Identity-Based Distributed Provable Data Possession in Multi-Cloud Storage

***Abstract*—Remote data integrity checking is of crucial importance in cloud storage. It can make the clients verify whether their outsourced data is kept intact without downloading the whole data. In some application scenarios, the clients have to store their data on multi-cloud servers. At the same time, the integrity checking protocol must be efficient in order to save the verifier's cost. From the two points, we propose a novel remote data integrity checking model: ID-DPDP (identity-based distributed provable data possession) in multi-cloud storage. The formal system model and security model are given. Based on the bilinear pairings, a concrete ID-DPDP protocol is designed. The proposed ID-DPDP protocol is provably secure under the hardness assumption of the standard CDH (computational Diffie-Hellman) problem. In addition to the structural advantage of elimination of certificate management, our ID-DPDP protocol is also efficient and flexible. Based on the client's authorization, the proposed ID-DPDP protocol can realize private verification, delegated verification and public verification.**

授权

***Index Terms*—Cloud computing, Provable data possession, Identity-based cryptography, Distributed computing, Bilinear pairings**

I. Introduction

Over the last years, cloud computing has become an important theme in the computer field. Essentially, it takes the information processing as a service, such as storage, computing. It relieves of the burden for storage management, universal data access with independent geographical locations. At the same time, it avoids of capital expenditure on hardware, software, and personnel maintenances, etc. Thus, cloud computing attracts more intention from the enterprise.

The foundations of cloud computing lie in the outsourcing of computing tasks to the third party. It entails the security risks in terms of confidentiality, integrity and availability of data and service. The issue to convince the cloud clients that their data are kept intact is especially vital since the clients do not store these data locally. Remote data integrity checking is a primitive to address this issue. For the general case, when the client stores his data on multi-cloud servers, the distributed storage and integrity checking are indispensable. On the other hand, the integrity checking protocol must be efficient in order to make it suitable for capacity-limited end devices. Thus, based on distributed computation, we will study distributed remote data integrity checking model and present the corresponding concrete protocol in multi-cloud storage.

A. Motivation

We consider an ocean information service corporation Cor in the cloud computing environment. Cor can provide the following services: ocean measurement data, ocean environment monitoring data, hydrological data, marine biological data, GIS information, etc. Besides of the above services, Cor has also some private information and some public information, such as the corporation's advertisement. Cor will store these different ocean data on multiple cloud servers. Different cloud service providers have different reputation and charging standard. Of course, these cloud service providers need different charges according to the different security-levels. Usually, more secure and more expensive. Thus, Cor will select different cloud service providers to store its different data. For some sensitive ocean data, it will copy these data many times and store these copies on different cloud servers. For the private data, it will store them on the private cloud server. For the public advertisement data, it will store them on the cheap public cloud server. At last, Cor stores its whole data on the different cloud servers according to their importance and sensitivity. Of course, the storage selection will take account into the Cor's profits and losses. Thus, the distributed cloud storage is indispensable. In multi-cloud environment, distributed provable data possession is an important element to secure the remote data.

In PKI (public key infrastructure), provable data possession protocol needs public key certificate distribution and management. It will incur considerable overheads since the verifier will check the certificate when it checks the remote data integrity. In addition to the heavy certificate verification, the system also suffers from the other complicated certificates management such as certificates generation, delivery, revocation, renewals, etc. In cloud computing, most verifiers only have low computation capacity. Identity-based public key cryptography can eliminate the complicated certificate management. In order to increase the efficiency, identity-based provable data possession is more attractive. Thus, it will be very meaningful to study the ID-DPDP.

B. Related work

In cloud computing, remote data integrity checking is an important security problem. The clients' massive data is outside his control. The malicious cloud server may corrupt the clients' data in order to gain more benefits. Many researchers proposed the corresponding system model and security model. In 2007, provable data possession (PDP) paradigm was proposed by Ateniese *et al.* [1]. In the PDP model, the verifier can check remote data integrity with a high probability. Based on the RSA, they designed two provably secure PDP schemes. After that, Ateniese *et al.* proposed dynamic PDP model and concrete scheme [2] although it does not support insert operation. In order to support the insert operation, in 2009, Erway *et al.* proposed a full-dynamic PDP scheme based on the authenticated flip table [3]. The similar work has also been done by F. Sebé *et al.* [4]. PDP allows a verifier to verify the remote data integrity without retrieving or downloading the whole data. It is a probabilistic proof of possession by sampling random set of blocks from the server, which drastically reduces I/O costs. The verifier only maintains small metadata to perform the integrity checking. PDP is an interesting remote data integrity checking model. In 2012, Wang proposed the security model and concrete scheme of proxy PDP in public clouds [5]. At the same time, Zhu *et al.* proposed the cooperative PDP in the multi-cloud storage [6].

Following Ateniese *et al.*'s pioneering work, many remote data integrity checking models and protocols have been proposed [7], [8], [9], [10], [11], [12]. In 2008, Shacham presented the first proof of retrievability (POR) scheme with provable security [13]. In POR, the verifier can check the remote data integrity and retrieve the remote data at any time. The state of the art can be found in [14], [15], [16], [17]. On some cases, the client may delegate the remote data integrity checking task to the third party. It results in the third party auditing in cloud computing [18], [19], [20], [21]. One of benefits of cloud storage is to enable universal data access with independent geographical locations. This implies that the end devices may be mobile and limited in computation and storage. Efficient integrity checking protocols are more suitable for cloud clients equipped with mobile end devices.

C. Contributions

In identity-based public key cryptography, this paper focuses on distributed provable data possession in multi-cloud storage. The protocol can be made efficient by eliminating the certificate management. We propose the new remote data integrity checking model: ID-DPDP. The system model and security model are formally proposed. Then, based on the bilinear pairings, the concrete ID-DPDP protocol is designed. In the random oracle model, our ID-DPDP protocol is provably secure. On the other hand, our protocol is more flexible besides the high efficiency. Based on the client's authorization, the proposed ID-DPDP protocol can realize private verification, delegated verification and public verification.

D. Paper Organization

The rest of the paper is organized as follows. Section II formalizes the ID-DPDP model. Section III presents our ID-DPDP protocol with a detailed performance analysis. Section IV evaluates the security of the proposed ID-DPDP protocol. Finally, Section V concludes the paper.

II. System model and security model of ID-DPDP

The ID-DPDP system model and security definition are presented in this section. An ID-DPDP protocol comprises four different entities which are illustrated in Figure 1. We describe them below:

1) *Client*: an entity, which has massive data to be stored on the multi-cloud for maintenance and computation, can be either individual consumer or corporation.
2) *CS* (Cloud Server): an entity, which is managed by cloud service provider, has significant storage space and computation resource to maintain the clients' data.

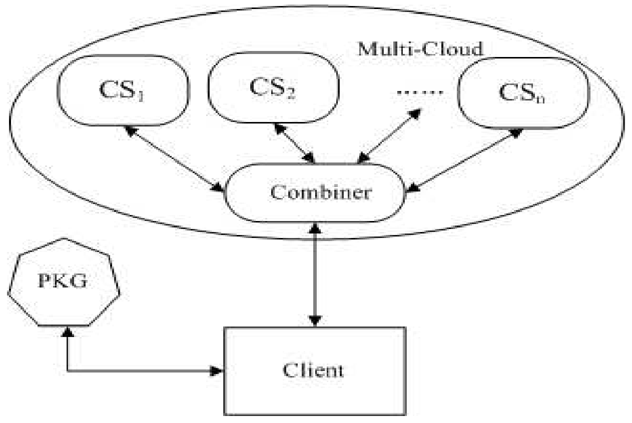

Fig. 1. The System Model of ID-DPDP

3) *Combiner*: an entity, which receives the storage request and distributes the block-tag pairs to the corresponding cloud servers. When receiving the challenge, it splits the challenge and distributes them to the different cloud servers. When receiving the responses from the cloud servers, it combines them and sends the combined response to the verifier.
4) *PKG* (Private Key Generator): an entity, when receiving the identity, it outputs the corresponding private key.

First, we give the definition of interactive proof system. It will be used in the definition of ID-DPDP. Then, we present the definition and security model of ID-DPDP protocol.

Definition 1 (Interactive Proof System): [22] Let $c, s : \mathbb{N} \rightarrow \mathbb{R}$ be functions satisfying $c(n) > s(n) + \frac{1}{p(n)}$ for some polynomial $p(\cdot)$. An interactive pair (P, V) is called a interactive proof system for the language L, with completeness bound $c(\cdot)$ and soundness bound $s(\cdot)$, if

1) Completeness: for every $x \in L$, $\Pr[< P, V > (x) = 1] \geq c(|x|)$.
2) Soundness: for every $x \notin L$ and every interactive machine B, $\Pr[< B, V > (x) = 1] \leq s(|x|)$.

Interactive proof system is used in the definition of ID-DPDP, *i.e.*, Definition 2.

Definition 2 (ID-DPDP): An ID-DPDP protocol is a collection of three algorithms (Setup, Extract, TagGen) and an interactive proof system (Proof). They are described in detail below.

1) $\mathsf{Setup}(1^k)$: Input the security parameter k, it outputs the system public parameters *params*, the master public key *mpk* and the master secret key *msk*.
2) $\mathsf{Extract}(1^k, params, mpk, msk, ID)$: Input the public parameters $params$, the master public key mpk, the master secret key msk, and the identity ID of a client, it outputs the private key sk_{ID} that corresponds to the client with the identity *ID*.
3) $\mathsf{TagGen}(sk_{ID}, F_i, \mathcal{P})$: Input the private key sk_{ID}, the block F_i and a set of *CS* $\mathcal{P} = \{CS_j\}$, it outputs the tuple $\{\phi_i, (F_i, T_i)\}$, where ϕ_i denotes the i-th record of metadata, (F_i, T_i) denotes the i-th block-tag pair. Denote all the metadata $\{\phi_i\}$ as ϕ.

4) $\mathsf{Proof}(\mathcal{P}, C(Combiner), V(Verifier))$: is a protocol among $\mathcal{P}$, C and V. At the end of the interactive protocol, V outputs a bit $\{0|1\}$ denoting false or true.

Besides of the high efficiency based on the communication and computation overheads, a practical ID-DPDP protocol must satisfy the following security requirements:

1) The verifier can perform the ID-DPDP protocol without the local copy of the file(s) to be checked.
2) If some challenged block-tag pairs are modified or lost, the response can not pass the ID-DPDP protocol even if $\mathcal{P}$ and C collude.

To capture the above security requirements, we define the security of an ID-DPDP protocol as follows.

Definition 3 (Unforgeability): An ID-DPDP protocol is unforgeable if for any (probabilistic polynomial) adversary $\mathcal{A}$ (malicious CS and combiner) the probability that $\mathcal{A}$ wins the ID-DPDP game on a set of file blocks is negligible. The ID-DPDP game between the adversary $\mathcal{A}$ and the challenger $\mathcal{C}$ can be described as follows:

1) Setup: The challenger $\mathcal{C}$ runs $Setup(1^k)$ and gets $(params, mpk, msk)$. It sends the public parameters and master public key $(params, mpk)$ to $\mathcal{A}$ while it keeps confidential the master secret key msk.
2) First-Phase Queries: The adversary $\mathcal{A}$ adaptively makes $\mathsf{Extract}$, Hash, TagGen queries to the challenger $\mathcal{C}$ as follows:
   - $\mathsf{Extract}$ queries. The adversary $\mathcal{A}$ queries the private key of the identity ID. By running $\mathsf{Extract}(params, mpk, msk, ID)$, the challenger $\mathcal{C}$ gets the private key sk_{ID} and forwards it to $\mathcal{A}$. Let S_1 denote the extracted identity set in the first-phase.
   - Hash queries. The adversary $\mathcal{A}$ queries *hash* function adaptively. $\mathcal{C}$ responds the *hash* values to $\mathcal{A}$.
   - TagGen queries. The adversary $\mathcal{A}$ makes block-tag pair queries adaptively. For a block tag query F_i, the challenger calculates the tag T_i and sends it back to the adversary. Let (F_i, T_i) be the queried block-tag pair for index $i \in \mathbb{I}_1$, where $\mathbb{I}_1$ is a set of indices that the corresponding block tags have been queried in the first-phase.
3) Challenge: $\mathcal{C}$ generates a challenge $chal$ which defines a ordered collection $\{ID^*, i_1, i_2, \cdots, i_c\}$, where $ID^* \notin S_1$, $\{i_1, i_2, \cdots, i_c\} \nsubseteq \mathbb{I}_1$, and c is a positive integer. The adversary is required to provide the data possession proof for the blocks $F_{i_1}, \cdots, F_{i_c}$.
4) Second-Phase Queries: Similar to the First-Phase Queries. Let the $\mathsf{Extract}$ query identity set be S_2 and the TagGen query index set be $\mathbb{I}_2$. The restriction is that $\{i_1, i_2, \cdots, i_c\} \nsubseteq (\mathbb{I}_1 \cup \mathbb{I}_2)$ and $ID^* \notin (S_1 \cup S_2)$.
5) Forge: The adversary $\mathcal{A}$ responses θ for the challenge $chal$.

We say that the adversary $\mathcal{A}$ wins the ID-DPDP game if the response θ can pass $\mathcal{C}$'s verification.

Definition 3 states that, for the challenged blocks, the malicious CS or C cannot produce a proof of data possession if the blocks have been modified or deleted. On the other hand, the definition does not state clearly the status of the blocks that are not challenged. In practice, a secure ID-DPDP protocol also needs to convince the client that all of his outsourced data is kept intact with a high probability. We give the following security definition.

TABLE I
NOTATIONS AND DESCRIPTIONS

Notations	Descriptions
$\mathcal{G}_1$	Cyclic multiplicative group with order q
$\mathcal{G}_2$	Cyclic multiplicative group with order q
$\mathcal{Z}_q^*$	$\{1, 2, \cdots, q-1\}$
g	A generator of $\mathcal{G}_1$
d	A generator of $\mathcal{G}_2$
H, h, h_1	Three cryptographic hash functions
f	Pseudo-random function
π	Pseudo-random permutation
(x, Y)	Master secret/public key pair
(ID, sk_{ID})	Client's identity-private key pair
(R, σ)	Client's private key, $sk_{ID} = (R, \sigma)$
n	The block number
s	The sector number
$F = (F_1, \cdots, F_n)$	The stored file F is split into n blocks
$F_i = (\tilde{F}_{i1}, \cdots, \tilde{F}_{is})$	The block F_i is split into s blocks
F_{ij}	$F_{ij} = h_1(\tilde{F}_{ij})$
CS	Cloud server
$\hat{n}$	The cloud server number
l_i	The index of the CS which stores the i-th block-tag pair
CS_{l_i}	The CS which stores the i-th block
$\mathcal{P} = \{CS_i, 1 \leq l \leq \hat{n}\}$	The CS set
T_{cl}	The client's table which lists the storage metadata
T_o	The combiner's table
$u = \{u_1, \cdots, u_s\}$	The parameters picked by the client
$\phi_i = (i, u, N_i, CS_{l_i})$	The record where i denotes the i-th block, N_i denotes the block F_i's name
(F_i, T_i)	The i-th block-tag pair
C	The combiner
V	The verifier
v_i	The permutated index $v_i = \pi_{k_1}(i)$ of i
$\theta_i = (\hat{F}^{(i)}, T^{(i)})$, where $\hat{F}^{(i)} = (\hat{F}_1^{(i)}, \cdots, \hat{F}_s^{(i)})$	CS$_i$'s response to the combiner C
$\theta = (\hat{F}, T)$, where $\hat{F} = (\hat{F}_1, \hat{F}_2, \cdots, \hat{F}_s)$	The combiner's response to the verifier V

Definition 4 ((ρ, δ) security): An ID-DPDP protocol is (ρ, δ) secure if the CS corrupted ρ fraction of the whole blocks, the probability that the corrupted blocks are detected is at least δ.

The notations throughout this paper are listed in Table I.

III. The Proposed ID-DPDP Protocol

In this section, we present an efficient ID-DPDP protocol. It is built from bilinear pairings which will be briefly reviewed below.

A. *Bilinear pairings*

Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be two cyclic multiplicative groups with the same prime order q. Let $e : \mathcal{G}_1 \times \mathcal{G}_1 \rightarrow \mathcal{G}_2$ be a bilinear map [25] which satisfies the following properties:

1) Bilinearity: $\forall g_1, g_2, g_3 \in \mathcal{G}_1$ and $a, b \in \mathcal{Z}_q$,
$$e(g_1, g_2g_3) = e(g_2g_3, g_1) = e(g_2, g_1)e(g_3, g_1)$$
$$e({g_1}^a, {g_2}^b) = e(g_1, g_2)^{ab}$$
2) Non-degeneracy: $\exists g_4, g_5 \in \mathcal{G}_1$ such that $e(g_4, g_5) \neq 1_{\mathcal{G}_2}$.
3) Computability: $\forall g_6, g_7 \in \mathcal{G}_1$, there is an efficient algorithm to calculate $e(g_6, g_7)$.

Such a bilinear map e can be constructed by the modified Weil [23] or Tate pairings [24] on elliptic curves. Our ID-DPDP scheme relies on the hardness of CDH (Computational Diffie-Hellman) problem and the easiness of DDH (Decisional Diffie-Hellman) problem. They are defined below.

Definition 5 (CDH Problem on $\mathcal{G}_1$): Let g be the generator of $\mathcal{G}_1$. Given $g, g^a, g^b \in \mathcal{G}_1$ for randomly chosen $a, b \in \mathcal{Z}_q$, calculate $g^{ab} \in \mathcal{G}_1$.

Definition 6 (DDH Problem on $\mathcal{G}_1$): Let g be the generator of $\mathcal{G}_1$. Given $(g, g^a, g^b, \hat{g}) \in \mathcal{G}_1^4$ for randomly chosen $a, b \in \mathcal{Z}_q^*$, decide whether $g^{ab} \overset{?}{=} \hat{g}$.

In the paper, the chosen group $\mathcal{G}_1$ satisfies that CDH problem is difficult but DDH problem is easy. The DDH problem can be solved by making use of the bilinear pairings. Thus, $(\mathcal{G}_1, \mathcal{G}_2)$ are also defined as GDH (Gap Diffie-Hellman) groups.

B. *The Concrete ID-DPDP Protocol*

This protocol comprises four procedures: Setup, Extract, TagGen, and Proof. Its architecture can be depicted in Figure 2. The figure can be described as follows: 1. In the phase Extract, PKG creates the private key for the client. 2. The client creates the block-tag pair and uploads it to combiner. The combiner distributes the block-tag pairs to the different cloud servers according to the storage metadata. 3. The verifier sends the challenge to combiner and the combiner distributes the challenge query to the corresponding cloud servers according to the storage metadata. 4. The cloud servers respond the challenge and the combiner aggregates these responses from the cloud servers. The combiner sends the aggregated response to the verifier. Finally, the verifier checks whether the aggregated response is valid.

The concrete ID-DPDP construction mainly comes from the signature, provable data possession and distributed computing. The signature relates the client's identity with his private key. Distributed computing is used to store the client's data on multi-cloud servers. At the same time, distributed computing is also used to combine the multi-cloud servers' responses to respond the verifier's challenge. Based on the provable data possession protocol [13], the ID-DPDP protocol is constructed by making use of the signature and distributed computing.

Without loss of generality, let the number of stored blocks be n. For different block F_i, the corresponding tuple (N_i, CS_{l_i}, i) is also different. F_i denotes the i-th block. Denote N_i as the name of F_i. F_i is stored in CS_{l_i} where

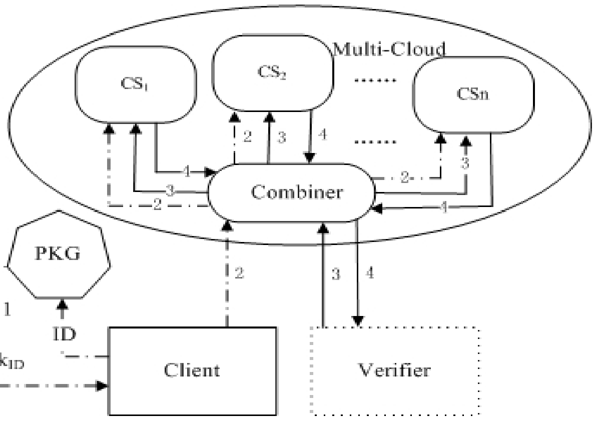

Fig. 2. Architecture of our ID-DPDP protocol

l_i is the index of the corresponding CS. (N_i, CS_{l_i}, i) will be used to generate the tag for the block F_i. The algorithms can be described in detail below.

- Setup: Let g be a generator of the group $\mathcal{G}_1$ with the order q. Define the following cryptographic hash functions:
$$H : \{0,1\}^* \rightarrow \mathcal{Z}_q^*$$
$$h : \{0,1\}^* \times \mathcal{Z}_q^* \rightarrow \mathcal{G}_1$$
$$h_1 : \{0,1\}^* \rightarrow \mathcal{Z}_q^*$$
Let f be a pseudo-random function and π be a pseudo-random permutation. They can be described in detail below:
$$f : \mathcal{Z}_q^* \times \{1, 2, \cdots, n\} \rightarrow \mathcal{Z}_q^*$$
$$\pi : \mathcal{Z}_q^* \times \{1, 2, \cdots, n\} \rightarrow \{1, 2, \cdots, n\}$$
公开参数
PKG picks a random number $x \in \mathcal{Z}_q^*$ and calculates $Y = g^x$. The parameters $\{\mathcal{G}_1, \mathcal{G}_2, e, q, g, Y, H, h, h_1, f, \pi\}$ are made public. PKG keeps the master secret key x confidential.
- Extract: Input the identity ID, PKG picks $r \in \mathcal{Z}_q^*$ and calculates
$$R = g^r, \qquad \sigma = r + xH(ID, R) \bmod q$$
PKG sends the private key $sk_{ID} = (R, \sigma)$ to the client by the secure channel. The client can verify the correctness of the received private key by checking whether the following equation holds.
$$g^{\sigma} \overset{?}{=} RY^{H(ID,R)} \tag{1}$$
If the formula (1) holds, the client ID accepts the private key; otherwise, the client ID rejects it.
- TagGen$(sk_{ID}, F, \mathcal{P})$: Split the whole file F into n blocks, *i.e.*, $F = (F_1, F_2, \cdots, F_n)$. The client prepares to store the block F_i in the cloud server CS_{l_i}. Then, for $1 \leq i \leq n$, each block F_i is split into s sectors, *i.e.*, $F_i = \{\tilde{F}_{i1}, \tilde{F}_{i2}, \cdots, \tilde{F}_{is}\}$. Thus, the client gets $n \times s$ sectors $\{\tilde{F}_{ij}\}_{i\in[1,n], j\in[1,s]}$. Picks s random $u_1, u_2, \cdots, u_s \in \mathcal{G}_1$.

Denote $u = \{u_1, u_2, \cdots, u_s\}$. For F_i, the client performs the procedures below:

1) The client calculates $F_{ij} = h_1(\tilde{F}_{ij})$ for every sector $\tilde{F}_{ij}, 1 \le j \le s$.
2) The client calculates
$$T_i = (h(N_i, CS_{l_i}, i) \prod_{j=1}^{s} u_j^{F_{ij}})^{\sigma}$$
3) The client adds the record $\phi_i = (i, u, N_i, CS_{l_i})$ to the table T_{cl}. It stores T_{cl} locally.
4) The client sends the metadata table T_{cl} to the combiner. The combiner adds the records of T_{cl} to its own metadata table T_o.
5) The client outputs T_i and stores (F_i, T_i) in CS$_{l_i}$.

- Proof ($\mathcal{P}$, C, V): This is a 5-move protocol among $\mathcal{P} = \{CS_i\}_{i \in [1, \hat{n}]}$), C, and V with the input (*public parameters*, T_{cl}). If the client delegates the verification task to some verifier, it sends the metadata table T_{cl} to the verifier. Of course, the verifier may be the third auditor or the client's proxy. The interaction protocol can be given in detail below.
  1) Challenge 1 (C $\leftarrow$ V): the verifier picks the challenge $chal = (c, k_1, k_2)$ where $1 \le c \le n, k_1, k_2 \in \mathcal{Z}_q^*$ and sends $chal$ to the combiner C;
  2) Challenge 2 ($\mathcal{P} \leftarrow$ C): the combiner calculates $v_i = \pi_{k_1}(i), 1 \le i \le c$ and looks up the table T_o to get the records that correspond to $\{v_1, v_2, \cdots, v_c\} = \mathcal{M}_1 \bigcup \mathcal{M}_1 \bigcup \cdots \bigcup \mathcal{M}_{\hat{n}}$. $\mathcal{M}_i$ denotes the index set where the corresponding block-tag pair is stored in CS$_i$. Then, C sends $(\mathcal{M}_i, k_2)$ to CS$_i \in \mathcal{P}$.
  3) Response1 ($\mathcal{P} \rightarrow$ C): For CS$_i \in \mathcal{P}$, it performs the procedures below:
     a) For $v_l \in \mathcal{M}_i$, CS$_i$ splits F_{v_l} into s sectors $F_{v_l} = \{\tilde{F}_{v_l 1}, \tilde{F}_{v_l 2}, \cdots, \tilde{F}_{v_l s}\}$ and calculates $F_{v_l j} = h_1(\tilde{F}_{v_l j})$ for $1 \le j \le s$.
     (*Note*: $F_{v_l j} = h_1(\tilde{F}_{v_l j})$ can also be precomputed and stored by CS)
     b) CS$_i$ calculates $a_l = f_{k_2}(l), v_l \in \mathcal{M}_i$ and
     $$T^{(i)} = \prod_{v_l \in \mathcal{M}_i} {T_{v_l}}^{a_l}$$
     c) For $1 \le j \le s$, CS$_i$ calculates
     $$\hat{F}_j^{(i)} = \sum_{v_l \in \mathcal{M}_i} a_l F_{v_l j}$$
     Denote $\hat{F}^{(i)} = (\hat{F}_1^{(i)}, \cdots, \hat{F}_s^{(i)})$.
     d) CS$_i$ sends $\theta_i = (\hat{F}^{(i)}, T^{(i)})$ to C.
  4) Response2 (C $\rightarrow$ V): After receiving all the responses from CS$_i \in \mathcal{P}$, the combiner aggregates $\{\theta_i\}_{CS_i \in \mathcal{P}}$ into the final response as
  $$T = \prod_{CS_i} T^{(i)}, \quad \hat{F}_l = \sum_{CS_i} \hat{F}_l^{(i)}$$
  Denote $\hat{F} = (\hat{F}_1, \hat{F}_2, \cdots, \hat{F}_s)$. The combiner sends $\theta = (\hat{F}, T)$ to the verifier V.
  5) After receiving the response $\theta = (\hat{F}, T)$, the verifier calculates
  $$v_i = \pi_{k_1}(i)$$
  $$h_i = h(N_{v_i}, CS_{l_{v_i}}, v_i)$$
  $$a_i = f_{k_2}(i)$$
  Then, it verifies whether the following formula holds.
  $$e(T, g) \stackrel{?}{=} e(\prod_{i=1}^{c} h_i^{a_i} \prod_{j=1}^{s} u_j^{\hat{F}_j}, RY^{H(ID,R)}) \qquad (2)$$
  If the formula (2) holds, then the verifier outputs "success". Otherwise, the verifier outputs "failure".

Notes: In Proof($\mathcal{P}$, C, V), the verifier picks the challenge $chal = (c, k_1, k_2)$ where k_1, k_2 are randomly chosen from $\mathcal{Z}_q^*$. Based on the challenge $chal$, the combiner determines the challenged block index set as $\{v_1, v_2, \cdots, v_c\}$ where $v_i = \pi_{k_1}(i),\ 1 \le i \le c$. Since $\pi_{k_1}(\cdot)$ is a pseudo-random permutation determined by the random k_1, the challenged c block-tag pairs come from the random selection of the n stored block-tag pairs.

Correctness: An ID-DPDP protocol must be workable and correct. That is, if the PKG, C, V and $\mathcal{P}$ are honest and follow the specified procedures, the response θ can pass V's checking. The correctness follows from below:

$$\begin{aligned} & e(T, g) \\ = \ & e(\textstyle\prod_{CS_i} T^{(i)}, g) \\ = \ & e(\textstyle\prod_{CS_i} \prod_{v_l \in \mathcal{M}_i} T_{v_l}^{f_{k_2}(l)}, g) \\ = \ & e(\textstyle\prod_{CS_i} \prod_{v_l \in \mathcal{M}_i} (h_l^{a_l} \prod_{j=1}^{s} u_j^{a_l F_{v_l j}}), g^{\sigma}) \\ = \ & e((\textstyle\prod_{i=1}^{c} h_i^{a_i}) \prod_{j=1}^{s} u_j^{\hat{F}_j}, RY^{H(ID,R)}) \end{aligned}$$

C. Performance analysis

First, we analyze the performance of our proposed ID-DPDP protocol from the computation and communication overhead. We compare our ID-DPDP protocol with the other up-to-date PDP protocols. On the other hand, our protocol does not suffer from resource-consuming certificate management which is require by the other existing protocols. Second, we analyze our proposed ID-DPDP protocol's properties of flexibility and verification. Third, we give the prototypal implementation of the proposed ID-DPDP protocol.

Computation: Suppose there are n message blocks which will be stored in $\hat{n}$ cloud servers. The block's sector number is s. The challenged block number is c. We will consider the computation overhead in the different phases. On the group $\mathcal{G}_1$, bilinear pairings, exponentiation, multiplication, and the hash function h_1 (the input may be large data, such as, 1G byte) contribute most computation cost. Compared with them, the hash function h, the operations on $\mathcal{Z}_q$ and $\mathcal{G}_2$ are faster, the hash function H can be done once for all. Thus, we do not consider the hash functions h and H, the operations on $\mathcal{Z}_q$ and $\mathcal{G}_2$. On the client, the computation cost mainly comes from the procedures of $TagGen$ and $Verification$ (*i.e.*, the phase 5 in the protocol Proof($\mathcal{P}$, C, V)). In the phase *TagGen*, the client

TABLE II
COMPARISON OF COMPUTATION COST

Protocols	TagGen	Verify	$\mathcal{P}$+C	Cert. Verify
Zhu [6]	(s+n(s+2))C_{exp}+nsC_{mul}	3C_e+(c+s)C_{exp}+(c+s-2)C_{mul}	$\hat{n}sC_e$+(3$\hat{n}$+c+2)C_{exp} +(2$\hat{n}$+c-1)C_{mul}	Yes
Zhu [21]	(s+2n)C_{exp}+nC_{mul}	3C_e+(c+s)C_{exp}+(c+s-2)C_{mul}	cC_{exp}+(c-1)C_{mul}	Yes
Barsoum [30]	n(s+1)C_{exp}+nsC_{mul}	2C_e+(c+s+1)C_{exp}+(c+s-1)C_{mul}	cC_{exp}+(c-1)C_{mul}	Yes
Our ID-DPDP	n(s+1)C_{exp}+nsC_{mul}+nsC_{h_1}	2C_e+(c+s+1)C_{exp}+(c+s)C_{mul}	cC_{exp}+(c-1)C_{mul}+csC_{h_1}	No

Notes: For the same file, h_1 can be used to generate less block-tag pairs which incur less computation cost.

performs $n(s+1)$ exponentiation, ns multiplication on $\mathcal{G}_1$, ns hash function h_1 and n hash function h. At the same time, for every file, the corresponding record ϕ_i is stored by the client and C. These stored metadata is small. In the phase *Proof*, in order to respond the challenge $chal = (c, k_1, k_2)$ and generate the response θ, $\mathcal{P}$ and the combiner C perform c exponentiation, $c-1$ multiplication on the group $\mathcal{G}_1$ and cs hash function h_1. In the verification of the response θ, V performs $c+s+1$ exponentiation, 2 pairings, $c+s$ multiplication on the group $\mathcal{G}_1$ and c hash function h. The exponentiation and multiplication operations are more efficient than pairing [26]. Based on the test presented in [27], [28], [29], a Tate pairing operation consumes about 10 ms on a platform with PIII 3.0 GHz, 30 ms on a platform with Pentium D 3.0 GHz and 170 ms on an iMote2 sensor working at 416 MHz. On the other hand, in 2012, Zhu et al. proposed the cooperative provable data possession for integrity in multicloud storage [6]. Almost at the same time, Zhu et al. proposed the dynamic audit services for outsourced storages in clouds [21]. In 2012, Barsoum et al. proposed a provable multi-copy data possession protocol [30]. These three PDP protocols are designed in the PKI. Thus, the certification verification is necessary when these three PDP protocols are performed. Compared with them, our proposed ID-DPDP scheme is more efficient in the computation cost. The computation comparison can be summarized in Table II. In Table II, C_{exp} denotes the time cost of exponentiation on the group $\mathcal{G}_1$; C_{mul} denotes the time cost of multiplication on the group $\mathcal{G}_1$; C_e denotes the time cost of bilinear pairing; C_{h_1} denotes the time cost of the hash function h_1. In other schemes, the sector must be in $\mathcal{Z}_q$. Our scheme only requires the hash function h_1's value lies in $\mathcal{Z}_q$. Thus, the hash function h_1 can be used to generate less block-tag pairs for the same file. Less block-tag pairs only incur less computation cost. It shows that our protocol can be implemented in mobile devices which have limited computation power.

Communication: National Bureau of Standards and ANSI X9 have determined the shortest key length requirements: RSA and DSA is 1024 bits, ECC is 160 bits [31]. Based on the requirements, we give our ID-DPDP protocol's communication overhead. In the phase Proof, the communication overhead mainly comes from the challenge $chal$ and response. The block-tag pairs are uploaded once and for all. After that, the phase Proof will be performed periodically. Thus, the communication overheads mainly come from the ID-DPDP queries and responses. Suppose there are n message blocks are stored in the CS. $\mathcal{G}_1$ and $\mathcal{G}_2$ have the same order q. In ID-DPDP query, the verifier sends the challenge $chal = (c, k_1, k_2)$ to combiner, *i.e.*, the communication overhead is $\log_2 n + 2\log_2 q$.

TABLE III
COMPARISON OF COMMUNICATION COST (BITS)

Protocols	Chal	Response	ID-Based
Zhu[6]	c($\log_2 n$+ $\log_2 q$)	1$\mathcal{G}_1$+1$\mathcal{G}_2$+s$\log_2 q$	No
Zhu[21]	c($\log_2 n + \log_2 q$)	1$\mathcal{G}_1$++s$\log_2 q$	No
Barsoum [30]	$\log_2 n$+2$\log_2 q$	1$\mathcal{G}_1$+ns$\log_2 q$	No
Our ID-DPDP	$\log_2 n$+2$\log_2 q$	1$\mathcal{G}_1$+s$\log_2 q$	Yes

In the response, the combiner responds 1 element in $\mathcal{G}_1$ and s elements in $\mathcal{Z}_q^*$ to the verifier. On the other hand, Zhu et al. [6], Zhu et al. [21] and Barsoum et al. [30] proposed three different provable data possession scheme. Compared with these three schemes, our ID-DPDP scheme is more efficient in the communication cost. The communication comparison can be summarized in Table III. In Table III, 1$\mathcal{G}_1$ denotes one element of $\mathcal{G}_1$ and 1$\mathcal{G}_2$ denotes one element of $\mathcal{G}_2$, .

Flexibility: In the ID-DPDP protocol, one block is split into s sectors. Then, the hash function h_1 will be used on these sectors. These hash values are shorter than the order q of $\mathcal{G}_1$. Denote $k = \log_2 q$. Let h_1 be $h_1 : \{0,1\}^{\tilde{k}} \rightarrow \{0,1\}^k$. This property makes our ID-DPDP protocol more flexible between storage overhead and computation overhead. For example, for the file F, the client divides it into n blocks which will be split into s sectors. The hash function h_1 will be used on these sectors. Thus, $\tilde{k}s = \frac{|F|}{n}$. In order to save the storage space, the client will divide F into less blocks because every block corresponds to one block-tag pair. Thus, $s = \frac{|F|}{n\tilde{k}}$ will increase when n decreases. The picked public parameters $u_j, 1 \leq j \leq s$ will become more. These parameters will incur more computation cost with less storage cost. And vice versa. When s decreases, n increases. The computation cost will decrease along with the storage cost increases. Thus, our ID-DPDP protocol has the flexibility to adjust the computation cost and storage cost.

Private verification, delegated verification and public verification: Our proposed ID-DPDP protocol satisfies the private verification and public verification. In the verification procedure, the metadata in the table T_{cl} and R are indispensable. Thus, it can only be verified by the client who has T_{cl} and R *i.e.*, it has the property of private verification. On some cases, the client has no ability to check its remote data integrity, for example, he takes part in the battle in the war. Thus, it will delegate the third party to perform the ID-DPDP protocol. The third party may be the third auditor or the proxy or other entities. The client will send T_{cl} and R to the consignee. The consignee can perform the ID-DPDP protocol. Thus, it has the property of delegated verification. On the other hand, if the client makes T_{cl} and R public, every entity can perform the ID-DPDP protocol by himself. Thus, it has also the property of public verification.

Notes: In the verification, the verifier must have T_{cl} and R. T_{cl} stores the metadata which has no any information on the private key (R, σ). On the other hand, even if R is made public, the private key σ still keeps secret. The private key extraction process $Extract$ is actually a modified ElGamal signature scheme which is existentially unforgeable. For the identity ID, the corresponding private key (R, σ) is a signature on ID. Since it is existentially unforgeable, the private key σ still keeps secret even if R is made public. Thus, the client can generate the tags for different files with the same private key σ even if it makes T_{cl} and R public.

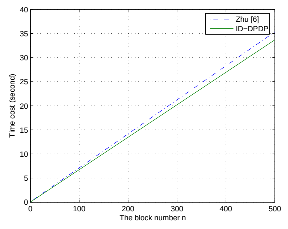

Fig. 3. Comparison on computation cost for TagGen based on s=20

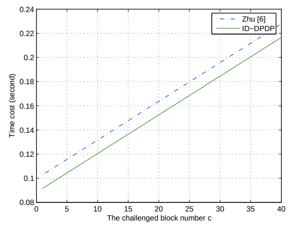

Fig. 4. Comparison on computation cost for Verify based on s=20

Simulation: In order to study its prototypal implementation, we have simulated the proposed ID-DPDP protocol by using C programming language with GMP Library (GMP-5.0.5) [33], Miracl Library [34] and PBC Library (pbc-0.5.13) [35]. In the simulation, CS works on an ASUS S56C Laptop with the following settings:

- CPU: Intel Core i7-3517U @ 1.90GHz
- Physical Memory: 4GB DDR3 1600MHz
- OS: Ubuntu 13.04 Linux 3.8.0-19-generic SMP i686

The client works on an PC Laptop with the following settings:

- CPU: CPU I PDC E6700 3.2GHz
- Physical Memory: DDR3 2G
- OS: Ubuntu 11.10 over VMware-workstation-full-8.0.0

In the simulation, we choose an elliptic curve with 160-bit group order, which provides a competitive security level with 1024-bit RSA. The certification verification scheme comes from the IEEE P1363 (Hybrid Schemes) [36].

In order to describe our ID-DPDP protocol's computation performance, we compare our ID-DPDP with Zhu's scheme [6]. Figure 3 depicts the comparison on computation cost for TagGen based on the sector number s=20. X-label denotes the block number n and Y-label denotes the time cost (second) in order to generate n tags. It is easily observed that our ID-DPDP protocol becomes more efficient along with the block number n increases. Figure 4 depicts the comparison on computation cost for Verify based on the same sector number s=20. X-label denotes the challenged block number c and Y-label denotes the time cost (second) in order to verify the response. Figure 5 depicts the comparison on CS computation cost for response based on the sector number s=20 and the CS number be $\hat{n} = 10$, *i.e.*, $|\mathcal{P}| = \hat{n}$. X-label denotes the challenged block number c and Y-label denotes the time cost (second) in order to create the response. They also show that our ID-DPDP scheme is more efficient than Zhu's scheme [6]. Then, we describe our ID-DPDP protocol's communication performance by comparing our ID-DPDP with Zhu's scheme [6]. Figure 6 depicts the comparison on communication cost for challenge based on the challenged block number c=10. X-label denotes the whole block number n and Y-label denotes the communication cost (bit) of challenge. Figure 7 depicts the comparison on communication cost for response. X-label denotes the sector number s and Y-label denotes the communication cost (bit) of response from the combiner. From Figures 6 and Figure 7, our ID-DPDP has shorter communication length.

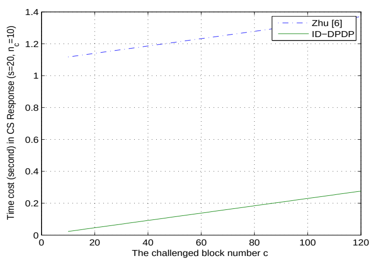

Fig. 5. Comparison on computation cost for CS's response based on s=20, $\hat{n} = 10$

In order to show the feasibility of our proposed scheme in concrete terms, we assume that the client will store 50 Tbyte data to CS. The order q of $\mathcal{G}_1$ and $\mathcal{G}_2$ is 160 bit. We take use of the hash function h which comes from the reference [37]. Let the hash functions h_1 be $SHA-1$ which maps 1G byte to 160 bit. Let the hash function H be also $SHA-1$. Let the sector number s be 1. Thus, 50 Tbyte data can be split into $50*1024$ blocks. The elliptic curve $\mathcal{G}_1$ is the Type A whose order is 160bit and point size is 128 byte [35]. In the phase *TagGen*, the whole computation cost is $50*1024*(2C_{exp}+C_{mul}+C_h+C_{h_1})$. Based on the 100 simulation, the average computation cost is 17.5535 hours. Now, there are 50*1024=101200 blocks are stored in CS. Suppose there are 1000 block-tag pairs are corrupted. If 230 block-tag pairs are challenged, the corrupted block-tag pairs can be detected with the probability at least 90% (Theorem 2). Thus, the challenge is $chal = (230, k_1, k_2)$. In the response, CS will perform $c = 230$ exponentiation and $c - 1 = 229$ multiplication on the group $\mathcal{G}_1$ (We omit the hash function h_1 since it can be pre-computed). The whole time cost is 0.5s. In the verification, the client will perform 2 pairings, $c+1 = 231$ exponentiation, $c+1 = 231$ addition on the group $\mathcal{G}_1$ and 1 hash function H. The whole time cost is 1.08s. On the other hand, we consider the communication cost. For 50 Tbyte data, the whole block-tag pairs are $50*1024^4 + 50*1024*128$ byte. The redundancy rate is $1.2*10^{-7}$. In the local area network with the bandwidth $1000Mb/s$, in order to upload these block-tag pairs, the time cost is 116.509 hours. The challenge size is $log_2(101200) + 160 + 160 = 337$ bit and the response size is $(1G + 128) * 8 = 8.6 * 10^9$ bit. The whole time cost is 8.2s. From the computation technology and communication technology, our proposed ID-DPDP scheme is feasible.

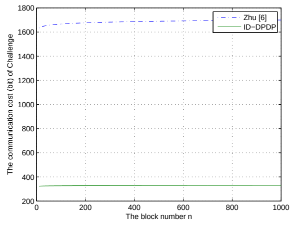

Fig. 6. Comparison on communication cost for Chal based on c=10

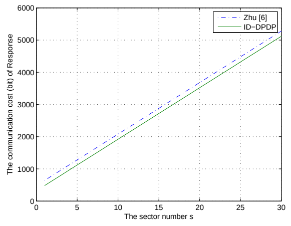

Fig. 7. Comparison on communication cost for Response

IV. Security Analysis

The security of our ID-DPDP protocol mainly consists of the correctness and unforgeability. The property of correctness has been shown in the subsection III-B. On the other hand, we study the universal unforgeability. It means that the attacker includes all the cloud servers $\mathcal{P}$ and the combiner C. If our ID-DPDP protocol is universally unforgeable, it can also prevent the collusion of malicious cloud servers $\mathcal{P}$ and C. Our proof comprises two parts: single block-tag pair is universally unforgeable; the response is universally unforgeable.

Definition 7: A forger $\mathcal{A}$ $(t, \epsilon, Q_H, Q_h, Q_{h_1}, Q_E, Q_T)$-breaks a single tag scheme if it runs in time at most t; $\mathcal{A}$ makes at most Q_H queries to the hash function H, at most Q_h queries to the hash function h, at most Q_{h_1} queries to the hash function h_1, at most Q_E queries to the extraction oracle $Extract$, at most Q_T queries to the tag oracle $TagGen$; and the probability that $\mathcal{A}$ forges a valid block-tag pair is as least ϵ. A single tag scheme is $(t, \epsilon, Q_H, Q_h, Q_{h_1}, Q_E, Q_T)$-existentially unforgeable under an adaptive chosen-message attack if no forger $(t, \epsilon, Q_H, Q_h, Q_{h_1}, Q_E, Q_T)$-breaks it.

The following Lemma 1 shows that the single tag scheme is secure.

Lemma 1: Let $(\mathcal{G}_1, \mathcal{G}_2)$ be a (t', ϵ')-GDH group pair of order q. Then the tag scheme on $(\mathcal{G}_1, \mathcal{G}_2)$ is $(t, \epsilon, Q_H, Q_h, Q_{h_1}, Q_E, Q_T)$-secure against existential forgery under an adaptive chosen-block attack (in the random oracle model) for all t and ϵ satisfying $\epsilon' \geq \frac{1}{9}$ and $t' \leq \frac{23(Q_H+Q_h)(t+O(Q_H)+O(Q_E)+O(Q_h)+O(Q_{h_1})+O(Q_T))}{\mu\delta(1-\mu)^{Q_E}(1-\delta)^{Q_T}\epsilon}$, where $0 < \mu, \delta < 1$. Based on the difficulty of the CDH problem, our single tag scheme is secure.

Proof: Let the stored index-block pair set be $\{(1, F_1), (2, F_2), \cdots, (n, F_n)\}$. For $1 \leq i \leq n$, each block F_i is split into s sectors, *i.e.*, $F_i = \{\tilde{F}_{i1}, \tilde{F}_{i2}, \cdots, \tilde{F}_{is}\}$. Then, the hash function h_1 is used on these sectors, *i.e.*, $F_{ij} = h_1(\tilde{F}_{ij})$. Then, we get the hash value set $\{F_{ij}\}$. Let the selected identity set be $\mathcal{I} = \{ID_1, ID_2, \cdots, ID_{\tilde{n}}\}$. We show how to construct a t'-algorithm $\mathcal{B}$ that solves CDH problem on $\mathcal{G}_1$ with probability at least ϵ'. This will contradict the fact that $(\mathcal{G}_1, \mathcal{G}_2)$ are GDH group pair.

Algorithm $\mathcal{B}$ is given $(g, g^a, g^b) \in \mathcal{G}_1^3$. Its goal is to output $g^{ab} \in \mathcal{G}_1$. Algorithm $\mathcal{B}$ simulates the challenger and interacts with forger $\mathcal{A}$ as follows.

Setup. Algorithm $\mathcal{B}$ starts by setting the master public key as $Y = g^a$ while a keeps unknown. It initializes the three tables Tab$_1$, Tab$_2$ and Tab$_3$. Then, it picks the two parameters μ and δ from the interval $(0, 1)$.

H-Oracle. At any time algorithm $\mathcal{A}$ can query the random oracle H. To respond to these queries, algorithm $\mathcal{B}$ maintains a list of tuples (ID_j, R_j, H_j) as explained below. We refer to this list as Tab$_1$. When $\mathcal{A}$ queries the oracle H at the pair (ID_j, R_j), algorithm $\mathcal{B}$ responds as follows:

1) If $(ID_j, R_j, *) \in$ Tab$_1$, then algorithm $\mathcal{B}$ retrieves the tuple (ID_j, R_j, H_j) and responds with H_j, *i.e.*, $H(ID_j, R_j) = H_j$.
2) Otherwise, $\mathcal{B}$ picks a random $H_j \in \mathcal{Z}_q$. Then, it adds the tuple (ID_j, R_j, H_j) to Tab$_1$ and responds to $\mathcal{A}$ by setting $H(ID_j, R_j) = H_j$.

Extract-Oracle. At any time algorithm $\mathcal{A}$ can query the oracle $Extract$. To respond to these queries, algorithm $\mathcal{B}$ maintains a list of tuples $(c_i, ID_i, R_i, H_i, \sigma_i)$ as explained below. We refer to this list as Tab$_2$. When $\mathcal{A}$ queries the oracle $Extract$ at the identity $ID_i \in \mathcal{I}$, algorithm $\mathcal{B}$ picks a bit $c_i \in \{0, 1\}$ according to the bivariate distribution function: $\Pr[c_i = 0] = \mu, \ \Pr[c_i = 1] = 1 - \mu$. Based on c_i, $\mathcal{B}$ responds as follows:

1) If $c_i = 1$, $\mathcal{B}$ independently picks the random $\sigma_i, H_i \in \mathcal{Z}_q$ and calculates $R_i = g^{\sigma_i} \cdot Y^{-H_i}$.
   a) If $(ID_i, R_i, *) \notin$ Tab$_1$, then algorithm $\mathcal{B}$ adds the tuple (ID_i, R_i, H_i) to Tab$_1$ and the tuple $(c_i, ID_i, R_i, H_i, \sigma_i)$ to Tab$_2$.
   b) If $(ID_i, R_i, *) \in$ Tab$_1$, then algorithm $\mathcal{B}$ picks the other random $\sigma_i, H_i \in \mathcal{Z}_q$ and calculates $R_i = g^{\sigma_i} \cdot Y^{-H_i}$ until $(ID_i, R_i, *) \notin$ Tab$_1$. Then, algorithm $\mathcal{B}$ adds the tuple (ID_i, R_i, H_i) to Tab$_1$ and the tuple $(c_i, ID_i, R_i, H_i, \sigma_i)$ to Tab$_2$.
   c) Finally, algorithm $\mathcal{B}$ responds with (R_i, σ_i).
2) If $c_i = 0$, algorithm $\mathcal{B}$ independently picks a random $R_i \in \mathcal{G}_1$ and calculates $H_i = H(ID_i, R_i)$ which satisfies $(ID_i, R_i, *) \notin$Tab$_1$ (otherwise, picks another R_i and calculates $H(ID_i, R_i)$). Then, $\mathcal{B}$ adds the tuple (ID_i, R_i, H_i) to Tab$_1$ and the tuple $(c_i, ID_i, R_i, H_i, \sigma_i)$ to Tab$_2$, where $\sigma_i = \perp$. $\mathcal{B}$ fails.

h-Oracle. At any time algorithm $\mathcal{A}$ can query the random oracle h. To respond to these queries, algorithm $\mathcal{B}$ maintains a list of tuples $(N_i, CS_{l_i}, i, z_i, b_i, d_i)$ as explained below. We refer to this list as Tab$_3$. When $\mathcal{A}$ queries the oracle h at the tuple (N_i, CS_{l_i}, i), $\mathcal{B}$ responds as follows:

1) If $(N_i, CS_{l_i}, i, z_i, b_i, d_i) \in$ Tab$_3$ and $1 \leq i \leq n$, then algorithm $\mathcal{B}$ responds with $z_i \prod_{j=1}^{s} u_j^{-F_{ij}}$, *i.e.*, $h(N_i, CS_{l_i}, i) = z_i \prod_{j=1}^{s} u_j^{-F_{ij}}$.
2) Otherwise, algorithm $\mathcal{B}$ picks a random $b_i \in \mathcal{Z}_q^*$ and a random coin $d_i \in \{0, 1\}$ according to the bivariate distribution $\Pr[d_i = 0] = \delta, \quad \Pr[d_i = 1] = 1 - \delta$.
   a) If $d_i = 1$, algorithm $\mathcal{B}$ calculates $z_i = g^{b_i}$. If $d_i = 0$, algorithm $\mathcal{B}$ calculates $z_i = (g^b)^{b_i}$.
   b) Algorithm $\mathcal{B}$ adds the tuple $(N_i, CS_{l_i}, i, z_i, b_i, d_i)$ to Tab$_3$ and responds with $z_i \prod_{j=1}^{s} u_j^{-F_{ij}}$, *i.e.*, $h(N_i, CS_{l_i}, i) = z_i \prod_{j=1}^{s} u_j^{-F_{ij}}$.

h_1-Oracle. Upon receiving the query $\hat{F}$, $\mathcal{B}$ takes a random value from $\mathcal{Z}_q$ and responds it to $\mathcal{A}$. It is a real hash function.

TagGen-Oracle. Let (N_i, CS_{l_i}, i, F_i) be a tag query issued by $\mathcal{A}$. Algorithm $\mathcal{B}$ responds to this query as follows:

1) Algorithm $\mathcal{B}$ runs the above algorithm for responding to h-queries to obtain $h(N_i, CS_{l_i}, i)$. Let $(N_i, CS_{l_i}, i, z_i, b_i, c_i)$ be the corresponding tuple in Tab$_3$. If $d_i = 0$, then algorithm $\mathcal{B}$ reports failure and terminates.
2) If $d_i = 1$, define $\sigma_i = (g^a)^{b_i}$. Observe that $T_i = (g^a)^{b_i} = (g^{b_i})^a = (h(N_i, CS_{l_i}, i)\prod_{j=1}^{s} u_j^{F_{ij}})^a$ and therefore T_i is a valid tag on F_i under the public key $(g^a, u_1, \cdots, u_s)$. Algorithm $\mathcal{B}$ gives T_i to algorithm $\mathcal{A}$.

Output. Eventually, on behalf of the client ID whose private key is (R, σ), algorithm $\mathcal{A}$ produces a block-tag pair (F_w, T_w) such that no tag query was issued for (N_w, CS_{l_w}, w, F_w). If $(N_w, CS_{l_w}, w, *) \notin$Tab$_3$, $\mathcal{B}$ issues a query itself for $h(N_w, CS_{l_w}, w)$ to ensure that such a tuple exists. We assume T_w is a valid tag on F_w under the given public key; if it is not, $\mathcal{B}$ reports failure and terminates. Next, $\mathcal{B}$ finds the tuple $(N_w, CS_{l_w}, j, z_w, b_w, d_w)$ in Tab$_3$. If $d_w = 1$, $\mathcal{B}$ reports failure and terminates. Otherwise, $d_w = 0$, (F_w, T_w) satisfies the following verification equation

$$e(T_w, g) = e(h(N_w, CS_{l_w}, w)\prod_{j=1}^{s} u_j^{h_1(F_{wj})}, RY^{H(ID,R)})$$

This completes the description of algorithm $\mathcal{B}$. It remains to show that $\mathcal{B}$ solves the given instance of the CDH problem with a high probability.

Breaking CDH: Based on the oracle-replay technique [32], $\mathcal{B}$ can get another block-tag pair $(F_w, \hat{T}_w)$ on the same block F_w by using different hash functions $\hat{h}$ and $\hat{H}$. Then, we can get $e(\hat{T}_w, g) = e(\hat{h}(N_w, CS_{l_w}, w)\prod_{j=1}^{s} u_j^{h_1(F_{wj})}, RY^{H(ID,R)})$.

According to the simulation, $\mathcal{B}$ knows the corresponding $b_w, \hat{b}_w$ that satisfy

$$h(N_w, CS_{l_w}, w) = (g^b)^{b_w}\prod_{j=1}^{s} u_j^{-h_1(F_{wj})}$$

$$\hat{h}(N_w, CS_{l_w}, w) = (g^b)^{\hat{b}_w}\prod_{j=1}^{s} u_j^{-h_1(F_{wj})}$$

Thus, we can get

$$e(T_w, g) = e((g^b)^{b_w}, RY^{H(ID,R)})$$

$$e(\hat{T}_w, g) = e((g^b)^{\hat{b}_w}, RY^{\hat{H}(ID,R)})$$

$$e(T_w\hat{T}_w^{\frac{-b_w}{\hat{b}_w}}, g) = e((g^b)^{b_w}, Y^{H(ID,R)-\hat{H}(ID,R)})$$

Finally, we get

$$g^{ab} = (T_w\hat{T}_w^{\frac{-b_w}{\hat{b}_w}})^{\frac{1}{b_w(H(ID,R)-\hat{H}(ID,R))}}$$

Probability analysis. To evaluate the success probability for algorithm $\mathcal{B}$, we analyze the four events needed for $\mathcal{B}$ to succeed:

$\mathcal{E}_1$: $\mathcal{B}$ does not abort as a result of any of $\mathcal{A}$'s Extract queries.

$\mathcal{E}_2$: $\mathcal{B}$ does not abort as a result of any of $\mathcal{A}$'s tag queries.

$\mathcal{E}_3$: $\mathcal{A}$ generates a valid block-tag forgery (F_w, T_w).

$\mathcal{E}_4$: Event $\mathcal{E}_3$, $c_w = 0$ for the tuple containing F_w in Tab$_2$ and $d_w = 0$ for the tuple containing F_w in Tab$_3$.

$\mathcal{B}$ succeeds if all of these events happen. The probability $\Pr[\mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \mathcal{E}_3 \wedge \mathcal{E}_4]$ decomposes as

$$\begin{aligned} & \Pr[\mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \mathcal{E}_3 \wedge \mathcal{E}_4] \\ = \ & \Pr[\mathcal{E}_1]\Pr[\mathcal{E}_2|\mathcal{E}_1]\Pr[\mathcal{E}_3|\mathcal{E}_1,\mathcal{E}_2]\Pr[\mathcal{E}_4|\mathcal{E}_1,\mathcal{E}_2,\mathcal{E}_3] \\ = \ & (1-\mu)^{Q_E}\cdot(1-\delta)^{Q_T}\cdot\epsilon\cdot\mu\cdot\delta \end{aligned}$$

Thus, in $\mathcal{B}$'s simulation environment, $\mathcal{A}$ can forge a valid block-tag pair with probability

$$\hat{\epsilon} \geq P_1\epsilon = \mu\delta(1-\mu)^{Q_E}(1-\delta)^{Q_T}\epsilon$$

. Algorithm $\mathcal{B}$'s running time is the same as $\mathcal{A}$'s running time plus the time which is taken to respond to Q_H H queries, Q_h h queries, Q_{h_1} h_1 queries, Q_E $Extract$ queries and Q_T tag queries. Hence, the total running time is at most $\hat{t} \leq t + O(Q_H) + O(Q_E) + O(Q_h) + O(Q_{h_1}) + O(Q_T)$. By taking use of the oracle replay technique [32], $\mathcal{A}$ can get two different tags on the same block and randomness with the probability $\epsilon' \geq \frac{1}{9}$ within the time $t' \leq 23(Q_H + Q_h)\hat{t}\hat{\epsilon}^{-1}$, *i.e.*, $t' \leq \frac{23(Q_H+Q_h)(t+O(Q_H)+O(Q_E)+O(Q_h)+O(Q_{h_1})+O(Q_T))}{\mu\delta(1-\mu)^{Q_E}(1-\delta)^{Q_T}\epsilon}$. After obtaining the two different tags on the same block and randomness, algorithm $\mathcal{B}$ can break the CDH problem. It contradicts the assumption that $(\mathcal{G}_1, \mathcal{G}_2)$ is a GDH group pair. This completes the proof. ■

Lemma 1 states that the untrusted CS has no ability to forge the single tag. But, can the untrusted CS forge the aggregated block-tag pair to cheat the verifier ? First, when all the stored block-tag pairs are challenged, we study whether the untrusted CS can aggregate the fake block-tag pairs to cheat the verifier. It corresponds to Lemma 2. Second, when some stored block-tag pairs are not challenged, we study whether the untrusted CS can substitute the valid unchallenged block-tag pairs for the modified block-tag pairs to cheat the verifier. It corresponds to Lemma 3.

Lemma 2: If all the stored block-tag pairs are challenged and some block-tag pairs are modified, the malicious CS aggregates the fake block-tag pairs which are different from the challenged block-tag pairs. The combined block-tag pair $(\hat{F}, T)$ (*i.e.*, response) only can pass the verification with negligible probability.

Proof: We can use proof by contradiction to prove Lemma 2. For the challenge $chal = (c, k_1, k_2)$, the following parameters can be calculated

$$v_i = \pi_{k_1}(i),\ h_i = h(N_{v_i}, CS_{l_{v_i}}, v_i),\ a_i = f_{k_2}(i)$$

Assume the combined block-tag pair $(\hat{F}, T)$ is forged and it can pass the verification, where $\hat{F} = (\hat{F}_1, \cdots, \hat{F}_s)$, *i.e.*,

$$\begin{aligned} e(T, g) &= e(\textstyle\prod_{i=1}^{c} h_i^{a_i}\prod_{j=1}^{s} u_j^{\hat{F}_j}, RY^{H(ID,R)}) \\ &= e(\textstyle\prod_{i=1}^{c} h_i^{a_i}\prod_{j=1}^{s} u_j^{\sum_{i=1}^{c} a_i\hat{F}_{v_ij}}, RY^{H(ID,R)}) \end{aligned}$$

Then,

$$\prod_{i=1}^{c} e(T_{v_i}, g)^{a_i} = \prod_{i=1}^{c} e(h_i\prod_{j=1}^{s} u_j^{\hat{F}_{v_ij}}, RY^{H(ID,R)})^{a_i}$$

Let d be a generator of $\mathcal{G}_2$. There exist $x_i, y_i \in \mathcal{Z}_q$ with the following properties:

$$e(T_{v_i}, g) = d^{x_i}, \quad e(h_i \prod_{j=1}^{s} u_j^{\hat{F}_{v_i j}}, RY^{H(ID,R)}) = d^{y_i}$$

We can get

$$d^{\sum_{i=1}^{c} a_i x_i} = d^{\sum_{i=1}^{c} a_i y_i}$$

$$\sum_{i=1}^{c} a_i(x_i - y_i) = 0 \tag{3}$$

Since the single tag is existentially unforgeable (Lemma 1), there exist at least two different indices i such that $x_i \neq y_i$. Suppose there exist $\bar{s} \leq c$ index pairs (x_i, y_i) with the property $x_i \neq y_i$. Then, there exist $q^{\bar{s}-1}$ tuples $(a_1, a_2, \cdots, a_c)$ which satisfy the above equation (3). Since $(a_1, a_2, \cdots, a_c)$ is a random vector, the equation (3) holds only with the probability less than $q^{\bar{s}-1}/q^c \leq q^{c-1}/q^c = q^{-1}$. It is negligible. Thus, if some fake block-tag pairs are aggregated, the combined block-tag pair $(\hat{F}, T)$ only can pass the verification with negligible probability. ■

When some modified block-tag pairs are challenged and some valid block-tag pairs are unchallenged, the malicious CS still can not cheat the verifier.

Lemma 3: If some challenged block-tag pairs are modified, the malicious CS substitutes the other valid block-tag pairs (which are not challenged) for the modified block-tag pairs (which are challenged). The combined block-tag pair $(\hat{F}, T)$ (*i.e.*, response) only can pass the verification with negligible probability.

Proof: Define the modified block-tag pair index set as $\mathcal{M}$ and the valid block-tag index set as $\bar{\mathcal{M}}$. Let the verifier's challenge be $chal = (c, k_1, k_2)$ and $\mathcal{S} = \{\pi_{k_1}(1), \cdots, \pi_{k_1}(c)\}$. For the CS set $\mathcal{P}$, suppose some challenged block-tag pairs $\{(F_l, T_l), l \in \mathcal{S}_1 \subseteq \mathcal{S}\}$ are modified, *i.e.* $\mathcal{S}_1 = \mathcal{S} \bigcap \mathcal{M} \neq \Phi$. Φ denotes the empty set. The corresponding CSs (whose modified block-tag pairs are challenged) substitute the other valid block-tag pairs $\{(F_{\hat{l}}, T_{\hat{l}}), \hat{l} \in \mathcal{S}_2 \subseteq \bar{\mathcal{M}}\}$ (which are not challenged) for $\{(F_l, T_l), l \in \mathcal{S}_1\}$ where $|\mathcal{S}_1| = |\mathcal{S}_2|$. By making use of the forged $\theta_i = (\hat{F}^{(i)}, T^{(i)})$, the combined response $\theta = (\hat{F}, T)$ only can pass verification with negligible probability.

If the challenged block-tag pairs $(F_l, T_l), l \in \mathcal{S}_1$ are modified, $\mathcal{P}$ substitutes the other valid block-tag pairs $(F_{\hat{l}}, T_{\hat{l}})$ for them. For $1 \leq i \leq c$, the following parameters are calculated

$$a_i = f_{k_2}(i), \; v_i = \pi_{k_1}(i), \; h_i = h(N_{v_i}, CS_{l_{v_i}}, v_i)$$

Based on the forgery process, we know the forged response $\theta = (\hat{F}, T)$ satisfies the formulas below

$$T = \textstyle\prod_{v_i \in \mathcal{S} \bigcap \bar{\mathcal{M}}} T_{v_i}^{a_i} \prod_{v_i \in \mathcal{S}_2} T_{v_i}^{a_i}$$
$$\hat{F}_j = \textstyle\sum_{v_i \in S \bigcap \bar{\mathcal{M}}} a_i F_{v_i j} + \sum_{v_i \in \mathcal{S}_2} a_i F_{v_i j}, \; 1 \leq j \leq s$$

where $\hat{F} = (\hat{F}_1, \hat{F}_2, \cdots, \hat{F}_s)$.

If the forged response θ can pass the verification, then

$$e(T, g) = e(\textstyle\prod_{i=1}^{c} h_i^{a_i} \prod_{j=1}^{s} u_j^{\hat{F}_j}, RY^{H(ID,R)})$$

i.e.,

$$\begin{aligned} & e(\prod_{v_i \in \mathcal{S} \bigcap \bar{\mathcal{M}}} T_{v_i}^{a_i} \prod_{v_i \in \mathcal{S}_2} T_{v_i}^{a_i}, g) \\ =& e((\prod_{v_i \in \mathcal{S} \bigcap \bar{\mathcal{M}}} h_i^{a_i} \prod_{j=1}^{s} u_j^{\sum_{v_i \in S \cap \bar{\mathcal{M}}} a_i F_{v_i j}}) \cdot (\prod_{v_i \in \mathcal{S}_1} h_i^{a_i} \\ & \prod_{j=1}^{s} u_j^{\sum_{v_i \in \mathcal{S}_2} a_i F_{v_i j}}), RY^{H(ID,R)}) \end{aligned} \tag{4}$$

For the index in the set $\mathcal{S} \bigcap \bar{\mathcal{M}}$, we can get

$$\begin{aligned} & e(\prod_{v_i \in \mathcal{S} \bigcap \bar{\mathcal{M}}} T_{v_i}^{a_i}, g) \\ =& e(\prod_{v_i \in \mathcal{S} \bigcap \bar{\mathcal{M}}} h_i^{a_i} \prod_{j=1}^{s} u_j^{\sum_{v_i \in S \cap \bar{\mathcal{M}}} a_i F_{v_i j}}, RY^{H(ID,R)}) \end{aligned} \tag{5}$$

According to the formulas (4)(5), we can get the formula below

$$e(\prod_{v_i \in \mathcal{S}_2} T_{v_i}^{a_i}, g) = e(\prod_{v_i \in \mathcal{S}_1} h_i^{a_i} \prod_{j=1}^{s} u_j^{\sum_{v_i \in \mathcal{S}_2} a_i F_{v_i j}}, RY^{H(ID,R)}) \tag{6}$$

On the other hand, we can get the following formula by making use of the tag scheme

$$e(\prod_{v_i \in \mathcal{S}_2} T_{v_i}^{a_i}, g) = e(\prod_{v_i \in \mathcal{S}_2} h_i^{a_i} \prod_{j=1}^{s} u_j^{\sum_{v_i \in \mathcal{S}_2} a_i F_{v_i j}}, RY^{H(ID,R)}) \tag{7}$$

From the formulas (6)(7), we can get

$$\prod_{v_i \in \mathcal{S}_1} h_i^{a_i} = \prod_{v_i \in \mathcal{S}_2} h_i^{a_i} \tag{8}$$

Since h is a cryptographic hash function which is collision-free, the probability that the equation (8) holds is $\frac{1}{q}$, *i.e.*, it is negligible. The combined block-tag pair $(\hat{F}, T)$ (*i.e.*, response) only can pass the verification with negligible probability. ■

Thus, from the above three lemmas (Lemma 1, Lemma 2, Lemma 3), we have the following claim.

Theorem 1: The proposed ID-DPDP protocol is existentially unforgeable in the random oracle model if the CDH problem on $\mathcal{G}_1$ is hard.

Proof: Let the stored index-block pair set be $\{(1, F_1), (2, F_2), \cdots, (n, F_n)\}$ and the CS set be $\mathcal{P}$. For $1 \leq i \leq n$, each block F_i is split into s sectors, *i.e.*, $F_i = \{\tilde{F}_{i1}, \tilde{F}_{i2}, \cdots, \tilde{F}_{is}\}$. Then, the hash function h_1 is used on these sectors, *i.e.*, $F_{ij} = h_1(\tilde{F}_{ij})$. Then, we get the hash value set $\{F_{ij}\}$. Let the selected identity set be $\mathcal{I} = \{ID_1, ID_2, \cdots, ID_{\tilde{n}}\}$. Define the challenger as $\mathcal{B}$ and the adversary as $\mathcal{A}$. Algorithm $\mathcal{B}$ is given $(g, g^a, g^b) \in \mathcal{G}_1$. Its goal is to output $g^{ab} \in \mathcal{G}_1$.

The Setup, Extract-Oracle, H-Oracle, h-Oracle, h_1-Oracle, TagGen-Oracle procedures are the same as the corresponding procedures in the proof of Lemma 1.

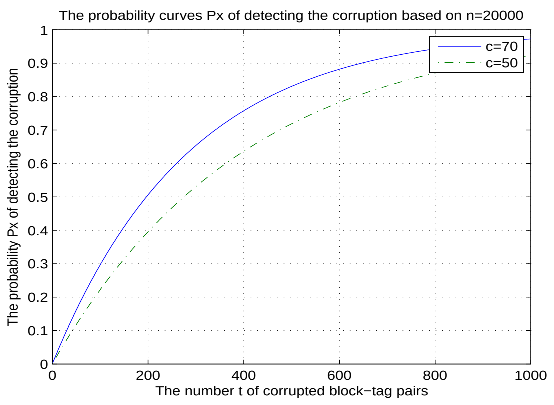

Fig. 8. The probability curve P_X of detecting the corruption

Let the challenge be $chal = (c, k_1, k_2)$ and the forged response be $\theta = (\hat{F}, T)$ where $\hat{F} = (\hat{F}_1, \hat{F}_2, \cdots, \hat{F}_s)$. Assume that θ can pass the verification, *i.e.*,

$$e(T, g) = e\left(\prod_{i=1}^{c} h_i^{a_i} \prod_{j=1}^{s} u_j^{\hat{F}_j}, RY^{H(ID,R)}\right) \quad (9)$$

where $h_i = h(N_{\pi_{k_1}(i)}, CS_{l_{\pi_{k_1}(i)}}, \pi_{k_1}(i))$, $a_i = f_{k_2}(i)$, $1 \leq i \leq c$.

According to Lemma 1 and Lemma 2, if some challenged block-tag pairs are modified, the verification formula (9) only holds with negligible probability. Since the verifier does not store block-tag pairs, $\mathcal{A}$ can substitute the other valid block-tag pairs for the modified block-tag pairs. According to Lemma 3, the substituted response only can pass the verification with negligible probability, *i.e.*, the formula (9) only holds with negligible probability.

Thus, in the random oracle, except with negligible probability no adversary can cause the verifier to accept the challenged response if some challenged block-tag pairs are modified. Theorem 1 is proved. ■

Theorem 2: Suppose that n block-tag pairs are stored, $\bar{d}$ block-tag pairs are modified and c block-tag pairs are challenged. Then, our proposed ID-DPDP protocol is $(\frac{\bar{d}}{n}, 1 - (\frac{n-\bar{d}}{n})^c)$-secure, *i.e.*,

$$1 - (\frac{n-\bar{d}}{n})^c \leq P_X \leq 1 - (\frac{n-c+1-\bar{d}}{n-c+1})^c$$

where P_X denotes the probability of detecting the modification.

Proof: Let the challenged block-tag pair set be $\mathcal{S}$ and the modified block-tag pair set be $\mathcal{M}$. Let the random variable X be $X = |\mathcal{S} \bigcap \mathcal{M}|$. According to the definition of P_X, we can get

$$\begin{aligned} P_X &= P\{X \geq 1\} \\ &= 1 - P\{X = 0\} \\ &= 1 - \frac{n-\bar{d}}{n} \frac{n-1-\bar{d}}{n-1} \cdot \cdots \cdot \frac{n-c+1-\bar{d}}{n-c+1} \end{aligned}$$

Thus, we can get

$$1 - (\frac{n-\bar{d}}{n})^c \leq P_X \leq 1 - (\frac{n-c+1-\bar{d}}{n-c+1})^c$$

This completes the proof of the theorem. ■

Figure 8 shows the probability curve of P_X. From the picture, we know that our protocol has high integrity checking probability.

V. Conclusion

In multi-cloud storage, this paper formalizes the ID-DPDP system model and security model. At the same time, we propose the first ID-DPDP protocol which is provably secure under the assumption that the CDH problem is hard. Besides of the elimination of certificate management, our ID-DPDP protocol has also flexibility and high efficiency. At the same time, the proposed ID-DPDP protocol can realize private verification, delegated verification and public verification based on the client's authorization.

References

[1] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, D. Song, "Provable Data Possession at Untrusted Stores", *CCS'07*, pp. 598-609, 2007.
[2] G. Ateniese, R. DiPietro, L. V. Mancini, G. Tsudik, "Scalable and Efficient Provable Data Possession", *SecureComm 2008*, 2008.
[3] C. C. Erway, A. Kupcu, C. Papamanthou, R. Tamassia, "Dynamic Provable Data Possession", *CCS'09*, pp. 213-222, 2009.
[4] F. Sebé, J. Domingo-Ferrer, A. Martínez-Ballesté, Y. Deswarte, J. Quisquater, "Efficient Remote Data Integrity checking in Critical Information Infrastructures", *IEEE Transactions on Knowledge and Data Engineering*, 20(8), pp. 1-6, 2008.
[5] H.Q. Wang, "Proxy Provable Data Possession in Public Clouds," *IEEE Transactions on Services Computing*, 2012. http://doi.ieeecomputersociety.org/10.1109/TSC.2012.35
[6] Y. Zhu, H. Hu, G.J. Ahn, M. Yu, "Cooperative Provable Data Possession for Integrity Verification in Multicloud Storage", *IEEE Transactions on Parallel and Distributed Systems*, 23(12), pp. 2231-2244, 2012.
[7] Y. Zhu, H. Wang, Z. Hu, G. J. Ahn, H. Hu, S. S. Yau, "Efficient Provable Data Possession for Hybrid Clouds", *CCS'10*, pp. 756-758, 2010.
[8] R. Curtmola, O. Khan, R. Burns, G. Ateniese, "MR-PDP: Multiple-Replica Provable Data Possession", *ICDCS'08*, pp. 411-420, 2008.
[9] A. F. Barsoum, M. A. Hasan, "Provable Possession and Replication of Data over Cloud Servers", CACR, University of Waterloo, Report2010/32,2010. Available at http://www.cacr.math.uwaterloo.ca/techreports /2010/cacr2010-32.pdf.
[10] Z. Hao, N. Yu, "A Multiple-Replica Remote Data Possession Checking Protocol with Public Verifiability", *2010 Second International Symposium on Data, Privacy, and E-Commerce*, pp. 84-89, 2010.
[11] A. F. Barsoum, M. A. Hasan, "On Verifying Dynamic Multiple Data Copies over Cloud Servers", *IACR eprint report 447, 2011*. Available at http://eprint.iacr.org/2011/447.pdf.
[12] A. Juels, B. S. Kaliski Jr., "PORs: Proofs of Retrievability for Large Files", *CCS'07*, pp. 584-597, 2007.
[13] H. Shacham, B. Waters, "Compact Proofs of Retrievability", *ASIACRYPT 2008*, LNCS 5350, pp. 90-107, 2008.
[14] K. D. Bowers, A. Juels, A. Oprea, "Proofs of Retrievability: Theory and Implementation", *CCSW'09*, pp. 43-54, 2009.
[15] Q. Zheng, S. Xu. Fair and Dynamic Proofs of Retrievability. *CODASPY'11*, pp. 237-248, 2011.
[16] Y. Dodis, S. Vadhan, D. Wichs, "Proofs of Retrievability via Hardness Amplification", *TCC 2009*, LNCS 5444, pp. 109-127, 2009.
[17] Y. Zhu, H. Wang, Z. Hu, G. J. Ahn, H. Hu, "Zero-Knowledge Proofs of Retrievability", *Sci China Inf Sci*, 54(8), pp. 1608-1617, 2011.
[18] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing", *INFOCOM 2010*, IEEE, March 2010.
[19] Q. Wang, C. Wang, K. Ren, W. Lou, J. Li, "Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing", *IEEE Transactions on Parallel And Distributed Systems* , 22(5), pp. 847-859, 2011.
[20] C. Wang, Q. Wang, K. Ren, N. Cao, W. Lou, "Toward Secure and Dependable Storage Services in Cloud Computing," *IEEE Transactions on Services Computing*, 5(2), pp. 220-232, 2012.

[21] Y. Zhu, G.J. Ahn, H. Hu, S.S. Yau, H.G. An, S. Chen, "Dynamic Audit Services for Outsourced Storages in Clouds," *IEEE Transactions on Services Computing*, 2011. http://doi.ieeecomputersociety.org/10.1109/TSC.2011.51

[22] O. Goldreich, "Foundations of Cryptography: Basic Tools", Publishing House of Electronics Industry, Beijing, 2003, pp. 194-195.

[23] D. Boneh, M. Franklin, "Identity-based Encryption from the Weil Pairing", *CRYPTO 2001*, LNCS 2139, 2001, 213-229.

[24] A. Miyaji, M. Nakabayashi, S. Takano "New Explicit Conditions of Elliptic Curve Traces for FR-reduction", *IEICE Transactions Fundamentals*, 5, pp. 1234-1243, 2001.

[25] D. Boneh, B. Lynn, H. Shacham, "Short Signatures from the Weil Pairing", *ASIACRYPT 2001*, LNCS 2248, pp. 514-532, 2001.

[26] H. W. Lim, "On the Application of Identity-based Cryptography in Grid Security", *Ph.D. dissertation*, University of London, London, U.K., 2006.

[27] S. Yu, K. Ren, W. Lou, "FDAC: Toward Fine-grained Distributed Data Access Control in Wireless Sensor Networks", *IEEE Trans. Parallel Distrib. Syst.*, 22(4), pp. 673-686, 2011.

[28] S. Yu, K. Ren, W. Lou, "Attribute-based On-demand Multicast Group Setup with Membership Anonymity", *Calculater Networks*, 54(3), pp. 377-386, 2010.

[29] P. S. L. M. Barreto, B. Lynn, M. Scott, "Efficient Implementation of Pairing-based Cryptosystems", *Journal of Cryptology*, 17(4), pp. 321-334, 2004.

[30] A. F. Barsoum, M. A. Hasan, "Integrity Verification of Multiple Data Copies over Untrusted Cloud Servers," *12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*, pp. 829-834, 2012.

[31] C. Research, "SEC 2: Recommended Elliptic Curve Domain Parameters", http://www.secg.org/collateral/sec_final.pdf

[32] D. Pointcheval, J. Stern, "Security Arguments for Digital Signatures and Blind Signatures", *Journal of Cryptology*, 13(3), pp. 361-396, 2000.

[33] The GNU Multiple Precision Arithmetic Library (GMP). Available: http://gmplib.org/.

[34] Multiprecision Integer and Rational Arithmetic C/C++ Library (MIRACL). Available: http://certivox.com/.

[35] The Pairing-Based Cryptography Library (PBC). Available: http://crypto.stanford.edu/pbc/howto.html.

[36] IEEE P1363: Hybrid Schemes. Available: http://grouper.ieee.org/groups/1363/StudyGroup/Hybrid.html

[37] B. Lynn, "On the Implementation of Pairing-Based Cryptosystems", *Ph.D. dissertation*, http://crypto.stanford.edu/pbc/thesis.pdf, Stanford University, 2008.