Analysis and Improvement of an Efficient and Secure Identity-Based Public Auditing for Dynamic Outsourced Data with Proxy

Jining Zhao[1], Chunxiang Xu[1(✉)], and Kefei Chen[2]

[1] Center for Cyber Security, School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610051, China
kevinchao86@gmail.com, chxxu@uestc.edu.cn

[2] Hangzhou Key Laboratory of Cryptography and Network Security, Hangzhou Normal University, Hangzhou 311121, China
kfchen@hznu.edu.cn

Abstract. In big data age, flexible cloud service greatly enhances productivity for enterprises and individuals in different applications. When cloud access is restricted, data owner could authorize a proxy to process the data, and upload them to enjoy the powerful cloud storage service. Meanwhile, outsourced data integrity breach becomes a serious security issue for cloud storage. Identity Based Provable Data Possession (PDP) as a critical technology, could enable each data owner to efficiently verify cloud data integrity, without downloading entire copy and complicated public key certificate management issue. But it remains a great challenge for multiple data owners to efficiently and securely perform batch data integrity checking on huge data on different storage clouds, with proxy processing. Yu et al. recently proposed an Identity-Based Public Auditing for Dynamic Outsourced Data with Proxy Processing (https://doi.org/10.3837/tiis.2017.10.019), which tried to address this problem. In this article, we first demonstrate that this scheme is insecure since malicious clouds could pass integrity auditing without original data. Additionally, malicious clouds are able to recover the proxys private key and thus impersonate proxy to arbitrarily forge tags for any modified data. Secondly, in order to repair these security flaws, we propose an improved scheme to enable secure identity based batch public auditing with proxy processing. Thirdly, the security of our improved scheme is proved under CDH hard problem in the random oracle model. The complexity analysis of its performance shows better efficiency over identity-based proxy-oriented data uploading and remote data integrity checking in public cloud on single owner effort on a single cloud, which will benefit big data storage if it is extrapolated in real application.

Keywords: Cloud storage · Provable data possession
Identity-based cryptography · Provable security

F. Li et al. (Eds.): FCS 2018, CCIS 879, pp. 121–137, 2018.
https://doi.org/10.1007/978-981-13-3095-7_10

1 Introduction

In the age of Big Data with critical Data that is big, powerful cloud storage increasingly contributes to individuals life and enterprises business, by offering flexible and accessible data management services. From IDG report, 127 billion USD is spent globally on public cloud in 2017, with data storage size swelling to trillion gigabytes in 2025 [1]. For infrastructure, application and business processing service, cloud technology increasingly makes critical contribution, shifting to approximately 28% of the total market revenue in 2021 [2]. By managing huge data on different storage clouds, a great number of data owners enjoy customized applications for their business or utilities. When the access to cloud is restricted or owners' mobile devices are of limited computation capacity, a proxy with authorizations could perform data processing tasks before outsourcing them to remote cloud. With some protections from privacy preserving technologies [3], data owners still have to confront with security risks of outsourcing data integrity, due to system failures and external attacks. Meanwhile, cloud storage providers might have the incentives to delete cloud data and keep the accident news off their owners, for the sake of cost and reputations. Therefore, it is imperative to enable secure and efficient remote integrity checking for multiple owners, especially for cloud data which is originally processed by owners authorized proxy in the access restricted scenario.

Provable Data Possession (PDP) [4] as a critical technology, which is proposed by Ateniese et al., could allow efficient data integrity checking without having to download the entire data copy. Meanwhile, Shacham et al. designed proof of retrievability [5] to allow polynomial time data recovering and integrity checking. Based on Public Key Infrastructure (PKI), Wang et al. enabled cloud data integrity public auditing [6] with third party auditor, by performing PDP for single data owner in a privacy preserving manner. In [7,8], PDP is extended to support integrity auditing for data with dynamic update. For scalability of integrity checking tasks, Zhu et al. designed cooperative PDP for distributed cloud data integrity [9], and Yang et al. made further effort of enabling the multiple clouds' data integrity auditing for the multiple data owners [10]. Some works were designed to support data auditing with special features, such as multiple data storage replica [11] and group user data share [12] and revocation. For recent years, continuous progress has been made on data auditing in [13–15]. However, these famous works were all built on PKI, where each owner's public key certificate is required to be transferred and verified.

To eliminate the complicated management issue of public key certificates, Zhao et al. proposed the first identity-based public auditing scheme [16], to enable PDP primi-tive with identity based cryptography [17], where the efficiency is optimized from cryptosystem level. In 2015, Wang et al. designed the identity based distributed PDP to support multi-cloud storage for single owner [20]. In 2016, Liu et al. considered generic identity-based PDP construction [19] by combining PKI based PDP and Identity Based Signature [18]. Later, Yu et al. enabled zero knowledge privacy integrity checking for identity based PDP in [21]. In the setting of restricted cloud access, Wang et al. for the first time

proposed an identity based PDP scheme, called Identity-based proxy-oriented data uploading and remote data integrity checking in public cloud (ID-PUIC), to support single owners authorized proxy to process data for single cloud [22]. Spontaneously, security flaws were found in some classic designs but luckily were repaired in [18,19,23]. So the challenging problem still remained to be unsolved, i.e., how to efficiently perform multiple clouds data integrity checking for multiple data owners with proxy processing data.

In 2017, Yu et al. designed an identity based batch public auditing scheme [25], to facilitate secure data integrity checking on multiple clouds for multiple owners, and support proxy data processing, without public key certificate managing issue. Unfortunately, after careful analysis, this work is not able to address the challenging problem of better efficiency and security simultaneously, when coming across malicious behaviors.

Contributions: Firstly, we demonstrate that this work [25] is vulnerable to data loss attack and proxy private key recovering attack. Especially, malicious clouds are able to use masked data rather than original data to pass integrity checking, and arbitrary two pairs of data and tags are sufficient to recover private key of the authorized proxy. Secondly, we propose an improved scheme, which could perform integrity checking and resist these above security flaws. Thirdly, we prove security of our scheme in random oracle under CDH assumption. In the end, our improved scheme illustrates better efficiency of complexity over identity-based proxy-oriented data uploading and remote data integrity checking scheme in public cloud [22] on single owner effort on single cloud, which will benefit big data storage if extrapolated to real application.

Paper Organization: The rest of the paper starts with notations and reviews of definition of identity-based batch public auditing with proxy processing scheme (ID-BPAP) along with its system and security model in Sect. 2. After revisiting of ID-BPAP scheme in Sect. 3, two security flaws are demonstrated in Sect. 4. We present our improved scheme in Sect. 5, and formally prove its security in Subsect. 5.1 under random oracle model. In Sect. 6, we compare our improved scheme with Wang et al.s ID-PUIC, in the context of overheads based on complexity analysis, to study the trend of efficiency for computation and communication. Section 7 concludes our paper.

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 $e : G_1 \times G_1 \rightarrow G_2$.
- (mpk, msk) are the Private Key Generator (PKG)'s master public and private keys pair. sk_i is ith data owner's corresponding identity-based private key.

- There are n_o data owners, outsourcing N blocks on n_J clouds. $\tilde{F}_{ijk}$ is i-th owner's k-th block on cloud CS_j, with proxy tag σ_{ijk} from its masked F_{ijk} or encrypted $\hat{F}_{ijk}$.
- f is a pseudo random function (PRF) $f : Z_q \times \{1, \cdots, N\} \rightarrow Z_q$; π is a pseudo random permutation $\pi : Z_q \times \{1, \cdots, N\} \rightarrow \{1, \cdots, N\}$.
- $chal$ is challenge token generated by third party auditor (TPA). $chal_j$ is the specific challenge token for CS_j. c_{ij} is challenged number of blocks for ith owner and $a_{ij} \in [1, c_{ij}]$ further specifies index of each block as $k = \pi_{v_{ij,1}}(a_{ij})$.
- C is the index set of challenged data picked by TPA. O is the index set of data owner's identities upon challenged blocks, and J is the index set of challenged clouds, where $|O| = n_1$ and $|J| = n_2$. P_j is the proof of storage generated by CS_j.

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 $a, b \in Z_q$.

2.2 Definition of ID-BPAP

In this section, we will present the definition of Identity-Based Batch Public Auditing scheme with Proxy Processing (ID-BPAP) from the original paper [25], in the seven algorithms below.

1. $\mathsf{Setup}(1^k) \rightarrow (params, mpk, msk)$ is initialized by PKG with security parameter k. It outputs the public parameters $params$, master key pairs (mpk, msk).
2. $\mathsf{Extract}(params, msk, ID_i) \rightarrow sk_i$ is executed by PKG with as input $params$, master private key msk and data owner's identity ID_i, it outputs the private key sk_i for the owner. It also extracts private key sk_p for proxy of ID_p.
3. $\mathsf{ProxyKeyGen}(params, ID_i, sk_i, ID_p, sk_p) \rightarrow u_{pi}$ is run by proxy ID_p with interaction of data owner ID_i. With input of parameters $params$ and its private key sk_i, data owner generates warrant and corresponding signature to send to proxy. Then proxy outputs the proxy secret key u_{pi} with its private key sk_p.
4. $\mathsf{TagGen}(params, ID_i, sk_p, u_{pi}, mpk, \{\tilde{F}_{ijk}\}) \rightarrow \{\sigma_{ijk}\}$ is run by proxy. It takes as input parameters $params$, owner's identity ID_i, its individual private key sk_p, corresponding secret key u_{pi}, master public key mpk and owners' blocks $\{\tilde{F}_{ijk}\}$ to be outsourced. Then proxy tags $\{\sigma_{ijk}\}$ of above blocks could be generated.
5. $\mathsf{Challenge}(\{(i, j, k)\}) \rightarrow (chal, \{chal_j\})$ is executed by TPA. It takes as input data index set $\{(i, j, k)\}$ and selects some index as challenge token $chal$ for this instance. According to the specified indexes $\{j\}$, the challenge token $chal$ is further divided into a set of tokens $\{chal_j\}$ and only forwarding $chal_j$ to the cloud CS_j.
6. $\mathsf{ProofGen}(params, chal_j, \{ID_i\}, \{\sigma_{ijk}\}, \{\tilde{F}_{ijk}\}) \rightarrow P_j$ is run by cloud CS_j. It takes as input the parameters $params$, the challenge token $chal_j$, the specified set of data owners' identities $\{ID_i\}$, the set of tags $\{\sigma_{ijk}\}$, and the blocks $\{\tilde{F}_{ijk}\}$. Then the proof P_j is generated for challenge token $chal_j$, and is sent back to TPA.

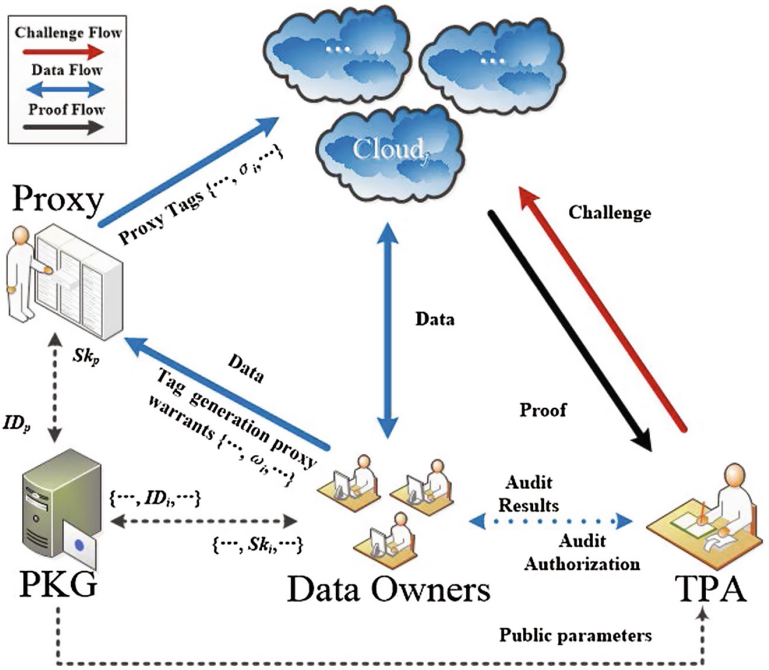

Fig. 1. Architecture of ID-BPAP

7. Verify$(params, chal, \{ID_i\}, \{P_j\}, mpk) \rightarrow \{0, 1\}$ is executed by TPA. 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

As it depicts in Fig. 1, there are five kinds of entities in an ID-BPAP scheme, i.e., the *PKG*, data *Owners*, *Proxy*, multiple *Clouds*, and a batch *TPA*. *PKG* initializes the system parameters and extracts private keys for data owners and proxy of their own identities. Data *Owners* delegate *Proxy* to process their massive data before storing them in multiple clouds. *Proxy* of abundant computation and bandwidth resource, helps data owners to generate proxy data tags and upload them to clouds, with data owners' special warrants. Multiple *Clouds* maintain powerful storage and computation resources to provide storage service for data owners. The batch *TPA* is a trusted third party auditor to offer the batch data integrity verification on multiple clouds for the data owners.

2.4 Security Model

In an ID-BPAP scheme, we assume PKG is trusted to execute the scheme, and proxy honestly generates tags but may have management fault of data before tag generation. Meanwhile, original data owners might generate data tag themselves without the delegated proxy. Clouds could also hide data accident for the sake of reputation and saving cost, and TPA is trusted but curious about the data content. A secure ID-BPAP scheme should satisfy three properties:

(1) Proxy-protection: Data owners themselves are not able to masquerade as proxy to generate tags. Only proxy with authorization warrant could generate proxy tags.
(2) Unforgeability: It is infeasible to fabricate valid data storage proofs to pass the auditing of TPA if any cloud data is modified or deleted.
(3) Privacy-preserving: Real data content will not be revealed during the process of auditing.

According to the security requirements, we review the three formal definitions as follows:

1. Definition of **Proxy-Protection:** The scheme is proxy-protected, if any probabilistic polynomial time (PPT) data owner wins proxy Tag-Forge game with negligible probability.
   Setup: Challenger $\mathcal{C}_1$ in the role of PKG and TPA, first generates master public/private key pair and system parameter. It runs Extract to generate private key sk_p for proxy of ID_p and keeps its secret. Those public and not secret parameters could be sent to adversary $\mathcal{A}_1$ as data owner.
   Queries: Besides all hash functions, $\mathcal{A}_1$ could adaptively query Extract for private key sk_i for identity ID_i except ID_p. Denote index set of identities as S_1 ($p \notin S_1$). It could also query proxy tag secret key $u_{p'i}$ for $(ID_{p'}, ID_i)$ except for pair having ID_p. Denote index set of pairs as $S_1^{'}$ ($(p,i) \notin S_1^{'}$). Upon block $\tilde{F}_{ijk}$, $\mathcal{A}_1$ could also adaptively query proxy tag $\sigma_{p'ijk}$ with the same identity requirement. Let us denote tuples set of indexes and corresponding block as $S_1^{''}, (p,i,j,k,\tilde{F}_{ijk}) \notin S_1^{''}$.
   Output: $\mathcal{A}_1$ wins the game if it creates a valid proxy tag $\sigma_{i^*j^*k^*}$ for data block $\tilde{F}_{i^*j^*k^*}$ by itself , for which it has neither extracted private key nor proxy tag secret key for proxy ID_p, i.e., where $p \notin S_1$, $(p,i^*) \notin S_1^{'}$, and $(p,i^*,j^*,k^*,\tilde{F}_{i^*j^*k^*}) \notin S_1^{''}$.
2. Definition of **Unforgeability:** The scheme is unforgeable if any PPT clouds win the Proofs-Forge game below, with negligible probability.
   Setup: Challenger $\mathcal{C}_2$ in the role of PKG and TPA, first generates master public/private key pair and system parameter. It runs Extract to generate private key sk_p for proxy of ID_p and keeps its secret. Those public and not secret parameters could be sent to adversary $\mathcal{A}_2$ as clouds.
   First phase queries: Besides all hash functions, $\mathcal{A}_2$ could adaptively query Extract for private key sk_i for identity ID_i except ID_p. Denote index set

of identities as S_2 ($p \notin S_2$). It could also query proxy tag secret key $u_{p'i}$ for $(ID_{p'}, ID_i)$ except for pair having ID_p. Denote index set of pairs as $S_2^{'}$ ($(p,i) \notin S_2^{'}$). Upon block $\tilde{F}_{ijk}$, $\mathcal{A}_2$ could also adaptively query proxy tag $\sigma_{p'ijk}$ with the same identity requirement. Let us denote tuples set of indexes and corresponding block as $S_2^{''}$, $(p,i,j,k,\tilde{F}_{ijk}) \notin S_2^{''}$.

Challenge: $\mathcal{C}_2$ generates challenge set *chal* with ordered number set $\{c_{i^*j^*}\}$ to specify every block $\tilde{F}_{i^*j^*k^*}$ on the j^*th cloud for owner of ID_{i^*}, where $\{(p,i^*,j^*,k_n^*)|\ 1 \le n \le c_{i^*j^*}\}$, $i^* \neq p$, $(p,i^*) \notin S_2^{'}$, and $(p,i^*,j^*,k_n^*,\tilde{F}_{i^*j^*k_n^*}) \notin S_2^{''}$. *chal* will be sent to $\mathcal{A}_2$.

Second phase queries: Similar to First phase queries, denote index set of identities for Extract private keys as S_3, index set of identity pairs for proxy tag secret key queries as $S_3^{'}$, tuple set of index and data for proxy tag queries as $S_3^{''}$. We require that $p \notin S_2 \cup S_3$, $(p,i) \notin S_2^{'} \cup S_3^{'}$, and $(p,i^*,j^*,k_n^*,\tilde{F}_{i^*j^*k_n^*}) \notin S_2^{''} \cup S_3^{''}$.

Output: $\mathcal{A}_2$ wins the game if it fabricates valid proofs $\{P_{j^*}\}$ for the same challenge *chal* on the specified set of blocks.

3. Definition of **Privacy-Preserving:** Proofs are privacy-preserving if TPA cannot retrieve original value about the cloud data during the auditing.

3 Revisiting of ID-BPAP

In this section,we will revisit the ID-BPAP scheme of seven algorithms in [25].

1. 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 $H_1: \{0,1\}^* \to G_1$, $H_2: \{0,1\}^* \to Z_q$, $H_3: Z_q \times \{0,1\}^* \to Z_q$, a pseudo permutation $\pi: Z_q \times \{1,\cdots,N\} \to \{1,\cdots,N\}$ and a pseudo random function $f: Z_q \times \{1,\cdots,n\} \to Z_q$. It picks random $x \in Z_q$ as master private key msk and computes g^x as master public key mpk. The global parameters are $(e, G_1, G_2, g, mpk, H_1, H_2, H_3, \pi, f)$.
2. 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. For proxy, $sk_p = H_1(ID_p)^x$.
3. ProxyKeyGen: For data owner ID_i, it picks up random $r_i \in Z_q$ and creates its proxy warrant ω_i with its signature $U_i = sk_i^{r_i H_2(\omega_i||R_i)}$, $\xi_i = g^{r_i}$, where $R_i = H_1(ID_i)^{r_i}$. $(\omega_i, U_i, R_i, \xi_i)$ are sent to proxy, clouds and TPA. Upon the warrant ω_i, TPA and proxy could verify it with signature as $e(U_i, g) = e(R_i^{H_2(\omega_i||R_i)}, mpk)$, $e(R_i, g) = e(H_1(ID_i), \xi_i)$, and notify the data owner if any equations does not hold. Proxy generates the proxy secret key as $u_{pi} = U_i \cdot sk_p^{r_{pi}} = H_1(ID_i)^{x r_i H_2(\omega_i||R_i)} \cdot H_1(ID_p)^{x r_{pi}}$ by picking up random $r_{pi} \in Z_q$. It also computes the not secret $R_{pi} = H_1(ID_p)^{r_{pi}}$, which is sent to TPA for future verification.
4. TagGen: Data owner of ID_i first divides original data $\tilde{F}_i$ into blocks $\{\tilde{F}_{ijk}\}$, and computes each $F_{ijk} = \tilde{F}_{ijk} + H_2(\tilde{F}_{ijk})$. Data blocks $\{\tilde{F}_{ijk}\}$ are outsourced

to corresponding clouds while masked $\{F_{ijk}\}$ are sent to proxy. Then proxy generates proxy tag for each data block as

$$\sigma_{ijk} = sk_p^{H_3(i||j||k,name_{ijk}||time_{ijk})} \cdot u_{pi}^{F_{ijk}} \tag{1}$$

where $name_{ijk}$ is the name of block $\tilde{F}_{ijk}$, and $time_{ijk}$ is the time stamp when proxy generates the tag. All the tags $\{\sigma_{ijk}\}$ and not secret R_{pi} will be transferred to corresponding clouds, which will not accept them and inform the owner unless the warrant ω_i and the proxy tag σ_{ijk} could be verified by having the following equations holds as

$$\begin{aligned} e(R_i, g) = e(H_1(ID_i), \xi_i), e(U_i, g) = e(R_i^{H_2(\omega_i||R_i)}, mpk) \\ e(\sigma_{ijk}, g) = e(H_1(ID_p)^{H_3(i||j||k,name_{ijk}||time_{ijk})} \cdot (R_i^{H_2(\omega_i||R_i)} \cdot R_{pi})^{F_{ijk}}, mpk) \end{aligned} \tag{2}$$

5. Challenge: For data owner of ID_i on jth cloud's data, TPA picks up number of challenged blocks as c_{ij} and random $v_{ij,1}$ and $v_{ij,2} \in Z_q$. Denote O_j as index set of identities for owners having data on jth cloud. It generates the challenge token $chal_j = \{(c_{ij}, v_{ij,1}, v_{ij,2})\}_{i \in O_j}$, and sends it to the cloud.
6. ProofGen: According to the challenge token $chal_j = \{(c_{ij}, v_{ij,1}, v_{ij,2})\}_{i \in O_j}$, cloud CS_j first generates index set δ_{ij} of challenged blocks for owner of ID_i where each index $k = \pi_{v_{ij,1}}(a_{ij})$ $(1 \leq a_{ij} \leq c_{ij})$ with specified challenge number c_{ij} and then the corresponding co-efficient $h_{ijk} = f_{v_{ij,2}}(i, j, k) \in Z_q$. The proof of storage P_j includes aggregate tag $T_j^{'}$ and masked data proof $\{F_{ij}^{'}\}$ for the data owners of identities with index set O_j:

$$T_j' = \prod_{i \in O_j} \prod_{k \in \delta_{ij}} \sigma_{ijk}^{h_{ijk}}, F_{ij}' = \sum_{k \in \delta_{ij}} h_{ijk} \cdot F_{ijk} \tag{3}$$

where $F_{ijk} = \tilde{F}_{ijk} + H_2(\tilde{F}_{ijk})$. $P_j = (T_j^{'}, \{F_{ij}'\}_{i \in O_j})$ will be sent to TPA.
7. Verify: After receiving all the proofs $\{P_j\}$ from challenged clouds, TPA denotes $O = \cup_{j \in J} O_j$ as identity index set of all the challenged data owners from challenge tokens $\{chal_j = \{(c_{ij}, v_{ij,1}, v_{ij,2})\}_{i \in O_j}\}_{j \in J}$, and computes index set of all challenged blocks by $\{k\} = \{\pi_{v_{ij,1}}(a_{ij}) | 1 \leq a_{ij} \leq c_{ij}\}$ and co-efficient set $\{h_{ijk}\} = \{f_{v_{ij,2}}(i, j, k)\}$, as in ProofGen. With all valid set of warrant $\{\omega_i\}$ and corresponding signatures $\{(U_i, R_i, \xi_i)\}$ from data owners, together with blocks names and time stamps $\{(name_{ijk}, time_{ijk})\}$, TPA is able to verify data integrity as:

$$\begin{aligned} e(\prod_{i \in O} (R_i^{H_2(\omega_i||R_i)} \cdot R_{pi})^{\sum_{j \in J} F_{ij}^{'}} \cdot H_1(ID_p)^{\sum_{i \in O} \sum_{j \in J} \sum_{k \in \delta_{ij}} h_{ijk} \cdot H_3(i||j||k,name_{ijk}||time_{ijk})}, mpk) \\ = e(\prod_{j \in J} T_j', g) \end{aligned} \tag{4}$$

It will output 1 (valid) if the above equation holds and 0 (invalid) otherwise.

4 On the Security of ID-BPAP

With security analysis in [25], Yu et al.'s ID-Batch Batch Public Auditing with Proxy Processing (ID-BPAP) should satisfy security properties for data proof with unforge-ability and tag generation with proxy-protection. However, their proposed ID-BPAP in [25], may suffer from two security issues, as the analysis in the following.

4.1 First Issue: Generating Valid Proof Without Original Data

In Yu et al.'s ID-BPAP scheme, the TPA utilizes masked data proof to evaluate the original data integrity on the cloud. This design indeed helps to prevent TPA obtain original data content, but also leaves the room for malicious clouds to launch data attack as follows.

In the ProofGen, for the output $P_j = (T_j^{'}, \{F_{ij}^{'}\}_{i \in O_j})$, honest cloud takes original data $\tilde{F}_{ijk}$ as input to get masked data $F_{ijk} = \tilde{F}_{ijk} + H_2(\tilde{F}_{ijk})$, and do the combination with the fresh challenge co-efficient $\{h_{ijk}\}$, as $F_{ij}^{'} = \sum_{k \in \delta_{ij}} h_{ijk} \cdot F_{ijk}$. That is to say, the data integrity proof, is generated by combining of fresh challenge co-efficient and masked data, rather than directly with the original data itself. Therefore, for malicious clouds, by pre-computing and storing masked data F_{ijk}, it is able to directly generate valid integrity proof $P_j = (T_j^{'}, \{F_{ij}^{'}\}_{i \in O_j})$, without having to store the original data $\tilde{F}_{ijk}$. In this way, malicious clouds could modify original data $\tilde{F}_{ijk}$ as $\tilde{F}_{ijk}^{*}$ or even delete it, and successfully pass TPAs integrity checking.

4.2 Second Issue: Recovering Private Key of Proxy and Proxy Tag Secret Key

With proxy-protection property, only proxy with authorization could generate the data tags for integrity verification. As analysis below, we could find that it is feasible to recover proxys private key and thus impersonate proxy to generate data tag, for those who could access the data and tags.

In TagGen, for data $\tilde{F}_{ijk}$, tag $\sigma_{ijk} = sk_p^{H_3(i||j||k, name_{ijk}||time_{ijk})} \cdot u_{pi}^{F_{ijk}}$ is generated by proxy, with its individual private key sk_p and proxy tag secret key u_{pi}, and then uploads tag on the cloud. Afterwards, malicious clouds or curious data owner of ID_i, retrieve two arbitrary data blocks $(\tilde{F}_{ijk_1}, \tilde{F}_{ijk_2})$ with corresponding tags $(\sigma_{ijk_1}, \sigma_{ijk_2})$, and do the computation:

$$sk_p = \left(\frac{\sigma_{ijk_1}^{\frac{1}{F_{ijk_1}}}}{\sigma_{ijk_2}^{\frac{1}{F_{ijk_2}}}}\right)^{\frac{F_{ijk_1}F_{ijk_2}}{H_3(i||j||k_1, name_{ijk_1}||time_{ijk_1})F_{ijk_2} - H_3(i||j||k_2, name_{ijk_2}||time_{ijk_2})F_{ijk_1}}}, u_{pi} = \left(\frac{\sigma_{ijk_1}^{\frac{1}{H_3(i||j||k_1, name_{ijk_1}||time_{ijk_1})}}}{\sigma_{ijk_2}^{\frac{1}{H_3(i||j||k_2, name_{ijk_2}||time_{ijk_2})}}}\right)^{Ep}$$

Where $Ep = \frac{H_3(i||j||k_2, name_{ijk_2}||time_{ijk_2})H_3(i||j||k_1, name_{ijk_1}||time_{ijk_1})}{F_{ijk_1}H_3(i||j||k_2, name_{ijk_2}||time_{ijk_2}) - F_{ijk_2}H_3(i||j||k_1, name_{ijk_1}||time_{ijk_1})}$, and masked data $(F_{ijk_1}, F_{ijk_2}) = (\tilde{F}_{ijk_1} + H_2(\tilde{F}_{ijk_1}), \tilde{F}_{ijk_2} + H_2(\tilde{F}_{ijk_2}))$. With the recovered proxy private key sk_p and proxy tag secret key u_{pi}, three kinds of

security problems could happen. First, for new block $\tilde{F}_{ijk_3}$, the proxy tag could be fabricated as $\sigma_{ijk_3} = sk_p^{H_3(i||j||k_3,name_{ijk_3}||time_{ijk_3})} \cdot u_{pi}^{F_{ijk_3}}$ by the data owner itself. This valid tag will keep Eqs. (2) (3) hold and finally help data to pass the TPA auditing in Eq. (4). And thus proxy-protection security property cannot be guaranteed. Second, if the original block is modified to $\tilde{F}^*_{ijk_3}$, malicious clouds could generate valid tag as $\sigma^*_{ijk_3} = sk_p^{H_3(i||j||k_3,name_{ijk_3}||time_{ijk_3})} \cdot u_{pi}^{F^*_{ijk_3}}$, where $F^*_{ijk_3} = \tilde{F}^*_{ijk_3} + H_2(\tilde{F}^*_{ijk_3})$, without awareness of data owner and proxy. Certainly this modified data-tag pair will also keep Eqs. (2) (3) hold and help to generate valid integrity proof in Eq. (4), but unforgeability property cannot be guaranteed for falling to check data modification. Third, the digital property belonging to proxy, will be in great risk of illegal access, due to the recovered proxy individual private key by other entities.

5 Improved Scheme

1. Setup: 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 four cryptographic hash functions $H_1 : \{0,1\}^* \rightarrow G_1$, $H_2 : \{0,1\}^* \rightarrow Z_q$, $H_3 : Z_q \times \{0,1\}^* \rightarrow Z_q$, $H_4 : \{0,1\}^* \rightarrow G_1$, a pseudo permutation $\pi : Z_q \times \{1, \cdots, N\} \rightarrow \{1, \cdots, N\}$ and a pseudo random function $f : Z_q \times \{1, \cdots, n\} \rightarrow Z_q$. It picks random $x \in Z_q$ as master private key msk and computes g^x as master public key mpk. The global parameters are $(e, G_1, G_2, g, mpk, H_1, H_2, H_3, H_4, \pi, f)$.
2. 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. For proxy, $sk_p = H_1(ID_p)^x$.
3. ProxyKeyGen: For data owner ID_i, it picks up random $r_i \in Z_q$ and creates its proxy warrant ω_i with its signature $U_i = sk_i^{r_i H_2(\omega_i||R_i)}$, $\xi_i = g^{r_i}$, where $R_i = H_1(ID_i)^{r_i}$. $(\omega_i, U_i, R_i, \xi_i)$ are sent to proxy, clouds and TPA. Upon the warrant ω_i, TPA and proxy could verify it with signature as $e(U_i, g) = e(R_i^{H_2(\omega_i||R_i)}, mpk)$, $e(R_i, g) = e(H_1(ID_i), \xi_i)$, and notify the data owner if any equations does not hold. Proxy generates the proxy secret key as $u_{pi} = U_i \cdot sk_p^{r_{pi}} = H_1(ID_i)^{xr_i H_2(\omega_i||R_i)} \cdot H_1(ID_p)^{xr_{pi}}$ by picking up random $r_{pi} \in Z_q$. It also computes the not secret $R_{pi} = H_1(ID_p)^{r_{pi}}$, $\phi_{pi} = g^{r_{pi}}$, which are sent to TPA for future verification.
4. TagGen: Data owner of ID_i first divides original data $\tilde{F}_i$ into blocks $\{\tilde{F}_{ijk}\}$. To ensure the data privacy, each block is converted in to its cipher text $\hat{F}_{ijk} \in Z_q$ by symmetric encryption. Cipher text blocks $\{\hat{F}_{ijk}\}$ are outsourced to corresponding clouds and sent to the proxy. For each data block, proxy generates tag $\sigma_{ijk} = (T_{ijk}, S)$ as

$$T_{ijk} = (sk_p, u_{pi})^{H_3(i||j||k,name_i||time_{ijk}) + \hat{F}_{ijk}} \cdot H_4(i||j||k, name_i||time_{ijk}||S)^{\eta}, S = g^{\eta} \tag{4}$$

where $name_i$ is the name of file $\tilde{F}_i$, and $time_{ijk}$ is the time stamp when proxy generates the tag. All the tags $\{\sigma_{ijk}\}$ and not secret R_{pi} will be transferred to

corresponding clouds, which will not accept them and inform the owner unless the warrant ω_i and the proxy tag σ_{ijk} could be verified by having the following equations holds as $e(R_i, g) = e(H_1(ID_i), \xi_i)$, $e(R_{pi}, g) = e(H_1(ID_i), \phi_{pi})$, $e(U_i, g) = e(R_i^{H_2(\omega_i||R_i)}, mpk)$

$$e(T_{ijk}, g) = e((H_1(ID_p) \cdot (R_i^{H_2(\omega_i||R_i)} \cdot R_{pi}))^{H_3(i||j||k, name_i||time_{ijk}) + \hat{F}_{ijk}}, mpk) \cdot e(H_4(i||j||k, name_i||time_{ijk}||S), S) \tag{5}$$

5. Challenge: For data owner of ID_i on jth cloud's data, TPA picks up number of challenged blocks c_{ij} and random $v_{ij,1}$ and $v_{ij,2} \in Z_q$. Denote O_j as index set of identities for owners having data on cloud CS_j. It generates the challenge token $chal_j = \{(c_{ij}, v_{ij,1}, v_{ij,2})\}_{i \in O_j}$, and sends it to the cloud.
6. ProofGen: According to the challenge token $chal_j = \{(c_{ij}, v_{ij,1}, v_{ij,2})\}_{i \in O_j}$, cloud CS_j first generates index set δ_{ij} of challenged blocks for owner of ID_i where each index $k = \pi_{v_{ij,1}}(a_{ij})$ $(1 \leq a_{ij} \leq c_{ij})$ with specified challenge number c_{ij} and then the corresponding co-efficient $h_{ijk} = f_{v_{ij,2}}(i, j, k) \in Z_q$. The proof of storage P_j includes aggregate tag $T_j^{'}, S^{'}$ and data proof $\{F_{ij}^{'}\}$ for the data owners of identities with index set O_j:

$$T_j' = \prod_{i \in O_j} \prod_{k \in \delta_{ij}} T_{ijk}^{h_{ijk}}, S^{'} = S, F_{ij}' = \sum_{k \in \delta_{ij}} h_{ijk} \hat{F}_{ijk} \tag{6}$$

$P_j = (T_j^{'}, S^{'}, \{F_{ij}'\}_{i \in O_j})$ will be sent to TPA.

7. Verify: After receiving all the proofs $\{P_j\}$ from challenged clouds, TPA denotes $O = \cup_{j \in J} O_j$ as identity index set of all the challenged data owners from challenge tokens $\{chal_j = \{(c_{ij}, v_{ij,1}, v_{ij,2})\}_{i \in O_j}\}_{j \in J}$, and computes index set of all challenged blocks by $\{k\} = \{\pi_{v_{ij,1}}(a_{ij}) | 1 \leq a_{ij} \leq c_{ij}\}$ and co-efficient set $\{h_{ijk}\} = \{f_{v_{ij,2}}(i, j, k)\}$, as in ProofGen. With all valid set of warrant $\{\omega_i\}$ and corresponding signatures $\{(U_i, R_i, \xi_i)\}$ from data owners, together with files' names and time stamps $\{(name_i, time_{ijk})\}$, TPA is able to verify data integrity as:

$$e(\prod_{j \in J} T_j', g) = e(\prod_{i \in O} ((H_1(ID_p) \cdot (R_i^{H_2(\omega_i||R_i)} \cdot R_{pi})))^{L_i}, mpk) \cdot e(\prod_{i \in O_j} \prod_{j \in J} \prod_{k \in \delta_{ij}} (H_4(i||j||k, name_i||time_{ijk}||S^{'}))^{h_{ijk}}, S^{'}) \tag{7}$$

where $L_i = \sum_{j \in J} F_{ij}^{'} + \sum_{j \in J} \sum_{k \in \delta_{ij}} h_{ijk} \cdot H_3(i||j||k, name_i||time_{ijk})$. It will output 1 (valid) if the above equation holds and 0 (invalid) otherwise.

5.1 Security Analysis of Improved Scheme

Based on the formal definition of ID-BPAP scheme (Subsect. 2.2) and corresponding system model (Subsect. 2.3) and security model (Subsect. 2.4), in this

section, we prove security from proxy-protection of tag generation and unforgeability of proofs, in our improved scheme Sec-ID-BPAP. With data block outsourced in cipher text form by symmetric encryption, our Sec-ID-BPAP is privacy-preserving in TPA auditing. Compared with [22]s security analysis, we also utilize Corons random oracle model [24] to define the interactions between adversary of our scheme and challenger.

Theorem 1 *(Proxy-Protection). If there exists Probabilistic Polynomial Time (PPT) adversary $\mathcal{A}_1$ who could generate valid proxy tag without proxy individual private key in our Sec-ID-BPAP, then our scheme is proxy-protective when challenger $\mathcal{C}_1$ could solve CDH problem with non-negligibility within PPT time.*

Proof: There are $\hat{N}$ number of selected identities $\{ID_i\}_{i\in O}$ having the proxy ID_p. The original data block $\{\tilde{F}_{ijk}\}_{i\in O, j\in J, k\in\delta_{ij}}$ will be encrypted into corresponding ciphertext blocks $\{\hat{F}_{ijk}\}$ before being outsourced on clouds $\{CS_j\}_{j\in J}$. Certainly, the integrity of ciphertext block is equivalent to integrity of original block.

1. Setup: Simulator $\mathcal{C}_1$ plays in the role of PKG to choose random $a \in Z_q$, then the master private/public keys pair $(msk, mpk) = (a, g^a)$ upon generator $g \in G_1$. It also picks random $b \in Z_q$. CDH instance is g^a, $g^b \in G_1$, computing g^{ab}. Although $\mathcal{A}_1$ is not allowed to query the target proxy tag secret keys u_{pi}, the R_{pi} could be accessed as $H_1(ID_p)^{r_{pi}}$ by $\mathcal{C}_1$ picking up $r_{pi} \in Z_q$.
2. $\mathcal{C}_1$ answers query by maintaining input and output list for every oracle.
3. Hash function Oracle: H_2 and H_3 work as normal hash functions.
   (a) H_1-oracle: $\mathcal{C}_1$ answers with g^{y_i} for $y_i \in Z_q$ if $i \neq p$, and $y_i = b$ for $i = p$.
   (b) H_4-oracle: $\mathcal{C}_1$ answers with $g^{z_{ijk}}$ for $z_{ijk} \in Z_q$.
4. Extract-oracle: $\mathcal{C}_1$ answers $sk_i = (g^a)^{y_i}$ from H_1, if $i \neq p$; else aborts. Denote index set of identities extracting private keys as $S_1 (p \notin S_1)$.
5. ProxyKeygen-oracle: $\mathcal{C}_1$ answers $u_{p'i} = U_i \cdot (g^a)^{y_{p'} r_{p'i}}$ from H_1 and $r_{p'i} \in Z_q$, if $i \neq p$; else aborts. Denote index pair set of identities as $S_1^{'}$ $((p, i) \notin S_1^{'})$.
6. Tag-oracle: $\mathcal{C}_1$ answers $T_{ijk} = ((g^a)^{y_{p'}} \cdot u_{p'i})^{H_{3,ijk} + \hat{F}_{ijk}} \cdot S_{ijk}^{z_{ijk}}$ with $S_{ijk} \in G_1$ from H_1,H_4, if $p^{'} \neq p$; else aborts. Denote query input as set $S_1^{''}$ $((p, i, j, k, \hat{F}_{ijk}) \notin S_1^{''})$.

Forgery Output: Finally $\mathcal{A}_1$ itself outputs a valid tag $\sigma_{i^*j^*k^*} = (T_{i^*j^*k^*}, S^{'})$ for data ciphertext block $\hat{F}_{i^*j^*k^*}$ generated by proxy ID_p with warrant ω_{i^*} and its signature $(U_{i^*}, R_{i^*}, \xi_{i^*})$. $\mathcal{C}_1$ looks up lists of all oracles. It will not abort and terminate only when none of corresponding records exists, i.e., requiring $ID_{i^*} \neq ID_p$, $(p, i^*) \notin S_1^{'}$, $(p, i^*, j^*, k^*, \hat{F}_{i^*j^*k^*}) \notin S_1^{''}$. If game could proceed, $\mathcal{C}_1$ keeps on checking all hash function oracles and makes queries itself if there is no relative record in their lists. $R_{pi^*} = H_1(ID_p)^{r_{pi^*}}$ in Setup and $U_{i^*} = (g^a)^{y_{i^*} r_{i^*} H_2(\omega_{i^*} || R_{i^*})}$ for validity of warrant ω_{i^*}.

Since $\sigma_{i^*j^*k^*} = (T_{i^*j^*k^*}, S^{'})$ satisfies Eq. (5) as valid tag, we will have a solution of CDH problem after simplification with corresponding records of oracles

and properties of bilinear mapping:

$$g^{ab} = (T_{i^*j^*k^*} \cdot S^{'-z_{i^*j^*k^*}} \cdot U_{i^*}^{-H_3(i^*||j^*||k^*, name_{i^*}||time_{i^*j^*k^*}) - \hat{F}_{i^*j^*k^*}})^{\frac{1}{W}} \quad (8)$$

Where $W = (1 + r_{pi^*})(H_3(i^*||j^*||k^*, name_{i^*}||time_{i^*j^*k^*}) + \hat{F}_{i^*j^*k^*})$.

Due to the limitation of space, we will give detailed analysis for the non-negligible probability and polynomial of time in the full version.

Theorem 2 *(Unforgeability). If there exists PPT time adversary $\mathcal{A}_2$ who could forge valid proof of our Sec-ID-BPAP, then our scheme is unforgeable when challenger $\mathcal{C}_2$ could solve CDH problem with non-negligibility within PPT time.*

Proof: There are $\hat{N}$ number of selected identities $\{ID_i\}_{i\in O}$ having the proxy ID_p. The original data block $\{\tilde{F}_{ijk}\}_{i\in O, j\in J, k\in\delta_{ij}}$ will be encrypted into corresponding ciphertext blocks $\{\hat{F}_{ijk}\}$ before being outsourced on clouds $\{CS_j\}_{j\in J}$. Certainly, the integrity of ciphertext block is equivalent to integrity of original block.

1. Setup: Like Theorem 1, $\mathcal{C}_2$ in the role of PKG, generates master private/public keys pair $(msk, mpk) = (a, g^a)$ upon generator g, and CDH instance is g^a, $g^b \in G_1$, computing g^{ab}. It also allows $\mathcal{A}_2$ to access R_{pi} as $H_1(ID_p)^{r_{pi}}$ where $r_{pi} \in Z_q$.
2. H_1-oracle, H_2-oracle, H_3-oracle,H_4-oracle, Extract-oracle, ProxyKeygen-oracle, TagGen-oracle remain the same as Theorem 1.
3. First phase queries: $\mathcal{A}_2$ could access all the hash oracles. Let us denote index set $\{ID_i\}$ of private key extracting as S_2 ($p \notin S_2$), index pair set of $\{(ID_{p'}, ID_i)\}$ of proxy tag secret key query as $S_2^{'}$ ($(p,i) \notin S_2^{'}$), the tuple set of index and data for proxy tag query as $S_2^{''}$ ($(p,i,j,k,\hat{F}_{ijk}) \notin S_2^{''}$).
4. Challenge phase: $\mathcal{C}_2$ generates challenge set *chal* with ordered $\{c_{i^*j^*}\}$ to specify every cipher text block $\hat{F}_{i^*j^*k_n^*}$ on CS_{j^*} for ID_{i^*}, where $\{(p, i^*, j^*, k_n^*)|\ 1 \le n \le c_{i^*j^*}\}$, and $i^* \neq p$, $(p, i^*) \notin S_2^{'}$, $(p, i^*, j^*, k_n^*, F_{i^*j^*k_n^*}) \notin S_2^{''}$. *chal* will be sent to TPA.
5. Second phase queries: $\mathcal{A}_2$ makes queries similar to First phase queries. Denote index set of identities for Extract private key queries as S_3, index set of identity pairs for proxy tag secret key queries as $S_3^{'}$, tuple set of index and data for proxy taga queries as $S_3^{''}$. We requires that $p \notin S_2 \cup S_3$, $(p,i) \notin S_2^{'} \cup S_3^{'}$ and $(p,i,j,k,\hat{F}_{ijk}) \notin S_2^{''} \cup S_3^{''}$.

Forgery Output: Finally, $\mathcal{A}_2$ itself outputs valid proof $\{P_{j^*}\}_{j^*\in J}$ for data cipher text blocks $\{\hat{F}_{i^*j^*k_n^*}\}_{1\le n\le c_{i^*j^*}}$ and tags generated by proxy ID_p with warrants $\{\omega_{i^*}\}_{i^*\in O}$ and signatures $\{(U_{i^*}, R_{i^*}, \xi_{i^*})\}_{i^*\in O}$. $\mathcal{C}_2$ looks up lists of all oracles and it will abort and terminate unless none of corresponding records exists. If game could proceed, $\mathcal{C}_2$ keeps on checking all hash function oracles and makes queries itself if there is no relative record in their lists. $R_{pi^*} = H_1(ID_p)^{r_{pi^*}}$ in Setup and $U_{i^*} = (g^a)^{y_{i^*}r_{i^*}H_2(\omega_{i^*}||R_{i^*})}$ for validity of warrant ω_{i^*}.

Since $\{P_{j^*}\}_{j^* \in J} = \{(T'_{j^*}, S', \{F'_{i^* j^*}\}_{i^* \in O_j})\}_{j^* \in J}$ satisfies Eq. (7), the CDH problem solution is obtained after simplification with corresponding records of oracles and properties of bilinear mapping:

$$g^{ab} = (W'_1 \cdot W'_2)^{\frac{1}{\sum_{i^* \in O}(1 + r_{pi^*})E_{i^*}}} \tag{9}$$

where $E_{i^*} = \sum_{j^* \in J} F'_{i^* j^*} + \sum_{j^* \in J} \sum_{n \in [1, c_{i^* j^*}]} h_{i^* j^* k_n^*} \cdot H_3(i^* || j^* || k_n^*, name_{i^*} || time_{i^* j^* k_n^*})$

$$W'_1 = \prod_{j^* \in J} T'_j \cdot S'^{-\sum_{i^* \in O} \sum_{j^* \in J} \sum_{n \in [1, c_{i^* j^*}]} z_{i^* j^* k_n^*} \cdot h_{i^* j^* k_n^*}}$$

$$W'_2 = \prod_{i^* \in O} U_{i^*}^{-\sum_{j^* \in J}(F'_{i^* j^*} + \sum_{n \in [1, c_{i^* j^*}]} h_{i^* j^* k_n^*} \cdot H_3(i^* || j^* || k_n^*, name_{i^*} || time_{i^* j^* k_n^*}))}$$

Due to the limitation of space, we will give detailed analysis for the non-negligible probability and polynomial of time in the full version.

6 Efficiency Analysis

In this section, we compare cost of computation and communication of our improved scheme Sec-ID-BPAP, with Wang et al. 's ID-PUIC [22], summarized in Tables 1, and 2, respectively.

Table 1. Computation cost comparison for multiple owners and multiple clouds

Schemes	TagGen	ProofGen	Verify	Security
ID-PUIC[22]	$2NC_{exp}$	cC_{exp}	$2n_1n_2C_e + (c + n_1n_2)C_{exp}$	Secure
Sec-ID-BPAP	$(2N + n_O)C_{exp}$	cC_{exp}	$3C_e + (c + n_1)C_{exp}$	Secure

Table 2. Communication cost comparison for multiple owners and multiple clouds

Schemes	Challenge	Proof	Security
ID-PUIC[22]	$n_1n_2 \log_2 N + 2n_1n_2 \log_2 q$	$n_1n_2\mathcal{G}_1 + n_1n_2 \log_2 q$	Secure
Sec-ID-BPAP	$n_1n_2 \log_2 N + 2n_1n_2 \log_2 q$	$2n_2\mathcal{G}_1 + n_1n_2 \log_2 q$	Secure

1. Assume there are n_O data owners storing N blocks $\{\hat{F}_{ijk}\}$ on n_J clouds, by only *one-off* TagGen and upload. To audit data integrity, *periodical* Challenge and Verify will be executed between clouds and TPA, upon randomly selected c data blocks and tags of n_1 data owners on n_2 clouds, element size of group G_1 is $\mathcal{G}_1$. The dominant cost of this scheme is mostly contributed by ProofGen and Verify.

2. 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, operations 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-PUIC only offers single owner's auditing on one cloud, we consider repeating n_1n_2 loops of ID-PUIC instances, with N/n_1n_2 outsourced blocks and only challenged c/n_1n_2 blocks per loop.

Analysis for Computation: In order to fully protect tags $\{\sigma_{ijk} = (S_{ijk}, T_{ijk})\}$ from being 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 flaw, i.e., private key recovery of ID-BPAP, we need 3 bilinear pairing computation to allow batch auditing at one time, which thus achieves enhanced security and still outperforms $2n_1n_2$ pairings in Wang et al.'s ID-PUIC [22], if applied to the multiple clouds and multiple owners scenario.

Analysis for Communication: Communication for Challenge remains the same as ID-BPAP [25]. The total overhead of transmission is still smaller than Wang et al.s ID-PUIC if applied to the multiple clouds and multiple owners setting. Meanwhile, our improved scheme does not suffer from private key recovery as ID-BPAP.

Above all, the enhanced efficiency will become demonstrative if applied to huge data storage utilities like big data analysis. We will provide the analysis in detail upon extensive simulation in the full version of paper.

7 Conclusions

In this paper, we revisited an identity-based batch public auditing scheme with proxy processing (ID-BPAP) scheme designed by Yu et al. in [25], and demonstrated that any cloud server could generate valid data integrity proof without original data. Meanwhile, it is also feasible to recover Proxys private key and generate valid proxy tags for any modified data without Proxys awareness. Therefore, we propose our solution to repair the security flaws and thus enhance the security, at the expense of reasonable overheads while still enjoying better efficiency over Wang et al.'s scheme [22]. As a future work, we will keep on seeking to improve the efficiency of our proposed scheme of enhanced security and privacy, and evaluate it based on real-world multiple clouds storage system, with sound security for data integrity.

Acknowledgments. This work was supported by the National Key R&D Program of China under Grant 2017YFB0802000 and the National Natural Science Foundation of China under Grant 61872060 and 61370203, State Scholarship Fund Program of China Scholarship Council under Grant 201506070077. We also appreciate valuable comments from anonymous reviewers, which helps to contribute to a paper of good quality.

References

1. IDC.com: Worldwide Public Cloud Services Spending Forecast to Reach $122.5 Billion in 2017, According to IDC. http://www.idc.com/getdoc.jsp?containerId=prUS42321417. Accessed 20 Feb 2017
2. Gartner.com: Gartner Forecasts Worldwide Public Cloud Services Revenue to Reach $260 Billion in 2017. https://www.gartner.com/newsroom/id/3815165. Accessed 12 Oct 2017
3. Fu, Z., Wu, X., Guan, C., Sun, X., Ren, K.: Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement. IEEE Trans. Inf. Forensics Secur. **11**(12), 2706–2716 (2016)
4. Ateniese, G., et al.: Provable data possession at untrusted stores. In: Proceedings of ACM CCS 2007, pp. 598–609 (2007)
5. Shacham, H., Waters, B.: Compact proofs of retrievability. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 90–107. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89255-7_7
6. Wang, C., Chow, S.S.M., Wang, Q., Ren, K., Lou, W.: Privacy-preserving public auditing for secure cloud storage. IEEE Trans. Comput. **62**(2), 362–375 (2013)
7. Wang, Q., Wang, C., Ren, K., Lou, W., Li, J.: Enabling public audibility and data dy-namics for storage security in cloud computing. IEEE Trans. Parallel Distrib. Syst. **22**, 847–859 (2011)
8. Erway, C.C., Kp, A., Papamanthou, C., Tamassia, R.: Dynamic provable data possession. ACM Trans. Inf. Syst. Secur. **17**(4), 15 (2015)
9. Zhu, Y., Hu, H., Ahn, G.J., Yu, M.: Cooperative provable data possession for integrity verification in MultiCloud storage. IEEE Trans. Parallel Distrib. Syst. **23**(12), 2231–2244 (2012)
10. Yang, K., Jia, X.: An efficient and secure dynamic auditing protocol for data storage in cloud computing. IEEE Trans. Parallel Distrib. Syst. **24**(9), 1717–1726 (2013)
11. Curtmola, R., Khan, O., Burns, R., Ateniese, G.:MR-PDP: multiple-replica provable data possession. In: Proceedings of the 28th International Conference on Distributed Computing Systems, pp. 411–420 (2008)
12. Wang, B., Li, B., Li, H.: Panda: public auditing for shared data with efficient user revocation in the cloud. IEEE Trans. Serv. Comput. **8**(1), 92–106 (2015)
13. Liu, C., Ranjan, R., Yang, C., Zhang, X., Wang, L., Chen, J.: MuRDPA: top-down levelled multi-replica merkle hash tree based secure public auditing for dynamic big data storage on cloud. IEEE Trans. Comput. **64**(9), 2609–2622 (2015)
14. Barsoum, A.F., Hasan, M.A.: Provable multicopy dynamic data possession in cloud computing systems. IEEE Trans. Inf. Forensics Secur. **10**(3), 485–497 (2015)
15. Wang, J., Chen, X., Huang, X., You, I., Xiang, Y.: Verifiable auditing for outsourced database in cloud computing. IEEE Trans. Comput. **64**(11), 3293–3303 (2015)
16. Zhao, J., Xu, C., Li, F., Zhang, W.: Identity-based public verification with privacy preserving for data storage security in cloud computing. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **96**(12), 2709–2716 (2013)
17. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_13
18. Yu, Y., Zhang, Y., Mu, Y., Susilo, W., Liu, H.: Provably secure identity based provable data possession. In: Au, M.-H., Miyaji, A. (eds.) ProvSec 2015. LNCS, vol. 9451, pp. 310–325. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-26059-4_17

19. Liu, H., Mu, Y., Zhao, J., Xu, C., Wang, H., Chen, L.: Identity-based provable data possession revisited: security analysis and generic construction. Comput. Stand. Interfaces **54**(1), 10–19 (2017)
20. Wang, H.: Identity-based distributed provable data possession in multicloud storage. IEEE Trans. Serv. Comput. **8**(2), 328–340 (2015)
21. Yu, Y.: Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage. IEEE Trans. Inf. Forensics Secur. **12**(4), 767–778 (2017)
22. Wang, H., He, D., Tang, S.: Identity-based proxy-oriented data uploading and remote data integrity checking in public cloud. IEEE Trans. Inf. Forensics Secur. **11**(6), 1165–1176 (2016)
23. Peng, S., Zhou, F., Xu, J., Xu, Z.: Comments on identity-based distributed provable data possession in Multicloud storage. IEEE Trans. Serv. Comput. **9**(6), 996–998 (2016)
24. Coron, J.-S.: On the exact security of full domain hash. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 229–235. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44598-6_14
25. Yu, H., Cai, Y., Kong, S.: Efficient and secure identity-based public auditing for dynamic outsourced data with proxy. KSII Trans. Internet and Inf. Syst. **11**(10), 5039–5061 (2017)