}$ retrieves $(ID_i, d_i, y_i, h_{1,i})$ from H_1-list, and extracts the identity-based private key $sk_i = h_{1,i}^a = (g^a)^{y_i}$ if $d_i = 0$. Meanwhile, (ID_i, sk_i) is added in the $Extract$ -list. If $d_i = 1$, $\boldsymbol{B}$ aborts.

· TagGen-oracle: To ensure $\boldsymbol{A}$'s view as real instances, simulator $\boldsymbol{B}$ maintains Tag -list as $\{(name_i, i, j, k, M_{ijk}, S_{t,ijk}, \sigma_{ijk})\}$ to responds $\boldsymbol{A}$'s each query for blocks M_{ijk}, of the file named $name_i$, as follows:

- If $(name_i, i, j, k, M_{ijk})$ is already in the Tag -list, $\boldsymbol{B}$ retrieves σ_{ijk} from the record and responds as tag for corresponding $(name_i, i, j, k, M_{ijk})$.
- Else $\boldsymbol{B}$ looks up H_1-list for $H_1(ID_i)$, H_2-list for $H_2(name_i \| i \| j \| k)$, makes query for itself if any corresponding record does not exist.
  · If $d_i = 0$ for corresponding record in the H_1 -list, $\boldsymbol{B}$ selects a random $S_{t,ijk} \in G_1$ ($t \in [1, \cdots, q_T]$) and queries H_3-oracle for $H_3(S_{t,ijk} \| name_i \| i \| j \| k)$, where q_T is the upper bounding number of tag query. The tag is generated as follows:

$$\sigma_{ijk} = (S_{t,ijk}, (g^a)^{y_i \sum_{l=1}^{s} v_l F_{ijkl} + H_2(name_i||i||j||k)} \cdot S_{t,ijk}^{z_{name_i ijk}})$$

and the new record ($name_i$, i, j, k, M_{ijk}, $S_{t,ijk}$, σ_{ijk}) are added into *Tag*-list.

· Else ***B*** aborts.

Forgery Output: Finally, ***A*** itself outputs a valid tag $\sigma_{i^* j^* k^*}$ satisfying equation (3), where neither has it queried TagGen-oracle for tag of the corresponding data block $M_{i^* j^* k^*}$ of file named $name_{i^*}$ nor has it retrieved the private key for identity ID_{i^*} from Extract-oracle. It means that there could be queried records of the specific ID_{i^*} with corresponding d_{i^*} only in H_1-oracle's list, but no queried records existed for the same ID_{i^*} with same d_{i^*} for TagGen-oracle and Extract-oracle's lists. Then, for the above ($name_{i^*}, i^*, j^*, k^*$), ***B*** looks up H_1-list for $H_1(ID_{i^*})$, H_2-list for $H_2(name_{i^*} \| i^* \| j^* \| k^*)$, H_3-list for $H_3(S_{i^* j^* k^*} \| name_{i^*} \| i^* \| j^* \| k^*)$, makes query for itself if any corresponding record does not exist. If not the case where $d_{i^*} = 1$ for $H_1(ID_{i^*})$, ***B*** aborts. Otherwise, a solution of CDH problem is obtained.

Since $\sigma_{i^* j^* k^*} = (S_{i^* j^* k^*}, T_{i^* j^* k^*})$ satisfies equation (3) like:

$$e(T_{i^* j^* k^*}, g) = e(H_1(ID_{i^*})^{\sum_{l=1}^{s} v_l F_{i^* j^* k^* l^*} + H_2(name_{i^*}||i^*||j^*||k^*)}, mpk) \cdot e(H_3(S_{i^* j^* k^*} \| name_{i^*} \| i^* \| j^* \| k^*), S_{i^* j^* k^*})$$

With $d_{i^*} = 1$, $H_1(ID_{i^*}) = g^{by_{i^*}}$ in H_1-list, and $H_3(S_{i^* j^* k^*} \| name_{i^*} \| i^* \| j^* \| k^*) = g^{z_{name_{i^*} i^* j^* k^*}}$ in H_3-list, $mpk = g^a$:

$$e(T_{i^* j^* k^*}, g) = e((g^{by_{i^*}})^{\sum_{l=1}^{s} v_l F_{i^* j^* k^* l^*} + H_2(name_{i^*}||i^*||j^*||k^*)}, g^a) \cdot e(g^{z_{name_{i^*} i^* j^* k^*}}, S_{i^* j^* k^*})$$

$$= e((g^{ab})^{y_{i^*}(\sum_{l=1}^{s} v_l F_{i^* j^* k^* l^*} + H_2(name_{i^*}||i^*||j^*||k^*))} \cdot S_{i^* j^* k^*}^{z_{name_{i^*} i^* j^* k^*}}, g)$$

Therefore, the solution of CDH problem instance for given g^a, g^b, is obtained as:

$$g^{ab} = \left(T_{i^* j^* k^*} \Big/ S_{i^* j^* k^*}^{z_{name_{i^*} i^* j^* k^*}} \right)^{y_{i^*}^{-1} (\sum_{l=1}^{s} v_l F_{i^* j^* k^* l^*} + H_2(name_{i^*}||i^*||j^*||k^*))^{-1}}$$

Secondly, **Forgery of Proof of Storage** for *Proof-Forge* game:

· Setup: Simulator ***B*** plays in the role of PKG to choose random $a \in Z_q$ as master private key *msk* and upon generator $g \in G_1$ master public key $mpk = g^a$. It also picks random $b \in Z_q$, and thus the instance of CDH problem as given g^a, g^{ab}, computing g^{ab}.

· H_1-oracle, H_2-oracle, H_3-oracle, Extract-oracle, TagGen-oracle remain the same as above.

· Firstphase Queries: ***A*** could access all the hash oracles and query identity based private keys and tags.

· Challengephase: Let us denote the tuples of tag queries for TagGen-oracle in Firstphase Queries as $I_1=\{(name_i,i,j,k,M_{ijk})\}$, and S_1 as the set of identities $\{ID_i\}$ which have extracted private keys for Extract-oracle}. ***B*** generates challenge token $chal^*=(I^*,K^*)$ for $I^*=\{(name_{i_n^*}, i_n^*, j_n^*, k_n^*) \mid 1\le n\le c^*\}$ and $K^*=\{\kappa_{i_n^*} \mid 1\le n\le c^*\}$, where at least one $ID_{i_n^*}\notin S_1$ and at least one tuple $(name_{i_n^*}, i_n^*, j_n^*, k_n^*, M_{i_n^* j_n^* k_n^*})\notin I_1$. $chal^*$ is sent to ***A***.

· Secondphase Queries: ***A*** could access all the hash oracles and query identity based private keys and tags. Let us denote the tuples of tag queries for TagGen-oracle in this phase as $I_2=\{(name_i,i,j,k,M_{ijk})\}$ and S_2 as the set of identities $\{ID_i\}$which have extracted private keys for Extract-oracle. We require that there is at least one $ID_{i_n^*}\notin(S_1\cup S_2)$ and at least one tuple $(name_{i_n^*}, i_n^*, j_n^*, k_n^*, M_{i_n^* j_n^* k_n^*})\notin(I_1\cup I_2)$ for the challenge token $chal^*=(I^*,K^*)$.

Forgery Output: Finally, ***A*** generates a valid proof of storage $P^*=\{P_{j^*}\}=\{\{S'_{i_n^* j^*} \mid i_n^*\in chal_j\}, T'_{j^*}, \{F'_{i^* j^* l^*} \mid (i^*, j^*, k^*)\in I^*\}\}$ for challenge token $chal^*=(I^*,K^*)$ where $I^*=\{(name_{i_n^*}, i_n^*, j_n^*, k_n^*) \mid 1\le n\le c^*\}$ and $K^*=\{\kappa_{i_n^*} \mid 1\le n\le c^*\}$. For $\{(name_{i^*}, i_n^*, j_n^*, k_n^*, S'_{i_n^* j_n^* k_n^*})\}$, where $S'_{i_n^* j_n^* k_n^*}=S'_{i_n^* j^*}, \forall j_n=j$, ***B*** looks up H_1-list for $H_1(ID_{i^*})$, H_2-list for $H_2(name_{i_n^*}\|i_n^*\|j_n^*\|k_n^*)$, H_3-list for $H_3(S_{i_n^* j_n^* k_n^*}\|name_{i_n^*}\|i_n^*\|j_n^*\|k_n^*)$, makes query for itself if any corresponding record does not exist. Upon all the $\{d_{i_n^*} \mid 1\le n\le c^*\}$ from H_1-list, ***B*** proceeds as follows:

· If every $d_{i_n^*}=0$, aborts.

· Else there is at least one $d_{i_n^*}\neq 0$, ***B*** computes $\{r_n^*\}=\{f_{\kappa_{i_n^*}}(i_n^*, j_n^*, k_n^*) \mid n=1,\cdots,c^*\}$, and denotes index set $\{i^*\}$ for all the challenged data owners as O^* and index set $\{j^*\}$ for all challenged cloud servers as C^*. The valid proof of storage $P^*=\{P_{j^*}\}=\{\{S'_{i_n^* j^*}\}_{i_n^*\in chal_j}, T'_{j^*}, \{F'_{i^* j^* l^*} \mid (i^*, j^*, k^*)\in I^*\}\}$ certainly satisfies equation (4), and thus:

$$e\Big(\prod_{j^*\in C^*} T'_{j^*}, g\Big)=e\Big(\prod_{i^*\in O^*} H_1(ID_{i^*})^{\sum_{j^*\in C^*}\sum_{l^*=1}^{s} v_{l^*} F'_{i^* j^* l^*}+\sum_{i_n^*=i^*} H_2(name_{i_n^*}\|i_n^*\|j_n^*\|k_n^*) r_n^*}, mpk\Big)$$
$$\cdot\prod_{i^*\in O^*} e\Big(\prod_{j\in C^*}\prod_{i_n^*=i^*, chal_j} H_3(S'_{i_n^* j^* k_n^*}\|name_{i_n^*}\|i_n^*\|j^*\|k_n^*)^{r_n^*}, S'_{i^*}\Big)$$

where $S'_{i^*}=S'_{i_n^* j^*}=S'_{i_n^* j^* k_n^*}=g^{u_{i^*}}, \forall i_n^*=i$, $S'_{i_n^* j^* k_n^*}=S'_{i_n^* j^*}=S'_{i_n^* j_n^* k_n^*}$.

For the identical blocks index set $\{(i_n^*, j_n^*, k_n^*) \mid i_n^*\in chal_j, j\in C^*, n\in[1,c^*]\}$, ***B*** is able to extract all original blocks' sectors $\{F^*_{i_n^* j_n^* k_n^* l}\}$ by solving a system of c^* linear equations in exactly c^* times of challenge interactions with ***A***, since each $F^*_{i_n^* j_n^* k_n^* l}$ is combined with c^* number of $\{r_n^{*(t)}\}_{t\in[1,c^*]}$ as $\{F'^{(t)}_{i_n^* j_n^* l}\}_{t\in[1,c^*]}$, where $F'^{(t)}_{i_n^* j_n^* l}=\sum_{i_n^*\in chal_j} F^*_{i_n^* j_n^* k_n^* l}\cdot r_n^{*(t)}$ in Proofgen.

For simplicity, we do not additionally consider this polynomial extraction time.

Regarding the correctness of equation (4),

$$e(\prod_{j\in C^*} T'_{j^*}, g) = e(\prod_{n=1}^{c^*} H_1(ID_{i_n^*})^{[m^*_{i_n^* j_n^* k_n^*} + H_2(name_{i_n^*} \| i_n^* \| j_n^* \| k_n^*)]r_n^*}, mpk)$$

$$\cdot \prod_{n=1}^{c^*} e(H_3(S'_{i_n^* j_n^* k_n^*} \| name_{i_n^*} \| i_n^* \| j_n^* \| k_n^*)^{r_n^*}, S'_{i_n^* j_n^* k_n^*})$$

We let $N_0 = \{n \mid (ID_i, d_i, y_i, h_{1,i}) \in H_1\text{-list}, d_{i_n^*} = 0\}$, $N_1 = \{n \mid (ID_i, d_i, y_i, h_{1,i}) \in H_1\text{-list}, d_{i_n^*} = 1\}$, where $N_1 \neq \varnothing$, $m^*_{i_n^* j_n^* k_n^*} = \sum_{l^*=1}^{s} v_{l^*} F_{i_n^* j_n^* k_n^* l^*}$, and thus:

$$e(\prod_{j\in C^*} T'_{j^*}, g) = e(\prod_{n\in N_0 \cup n\in N_1} H_1(ID_{i_n^*})^{[m^*_{i_n^* j_n^* k_n^*} + H_2(name_{i_n^*} \| i_n^* \| j_n^* \| k_n^*)]r_n^*}, mpk)$$

$$\cdot \prod_{n=1}^{c^*} e(H_3(S'_{i_n^* j_n^* k_n^*} \| name_{i_n^*} \| i_n^* \| j_n^* \| k_n^*)^{r_n^*}, S'_{i_n^* j_n^* k_n^*})$$

$$= e(\prod_{n\in N_0} g^{w_{name_{i_n^*} i_n^* j_n^* k_n^*}} \cdot \prod_{n\in N_1} g^{bw_{name_{i_n^*} i_n^* j_n^* k_n^*}}, g^a) \cdot \prod_{n=1}^{c^*} e(g^{z_{name_{i_n^*} i_n^* j_n^* k_n^*} \cdot r_n^*}, S'_{i_n^* j_n^* k_n^*})$$

$$= e((g^a)^{\sum_{n\in N_0} w_{name_{i_n^*} i_n^* j_n^* k_n^*}} \cdot (g^{ab})^{\sum_{n\in N_1} w_{name_{i_n^*} i_n^* j_n^* k_n^*}} \cdot \prod_{n=1}^{c^*} S'^{\,z_{name_{i_n^*} i_n^* j_n^* k_n^*} \cdot r_n^*}_{i_n^* j_n^* k_n^*}, g)$$

where $w_{name_{i_n^*} i_n^* j_n^* k_n^*} = y_{i_n^*}([m^*_{i_n^* j_n^* k_n^*} + H_2(name_{i_n^*} \| i_n^* \| j_n^* \| k_n^*)]r_n^*)$, and $\forall i_n^* = i$, $S'_{i_n^* j_n^* k_n^*} = S'_{i_n^* j^* k_n^*} = S'_{i_n^* j^*}$. Therefore, the solution of CDH problem for given g^a, g^b, is obtained as:

$$g^{ab} = \left(\prod_{j\in C^*} T'_{j^*} \Big/ \left(\prod_{n=1}^{c^*} S'^{\,z_{name_{i_n^*} i_n^* j^* k_n^*} \cdot r_n^*}_{i_n^* j^*} \cdot (g^a)^{\sum_{n\in N_0} w_{name_{i_n^*} i_n^* j_n^* k_n^*}} \right) \right)^{[\sum_{n\in N_1} w_{name_{i_n^*} i_n^* j_n^* k_n^*}]^{-1}}$$

Probability and Time Analysis

· We analyze ***B*** 's probability and time of solving CDH problem with the ***A*** 's ability to forge tag of our improved scheme. For the following four events:

- $\boldsymbol{E}_1$: ***B*** does not abort for any ***A*** 's Extract queries.
- $\boldsymbol{E}_2$: ***B*** does not abort for any ***A*** 's TagGen queries.
- $\boldsymbol{E}_3$: ***A*** generates a valid tag $\sigma_{i^* j^* k^*}$ for $(name_{i^*}, i^*, j^*, k^*, M_{i^* j^* k^*})$.
- $\boldsymbol{E}_4$: Event $\boldsymbol{E}_3$, $d_{i^*} = 1$ in H_1-list, for the above $name_{i^*}, i^*, j^*, k^*$.

If ***A*** succeeds in all the events, ***B*** 's probability of solving CDH problem is: $\Pr[\boldsymbol{E}_1 \wedge \boldsymbol{E}_2 \wedge \boldsymbol{E}_3 \wedge \boldsymbol{E}_4] = \delta^{qE}\delta^{qT}\varepsilon_1(1-\delta)$. With $\delta=(q_E+q_T)/(q_E+q_T+1)$, the probability is at least $\varepsilon_1/(\hat{e}(q_E+q_T+1))$, where $\hat{e}$ is the natural logarithm.

The total running time of ***B*** comprises of ***A***'s running time t_1 and additional time, where ***B*** responds with (q_H+q_T) hash queries, q_E Extract queries, q_T TagGen queries and final CDH problem transforming time. Both hash response and Extract require at most once exponentiation on group G_1 for each query, while it takes twice exponentiation for TagGen oracle query and once inversion and twice exponentiation on G_1 for final output of CDH

solution. Since among the TagGen oracle, tag σ_{ijk} is calculated by multiply power of g^a and $S_{t,ijk}$ respectively. Then, it takes twice exponentiation for TagGen oracle query on group on G_1.

In the final CDH solution, $g^{ab} = \left(T_{i^*j^*k^*} \Big/ S_{i^*j^*k^*}^{z_{name_{i^*}\|i^*j^*k^*}} \right)^{y_{i^*}^{-1}(\sum_{l=1}^{s} v_l F_{i^*j^*k^*l^*} + H_2(name_{i^*}\|i^*\|j^*\|k^*))^{-1}}$. For the inner base, $S_{i^*j^*k^*}$ first takes one exponentiation and one inversion to be $\left(S_{i^*j^*k^*}^{z_{name_{i^*}\|i^*j^*k^*}} \right)^{-1}$. Then the base $T_{i^*j^*k^*} \Big/ S_{i^*j^*k^*}^{z_{name_{i^*}\|i^*j^*k^*}}$ could be calculated, which requires another exponentiation G_1 to generate g^{ab}. Therefore, final CDH solution will cost once inversion of group element and twice exponentiation on G_1.

We assume both inversion and exponentiation operations on G_1 require t_{G_1}. Therefore, the total running time is at most $t_1 + t_{G_1} \cdot (q_H + q_E + 3q_T + 3)$.

· We analyze ***B*** 's probability and time of solving CDH problem with the ***A*** 's ability to forge proof of our improved scheme. For the following four events:

- $\boldsymbol{E}_1$: ***B*** does not abort for any ***A*** 's Extract queries.
- $\boldsymbol{E}_2$: ***B*** does not abort for any ***A*** 's TagGen queries.
- $\boldsymbol{E}_3$: ***A*** generates a valid proof of $P^* = \{P_{j^*}\} = \{\{S'_{i_n^* j^* k_n^*}\}_{i_n^* \in chal_j}, T'_{j^*}, \{F'_{i^* j^* l^*} \mid (i^*, j^*, k^*) \in I^*\}\}$ for challenge token $chal^* = (I^*, K^*)$ where $I^* = \{(name_{i_n^*}, i_n^*, j_n^*, k_n^*) \mid 1 \le n \le c^*\}$ and $K^* = \{\kappa_{i_n^*} \mid 1 \le n \le c^*\}$.
- $\boldsymbol{E}_4$: Event $\boldsymbol{E}_3$, at least one $d_{i_n^*} = 1$ in H_1 -list, for above tuples $\{(name_{i_n^*}, i_n^*, j_n^*, k_n^*) \mid 1 \le n \le c^*\}$

If ***A*** succeeds in all the events, ***B*** 's probability of solving CDH problem is: $\Pr[\boldsymbol{E}_1 \wedge \boldsymbol{E}_2 \wedge \boldsymbol{E}_3 \wedge \boldsymbol{E}_4] = \delta^{qE}\delta^{qT}\varepsilon_2(1-\delta^{c^*}) \geqslant \delta^{c^*(qE+qT)}\varepsilon_2(1-\delta^{c^*})$. With $\delta = \left((q_E + q_T) / (q_E + q_T + 1) \right)^{\frac{1}{c^*}}$, the probability is at least $\varepsilon_2/(\hat{e}(q_E+q_T+1))$, where $\hat{e}$ is the natural logarithm.

The total running time of ***B*** comprises of ***A*** 's running time t_2 and additional time, where ***B*** responds with (q_H+q_T) hash queries, q_E Extract queries, q_T TagGen queries and final CDH problem transforming time. Both hash response and Extract require at most once exponentiation on group G_1 for each query, while it takes twice exponentiation for TagGen oracle query and once inversion and (c^*+2) exponentiation on G_1 for final output of CDH solution. For this CDH solution, the analysis of cost is similar to previous method.We assume both inversion and exponentiation operations on G_1 require t_{G_1}. Therefore, the total running time is at most $t_2 + t_{G_1} \cdot (q_H + q_E + 3q_T + c^* + 3)$.

We complete the proof.

6. Efficiency Analysis

In this section, we compare cost of computation and communication of our improved scheme, with Peng et al.'s secure ID-DPDP [18] and Zhou et al.'s original ID-BPDP [17], summarized in **Table 1**, and **Table 2**, respectively. In addition, the performance comparison on computation is depicted in **Fig. 2**, based on results from simulation of Peng et al.'s improved ID-DPDP [18] and our proposed scheme on a laptop, to evaluate efficiency trend when number of data owners, clouds and data amount increases.

Table 1. Computation Cost Comparison for Multiple Owners and Multiple Clouds

Schemes	TagGen	ProofGen	Verify	Security
Peng's ID-DPDP [18]	$N(s+1)C_{exp}+NsC_h$	$cC_{exp}+csC_h$	$2n_1C_e+[(s+1)n_1+c]C_{exp}$	Secure
Zhou's ID-BPDP [17]	$(N+2n_O)C_{exp}$	$2cC_{exp}$	$3C_e+n_1C_{exp}$	Not secure
Our scheme	$(2N+n_O)C_{exp}$	cC_{exp}	$(n_1+2)C_e+(n_1+c)C_{exp}$	Secure

Table 2. Communication Cost Comparison for Multiple Owners and Multiple Clouds

Schemes	Challenge	ProofGen	Security
Peng's ID-DPDP [18]	$(n_1+c)\log_2 N+(2n_1+c)\log_2 q$	$n_1(n_2+1)\mathcal{G}_1+n_1(n_2+1)s\log_2 q$	Secure
Zhou's ID-BPDP [17]	$c\log_2 N+c\log_2 q$	$2n_2\mathcal{G}_1+cs\log q$	Not Secure
Our scheme	$c\log_2 N+c\log_2 q$	$n_2(n_1+1)\mathcal{G}_1+n_1n_2s\log_2 q$	Secure

- Assume there are n_O data owners storing total N blocks $\{M_{ijk}\}$ on n_J clouds, by only *one-off* TagGen and upload. To prove data integrity, *periodical* Challenge and Verify will be executed between clouds and verifier, upon randomly selected c data blocks and tags of n_1 data owners on n_2 clouds, for s sectors per block and element size of group G_1 is $\mathcal{G}_1$. Consequently, the dominant cost of this scheme is mostly contributed by ProofGen and Verify.
- Among all the operations, bilinear pairings C_e, exponentiation C_{exp} on group G_1, and hash C_h on blocks are most expensive, compared with multiplication on G_1 and G_2, operation on Z_q, and other hash operations, which are efficient or can be done for only once. That is why we do not consider computation cost of Challenge mostly relying on efficient operations. Additionally, since ID-DPDP only offers single owner's PDP on multiple clouds, we consider repeating n_1 loops of ID-DPDP instances, with N/n_1 outsourced blocks and only challenged c/n_1 blocks per loop.

Analysis for computation: In order to fully protect tags $\{\sigma_{ijk}=(S_{ijk},T_{ijk})\}$ from being tampered and utilized to recover private keys by adversaries, n_O data owners initially require $(2N+n_O)C_{exp}$ operation in TagGen. Luckily, these could be performed off line for owners as one-off task, although a little bit expensive. In ProofGen, computation is cC_{exp} for all $\{P_j\}$. In Verify, to remedy security flaws, i.e., tag universal forgery and private key recovery of ID-BPDP, we need (n_1+2) bilinear pairing computation to allow batch verification at one time, which thus achieves enhanced security and still outperforms $2n_1$ pairings in Peng et al.'s secure ID-DPDP [18], if applied to the multiple clouds and multiple owners scenario.

Analysis for communication: Communication for Challenge remains the same as ID-BPDP [17]. In order to fully protect tags $\{\sigma_{ijk}=(S_{ijk},T_{ijk})\}$ from being tampered by adversaries in [17], we need to increase output of ProofGen to further help Verify to check

authenticated $\{S_{ijk}\}$ and file names $\{name_i\}$. But the total overhead of transmission is still smaller than Peng et al.' s secure ID-DPDP [18] if applied to the multiple clouds and multiple owners' setting in **Table 2**. Meanwhile, our improved scheme does not suffer from the tag universal forgery and private key recovery as ID-BPDP [17].

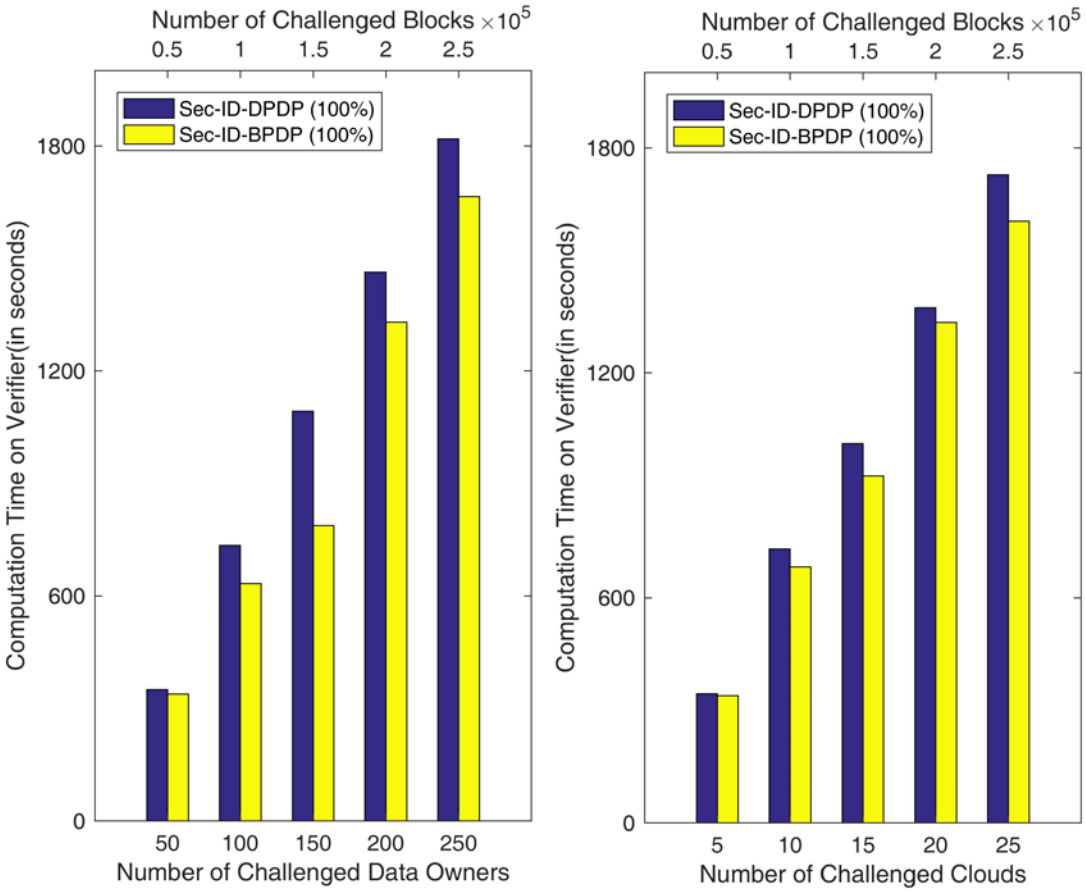


Fig. 2. Comparison of computation on Verifier:
1) as number of Owners increases: Total 10 Clouds of each stores 100 blocks per owner;
2) as number of Clouds increases: Total 100 Owners of each outsources 100 blocks per cloud

Simulation: In order to compare the performance about Peng et al.'s scheme [18] versus our scheme, we simulate data owners, storage clouds, and verifier on a laptop of Intel core i5 480 M at 2.67 GHz and 4G RAM running Linux operation system (Ubuntu 17.04 64bit with kernel 4.10.0-37-generic), in C programming language. Both of schemes are based on Pairing-Based Cryptography Library (PBC 0.5.14) [19], GNU Multiple Precision Arithmetic Library (GMP 6.1.2) [20] and OpenSSL Library (OpenSSL-1.1.0) [21].

To achieve 80-bit AES level of security, the elliptic curve we are using is of 160 bit group order with 512 bit length finite field element, from Type-A pairing in PBC library. Therefore, the size of element is $\mathcal{G}_1$=64 Bytes, and q is 20 Bytes length prime. For generating challenging co-efficiency $\{r_n\}$, we consider HMAC-SHA256 as pseudo random function f in OpenSSL library. We set each data block M_{ijk} split into s=50 sectors $\{F_{ijkl}\}$, and thus a sector is 20 B and a block is 1000B. The simulation has run 10 trials and collected their mean values as results.

For computation, our improved scheme acquires less cost than Peng et al. scheme in simulating TagGen, ProofGen. For generating tags of total 100000 blocks for 100 data owners, our scheme with TagGen costs 733.927 seconds while Peng et al.'s scheme requires 1148.584 seconds. In order to prove total 100000 blocks outsourced on 10 clouds for 100 data owners, running time of ProofGen is 258.048 seconds of our scheme versus 276.136 seconds in Peng et al's scheme.

For complicated multi-cloud & multi-owners storage scenario, we will study the critical computation performance comparison on verifier, for Peng et al. scheme (as Sec-ID-DPDP) and our improved scheme (as Sec-ID-BPDP).

On the left half of **Fig. 2**, the computation time on verifier's side is depicted for Peng et al.'s scheme [18] (marked in blue bar) and our improved scheme (in yellow bar), when challenged data owners increases from 50 to 250. For the fairness of evaluation, we repeat Peng et al.'s scheme to achieve the same number of data owners. Assume there are 10 clouds, each of which has 100 blocks for every data owner, and the total number of challenged data blocks will range from 0.5×10^5 to 2.5×10^5 (marked on the top X-axis in red color), i.e. from 50MB to 250MB, based on 100% probability to detect 1% rate of modification. It is illustrated that our improved scheme has less computation overheads on verifier's side versus Peng et al.'s scheme in this scenario.

Followed up with the right half, in **Fig. 2**, we present the computation time of on verifier's side as the number of challenged clouds increases from 5 to 25, for Peng et al. scheme [18] (marked in blue bar) and our improved scheme (in yellow bar), based on 100% probability to detect 1% rate of modification. Imagine there are 100 data owners, each of which outsources 100 blocks on every cloud, and thus the total number of challenged data blocks will range from 0.5×10^5 to 2.5×10^5 (shown on the top X-axis), i.e. from 50MB to 250MB. For the fairness of evaluation, we also repeat Peng et al.'s scheme to achieve the same number of data owners. It is shown that our improved scheme introduces less computation overheads on verifier's side versus Peng et al.'s scheme also in this scenario.

The difference illustrated in the **Fig. 2** is able to predict their trend of performance upon extrapolation to real multiple clouds storage system, which are equipped with powerful CPUs and huge memories, even if the performance of two schemes are temporarily limited by our simulated laptop. Therefore, our improved scheme is more efficient than Peng at al.'s scheme for the secure big data storage, which might have billion number of data owners, large number of storage clouds and large volume of data, in terms of storage integrity.

For communication overheads, our scheme outperforms Peng et al.'s scheme [18] both in Challenge and ProofGen, since the number of challenged clouds n_2 is usually much smaller than the number of challenged data owners n_1. Our scheme requires $(c/8\log_2 N+20c)$ B and $(64n_2(n_1+1)+1000n_1n_2)$ B while Peng et al.'s scheme costs $((n_1+c)/8\log_2 N+20(2n_1+c))$ B and $(64n_1(n_2+1)+1000n_1(n_2+1))$ B, for Challenge and ProofGen respectively, upon 64 B element size of group G_1, 20B sector, and 50 sectors per block. Especially, in real multiple clouds, for large scale of data like big data storage, it could be more economic and efficient to adopt sampling checking technology to guarantee high modification detecting probability with relative small number of challenged blocks, e.g. as spot checking technology demonstrated in [7], the verifier is able to detect the 1% modified blocks with at 99 % probability, by challenging 460 number of blocks among entire multiple clouds storage system.

7. Conclusions and Open Problem

In this paper, we revisited an identity-based batch provable data possession (ID-BPDP) scheme [17] designed by Zhou et al. on conference ProvSec 2016, and demonstrated that any cloud server could deceive verifier by universally forging valid tags with only its two block-tag pairs. In particular, it is also feasible to recover a data owner's private key to generate tags by malicious cloud itself. This will inevitably incur potential impersonation, and might be leveraged to threaten digital and material properties bound to personal identity. Therefore, we propose our solution to repair the security flaws and thus enhance the security, at the expense of reasonable overheads while still enjoy better efficiency over Peng et al.'s scheme [18].

Despite these above security flaws, it is still of great value for Zhou et al. to tackle the batch PDP problem with identity based cryptography infrastructure. Exquisitely designing tags, might be promising work to allow constant pairing computation in verification, if sharing reasonable commons of tags among data owners and not affecting security spontaneously. As a future work, we will keep on seeking to improve the efficiency of our proposed scheme of enhanced security, and evaluate it based on real-world multiple owners & multiple clouds storage system, with sound security for data integrity.

Acknowledgments

This work is supported by the NSFC of China under Grant Number 61370203, State Scholarship Fund Program (No. 201506070077) of China Scholarship Council and the Science and Technology on Communication Security Laboratory Foundation under Grant 9140C1103 01110C1103. Part of this research is done during the first author’s visit to Professor Li Xiong’s Assured Information Management and Sharing (AIMS) Lab, at Emory University. The academic and language training are greatly appreciated from AIMS Lab.

Reference

[1] Seagate.com, “Data Age 2025: The Evolution of Data to Life-Critical. Don't Focus on Big Data; Focus on the Data That's Big,” March 2017.Article (CrossRef Link)

[2] IDC.com, “Worldwide Public Cloud Services Spending Forecast to Reach $122.5 Billion in 2017, According to IDC,” February 20, 2017.Article (CrossRef Link)

[3] S. Yu, C. Wang, K. Ren, W. Lou, “Achieving secure, scalable, and fine-grained data access control in cloud computing,” in *Proc. of Proceedings of INFOCOM*, 2010. Article (CrossRef Link)

[4] Z. Fu, X. Wu, C. Guan, X. Sun, K. Ren, “Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement,” *IEEE Trans. Inf. Forensics Secur.*, 11(12), 2706-2716, 2016. Article (CrossRef Link)

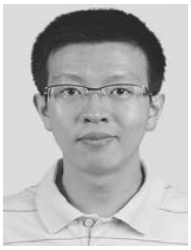
[5] Q. Wang, C. Wang, K. Ren, W. Lou, J. Li, “Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing,” *IEEE Trans. Parallel and Distributed Systems*, 22(5), 847-859, 2011. Article (CrossRef Link)

[6] C. Erway, A. Kupcu, C. Papamanthou, R. Tamassia, “Dynamic Provable Data Possession,” in *Proc. of Proceedings of ACM CCS 2009*, pp. 213-222, 2009. Article (CrossRef Link)

[7] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, D. Song, “Provable Data Possession at Untrusted Stores,” in *Proc. of Proceedings of ACM CCS 2007*, pp. 598-609, 2007. Article (CrossRef Link)

[8] H. Shacham, B. Waters , “Compact proofs of retrievability,” in *Proc. of Proceedings of ASIACRYPT 2008*, pp. 90-107, 2008. Article (CrossRef Link)

[9] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, W. Lou, "Privacy-Preserving Public Auditing for Secure Cloud Storage," *IEEE Transactions on Computers*, 62(2), 362-375, Feb. 2013. Article (CrossRef Link)
[10] Y. Zhu, H. Hu, G. J. Ahn, M.Yu, "Cooperative Provable Data Possession for Integrity Verification in MultiCloud Storage," *IEEE Trans. Parallel and Distributed Systems*, 23(12), 2231-2244, Dec., 2012. Article (CrossRef Link)
[11] Y. Yu, Y. Zhang, Y. Mu, W. Susilo, "Provably Secure Identity based Provable Data Possession," in *Proc. of Proceedings of ProvSec 2015*, LNCS 9451, pp. 1-16, Springer, Heidelberg, 2015. Article (CrossRef Link)
[12] Y. Yu, M. H. A. Au, G. Ateniese, X. Huang, W. Susilo, Y. Dai, G. Min , "Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage," *IEEE Transactions on Information Forensics and Security*, 2016. Article (CrossRef Link)
[13] J. Zhao, C. Xu, F. Li, W. Zhang, "Identity-based public verification with privacy preserving for data storage security in cloud computing," *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* vol. 96(12), 2709-2716, 2013. Article (CrossRef Link)
[14] J. Coron, "On the exact security of full domain hash," in *Proc. of Bellare, M. (ed.) CRYPTO 2000*. LNCS, vol. 1880, pp. 220-235. Springer, Heidelberg, 2000. Article (CrossRef Link)
[15] H. Wang, "Identity-Based Ditributed Provable Data Possession in Multicloud Storage," *IEEE Transactions on Services Computing*, Issue. 99, Mar, 2014. Article (CrossRef Link)
[16] B. Wang, B. Li, H. Li, "Panda: public auditing for shared data with efficient user revocation in the cloud," *IEEE Trans. Serv. Comput.*, 8(1), 92-106, 2015. Article (CrossRef Link)
[17] F. Zhou, S. Peng, J. Xu, Z. Xu, " Identity-based Batch Provable Data Possession," *in Proc. of Proceedings of Provable Security 2016*, Nanjing, China, pp. 112-129, October, 2016. Article (CrossRef Link)
[18] S. Peng, F. Zhou, J. Xu, Z. Xu, "Comments on "Identity-Based Distributed Provable Data Possession in Multicloud Storage" *IEEE Transactions on Services Computing*, 9(6), 996-998, Nov.-Dec, 2016. Article (CrossRef Link)
[19] The Pairing-Based Cryptography Library (PBC). Article (CrossRef Link)
[20] The GNU Multiple Precision Arithmetic Library (GMP). Article (CrossRef Link)
[21] OpenSSL: cryptography and SSL/TLS Toolkit. Article (CrossRef Link).
[22] L. Zhou, D. Wu, B. Zheng, and M. Guizani, "Joint physical-application layer security for wireless multimedia delivery,"IEEE Communications Magazine, vol. 52, no. 3, pp. 66-72, 2014. Article (CrossRef Link)
[23] L. Zhou, H.-C.Chao. "Multimedia traffic security architecture forinternet of things", IEEE Network, vol. 25, no. 3, pp. 35-40, 2011. Article (CrossRef Link)



Jining Zhao received his M.Sc. degree in University of Electronic Science and Technology of China (UESTC) in 2013, P.R. China. He is a Ph.D. degree candidate in information security at University of Electronic Science and Technology of China (UESTC). From 2016/03 to 2018/03, he worked as visiting research scholar in Department of Mathematics and Computer Science, Emory University, USA. He is presently engaged in cloud computing security, network security and cryptography.

Chunxiang Xu received her Ph.D. degrees at Xidian University, in 2004 , P.R. China. Dr. Xu is presently engaged in information security, cloud computing security and cryptography as a professor at University of Electronic Science and Technology of China (UESTC).

Kefei Chen received his Ph.D. degree from Justus Liebig University Giessen, Germany, in 1994. From 1996 to 2013, he was a professor in Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. Currently, he is a Distinguished Professor in School of Science, Hangzhou Normal University, China. His research interests include cryptography and network security.