7-2015

A new public remote integrity checking scheme with user privacy

Yiteng FENG

Yi MU

Guomin YANG
Singapore Management University, gmyang@smu.edu.sg

Joseph LIU

A New Public Remote Integrity Checking Scheme with User Privacy

Yiteng Feng[1], Yi Mu[1], Guomin Yang[1(✉)], and Joseph K. Liu[2]

[1] Centre for Computer and Information Security Research,
School of Computing and Information Technology,
University of Wollongong, Wollongong, Australia
yf579@uowmail.edu.au, {ymu,gyang}@uow.edu.au
[2] Faculty of Information Technology, Monash University, Melbourne, Australia
joseph.liu@monash.edu

Abstract. With a cloud storage, users can store their data files on a remote cloud server with a high quality on-demand cloud service and are able to share their data with other users. Since cloud servers are not usually regarded as fully trusted and the cloud data can be shared amongst users, the integrity checking of the remote files has become an important issue. A number of remote data integrity checking protocols have been proposed in the literature to allow public auditing of cloud data by a third party auditor (TPA). However, user privacy is not taken into account in most of the existing protocols. We believe that preserving the anonymity (i.e., identity privacy) of the data owner is also very importa nt in many applications. In this paper, we propose a new remote integrity checking scheme which allows the cloud server to protect the identity information of the data owner against the TPA. We also define a formal security model to capture the requirement of user anonymity, and prove the anonymity as well as the soundness of the proposed scheme.

Keywords: Data integrity · Identity privacy · Public auditing · Cloud storage

1 Introduction

Cloud computing offers various kinds of computation and storage services to end users via computer networks, and is becoming very popular nowadays. One of the major services is cloud storage which allows users to store their data files remotely and access the files anywhere any time. It has greatly reduced the burden for local storage management and maintenance. Some commerical products such as Google Drive and Dropbox have become very popular for both individuals and enterprises.

However, after the data owners outsource their files to the cloud server and delete the local copies, the server may not store the files correctly due to various

J.K. Liu—Joseph K. Liu is supported by National Natural Science Foundation of China (61472083).

E. Foo and D. Stebila (Eds.): ACISP 2015, LNCS 9144, pp. 377–394, 2015.
DOI: 10.1007/978-3-319-19962-7_22

reasons such as system crash or deliberately deleting some data blocks that are seldom or never used in order to save the cost. Moreover, the server may try to hide the accidents or misbehaviours to the end users. In order to detect or prevent this kind of situations, we need to audit the integrity of remote files either periodically or before downloading. Many protocols have been proposed for auditing the file integrity and retrievability in remote storage, such as Proof of Data Possession (PDP) [1] and Proof of Retrievability (PoR) [6,7]. These protocols utilised spot-checking techniques such as homomorphic authenticators to make the auditing more efficient.

Recently, some additional security and usability properties have been proposed and formalised for Remote Integrity Checking (RIC) protocols [4,11,12], including public auditing (i.e., the data auditing can be performed by a third-party auditor), batch auditing, and privacy-preserving auditing (i.e., the third-party auditor cannot learn any information about the file during the auditing process). These properties are achieved by extending the previous work on homomorphic authenticators and zero-knowledge proof systems.

Since users may frequently modify the files by inserting, updating and deleting the file blocks, data dynamic operations have also been considered in some recent RIC protocols. In [13,14], a Merkle Hash Tree (MHT) is employed to achieve this goal where the structure of a file is represented using a MHT. Later, a more efficient scheme was proposed by Yang and Jia in [15] where an index table is maintained by a *fixed* TPA in order to keep track of the data changes. Although Yang and Jia's scheme is more efficient than the MHT approach proposed in [14], the TPA must be fixed for a specific file, which can be considered as a limitation of the scheme. Some RIC protocols [9] have also considered file sharing among a group of users, where the technique of proxy-resign is used to allow authenticators to be transferred from a leaving member to another staying member in the group.

In this paper, we focus on another important and desirable property of RIC protocols: identity privacy (i.e., anonymity) of the data owner against outsiders and the TPA. Such a problem has been studied in [10] where a ring signature (i.e., authenticator) rather than an ordinary one is generated for each data block by the data owner. Nevertheless, there is an issue in this protocol: in order to allow data dynamic operations, a virtual index is used for each file block. Since the cloud server maintains the virtual index table, it can always redirect the challenge for a corrupted file block to an uncorrupted one in order to pass the verification. Details of this issue will be shown in Appendix A. A similar problem has also been considered in [8] where a group signature is used as the authenticator for each block. However, the group manager is still able to reveal the identity of signer and it is not easy to set up that kind of group.

Identity privacy is essential in many scenarios. For example, a group of users may prefer anonymous file sharing for some reasons such as fear of retribution for disclosure. Besides, censorship is usually required in local, organizational or national level applications such as anonymous bidding. A group of bidders will place their bids which are actually files on the cloud storage and the identity of

bidders should be confidential during assessment. At the same time, the assessment community would like to ensure the integrity of files periodically or before they retrieve a file.

In this paper, we propose a new RIC protocol with user anonymity. Differing from the protocol in [10], we consider the user anonymity in file level instead of data block level in order to save the storage and communication overheads. In the current work, we only focus on static files and leave the support of data dynamic operations as the future work.

Our Contribution. We formalise the notion of user anonymity for RIC protocols. In our RIC anonymity model, we consider the TPA as the adversary who aims to discover the identity of the file owner during the RIC proof with the cloud server. Then we construct an identity privacy-preserving RIC protocol based on the ring signature scheme proposed by Boneh et al. in [2], which is a variant of BLS signature [3]. The protocol is very practical because of the homomorphic property of the underlying signature scheme. Moreover, the protocol also supports batch verification such that the TPA can verify several proofs simultaneously using one verification equation. We prove that the proposed protocol can ensure data integrity while preserving identity privacy. In addition, this protocol can be also extended to support the privacy of audited files (which is defined as IND-Privacy in [4]).

Paper Organisation. The rest of the paper is organised as follows. In Section 2, we give the system model and security definition of the protocol. In Section 3, we present our identity privacy preserving remote integrity check protocol. We analyse the security of the proposed scheme in Section 4. We provide some extensions in Section 5. We conclude the paper in Section 6. The details of an attack on an existing scheme and the security proofs of the proposed scheme are given in Appendices.

2 System Model and Security Definition

Fig. 1 is an overview of the system. The basic case is that d users stored d files on the cloud server, and each of them owns one file. None of the users wants the TPA to link their identity with the file. The system can be easily extended to a more general setting where one user stores multiple files on the cloud server.

Users first form a group to preserve anonymity from the TPA. Then they outsource their files to the cloud individually. From the file level point of view, a file has one file tag, one glue value, some auxiliary information. From the block level point of view, one authenticator is attached to each block. During the auditing process, the file tags are public to the TPA but other information is kept secret.

After that, this group of users delegate the auditing task for the outsourced files to any TPA. Either periodically or at specific times, the TPA will challenge the cloud server for the storage of the outsourced files. The cloud server will then generate a short response for the challenge based on the information it has.

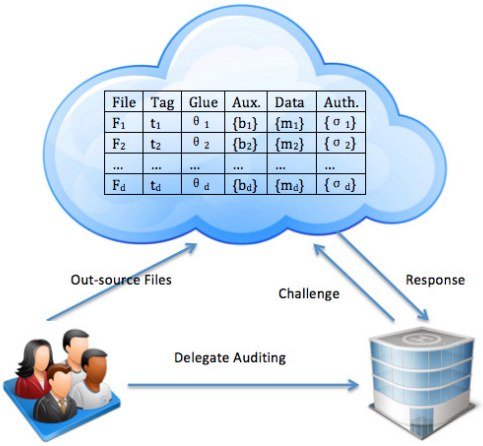

Fig. 1. System Model

Now we are going to describe the formal definition of an RIC protocol as well as the soundness and anonymity game.

2.1 Anonymous RIC Protocols

There are three types of parties in an anonymous RIC protocol: a group of users, the cloud server, and the TPA. An anonymous RIC scheme consists of the following algorithms:

- **Setup:** This algorithm takes a security parameter λ and generates the public parameters mpk for the whole system.
- **KeyGen:** This algorithm takes as input mpk and generates the public and private key pair (pk_i, sk_i) for each user.
- **TokenGen:** This algorithm takes as input a file F, and the owner's private key sk, to generate a file tag t, and the authenticator σ for each file block.
- **GlueGen:** This algorithm takes as input the public keys of all group users $\{pk_i\}$ and the owner's private key sk to generate a glue value θ and auxiliary information $\{b\}$ for anonymous public auditing.
- **Challenge:** Given the file tag t, this algorithm generates the challenge $chal$ for cloud server.
- **Response:** Taking as input $(F, t, \{\sigma\}, \theta, \{b\}, chal, mpk)$, this algorithm outputs a proof P to prove the integrity of the file.
- **Verify:** Taking as input $(\{pk_i\}, mpk, t, chal, P)$, this algorithm outputs true or false to indicate whether the file is stored correctly.

2.2 RIC Anonymity

Since we want to ensure that the TPA cannot distinguish the identity of the real file owner from other users in the group during multiple runs of the protocol,

we fix the owner at the beginning of the adversarial game, and then allow the adversary to ask multiple proof queries. The Anonymity for an RIC scheme is defined via an *Anonymity* game between a challenger $\mathcal{C}$ (i.e. the cloud server or prover) and an Adversary $\mathcal{A}$ (i.e. the TPA or verifier) as follows.

- **Setup:** The challenger $\mathcal{C}$ runs **Setup** and **KeyGen** to generate the public parameters and the key pairs for a group of users $U_1, U_2, ..., U_d$ and passes all the public keys to the Adversary $\mathcal{A}$. Also, $\mathcal{C}$ randomly chooses a user $b \in [1, d]$ to be the real signer.
- **Phase 1:** $\mathcal{A}$ can ask for tag queries. In each query, $\mathcal{A}$ can choose a file F to be stored and $\mathcal{C}$ generates the file tag, the authenticators, the glue value and the auxiliary information for the file. Only the file tag t is returned to $\mathcal{A}$. $\mathcal{A}$ can ask multiple tag queries for different files.
- **Phase 2:** $\mathcal{A}$ uses the file tag t of file F which appeared in **Phase 1** to generates the *chal* to ask for proof from $\mathcal{C}$. Then, $\mathcal{C}$ generates the proof P and sends P to $\mathcal{A}$. In this phase, $\mathcal{A}$ can also ask multiple proof queries.
- **Decision:** $\mathcal{A}$ output b' as the guess of b.

Define the advantage of adversary $\mathcal{A}$ as $Adv_{\mathcal{A}}(\lambda) = |\Pr[b' = b] - 1/d|$.

Definition 1. *We say an RIC scheme has anonymity if for any polynomial-time algorithm $\mathcal{A}$, $Adv_{\mathcal{A}}(\lambda)$ is a negligible function of the security parameter λ.*

2.3 RIC Soundness

We say a protocol is sound if it is infeasible for the cloud server to modify the content of a file without being detected by the TPA in the auditing. We define the soundness games between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ as follows:

- **Setup:** $\mathcal{C}$ runs **Setup** and **KeyGen** to generate the key pairs for a group of users. $\mathcal{C}$ then chooses the first user as the real signer and passes all the public keys to the Adversary.
- **Phase 1:** $\mathcal{A}$ makes multiple queries to $\mathcal{C}$ for **TokenGen** and **GlueGen** on any file F. $\mathcal{C}$ generates the file tag t, authenticators σ, and the glue value θ and auxiliary information $\{b\}$ using the secret key of the first user, and then returns $(t, \{\sigma\}, \theta, \{b\})$ to $\mathcal{A}$.
- **Phase 2:** $\mathcal{A}$ outputs a file F' and a file tag t' such that $t' = t$ but $F' \neq F$ for some (F, t) appeared in Phase 1 (i.e., at least one block of F is corrupted). $\mathcal{C}$ then plays the role as the verifier and executes the RIC protocol with $\mathcal{A}$ by sending a challenge *chal* which contains at least one corrupted block.
- **Decision:** $\mathcal{C}$ makes a decision which is either True or False after verifying the proof P' generated by the adversary $\mathcal{A}$.

Definition 2. *We say a RIC protocol is ε-sound if* $\Pr[\mathcal{C}$ *outputs True*$] \leq \varepsilon$.

3 The Proposed Scheme

3.1 Notation

Let G_1, G_2 and G_T be three multiplicative cyclic groups of prime order p. Let g_1 and g_2 be the generators of G_1 and G_2 respectively. The bilinear map $e : G_1 \times G_2 \rightarrow G_T$ should have the following properties:

- Bilinear: We say that the map is bilinear if $e(u^a, h^b) = e(u, h)^{ab}$ for all $u \in G_1, h \in G_2, a, b \in Z_p$.
- Non-degenerate: The map does not send all pairs in $G_1 \times G_2$ to the identity in G_T. As they are prime order groups, $e(g_1, g_2)$ is a generator of G_T.
- Computable: There is an efficient algorithm to compute $e(u, h)$ for any $u \in G_1, h \in G_2$.
- Isomorphism: In addition, there should be a computable isomorphism ψ from G_2 to G_1, with $\psi(g_2) = g_1$ and $e(\psi(u), h) = e(\psi(h), u)$ for any $u, h \in G_2$.

3.2 Our Anonymous RIC Scheme

We present the details of our anonymous RIC scheme.

Setup: Let $(e, p, G_1, G_2, G_T, g_1, g_2, \psi)$ be the bilinear map and associated group parameters defined as above. Choose a cryptographic hash functions $H : \{0,1\}^* \rightarrow G_1$. Choose another generator $h \in G_1$. Choose two random numbers $\alpha, \tau \in Z_p$ to generate the commitment keys: $\boldsymbol{u} = (u_1, u_2)$ where $u_1 = (g_1, g_1^{\alpha}), u_2 = (g_1^{\tau}, g_1^{\tau\alpha})$. The public parameters are $mpk = (e, p, G_1, G_2, G_T, g_1, g_2, \psi, H, h, u_1, u_2)$.

KeyGen: Each user U_i randomly chooses $x_i \in Z_p$ and computes $y_i = g_2^{x_i}$. Besides, each user generates a key pair (spk_i, ssk_i) for a secure ring signature scheme. The user U_i's public key is (y_i, spk_i), and the private key is (x_i, ssk_i).

TokenGen: The outsourced file F is first divided into n blocks, and the file owner s chooses a random file name $name$ from some sufficiently large domain (e.g., Z_p). Let t_0 be "$name||n$"; the file tag t is t_0 together with a ring signature on t_0 under the private key ssk_s and the public keys $\{spk_i\}_{i \neq s}$: $t \leftarrow t_0 || Sig_{ssk_s}(t_0)$. For each block m_j, the signer computes: $\sigma_j = \left(H\left(name||j\right) \cdot h^{m_j}\right)^{1/x_s}$.

GlueGen: For each user $U_i \neq U_s$, the signer chooses $a_i \in Z_p$, and computes the auxiliary information: $b_i = g_1^{a_i}$. Then the signer computes a glue value $\theta = 1 \Big/ \psi\left(\prod_{i \neq s}^{n} y_i^{a_i}\right)^{1/x_s}$. Finally, the user sends $\{F, t, \{\sigma_j\}_{j \in [1,n]}, \{b_i\}_{i \in [1,d]}, \theta\}$ to the cloud server. We shows the computation and communication between client and server in Fig. 2.

Challenge: The TPA first retrieves the file tag t and verifies the ring signature $Sig_{ssk_s}(t_0)$ based on the user public keys of all the users in the ring. The TPA quits by emitting False if the verification fails. Otherwise, the TPA recovers

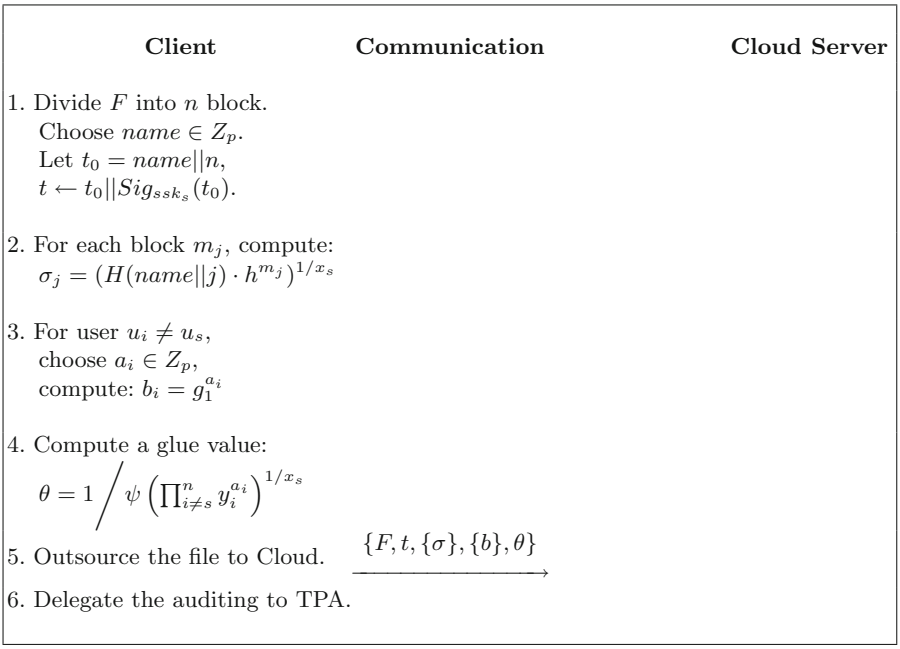

Fig. 2. Storing a file

$name, n$. Then the TPA picks c elements in set $[1, n]$, where n is the total number of file blocks, to indicate the blocks that will be checked. J denotes the subset of $[1, n]$ that contains all the chosen indices. For $j \in J$, the TPA randomly chooses $c_j \in Z_p$. Finally the TPA sends $\{(j, c_j)\}_{j \in J}$ to the cloud server as a challenge.

Response: Upon receiving the challenge $\{(j, c_j)\}_{j \in J}$, the cloud server computes $\mu = \sum_{j \in J} c_j m_j$, and $\Sigma_s = \prod_{j \in J} \sigma_j^{c_j} \cdot \theta$, $\Sigma_i = b_i$ for all $i \neq s$. Given $(\mu, \{\Sigma\})$, the TPA can use the following equation to check the integrity of the file $e\left(\prod_{j \in J} H(name||j)^{c_j} \cdot h^{\mu}, g_2\right) = \prod_{i=1}^{d} e\left(\Sigma_i, y_i\right)$.

However, if we run the above protocol for multiple times, the values of μ and Σ_s will change depending on the challenge, while other Σ_i will remain static, which will reveal the identity of the file owner. To address this issue, we make use of the Groth-Sahai Proof System [5] to hide all the $\{\Sigma\}$ while convincing the TPA they are correct. So in the modified **Response** algorithm, using the system public key $\boldsymbol{u}$ as the commitment key, the cloud server computes commitments as: $\boldsymbol{c} = (c_1, c_2, ..., c_d)$, where

$$
\begin{aligned}
c_1 &= (c_{11}, c_{12}) = (g_1^{r_{11}+r_{12}\tau}, g_1^{\alpha(r_{11}+r_{12}\tau)}\Sigma_1) = (u_{11}^{r_{11}} u_{12}^{r_{12}}, u_{21}^{r_{11}} u_{22}^{r_{12}} \Sigma_1) \\
c_2 &= (c_{21}, c_{22}) = (g_1^{r_{21}+r_{22}\tau}, g_1^{\alpha(r_{21}+r_{22}\tau)}\Sigma_2) = (u_{11}^{r_{21}} u_{12}^{r_{22}}, u_{21}^{r_{21}} u_{22}^{r_{22}} \Sigma_2) \\
&... \\
c_d &= (c_{d1}, c_{d2}) = (g_1^{r_{d1}+r_{d2}\tau}, g_1^{\alpha(r_{d1}+r_{d2}\tau)}\Sigma_d) = (u_{11}^{r_{d1}} u_{12}^{r_{d2}}, u_{21}^{r_{d1}} u_{22}^{r_{d2}} \Sigma_d)
\end{aligned}
$$

where $r_{ij} (i \in [1, d], j \in \{1, 2\})$ is randomly selected from Z_p.

The proof becomes: $\boldsymbol{\pi} = (\pi_1, \pi_2)$ where $\pi_1 = (1, y_1^{r_{11}} y_2^{r_{21}} ... y_d^{r_{d1}})$ and $\pi_2 = (1, y_1^{r_{12}} y_2^{r_{22}} ... y_d^{r_{d2}})$. Finally, $(\mu, \boldsymbol{c}, \boldsymbol{\pi})$ are sent to TPA instead of $(\mu, \{\Sigma\})$.

Verify: First, we define the $\star$ operator as $\boldsymbol{x} \star \boldsymbol{y} = F(x_1, y_1)F(x_2, y_2)...F(x_n, y_n)$, where

$$F(x_i, y_i) = \begin{pmatrix} e(x_{i1}, y_{i1}) & e(x_{i1}, y_{i2}) \\ e(x_{i2}, y_{i1}) & e(x_{i2}, y_{i2}) \end{pmatrix}$$

for $(x_{i1}, x_{i2}) \in G_1^2, (y_{i1}, y_{i2}) \in G_2^2, i \in [1, n]$.

To do the verification, TPA first performs the transformation on the original verification equation as follows.

- Transform Left Hand Side: $\boldsymbol{l} = (l_1, l_2)$ where $l_{11} = (1, 1)$ and $l_{12} = (1, L)$, where $L = e\left(\prod_{j \in J} H(name||j)^{c_j} \cdot h^{\mu}, g_2\right)$.
- Transform Public Keys: $\boldsymbol{k} = (k_1, k_2, ..., k_d)$ where $k_1 = (1, y_1)$ and $k_2 = (1, y_2)$ ... $k_d = (1, y_d)$

The verifier then performs the verification via the following equation: $\boldsymbol{l}(\boldsymbol{u} \star \boldsymbol{\pi}) = \boldsymbol{c} \star \boldsymbol{k}$. Please note that here we use the Hadamard product of the matrix. We show the computations and message flows between TPA and cloud server in Fig. 3. The correctness of both the non-anonymous and anonymous verification equations are proved in Appendix B.

4 Security Analysis

4.1 Anonymity of the Protocol

External Diffie-Hellman (XDH) Assumption: Groth-Sahai proof system in [5] is under the assumption SXDH, where the DDH problem is hard in both G_1 and G_2. But we need to have isomorphism to generate the glue value, so we will just assume DDH is hard in G_1. Let $gk = (\lambda, p, G_1, G_2, G_T, e, g_1, g_2)$, define a bilinear map $e : G_1 \times G_2 \rightarrow G_T$ and isomorphism from G_2 to G_1, with $\psi(g_2) = g_1$. The XDH assumption holds if we have

$$|\Pr[A(gk, \psi, g_1^x, g_1^y, g_1^{xy}) = 1] - \Pr[A(gk, \psi, g_1^x, g_1^y, g_1^z) = 1]| \leq \epsilon$$

where $x, y, z \leftarrow Z_p^*$.

Theorem 1. *Our new anonymous RIC protocol has identity privacy if XDH assumption holds.*

4.2 Soundness of the Protocol

Co-CDH Problem: Given $(G_1, G_2, g_1, g_2, h = g_1^a, u = g_2^b)$, Compute g_1^{ab}. Similar to [2], we can solve the Co-CDH problem by solving two instances of the following problem: given $(G_1, G_2, g_1, g_2, u = g_2^a, h = g_1^{ab})$, compute g_1^b. We will use this version of the Co-CDH problem to prove the soundness of our protocol.

Theorem 2. *The proposed RIC scheme is negl(λ)-sound, where negl(λ) is a negligible function of the security parameter λ, if the Co-CDH problem is hard.*

The proofs of the theorems are presented in Appendix C and D.

TPA	**Communication**	**Cloud Server**
1. Retrieve the file tag t. Verify the signature, and recover $name, n$.		
2. Pick c elements in $[1, n]$ to form a subset J. For each $j \in J$, choose $c_j \in Z_p$.		
3. Send Challenge to Cloud.	$\{(j, c_j)\}_{j \in J} \longrightarrow$	
		1. Compute: $\mu = \sum_{j \in J} c_j m_j$
		2. Compute: $\Sigma_s = \prod_{j \in J} \sigma_j^{c_j} \cdot \theta$ and $\Sigma_i = b_i$ for all $i \neq s$.
		3. Commit $\{\Sigma\}$ to $\boldsymbol{c}$: $c_i = (c_{i1}, c_{i2}) = (u_{11}^{r_{i1}} u_{12}^{r_{i2}}, u_{21}^{r_{i1}} u_{22}^{r_{i2}} \Sigma_i)$ where $r_{ij} (i \in [1, d], j \in \{1, 2\})$ is randomly selected from Z_p.
		4. Compute the Proof: $\boldsymbol{\pi} = (\pi_1, \pi_2)$. $\pi_1 = (1, y_1^{r_{11}} y_2^{r_{21}} ... y_d^{r_{d1}})$ $\pi_2 = (1, y_1^{r_{12}} y_2^{r_{22}} ... y_d^{r_{d2}})$
	$\{\mu, \boldsymbol{c}, \boldsymbol{\pi}\} \longleftarrow$	5. Send Response to TPA.
4. Transform left hand side: $\boldsymbol{l} = (l_1, l_2)$. $l_{11} = (1, 1)$, $l_{12} = (1, L)$, while $L = e\left(\prod_{j \in J} H(name \| j)^{c_j} \cdot h^{\mu}, g_2\right)$.		
5. Transform Public Keys: $\boldsymbol{k} = (k_1, k_2, ..., k_d)$, where $k_i = (1, y_i)$.		
6. Verify the proof with commitment key $\boldsymbol{u}$: $\boldsymbol{l}(\boldsymbol{u} \star \boldsymbol{\pi}) = \boldsymbol{c} \star \boldsymbol{k}$.		

Fig. 3. Auditing

5 Extension

5.1 Batch Auditing

Our scheme can be easily extended to support batch auditing, which can audit multiple files for the group at the same time and save the pairing computations. For example, for non-anonymous auditing, given two challenges for different files $\{(j, c_j)\}_{j\in J}$ and $\{(j', c'_j)\}_{j'\in J'}$, the cloud server can calculate $(\mu, \{\Sigma\}$ and $(\mu', \{\Sigma'\})$, and verification equation becomes

$$e\left(\left(\prod_{j\in J} H(name||j)^{c_j}\right)\cdot\left(\prod_{j'\in J'} H(name'||j')^{c'_j}\right)\cdot h^{\mu+\mu'}, g_2\right) = \prod_{i=1}^{d} e\left(\Sigma_i\cdot\Sigma'_i, y_i\right).$$

For anonymous auditing, the equation for the verification is still

$$\boldsymbol{l}(\boldsymbol{u}\star\boldsymbol{\pi}) = \boldsymbol{c}\star\boldsymbol{k}\ .$$

As a result, the response from the cloud server still contains three parts $(\mu+\mu', \boldsymbol{c}, \boldsymbol{\pi})$ where the commitment $\boldsymbol{c}$ will hide $\Sigma_i\cdot\Sigma'_i$ instead of Σ_i.

It is also worth noting that during TPA verification, L inside $\boldsymbol{l}$ will be changed to

$$\left(\prod_{j\in J} H(name||j)^{c_j}\right)\cdot\left(\prod_{j'\in J'} H(name'||j')^{c'_j}\right)\cdot h^{\mu+\mu'}\ .$$

5.2 IND-Privacy

We have addressed the identity privacy of users, and we can also extend the scheme to provide the privacy of the audited files, which is defined as IND-Privacy in [4]. With IND-Privacy, the TPA is not able to distinguish which file is being audited. The Groth-Sahai proof system [5] is also used in [4] to achieve this goal.

The non-anonymous verification equation of our RIC scheme

$$e\left(\prod_{j\in J} H(name||j)^{c_j}\cdot h^{\mu}, g_2\right) = \prod_{i=1}^{d} e\left(\Sigma_i, y_i\right)$$

can be expressed as

$$e\left(\prod_{j\in J} H(name||j)^{c_j}, g_2\right) = \prod_{i=1}^{d} e\left(\Sigma_i, y_i\right)\cdot e(h^{\mu}, g_2^{-1})\ .$$

Similar to [4], we can hide h^{μ} as well as all the $\{\Sigma\}$ in the commitment to achieve both Anonymity and IND-Privacy. The equation is still in the form of

$$\boldsymbol{l}(\boldsymbol{u}\star\boldsymbol{\pi}) = \boldsymbol{c}\star\boldsymbol{k}\ ,$$

where $\boldsymbol{c}$ and $\boldsymbol{k}$ will have extra components, and $\boldsymbol{l}, \boldsymbol{\pi}$ need to be changed accordingly.

5.3 Data Dynamics

There is no ideal solution for supporting data dynamics at the moment. However, if the TPA is fixed, then we can make the TPA keep the index tables for the files to keep track of all the data dynamic operations as [15] did. The cloud users have to communicate with TPA after they update the files on cloud server.

Another solution for achieving data dynamic operations is to replace the index information i of a data block m in the computation of the authenticator, and use the Merkle Hash Tree (MHT) for the block sequence enforcement. This approach has been used in [14]. The authenticator will become $\sigma_j = \left(H\left(m_j\right) \cdot h^{m_j}\right)^{1/x_s}$, and every time before auditing, the root stored on the cloud server and signed by the user will be verified first using some auxiliary information that allows reconstruction the MHT. The rest of our scheme remains unchanged.

6 Conclusion

In this paper, we introduced a new security notion – RIC Anonymity for RIC protocols. We also proposed a new RIC protocol that can preserve the identity privacy of the file owner when the TPA audits the files stored on the cloud server. We proved the soundness and the anonymity of the proposed protocol, and also showed that the protocol can support batch verifications of multiple RIC proofs, IND-Privacy, and data dynamic operations.

References

1. Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., Song, D.: Provable data possession at untrusted stores. In: ACM CCS, pp. 598–609 (2007)
2. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: EUROCRYPT, pp. 416–432 (2003)
3. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. J. Cryptology **17**(4), 297–319 (2004)
4. Fan, X., Yang, G., Mu, Y., Yu, Y.: On indistinguishability in remote data integrity checking. The Computer Journal (2013)
5. Groth, J., Sahai, A.: Efficient Non-interactive Proof Systems for Bilinear Groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008)
6. Juels, A., Kaliski Jr., B.S.: Pors: proofs of retrievability for large files. In: ACM CCS, pp. 584–597 (2007)
7. Shacham, H., Waters, B.: Compact Proofs of Retrievability. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 90–107. Springer, Heidelberg (2008)
8. Wang, B., Li, B., Li, H.: Knox: Privacy-Preserving Auditing for Shared Data with Large Groups in the Cloud. In: Bao, F., Samarati, P., Zhou, J. (eds.) ACNS 2012. LNCS, vol. 7341, pp. 507–525. Springer, Heidelberg (2012)

9. Wang, B., Li, B., Li, H.: Public auditing for shared data with efficient user revocation in the cloud. In: Joint Conference of the IEEE Computer and Communications Societies, pp. 2904–2912 (2013)
10. Wang, B., Li, B., Li, H.: Oruta: Privacy-preserving public auditing for shared data in the cloud. IEEE Transactions on Cloud Computing **2**(1), 43–56 (2014)
11. Wang, C., Chow, S.S.M., Wang, Q., Ren, K., Lou, W.: Privacy-preserving public auditing for secure cloud storage. IEEE Transaction on Computers 62(2), 362–375 (2013)
12. Wang, C., Wang, Q., Ren, K., Lou, W.: Privacy-preserving public auditing for data storage security in cloud computing. In: 2010 Proceedings IEEE INFOCOM, pp. 1–9, March 2010
13. Wang, Q., Wang, C., Li, J., Ren, K., Lou, W.: Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing. In: Backes, M., Ning, P. (eds.) ESORICS 2009. LNCS, vol. 5789, pp. 355–370. Springer, Heidelberg (2009)
14. Wang, Q., Wang, C., Ren, K., Lou, W., Li, J.: Enabling public auditability and data dynamics for storage security in cloud computing. IEEE Transactions on Parallel and Distributed System **22**(5), 847–859 (2011)
15. Yang, K., Jia, X.: An efficient and secure dynamic auditing protocol for data storage in cloud computing. IEEE Transactions on Parallel and Distributed Systems **24**(9), 1717–1726 (2013)

A A Security Issue in Oruta

Recently, a new public auditing protocol was proposed by Wang et al. in [10] to achieve the desirable features like public auditing, group anonymity and data dynamics, which is described in the Literature Review. Nevertheless, we review the protocol in [10] and show that there is a security issue in the protocol: a dishonest cloud server can delete some data blocks in a file without being caught by the third-party auditor. The reason is that the cloud server maintains an index hash table in order to achieve data dynamics. During auditing process, the cloud server answers to the identifier query for challenged indices, which will be used in the verification, but the integrity of the identifier is not ensured.

A.1 Reviewing the Protocol

Similar to most of the protocols (e.g, [11,14,15], the protocol Oruta [10] is based on the homomorphic authenticators and spot-checking techniques. However, it uses a ring signature [2] as the authenticator to achieve anonymity for each block. As it is a variant of BLS signature [3], the ring signature still can be aggregated. Besides, it uses the index hash table to support data dynamic operations. We describe the five algorithms inside Oruta.

Setup. Let G_1, G_2 and G_T be three multiplicative cyclic groups of prime order p, and g_1 and g_2 be the generators of G_1 and G_2, respectively. Let $e : G_1 \times G_2 \rightarrow G_T$ be a bilinear map, and $\psi : G_2 \rightarrow G_1$ be a computable isomorphism with $\psi(g_2) = g_1$. Let $H_1 : \{0,1\}^* \rightarrow G_1, H_2 : \{0,1\}^* \rightarrow Z_q$, and $h : G_1 \rightarrow Z_p$

denote three cryptographic hash functions. The global system parameters are $(e, \psi, p, q, G_1, G_2, G_T, g_1, g_2, H_1, H_2, h)$.

An outsourced data file M is divided into n blocks, and each block m_j is divided into k elements in Z_p. Then, the data component M can be viewed as an $n \times k$ matrix. Also, let d denote the number of users in the group where the file is shared and they want to preserve identity privacy from TPA.

KeyGen. Each user u_i randomly chooses $x_i \in Z_p$ and computes $w_i = g_2^{x_i}$. Thus, the user u_i's public key is w_i, and the private key is x_i. Besides, the user who firstly creates the file should generate a public aggregate key $pak = \{\eta_1, ..., \eta_k\}$, where each η_l $(1 \leq l \leq k)$ is a random elements of G_1.

SignGen. Given all members' public keys $\{w_1, ..., w_d\}$, a block $m_j = \{m_{j,1}, ..., m_{j,k}\}$, the identifier of the block id_j, a public aggregate key $pak = \{\eta_1, ..., \eta_k\}$ and the private key of the signer x_s, the user u_s generates a ring signature for this block as follows.

1. The signer first aggregates block m_j with the public aggregate key pak, and computes $\beta_j = H_1(id_j) \prod_{l=1}^{k} \eta_l^{m_{j,l}} \in G_1$.
2. Then the signer randomly chooses $a_{j,i} \in Z_p$ and sets $\sigma_{j,i} = g_1^{a_{j,i}}$ for all $i \neq s$. And for $i = s$, he/she computes $\sigma_{j,i} = (\frac{\beta_j}{\psi(\prod_{i \neq s} w_i^{a_{j,i}})})^{1/x_s}$. Therefore, the authenticator which is actually a ring signature of block m_j is $\sigma_j = \{\sigma_{j,1}, ..., \sigma_{j,d}\}$.

After the owner outsourced the file to the cloud server, any group members can send the auditing request to a third-party auditor (TPA). With the challenge and response process, the TPA checks the integrity of files without revealing the identity of the owner.

ProofGen. TPA generates the challenge in the following way:

1. The TPA picks c elements in set $[1, n]$, where n is the total number of blocks, to indicate the blocks that will be checked. Let J denote the indices of the chosen blocks.
2. For $j \in J$, the TPA randomly chooses $y_j \in Z_q$. Then the TPA sends $\{(j, y_j)\}_{j \in J}$ to the cloud server as a challenge message.

After receiving $\{(j, y_j)\}_{j \in J}$, the cloud server generates the proof as follows:

1. For $l \in [1, k]$, the cloud server randomly chooses $r_l \in Z_q$, and computes $\lambda_l = \eta_l^{r_l} \in G_1$, and $\mu_l = \sum_{j \in J} y_j m_{j,l} + r_l h(\lambda_l) \in Z_p$.
2. For $i \in [1, d]$, the cloud server computes $\phi_i = \prod_{j \in J} \sigma_{j,i}^{y_j}$.

Then the cloud server sends the proof $\{\{\lambda_l\}_{l \in [1,k]}, \{\mu_l\}_{l \in [1,k]}, \{\phi_i\}_{i \in [1,d]}, \{id_j\}_{j \in J}\}$ to TPA.

ProofVeriy. After receiving the proof, and given the public aggregate key $pak = \{\eta_1, ..., \eta_k\}$ and all the public keys $\{w_1, ..., w_d\}$ of the group members, the TPA verifies the proof by checking:

$$e(\prod_{j\in J} H_1(id_j)^{y_j} \cdot \prod_{l=1}^{k} \eta_l^{\mu_l}, g_2) \stackrel{?}{=} (\prod_{i=1}^{d} e(\phi_i, w_i)) \cdot e(\prod_{i=1}^{k} \lambda_l^{h(\lambda_l)}, g_2)$$

If this equation holds, the TPA sends a positive audit report back to the request user and vice versa.

Remarks. Instead of using the index of the block as its identifier (e.g. the index of block m_j is j), this scheme utilises index hash tables. An identifier from this table is described as $id_j = \{v_j, r_j\}$, where v_j is the virtual index of block m_j, and $r_j = H_2(m_j||v_j)$ is generated using a collision-resistance hash function $H_2 : \{0,1\}^* \in Z_q$. Here q is a prime that is much smaller than p. If $v_i < v_j$, then block m_i is in front of m_j in the file. The initial virtual index of block m_j is set as $v_j = j \cdot \delta$, where δ indicates the number of data block that can be inserted into m_j and m_{j+1}. For example, if m'_j is inserted, then $v'_j = (v_{j-1} + v_j)/2, r'_j = H_2(m'_j||v'_j)$ is inserted into the index hash table; if m_j is deleted, then the corresponding entry is removed from the table. During the computation of the authenticators and the challenge/response process, the identifiers are used instead of the indices. However, the TPA does not know about this table, so the identifier needs to be returned by the cloud server.

A.2 On the Security of the Protocol

As the basic requirements of an auditing protocol, Yang and Jia [15] pointed out that three kinds of attack must be prevented: Replace, Forge and Replay. In the first attack, the adversary may choose another pair of uncorrupted data block and data tag to replace the challenged pair to generate a valid proof. In the second attack, the adversary may try to forge a data tag for a (modified) data block. Lastly, in the replay attack, the adversary may try to generate a valid proof based on previous proofs or other information without retrieving the data blocks and data tags honestly. Below we show that the Oruta scheme [10] is vulnerable to the first type of attack.

For simplicity, let's consider the case where the TPA challenges for only one block m_j, and sends (j, y_j) to the cloud. If m_j is deliberately deleted by the cloud server but not the user, the server can still use m_{j+1}, σ_{j+1} to generate a valid proof as follows.

1. For $l \in [1, k]$, the cloud server randomly chooses $r_l \in Z_q$, and computes $\lambda_l = \eta_l^{r_l} \in G_1$.
2. For $l \in [1, k]$, the cloud server also computes $\mu_l = y_j m_{j+1,l} + r_l h(\lambda_l) \in Z_p$.
3. For $i \in [1, d]$, the cloud server computes $\phi_i = \sigma_{j+1,i}^{y_j}$.
   Then the cloud server sends the proof $\{\{\lambda\}, \{\mu\}, \{\phi\}, \{id_{j+1}\}$ to the TPA.

The TPA verifies the proof by checking: $e(H_1(id_{j+1})^{y_j} \cdot \prod_{l=1}^{k} \eta_l^{\mu_l}, g_2) \stackrel{?}{=} (\prod_{i=1}^{d} e(\phi_i, w_i)) \cdot e(\prod_{i=1}^{k} \lambda_l^{h(\lambda_l)}, g_2)$. Actually, this is still a valid proof as

$$
\begin{aligned}
RHS &= (\prod_{i=1}^{d} e(\sigma_{j+1,i}^{y^j}, w_i) \cdot e(\prod_{l=1}^{k} \lambda_l^{h(\lambda_l)}, g_2) \\
&= \left(\prod_{i=1}^{d} e(\sigma_{j+1,i}, w_i) \right)^{y_j} \cdot e(\prod_{l=1}^{k} \eta^{r_l h(\lambda_l)}, g_2) \\
&= e(\beta_{j+1}, g_2)^{y_j} \cdot e(\prod_{l=1}^{k} \eta^{r_l h(\lambda_l)}, g_2) \\
&= e((H_1(id_{j+1}) \prod_{l=1}^{k} \eta_l^{m_{j+1,l}})^{y_j}, g_2) \cdot e(\prod_{l=1}^{k} \eta^{r_l h(\lambda_l)}, g_2) \\
&= e(H_1(id_{j+1})^{y_j} \cdot \prod_{l=1}^{k} \eta_l^{y_j m_{j+1,l} + r_l h(\lambda_l)}, g_2) \\
&= e(H_1(id_{j+1})^{y_j} \cdot \prod_{l=1}^{k} \eta_l^{\mu_l}, g_2)
\end{aligned}
$$

Back to the normal case, since the cloud server maintains the index hash table, it can always redirect the challenge on the corrupted blocks to uncorrupted blocks in order to pass the verification. The index hash table is used by Oruta to support data dynamic operations. However, our attack essentially shows that such a technique is not suitable since it makes the scheme vulnerable to the Replace attack.

In [10], the authors assumed that the cloud server will return the identifiers for challenged indices honestly in the soundness (Unforgeability of Response) model and proof. However, as is known, the cloud server is treated as a soundness adversary in Remote Integrity Check protocols. So the cloud server has the motivation not to return the correct identifiers.

A.3 Recommendation

Despite of many interesting features supported by this protocol, we showed that a dishonest cloud server can deliberately delete some data blocks in a file but still pass the verification. We also pointed out that the problem is caused by the technique used in "Oruta" for allowing data dynamic operations.

Based on Yang and Jia's scheme [15], we can let the TPA maintain the index hash table as the cloud server is treated as the soundness adversary of the RIC protocol. Nevertheless, the TPA becomes stateful if we adopt such an approach, and another verifier cannot perform the auditing without such a table. Another option to solve the problem is to use the Merkle Hash Tree (MHT) which has been used in [14]. It uses a tree structure to organise the file blocks, and the

roots of the tree, which are derived from the file blocks, need to be verified in every operation.

B Proof of Correctness

Correctness of the Non-anonymous Verification Equation: $e\left(\prod_{j\in J} H(name||j)^{c_j}\cdot h^{\mu}, g_2\right) = \prod_{i=1}^{d} e\left(\Sigma_i, y_i\right)$.

$$
\begin{aligned}
RHS &= \prod_{i\neq s} e(\Sigma_i, y_i)\cdot e(\Sigma_s, y_s)\\
&= \prod_{i\neq s} e\left(g_1^{a_i}, y_i\right)\cdot e\left(\prod_{j\in J}\sigma_j^{c_j} \Big/ \psi\left(\prod_{i\neq s} y_i^{a_i}\right)^{1/x_s}, y_s\right)\\
&= e\left(g_1, \prod_{i\neq s}^{n} y_i^{a_i}\right)\cdot e\left(1\Big/\psi\left(\prod_{i\neq s} y_i^{a_i}\right), g_2\right)\cdot e\left(\prod_{j\in J}\sigma_j^{c_j}, y_s\right)\\
&= e\left(\psi\left(\prod_{i\neq s}^{n} y_i^{a_i}\right), g_2\right)\cdot e\left(1\Big/\psi\left(\prod_{i\neq s} y_i^{a_i}\right), g_2\right)\cdot e\left(\prod_{j\in J}\sigma_j^{c_j}, y_s\right)\\
&= e\left(\prod_{j\in J}\sigma_j^{c_j}, y_s\right) = e\left(\prod_{j\in J}\left(H\left(name||j\right)\cdot h^{m_j}\right)^{c_j/x_s}, g_2^{x_s}\right)\\
&= e\left(\prod_{j\in J} H\left(name||j\right)^{c_j}\cdot h^{\sum_{j\in J} c_j m_j}, g_2\right)\\
&= e\left(\prod_{j\in J}(H\left(name||j\right)^{c_j}\cdot h^{\mu}, g_2\right) = LHS
\end{aligned}
$$

Correctness of the anonymous verification equation: $\boldsymbol{l}(\boldsymbol{u}\star\boldsymbol{\pi}) = \boldsymbol{c}\star\boldsymbol{k}$.

$$
\begin{aligned}
LHS &= \boldsymbol{l}\ F(u_1,\pi_1)F(u_2,\pi_2)\\
&= \begin{pmatrix}1 & 1\\ 1 & L\end{pmatrix}\begin{pmatrix}1 & e(g_1, y_1^{r_{11}}y_2^{r_{21}}...y_d^{r_{d1}})\\ 1 & e(g_1^{\alpha}, y_1^{r_{11}}y_2^{r_{21}}...y_d^{r_{d1}})\end{pmatrix}\\
&= \begin{pmatrix}1 & e(g_1^{\tau}, y_1^{r_{12}}y_2^{r_{22}}...y_d^{r_{d2}})\\ 1 & e(g_1^{\tau\alpha}, y_1^{r_{12}}y_2^{r_{22}}...y_d^{r_{d2}})\end{pmatrix}
\end{aligned}
$$

$$
\begin{aligned}
RHS &= F(c_1,k_1)F(c_2,k_d)...F(c_d,k_d)\\
&= \begin{pmatrix}1 & e(g_1^{r_{11}+r_{12}\tau}, y_1)\\ 1 & e(g_1^{\alpha(r_{11}+r_{12}\tau)}\Sigma_1, y_1)\end{pmatrix}\begin{pmatrix}1 & e(g_1^{r_{21}+r_{22}\tau}, y_2)\\ 1 & e(g_1^{\alpha(r_{21}+r_{22}\tau)}\Sigma_2, y_2)\end{pmatrix}\begin{pmatrix}1 & e(g_1^{r_{d1}+r_{d2}\tau}, y_d)\\ 1 & e(g_1^{\alpha(r_{d1}+r_{d2}\tau)}\Sigma_d, y_d)\end{pmatrix}
\end{aligned}
$$

Calculating Hadamard product:

$$\begin{aligned} & e(g_1, y_1^{r_{11}} y_2^{r_{21}} ... y_d^{r_{d1}}) e(g_1^{\tau}, y_1^{r_{12}} y_2^{r_{22}} ... y_d^{r_{d2}}) \\ = & e(g_1, y_1^{r_{11}+r_{12}\tau}) e(g_1, y_2^{r_{21}+r_{22}\tau}) ... e(g_1, y_d^{r_{d1}+r_{d2}\tau}) \\ = & e(g_1^{r_{11}+r_{12}\tau}, y_1) e(g_1^{r_{21}+r_{22}\tau}, y_2) ... e(g_1^{r_{d1}+r_{d2}\tau}, y_d) \end{aligned}$$

$$\begin{aligned} & L\ e(g_1^{\alpha}, y_1^{r_{11}} y_2^{r_{21}} ... y_d^{r_{d1}}) e(g_1^{\tau\alpha}, y_1^{r_{12}} y_2^{r_{22}} ... y_d^{r_{d2}}) \\ = & L\ e(g_1^{\alpha(r_{11}+r_{12}\tau)}, y_1) e(g_1^{\alpha(r_{21}+r_{22}\tau)}, y_2) e(g_1^{\alpha(r_{d1}+r_{d2}\tau)}, y_d) \\ & \text{(as L can be viewed as } e(\Sigma_1, y_1) e(\Sigma_2, y_2) ... e(\Sigma_d, y_d)) \\ = & e(g_1^{\alpha(r_{11}+r_{12}\tau)} \Sigma_1, y_1) e(g_1^{\alpha(r_{21}+r_{22}\tau)} \Sigma_2, y_2) e(g_1^{\alpha(r_{d1}+r_{d2}\tau)} \Sigma_d, y_d) \end{aligned}$$

C Proof of Anonymity (Theorem 1)

Proof. For the (unconditional) anonymity of the file tag, it directly follows that of the underlying ring signature (e.g., [2]). In the following we will focus on the anonymity of the responses returned from cloud server. Let $\mathcal{A}$ be the adversary who has a non-negligible advantage ϵ in winning the Anonymity game, we can construct a simulator $\mathcal{C}$ to solve the XDH problem.

Simulator $\mathcal{C}$ receives a challenge $gk, \psi, A = g_1^x, B = g_1^y, C = g_1^z$, where z is either xy or a random element ξ in Z_p. $\mathcal{C}$ simulates the game as follows.

Setup: Simulator $\mathcal{C}$ sets the commitment key to be $u_1 = (g_1, A)$, $u_2 = (B, C)$. $\mathcal{C}$ then uses the information in gk and ψ to generate all user public/private key pairs as described. Finally Simulator $\mathcal{C}$ randomly chooses a user $b \in [1, d]$ as the real owner of all the queried files.

Phase 1: Simulator $\mathcal{C}$ answers the tag queries honestly and stores the tokens, glue value and auxiliary information correctly.

Phase 2: To answer a proof query, $\mathcal{C}$ computes $\mu_b, \{\Sigma_b\}$ honestly. Then $\mathcal{C}$ uses $\{\Sigma_b\}$ to generate response as follows: Randomly choose $r_{11}, r_{12}, r_{21}, r_{22} ... r_{d1}, r_{d2}$ from Z_p. Then compute the commitment $\boldsymbol{c}$: $c_{11} = g_1^{r_{11}} B^{r_{12}}$, $c_{12} = A^{r_{11}} C^{r_{12}} \Sigma_{b1}$, $c_{21} = g_1^{r_{21}} B^{r_{22}}$, $c_{22} = A^{r_{21}} C^{r_{22}} \Sigma_{b2}$, ... $c_{d1} = g_1^{r_{d1}} B^{r_{d2}}$, $c_{d2} = A^{r_{d1}} C^{r_{d2}} \Sigma_{bd}$. Finally compute the proof $\boldsymbol{\pi} = (\pi_1, \pi_2)$ where $\pi_1 = (1, y_1^{r_{11}} y_2^{r_{21}} ... y_d^{r_{d1}})$ and $\pi_2 = (1, y_1^{r_{12}} y_2^{r_{22}} ... y_d^{r_{d2}})$.

Decision: Simulator $\mathcal{C}$ sends the response $(\mu, \boldsymbol{c}, \boldsymbol{\pi})$ to $\mathcal{A}$. $\mathcal{A}$ outputs b' as a guess of b. If $b' = b$, then simulator $\mathcal{C}$ outputs 1; otherwise $\mathcal{C}$ outputs 0 for the instance of the XDH problem.

Probability: Case 1: $z = xy$. In this case, the distribution of the response is identically to that of a real response, so we have $\Pr[b' = b] = 1/d + \epsilon$. **Case 2:** $z = \xi$. In this case, the commitment scheme is perfectly hiding and reveals no information about the value of b. Hence, under this case we have $\Pr[b' = b] = 1/d$. Combining both cases, $\mathcal{C}$ has advantage ϵ to solve the XDH problem.

D Proof of Soundness (Theorem 2)

Proof. We will prove this by contradiction. If there exists an adversary $\mathcal{A}$ who can win the soundness game with non-negligible probability, then we can construct a simulator to solve the variant of the Co-CDH problem also with a non-negligible probability:

Setup: Simulator $\mathcal{C}$ sets up the system parameters and chooses $\alpha, \tau \in Z_p$ to generate the commitment key $u_1 = (g_1, g_1^{\alpha})$ and $u_2 = (g_1^{\tau}, g_1^{\tau\alpha})$. $\mathcal{C}$ then sets $h = g_1^{ab}$ as the other generator in system parameter and constructs a table $< name, j, r_j, H(name||j) >$ for hash oracle. $\mathcal{C}$ also randomly chooses $x_i \in Z_p$ for each user and computes the public key $y_i = (g_2^a)^{x_i} = u^{x_i}$. (We can view the private key as ax_i.)

Phase 1: For simplicity, Simulator $\mathcal{C}$ chooses the first user as the signing user, and generates the file tag using the underlying ring signature scheme. $\mathcal{C}$ answers the token queries as follows:

- Hash Query: Simulator $\mathcal{C}$ randomly chooses $r_j \in Z_p$ and sets $H(name||j) = \psi(g_2^a)^{r_j}/h^{m_j}$. The entry $< name, j, r_j, H(name||j) >$ is stored into the table.
- Authenticator: $\sigma_j = (H(name||j) \cdot h^{m_j})^{1/ax_1} = \psi(g_2^a)^{r_j/ax_1} = g_1^{r_j/x_1}$

Simulator $\mathcal{C}$ answers the glue queries as follows:

- For each user $i \neq 1$ in the ring, $\mathcal{C}$ chooses $a_i \in Z_p$, and computes auxiliary information. $b_i = g_1^{a_i}$
- $\mathcal{C}$ computes a glue value $\theta = \frac{1}{\psi\left(\prod_{i\neq 1}^{d} y_i^{a_i}\right)^{1/ax_1}} = \frac{1}{\psi\left(\prod_{i\neq 1}^{n}((g_2^a)^{x_i})^{a_i}\right)^{1/ax_1}} = \frac{1}{g_1^{\left(\sum_{i\neq 1} a_i x_i\right)/x_1}}$.

Phase 2: Finally $\mathcal{A}$ outputs a proof $P' = (\mu', \boldsymbol{c}, \boldsymbol{\pi})$ for $t', \{m'_j\}_{j\in J}$ and challenge $\{c_j\}_{j\in J}$, where at least one m'_j has been modified by the adversary.

$\mathcal{C}$ uses the commitment secret key α to recover the $\Sigma'_i = c_{i2}/c_{i1}^{\alpha}$ and let $\mu = \sum_{j\in J} c_j m_j$, $\{\Sigma\}$ be the honestly generated component. Then we have the following equations: $e\left(\prod_{j\in J} H\left(name||j\right)^{c_j} \cdot h^{\mu'}, g_2\right) = \prod_{i=1}^{n} e\left(\Sigma'_i, y_i\right)$ then $e\left(\prod_{j\in J} H\left(name||j\right)^{c_j} \cdot h^{\mu}, g_2\right) = \prod_{i=1}^{n} e\left(\Sigma_i, y_i\right)$. Therefore, we can get $e(h^{\mu'-\mu},\ g_2) = \prod_{i=1}^{n} e\left(\Sigma'_i/\Sigma_i, y_i\right)$ then $e(h^{\mu'-\mu}, g_2) = e\left(\prod_{i=1}^{n}\left(\Sigma'_i/\Sigma_i\right)^{x_i}, g_2^a\right)$. As $\mathcal{C}$ chooses the challenges c_j at random and $h = g_1^{ab}$, with overwhelming probability $1 - 1/p, \mu' = \sum_{j\in J} c_j m'_j \neq \mu = \sum_{j\in J} c_j m_j$, we can obtain

$$g_1^b = \left(\prod_{i=1}^{n}\left(\Sigma'_i/\Sigma_i\right)^{x_i}\right)^{1/(\mu'-\mu)}.$$