Proxy Provable Data Possession with General Access Structure in Public Clouds

Huaqun Wang[1,2](✉) and Debiao He[3]

[1] Dalian Ocean University, Dalian, China
wanghuaqun@aliyun.com
[2] State Key Laboratory of Cryptology, Beijing, China
[3] State Key Lab of Software Engineering Computer School,
Wuhan University, Wuhan, China

Abstract. Since public clouds are untrusted by many consumers, it is important to check whether their remote data keeps intact. Sometimes, it is necessary for many clients to cooperate to store their data in the public clouds. For example, a file needs many clients' approval before it is stored in the public clouds. Specially, different files need different client subsets' approval. After that, these stored remote data will be proved possession by the verifier. In some cases, the verifier has no ability to perform remote data possession proof, for example, the verifier is in the battlefield because of the war. It will delegate this task to its proxy. In this paper, we propose the concept of proxy provable data possession (PPDP) which supports a general access structure. We propose the corresponding system model, security model and a concrete PPDP protocol from n-multilinear map. Our concrete PPDP protocol is provably secure and efficient by security analysis and performance analysis. Since our proposed PPDP protocol supports the general access structure, only the clients of an authorized subset can cooperate to store the massive data to PCS (Public Cloud Servers), and it is impossible for those of an unauthorized subset to store the data to PCS.

Keywords: Cloud computing · Provable data possession · Proxy cryptography · Access control

1 Introduction

Cloud computing is an emerging technology where the client can rent the storage and computing resource of cloud computing servers. The client only needs a terminal device, such as smart phone, tablet, *etc.* Cloud computing servers have huge storage space and strong computation capability. In order to apply for

H. Wang was partly supported by the Natural Science Foundation of China through projects (61272522, 61572379,61501333), by the Program for Liaoning Excellent Talents in University through project (LR2014021), and by the Natural Science Foundation of Liaoning Province (2014020147).

DOI: 10.1007/978-3-319-38898-4_17

data storing or remote computing, the end clients can access cloud computing servers via a web browser or a light weight desktop or mobile application, *etc.* In cloud computing, cloud servers can provide three types service: Infrastructure as a Service, Platform as a Service and Application as a Service. The end nodes are some capacity-limited electronic facilities, for example, personal computer, tablet, remote desktop, mini-note, mobile. These end nodes can access the cloud computing networking to get computing service by via a web browser, *etc.*

Generally, cloud computing can be divided into three different types: public cloud, private cloud and hybrid cloud. Public cloud service may be free or offered on a pay-per-usage model. The main benefits of public cloud service can be listed as follows: easy and inexpensive set-up due to the reason that the corresponding costs are covered by the provider; better scalability; cheaper due to pay-per-usage model; *etc.* Public clouds are external or publicly available cloud environments that are accessible to any client, whereas private clouds are internal or private cloud environments for particular organizations. Hybrid clouds are composed of public clouds and private clouds. More security responsibilities for the clients are indispensable to cloud service providers. It is more critical in public clouds for their own properties.

Public clouds' infrastructure and computational resources are owned and operated by outside public cloud service providers which deliver services to the general clients via a multi-tenant platform. Thus, the clients can not look into the public cloud servers' management, operation, technical infrastructure and procedures. This property incurs some security problems due to the reason that the clients can not control their remote data. For the clients, one of the main concerns about moving data to a public cloud infrastructure is security. Specially, the clients need to ensure their remote data is kept intact in public clouds. It is important to study remote data integrity checking since the public cloud servers (PCS) may modify the clients' data to save the storage space or other aims. Or, some inevitable faults make some data lost. Thus, it is necessary to design provable data possession protocol in public clouds.

1.1 Motivation

We consider the application scenario below.

In a big supermarket, the different managers will move the massive data to the public clouds. The data has to do with sale, capital, staff member, *etc.* These different data needs to get different approvals before they are moved to the public clouds. Such as, before sale data is moved, these data must be approved by salesman and sales manager; before staff member data is moved, these data must be approved by human resource manager and the chairman; capital data will have to be approved by the salesman, the chief financial officer and the chairman before they are moved to public clouds, *etc.*

There exist many application scenarios that the data must be approved by multi clients before they are moved to the public clouds. Since different data needs different client subset's approval, it is necessary to study provable data possession protocol which supports a general access structure. In order to ensure their data security, the verifier has to check their remote data possession at regular intervals. In some situations, the verifier is restricted to access the network, e.g., in prison because of comitting crime, in the battlefield because of the war, *etc.* Thus, the verifier has to delegate its remote data possession proof task to the proxy. After that, the proxy will perform the remote data possession proof protocol based on their warrant. This real social requirement motivates us to study proxy provable data possession with general access structure in public clouds.

1.2 Related Work

It is important to ensure the clients' remote data integrity since the clients do not control their own data. In 2007, a provable data possession (PDP) model was proposed by *G. Ateniese et al.* [1]. PDP is a lightweight probable remote data integrity checking model. After that, they proposed dynamic PDP model and designed the concrete dynamic PDP scheme based on symmetric cryptography algorithm [2]. In order to support data insert operation, *Erway et al.* proposed a full-dynamic PDP scheme from authenticated skip table [3]. *F. Sebe et al.* designed a provable data possession scheme by using factoring large numbers difficult problem [4]. *Wang* proposed the concept of proxy provable data possession [5]. After that, identity-based provable data possession were proposed [6,7]. In order to ensure critical data secure, some clients copy them and get their replications. Then, they move these original data and replicated data to multi PCS. In this case, client must ensure its remote data intact on multi PCS, *i.e.*, multi-replica provable data possession [8–11]. At the same time, as a stronger remote data integrity checking model, proofs of retrievability (PORs) was also proposed [12]. After that, *H. Shacham* gave the first PORs protocol with full security proofs in the strong security model [12,13]. It can be also applied into the fields, pay-TV [14], medical/health data [15], etc. Some research results have been gotten in the field of PORs [16–19]. Provable data possession is an important model which gives the solution of remote data integrity checking. At the same time, it is also very meaningful to study special PDP models according to different application requirements.

1.3 Private PDP and PPDP

From the role of the PDP verifier, it can be divided into two categories: private PDP and public PDP. In the *CheckProof* phase of private PDP, some private information is needed. On the contrary, private information is not needed in the *CheckProof* phase of public PDP. Public PDP provides no guarantee of privacy and can easily leak information. Private PDP is necessary in some cases.

A supermarket sells goods every day and stores the sale records in the public clouds. The supermarket can check these sale records integrity periodically by using PDP model. It would not like other entities to perform the checking task. If the competitors can perform the integrity checking, they can get the sale information by performing many times integrity queries. Without loss of generality, we assume that the queried block sequence is $\{m_{s_1}, m_{s_2}, \cdots, m_{s_c}\}$. The symbols $s_1, s_2, \cdots, s_c$ denote the queried block indices where $s_1 \leq s_2 \leq \cdots \leq s_c$. By making s_c bigger gradually until the PCS can not reply valid response, the competitors can get the biggest number $\hat{s}_c$. Making use of block size and $\hat{s}_c$, the competitors can get the supermarket's sale record data size. Then, they can evaluate its sale volume for every day. It is dangerous for the supermarket. In this case, private PDP is necessary.

In private PDP, when the verifier has no ability to perform PDP protocol, it will delegate the PDP task to the proxy according to the warrant. Thus, it is important and meaningful to study PPDP with the general access structure.

Table 1. Notations and descriptions

Notations	Descriptions
$\mathcal{A}$	General access structure
$\mathcal{A}_i$	Valid subset to move the file to PCS
U_{j_l}	the l-th member in the subset $\mathcal{A}_j$
(x_{j_l}, X_{j_l})	Private/public key pair of U_{j_l}
(y, Y)	Private/public key pair of PCS
(z, Z)	Private/public key pair of dealer
(m_i, T_i)	Block-tag pair
Σ	ordered collection of tags
$F = \{m_1, \cdots, m_n\}$	Stored file
$\mathcal{G}_1, \mathcal{G}_2$	two multiplicative groups
$\hat{e}$	the bilinear map from $\mathcal{G}_1$ to $\mathcal{G}_2$
q	the order of $\mathcal{G}_1$ and $\mathcal{G}_2$
π	pseudo-random permutation
H, h	cryptographic hash function
f, Ω	two pseudo-random functions
$chal = (c, k_1, k_2)$	the challenge, *i.e.*, c denotes the size of the challenged block set, k_1, k_2 are two different random numbers
$(\omega\ ,\ cert)$	warrant-certificate pair
PCS	public cloud server
PPDP	proxy provable data possession

1.4 Our Contribution

In this paper, we propose the concept, system model and security model of PPDP protocol with general access structure. Then, by making use of the n-multiinear pairings and some difficult problems, we design a concrete and provably secure PPDP protocol which supports general access structure. Finally, we give the formal security proof and performance analysis. Through security analysis and performance analysis, our protocol is shown secure and efficient.

1.5 Organization

The rest of the paper is organized as follows. Section 2 introduces the preliminaries. Section 3 describes our PPDP protocol with general access structure, the formal security analysis and performance analysis. Finally, Sect. 4 gives a conclusion.

The notations throughout this paper are listed in Table 1.

2 Preliminaries

In this section, we propose the system model and security model of PPDP with general access structure. Then, the bilinear pairing, multilinear map and some corresponding difficult problems are reviewed in this section.

2.1 System Model and Security Model

The system consists of four different network entities: *Client, PCS, Dealer, Proxy.* They can be shown as the following.

1. *Client*, who has massive data to be stored on PCS for maintenance and computation, can be either individual consumer or organization, such as desktop computers, laptops, tablets, smart phones, *etc.*;
2. *PCS*, which is managed by public cloud service provider, has significant storage space and computation resource to maintain *client*' massive data;
3. *Dealer* is delegated to store multi-clients' data to PCS where the multi-client subset belongs to the concrete general access structure. It is trusted by all the clients.
4. *Proxy*, which is delegated to check *Client*'s data possession, has the ability to check *Client*'s data possession according to the warrant ω.

In the system model, there exists a general access structure $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_{n'}\}$. In order to store some special files, all the clients in some subset $\mathcal{A}_j$ cooperate to approve and move the special files to PCS via the entity *Dealer*. The clients no longer store the special files locally. The clients can perform the remote data possession proof or delegate it to the proxy in special cases.

We start with the precise definition of PPDP with general access structure, followed by the formal security definition. Before that, we define the general access structure in our PPDP protocol.

Definition 1 (General Access Structure). *For the client set $\mathcal{U} = \{U_1, U_2, \cdots, U_n\}$, the clients in $\mathcal{U}$'s subset $\mathcal{A}_j = \{U_{j_1}, U_{j_2}, \cdots, U_{j_{n_j}}\}$ can cooperate to approve and store the file F to PCS where $j = 1, 2, \cdots, n'$ and $\mathcal{A}_j \subseteq \mathcal{U}$. Denote $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_{n'}\}$. Then, $\mathcal{A}$ is regarded as the general access structure.*

Without loss of generality, suppose the stored file F is divided into n blocks, *i.e.*, $F = \{m_1, m_2, \cdots, m_n\}$.

Definition 2 (PPDP with General Access Structure). *For general access structure, PPDP is a collection of six polynomial time algorithms (*SetUp*,* TagGen*,* CertVry*,* CheckTag*,* GenProof*,* CheckProof*) among* PCS*,* Client*,* Dealer *and* Proxy *such that:*

1. *SetUp$(1^k) \rightarrow (sk, pk)$ is a probabilistic polynomial time key generation algorithm. Input a security parameter k, it returns a private/public key pair for every running. Every client $U_{j_l} \in \mathcal{A}_j$ can get its private/public key pair (x_{j_l}, X_{j_l}). PCS can also get its private/public key pair (y, Y). On the other hand, the client set $\mathcal{A}_j$ also prepares the warrant ω_j and the corresponding certificate cert$_j$, where ω_j points out the restriction conditions that the Proxy can perform the remote data possession checking task. The warrant-certificate pair (ω_j, cert$_j$) is sent to the Proxy.*
2. *$TagGen(x_{j_l}, X_{j_l}, Y, m_i, U_{j_l} \in \mathcal{A}_j) \rightarrow T_i$ is a probabilistic polynomial time algorithm that is run by all members of $\mathcal{A}_j$ and* Dealer *to generate the block tag T_i. Input the private/public key pair (x_{j_l}, X_{j_l}) for all the $U_{j_l} \in \mathcal{A}_j$, PCS's public key Y and a file block m_i, this algorithm returns the block tag T_i.*
3. *$CertVry(\omega_j, cert_j) \rightarrow \{``success", ``failure"\}$ is run by the proxy in order to validate the warrant-certificate pair. If the pair is valid, it outputs "Success" and accepts the pair ; otherwise, it outputs "$failure$" and rejects the pair.*
4. *$CheckTag(m_i, T_i, y, X_{j_l}, Y, U_{j_l} \in \mathcal{A}_j) \rightarrow \{``success", ``failure"\}$ is a determined polynomial time algorithm that is run by the PCS to check whether the block-tag pair (m_i, T_i) is valid or not. Input the block-tag pair (m_i, T_i), PCS's private/public key pair (y, Y) and the clients' public key X_{j_l} for all $U_{j_l} \in \mathcal{A}_j$, the algorithm returns "success" or "$failure$" denoting the pair is valid or not respectively.*
   Notes: CheckTag *phase is important in order to prevent the malicious clients. If the malicious clients store invalid block-tag pairs to PCS, PCS will accept them if* CheckTag *phase does not exist. When the malicious clients check these data's integrity, PCS's response will not pass the verification. The malicious clients will require PCS to pay compensation. Thus, PCS's benefits will be harmed.*
5. *$GenProof(X_{j_l}, y, Y, F, \Sigma, chal, U_{j_l} \in \mathcal{A}_j) \rightarrow V$ is a polynomial time algorithm that is run by the PCS in order to generate a proof of data integrity, where $\Sigma = \{T_1, T_2, \cdots, T_n\}$ is the ordered collection of tags. Input the public keys $(X_{j_l}, Y, U_{j_l} \in \mathcal{A}_j)$, an ordered collection F of blocks, an ordered collection of tags Σ and a challenge $chal$. Upon receiving the challenge from the proxy,*

it returns a data integrity proof V for some blocks in F that are determined by the challenge chal.

6. *CheckProof*$(X_{j_l}, Y, chal, V, auxiliary\ data, U_{j_l} \in \mathcal{A}_j) \rightarrow \{\text{"}success\text{"}, \text{"}failure\text{"}\}$ *is a polynomial time algorithm that is run by the proxy in order to check the PCS's response V. Input the public keys X_{j_l}, Y for $U_{j_l} \in \mathcal{A}_j$, a challenge* chal*, PCS's response V and some auxiliary data, this algorithm returns "success" or "failure" denoting whether V is valid or not for the data integrity checking of the blocks determined by* chal*.*

For the general access structure, in order to ensure that PPDP protocol is secure and efficient, the following requirements must be satisfied:

1. For the general access structure, the PPDP protocol only be performed by the clients or the delegated proxy.
2. *Dealer* should not be required to keep an entire copy of the files and tags.
3. The protocol should keep secure even if the PCS is malicious. If the PCS has modified some block tag pairs that are challenged, the response V can only pass the *CheckProof* phase with negligible probability. In other words, PCS has no ability to forge the response V in polynomial time.

According to the above security requirements, for general access structure, we define what is a secure PPDP protocol against malicious PCS (security property (3)) below. Without loss of generality, suppose the stored file is F and it is grouped into n blocks, *i.e.*, $F = \{m_1, m_2, \cdots, m_n\}$. Let the general access structure be $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_{n'}\}$. Suppose the subset $\mathcal{A}_j = \{U_{j_1}, U_{j_2}, \cdots, U_{j_{n_j}}\} \in \mathcal{A}$ has the right to approve to store the file F to PCS.

Definition 3 (Unforgeability). *For general access structure, PPDP protocol is unforgeable if for any (probabilistic polynomial time) adversary $\mathbb{A}$ the probability that $\mathbb{A}$ wins the following PPDP game is negligible. For the general access structure, the PPDP game between the challenger $\mathcal{C}$ and the adversary $\mathbb{A}$ can be shown below:*

1. *SetUp: $\mathcal{C}$ generates system parameters params, clients' private/public key pairs (x_{j_l}, X_{j_l}) for all $U_{j_l} \in \mathcal{A}_j$, the proxy's private/public key pair (z, Z) and PCS's private/public key pair (y, Y). Then, it sends $(params, X_{j_l}, Y, y, Z, z, U_{j_l} \in \mathcal{A}_j)$ to the adversary $\mathbb{A}$. $\mathcal{C}$ keeps $(x_{j_l}, U_{j_l} \in \mathcal{A}_j)$ confidential and sends y, z to $\mathbb{A}$, i.e., y, z are known to $\mathbb{A}$. It is consistent with the real environment since the adversary $\mathbb{A}$ simulates PCS or the collusion of PCS and the proxy.*
2. *First-Phase Queries: $\mathbb{A}$ adaptively makes a number of different queries to $\mathcal{C}$. Each query can be one of the following.*
   - Hash *queries. $\mathbb{A}$ makes Hash function queries adaptively. $\mathcal{C}$ responds the Hash values to $\mathbb{A}$.*
   - Tag *queries. $\mathbb{A}$ makes block tag queries adaptively. For a query m_{1_1} queried by $\mathbb{A}$, $\mathcal{C}$ computes the tag $T_{1_1} \leftarrow \mathsf{TagGen}(x_{j_l}, y, z, X_{j_l}, Y, Z, m_{1_1}, U_{j_l} \in \mathcal{A}_j)$ and sends it back to $\mathbb{A}$. Without loss of generality, let $\{m_{1_1}, m_{1_2}, \cdots, m_{1_i}, \cdots, m_{1_{|\mathbb{I}_1|}}\}$ be the blocks which have been submitted for tag queries. Denote the index set as $\mathbb{I}_1$, $i, e.$, $1_i \in \mathbb{I}_1$.*

3. *Challenge: $\mathcal{C}$ generates a challenge chal which defines a ordered collection $\{j_1, j_2, \cdots, j_c\}$, where $\{j_1, j_2, \cdots, j_c\} \nsubseteq \mathbb{I}_1$ is a set of indexes and c is a positive integer. $\mathcal{C}$ is required to provide a data integrity proof for the blocks $m_{j_1}, \cdots, m_{j_c}$.*
4. *Second-Phase Queries: Similar to the First-Phase Queries. Without loss of generality, let $\{m_{2_1}, m_{2_2}, \cdots, m_{2_i}, \cdots, m_{2_{|\mathbb{I}_2|}}\}$ be the blocks which have been submitted for tag queries. Denote the index set as $\mathbb{I}_2$, i, e., $2_i \in \mathbb{I}_2$. The restriction is that $\{j_1, j_2, \cdots, j_c\} \nsubseteq \mathbb{I}_1 \cup \mathbb{I}_2$.*
5. *Forge: $\mathbb{A}$ returns a data integrity checking response V for the blocks indicated by chal.*

We say that $\mathbb{A}$ wins the above game if $CheckProof(X_{j_l}, Y, chal, V, auxiliary$ $data, U_{j_l} \in \mathcal{A}_j) \rightarrow$ "success" with nonnegligible probability.

Definition 3 states that, for the challenged blocks, a malicious PCS cannot produce a valid remote data integrity checking response if some challenged block tag pairs have been modified. It is a very important security property. On the other hand, if the response can pass the proxy's verification, what is the probability of all the data keeps intact ? The following definition states clearly the status of the blocks that are not challenged. In practice, a secure PPDP protocol also needs to guarantee that after validating the PCS's response, the proxy can be convinced that all of his outsourced data have been kept intact with a high probability. This observation gives the following security definition.

Definition 4 ((ρ, δ) Security). *For general access structure, a PPDP protocol is (ρ, δ) secure if PCS corrupted ρ fraction of the whole blocks, the probability that the corrupted blocks are detected is at least δ.*

In order to explain the definition 4, we give a concrete example. Suppose PCS stored $\ddot{n}$ block-tag pairs. The instrument troubles or malicious operations make $\ddot{l}$ block-tag pairs lost for PCS. Then, the corrupted fraction of the whole blocks is $\rho = \frac{\ddot{l}}{\ddot{n}}$. Suppose the clients query $\ddot{c}$ block-tag pairs' integrity checking. If the probability that the corrupted blocks can detected is at least δ, we call this scheme satisfies the (ρ, δ) security.

2.2 Bilinear Pairings, Multilinear Map and Difficult Problem

Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be two cyclic multiplicative groups with the same prime order q. Let $\hat{e} : \mathcal{G}_1 \times \mathcal{G}_1 \rightarrow \mathcal{G}_2$ be a bilinear map. The bilinear map $\hat{e}$ can be constructed by the modified Weil or Tate pairings [20,21] on elliptic curves. The group $\mathcal{G}_1$ with such a map $\hat{e}$ is called a bilinear group. The Computational Diffie-Hellman (CDH) problem is assumed hard while the Decisional Diffie-Hellman (DDH) problem is assumed easy on the group $\mathcal{G}_1$ [22]. We give their expression below.

Gap Diffie-Hellman Problem (GDH). Let g is the generator of $\mathcal{G}_1$. For instance, given unknown $a, b, c \in \mathcal{Z}_q^*$ and $g, g^a, g^b, g^c \in \mathcal{G}_1$, it is recognized that there exists an efficient algorithm to determine whether $ab = c \bmod q$ by verifying $\hat{e}(g^a, g^b) = \hat{e}(g, g)^c$ in polynomial time (DDH problem), while no efficient algorithm can compute $g^{ab} \in \mathcal{G}_1$ with non-negligible probability within polynomial time (CDH problem). An algorithm $\mathbb{A}$ is said to (t, ϵ)-break the CDH problem on $\mathcal{G}_1$ if $\mathbb{A}$'s advantage is at least ϵ in time t, , *i.e.*,

$$Adv_{\mathcal{A}}^{CDH} = \Pr[\mathcal{A}(g, g^a, g^b) = g^{ab} : \forall a, b \in \mathcal{Z}_q^*] \geq \epsilon$$

The probability is taken over the choice of a, b and $\mathbb{A}$'s coin tosses.

A group $\mathcal{G}_1$ is a (t, ϵ)-GDH group if the DDH problem on $\mathcal{G}_1$ is efficiently computable and there exists no algorithm can (t, ϵ)-break the CDH problem on $\mathcal{G}_1$.

We say that $\mathcal{G}_1$ satisfies the CDH assumption if for any randomized polynomial time (in k) algorithm $\mathbb{A}$ we have that $Adv_{\mathcal{A}}^{CDH}(k)$ is a negligible function. In this paper, our multi-client PDP protocol come from the GDH group $\mathcal{G}_1$.

Next, we give the definition of an n-multilinear map. Multilinear map was proposed in the Ref. [23]. Many experts have proposed the concrete implementation [24,25]. We view the groups $\mathcal{G}_1$ and $\mathcal{G}_n$ as multiplicative groups.

Definition 5. *A map $\hat{e}_n : \mathcal{G}_1^n \rightarrow \mathcal{G}_n$ is an n-multilinear map if it satisfies the following properties:*

1. *$\mathcal{G}_1$ and $\mathcal{G}_n$ are groups of the same prime order q;*
2. *If $a_1, \cdots, a_n \in \mathcal{Z}_q^*$ and $g_1, \cdots, g_n \in \mathcal{G}_1$ then*

$$\hat{e}_n(g_1^{a_1}, \cdots, g_n^{a_n}) = \hat{e}_n(g_1, \cdots, g_n)^{a_1 a_2 \cdots a_n}$$

3. *The map $\hat{e}_n$ is non-degenerate in the following sense: if $g \in \mathcal{G}_1$ is a generator of $\mathcal{G}_1$ then $\hat{e}_n(g, \cdots, g)$ is a generator of $\mathcal{G}_n$.*

Multilinear Diffie-Hellman Problem. Given $g, g^{a_1}, \cdots, g^{a_{n+1}}$ in $\mathcal{G}_1$, it is hard to compute $\hat{e}_n(g, \cdots, g)^{a_1 \cdots a_{n+1}}$ in $\mathcal{G}_n$.

n-multilinear map has been used in the encryption, key management, hash function *etc.* [26–28].

3 Our Proposed PPDP Protocol with General Access Structure

3.1 Construction of PPDP Protocol with General Access Structure

First, we introduce some additional notations which will be used in the construction of our PPDP protocol with general access structure. Let g be a generator of $\mathcal{G}_1$. Suppose the stored file F (maybe encoded by using error-correcting code, such as, Reed-Solomon code) is divided into n blocks $(m_1, m_2, \cdots, m_n)$ where $m_i \in \mathcal{Z}_q^*$, *i.e.*, $F = (m_1, m_2, \cdots, m_n)$. The following functions are given below:

$$\begin{array}{l} f : \mathcal{Z}_q^* \times \{1, 2, \cdots, n\} \rightarrow \mathcal{Z}_q^* \\ \Omega : \mathcal{Z}_q^* \times \{1, 2, \cdots, n\} \rightarrow \mathcal{Z}_q^* \\ \pi : \mathcal{Z}_q^* \times \{1, 2, \cdots, n\} \rightarrow \{1, 2, \cdots, n\} \\ H : \{0, 1\}^* \rightarrow \mathcal{Z}_q^* \\ h : \mathcal{Z}_q^* \times \mathcal{Z}_q^* \rightarrow \mathcal{G}_1^* \end{array}$$

where f and Ω are two pseudo-random functions, and π is a pseudo-random permutation, H and h are cryptographic hash functions. For general access structure, PPDP protocol construction consists of six phases: *SetUp, TagGen, CertVry, CheckTag, GenProof, CheckProof.*

SetUp: PCS picks a random number $y \in \mathcal{Z}_q^*$ as its private key and computes $Y = g^y$ as its public key. The proxy picks a random number $z \in \mathcal{Z}_q^*$ as its private key and computes $Z = g^z$ as its public key. Suppose there are $\bar{n}$ clients $\mathcal{U} = \{U_1, U_2, \cdots, U_{\bar{n}}\}$. Let the general access structure be $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_s\}$, where $\mathcal{A}_j = \{U_{j_1}, U_{j_2}, \cdots, U_{j_{n_j}}\} \subseteq \mathcal{U}, 1 \le j \le s$. For every $\mathcal{A}_j$, the dealer picks a random $u_j \in \mathcal{G}_1$ as $\mathcal{A}_j$'s public key. For any client $U_i \in \mathcal{U}$, it picks a random $x_i \in \mathcal{Z}_q^*$ as its private key and computes $X_i = g^{x_i}$ as its public key. $\mathcal{A}_j$'s warrant consists of the description ω_j of the constraints for which remote data possession proof is delegated together with a certificate $cert_j$. $cert_j$ is the multi-signature on ω_j of all the clients in $\mathcal{A}_j$ by using the concrete algorithms [29,30]. Once delegated, the proxy can perform the data possession proof by using its private key z and warrant-certification pair $(\omega_j, cert_j)$. The clients send $(\omega_j, cert_j)$ to the proxy. The system parameter set is $params = \{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_{n_j+1}, \hat{e}_{n_j+1}, \hat{e}, f, \Omega, \pi, H, h, q, u_j, X_i, \mathcal{A}_j \in \mathcal{A}, U_i \in \mathcal{U}\}$.

$TagGen(x_{j_l}, F, i, U_{j_l} \in \mathcal{A}_j)$: Suppose the valid client subset $\mathcal{A}_j$ generates the corresponding tags for the file $F = (m_1, m_2, \cdots, m_n)$. Denote the set $\bar{\mathcal{A}}_{j_l} = \mathcal{A}_j / U_{j_l}$. For every block m_i, the clients $\{U_{j_1}, U_{j_2}, \cdots, U_{j_{n_j}}\}$ in $\mathcal{A}_j$ and the dealer generate the tag T_i. In $\mathcal{A}_j$, all the clients cooperate to generate the multi-signature $cert_j$ on the warrant ω_j. The warrant-certification pair $(\omega_j, cert_j)$ are sent to the proxy. For $U_{j_l} \in \{U_{j_1}, U_{j_2}, \cdots, U_{j_{n_j}}\}$, it performs the following procedures:

1. U_{j_l} computes

$$t_j = H(\hat{e}_{n_j+1}(X_{j_1}, \cdots, X_{j_{l-1}}, X_{j_{l+1}}, \cdots, X_{j_{n_j}}, Y, Z)^{x_{j_l}}, \omega_j)$$

   $W_{i,j} = \Omega_{t_j}(i), \quad T_{i,j_l} = (h(t_j, W_{i,j}) u_j^{m_i})^{x_{j_l}};$
2. U_{j_l} sends the block-tag pair (m_i, T_{i,j_l}) and the corresponding warrant ω_j to dealer.

After receiving all the block-tag pairs (m_i, T_{i,j_l}), where $m_i \in F$, $U_{j_l} \in \mathcal{A}_j$, the dealer computes $T_i = \prod_{U_{j_l} \in \mathcal{A}_j} T_{i,j_l}$. Then it uploads the block-tag pair (m_i, T_i) and the corresponding warrant ω_j to PCS. When the above procedures are performed n times, all the block-tag pairs (m_i, T_i) are generated and uploaded to PCS for $1 \le i \le n$.

CertVry$(\{(\omega_j, cert_j), X_{j_i}, U_{j_i} \in \mathcal{A}_j\})$: Upon receiving the clients' warrant-certification pair $(\omega_j, cert_j)$, the proxy verifies its validity. If it is valid, the proxy accepts ω_j; otherwise, the proxy rejects it and queries the *Clients* for new warrant-certification pair.

CheckTag($(m_i, T_i), 1 \leq i \leq n$): Given $\{(m_i, T_i), 1 \leq i \leq n\}$, for every i and $\mathcal{A}_j \in \mathcal{A}$, PCS computes

$$\hat{t}_j = H(\hat{e}_{n_j+1}(X_{j_1}, \cdots, X_{j_{n_j}}, Z)^y, \omega_j), \quad \hat{W}_{i,j} = \Omega_{\hat{t}_j}(i)$$

Then, it verifies whether the following formula holds.

$$\hat{e}(T_i, g) \overset{?}{=} \hat{e}(h(\hat{t}_j, \hat{W}_{i,j})u_j^{m_i}, \prod_{U_{j_l} \in \mathcal{A}_j} X_{j_l})$$

If it holds, PCS accepts it; otherwise, it is rejected.

GenProof($pk, F, chal, \Sigma$): Let $F, chal, \Sigma$ denote $F = (m_1, m_2, \cdots, m_n), chal = (c, k_1, k_2), \Sigma = (T_1, \cdots, T_n)$ where *chal* is the proxy's challenge. In this phase, the dealer asks the PCS for remote data integrity checking of c file blocks whose indices are randomly chosen for each challenge. It can prevent the PCS from anticipating which blocks will be queried in each challenge. The number $k_1 \in \mathcal{Z}_q^*$ is the random key of the pseudo-random permutation π. The number $k_2 \in \mathcal{Z}_q^*$ is the random key of the pseudo-random function f. On the other hand, the proxy sends $(\omega_j, cert_j)$ to PCS. PCS verifies whether the signature $cert_j$ is valid. If it is valid, PCS compares this ω_j with its stored warrant ω_j'. When $\omega_j = \omega_j'$ and the proxys query complys with the warrant ω_j, PCS performs the procedures as follows. Otherwise, PCS rejects the proxys query.

1. For $1 \leq r \leq c$, PCS computes $i_r = \pi_{k_1}(r), a_r = f_{k_2}(r)$ as the indexes and coefficients of the blocks for which the proof is generated.
2. PCS computes $T = \prod_{r=1}^{c} T_{i_r}^{a_r}$, $\hat{m} = \sum_{r=1}^{c} a_r m_{i_r}$.
3. PCS outputs $V = (T, \hat{m})$ and sends V to the proxy as the response to the *chal* query.

$CheckProof(chal, X_{j_l}, V, U_{j_l} \in \mathcal{A}_j)$: Upon receiving the response V from PCS, the proxy performs the procedures below:

1. For $1 \leq r \leq c$, the proxy computes
$$t_j = H(\hat{e}_{n_j+1}(X_{j_1}, \cdots, X_{j_{n_j}}, Y)^z, \omega_j)$$
$$v_r = \pi_{k_1}(r), \quad a_r = f_{k_2}(r), \quad W_{v_r,j} = \Omega_{t_j}(v_r)$$
2. The proxy checks whether the following formula holds.
$$\hat{e}(T, g) \overset{?}{=} \hat{e}(\prod_{r=1}^{c} h(t_j, W_{v_r,j})^{a_r} u_j^{\hat{m}}, \prod_{U_{j_l} \in \mathcal{A}_j} X_{j_l})$$
If the above formula holds, then the proxy outputs "success". Otherwise the proxy outputs "failure".

Notes: In the subset $\mathcal{A}_j$, any client U_{j_l} can also perform the phase *CheckProof* since the following formula holds:

$$\begin{aligned} &\hat{e}_{n_j+1}(X_{j_1}, \cdots, X_{j_{l-1}}, X_{j_{l+1}}, \cdots, X_{j_{n_j}}, Y, Z)^{x_{j_l}} \\ &= \hat{e}_{n_j+1}(X_{j_1}, \cdots, X_{j_{n_j}}, Y)^z \end{aligned}$$

Thus, U_{j_l} can also calculate t_j and finish *CheckProof*.

3.2 Correctness Analysis and Security Analysis

The correctness analysis and security analysis of our proposed PPDP protocol can be given by the following lemmas and theorems:

Theorem 1. *If* Client, Dealer*, and* PCS *are honest and follow the proposed procedures, then the uploaded block-tag pairs can pass* PCS*'s tag checking.*

Proof. In the phases *TagGen* and *CheckTag*, for all $U_{j_l} \in \mathcal{A}_j$,

$$\begin{aligned}\bar{t}_j &= H(\hat{e}_{n_j+1}(X_{j_1}, \cdots, X_{j_{n_j}}, Z)^y, \omega_j)\\ &= H(\hat{e}_{n_j+1}(g, \cdots, g, g)^{yz \prod_{U_{j_l} \in \mathcal{A}_j} x_{j_l}}, \omega_j)\\ &= t_j\\ &= \hat{t}_j\end{aligned}$$

Then, we can get $W_{i,j} = \bar{W}_{i,j} = \hat{W}_{i,j}$. By using *TagGen*, we know that

$$\begin{aligned}\hat{e}(T_i, g) &= \hat{e}(\prod_{U_{j_l} \in \mathcal{A}_j} (h(t_j, W_{i,j})u_j^{m_i})^{x_{j_l}}, g)\\ &= \hat{e}(h(t_j, W_{i,j})u_j^{m_i}, g^{\sum_{U_{j_l} \in \mathcal{A}_j} x_{j_l}})\\ &= \hat{e}(h(t_j, W_{i,j})u_j^{m_i}, \prod_{U_{j_l} \in \mathcal{A}_j} X_{j_l})\end{aligned}$$

Theorem 2. *If the proxy and PCS are honest and follow the proposed procedures, the response V can pass the proxy's data integrity checking,* i.e.*, our PPDP protocol satisfies the correctness.*

Proof. Based on *TagGen* and *GenProof*, we know that $T = \prod_{r=1}^{c} T_{i_r}^{a_r}$. Thus,

$$\begin{aligned}\hat{e}(T, g) &= \hat{e}(\prod_{r=1}^{c} T_{i_r}^{a_r}, g)\\ &= \hat{e}(\prod_{r=1}^{c} (h(t_j, W_{i_r,j})u_j^{m_{i_r}})^{a_r}, \prod_{U_{j_l} \in \mathcal{A}_j} X_{j_l})\\ &= \hat{e}(\prod_{r=1}^{c} h(t_j, W_{i_r,j})^{a_r} u_j^{\sum_{r=1}^{c} a_r m_{i_r}}, \prod_{U_{j_l} \in \mathcal{A}_j} X_{j_l})\\ &= \hat{e}(\prod_{r=1}^{c} h(t_j, W_{i_r,j})^{a_r} u_j^{\hat{m}}, \prod_{U_{j_l} \in \mathcal{A}_j} X_{j_l})\end{aligned}$$

Lemma 1. *Let $(\mathcal{G}_1, \mathcal{G}_2)$ be a (T', ϵ')-GDH group pair of order q. Let $\mathcal{A}_j$ be the tag generating subset. Then the tag generation scheme* TagGen *is $(T, q_S, q_H, q_h, \epsilon)$-existentially unforgeable under the adaptive chosen-message attack for all T and ϵ satisfying $\epsilon' \geq \frac{\epsilon}{(q_s+1)e}$ and $T' \leq T + c_{\mathcal{G}_1}(q_h + 2q_S) + c_{\hat{e}_{n_j}} q_H$, where $c_{\mathcal{G}_1}$ is the time cost of exponentiation on $\mathcal{G}_1$, $c_{\hat{e}_{n_j}}$ is the time cost of n_j-linear map. Here e is the base of the natural logarithm, and q_S, q_H, q_h are the times of* Tag query*,* H-query *and* h-query *respectively. n_j is the cardinal number of the tag generating subset $\mathcal{A}_j$.*

Proof. It is similar with Ref. [5]. We omit the proof procedures due to the page limits.

Lemma 2. *Let the challenge be chal* $= (c, k_1, k_2)$*. Then, the queried block-tag pair set is* $\{(m_{\pi_{k_1}(i)}, T_{\pi_{k_1}(i)}), 1 \leq i \leq c\}$*. If some block tag pairs are modified, the grouped block tag pair* $(\hat{m}, T)$ *can pass the proxy's verification only with negligible probability.*

Proof. We will prove this lemma by contradiction. It is assumed that the forged block tag pair $(\hat{m}, \hat{T})$ can pass the dealer's integrity checking, *i.e.*,

$$\hat{e}(\hat{T}, g) = \hat{e}(\prod_{r=1}^{c} h(t_j, W_{i_r,j})^{a_r} u_j^{\hat{m}}, \sum_{U_{j_l} \in \mathcal{A}_j} X_{j_l})$$

We prove this lemma from two cases.

Case 1, PCS makes use of the modified block tag pair to generate the grouped block tag pair and the block indexes satisfy the challenge requirements:

$$\hat{e}(\prod_{r=1}^{c} \hat{T}_{i_r}^{a_r}, g) = \hat{e}(\prod_{r=1}^{c} h(t_j, W_{i_r,j})^{a_r} u_j^{\sum_{r=1}^{c} a_r \hat{m}_{i_r}}, \prod_{U_{j_l} \in \mathcal{A}_j} X_{j_l})$$

where $a_r = f_{k_2}(r)$ and $i_r = \pi_{k_1}(r)$ are pseudo random, $1 \leq r \leq c$. Then,

$$\prod_{r=1}^{c} \hat{e}(\hat{T}_{\hat{m}_{i_r}}^{a_r}, g) = \prod_{r=1}^{c} \hat{e}(h(t_j, W_{i_r,j}) u_j^{\hat{m}_{i_r}}, \prod_{U_{j_l} \in \mathcal{A}_j} X_{j_l})^{a_r}$$

Let the generator of $\mathcal{G}_2$ be d, and

$$\hat{e}(\hat{T}_{i_r}, g) = d^{\hat{y}_r}$$

$$\hat{e}(h(t_j, W_{i_r,j}) u_j^{\hat{m}_{i_r}}, \prod_{U_{j_l} \in \mathcal{A}_j} X_{j_l}) = d^{y_r}$$

Then we can get

$$d^{\sum_{r=1}^{c} a_r \hat{y}_r} = d^{\sum_{j=1}^{c} a_r y_r}$$

$$\sum_{r=1}^{c} a_r \hat{y}_r = \sum_{r=1}^{c} a_r y_r$$

$$\sum_{r=1}^{c} a_j (\hat{y}_r - y_r) = 0 \bmod (q-1) \qquad (1)$$

According to *Lemma* 1, a single block *Tag* is existential unforgeable. So, there exist at least two different indexes r such that $\hat{y}_r \neq y_r$. Suppose there are $s \leq c$ pairs $(\hat{y}_r, y_r)$ such that $\hat{y}_r \neq y_r$. Then, there exist q^{s-1} tuples $(a_1, a_2, \cdots, a_c)$ satisfying the above Eq. (1). Since $(a_1, a_2, \cdots, a_c)$ is a random vector, the probability that the tuple satisfies the Eq. (1) is not greater than $q^{s-1}/q^c \leq q^{c-1}/q^c = q^{-1}$. This probability is negligible.

Case 2, the PCS substitutes the other valid block-*Tag* pairs for modified block-*Tag* pairs:

To the challenge $chal = (c, k_1, k_2)$, PCS can get queried block tag pairs index set $\{i_1, i_2, \cdots, i_c\}$. Without loss of generality, we assume s block tag pairs are modified and their index set is $\{i_1, i_2, \cdots, i_s\}$ where $s \leq c$. PCS substitutes s valid block tag pairs for the s modified pairs. Without loss of generality, suppose the s valid block tag pairs indexes are $\mathcal{V} = \{v_1, v_2, \cdots, v_s\}$. PCS computes the grouped block tag pair as follows:

$$T = \prod_{r=s+1}^{c} T_{i_r}^{a_r} \prod_{v \in \mathcal{V}} T_v^{a_v}, \qquad \hat{m} = \sum_{r=s+1}^{c} a_r m_{i_r} + \sum_{v \in \mathcal{V}} a_v m_v$$

where $a_r = f_{k_2}(r)$ for all $1 \leq r \leq c$ and $a_{v_i} = a_i$ for $1 \leq i \leq s$.

Assume the forged group block tag pair can pass the dealer's checking, *i.e.*,

$$\hat{e}(T, g) = \hat{e}(\prod_{r=1}^{c} h(t_j, W_{i_r,j})^{a_r} u_j^{\hat{m}}, \prod_{U_{j_l} \in \mathcal{A}_j} X_{j_l})$$

Since some block tag pairs are valid, *i.e.*, for $s+1 \leq r \leq c$,

$$\hat{e}(T_{i_r}, g) = \hat{e}(h(t_j, W_{i_r,j}) u_j^{m_{i_r}}, \prod_{U_{j_l} \in \mathcal{A}_j} X_{j_l})$$

We can get the following formula:

$\hat{e}(\prod_{v \in \mathcal{V}} T_v^{a_v}, g) = \hat{e}(\prod_{r=1}^{s} h(t_j, W_{i_r,j})^{a_r} u_j^{\sum_{v \in \mathcal{V}} a_v m_v}, \prod_{U_{j_l} \in \mathcal{A}_j} X_{j_l})$ On the other hand,

$$\hat{e}(\prod_{v \in \mathcal{V}} T_v^{a_v}, g) = \hat{e}(\prod_{v \in \mathcal{V}} h(t_j, W_{v,j})^{a_v} u_j^{\sum_{v \in \mathcal{V}} a_v m_v}, \prod_{U_{j_l} \in \mathcal{A}_j} X_{j_l})$$

Thus,

$$\begin{aligned} &\hat{e}(\textstyle\prod_{r=1}^{s} h(t_j, W_{i_r,j})^{a_r} u_j^{\sum_{v \in \mathcal{V}} a_v m_v}, \prod_{U_{j_l} \in \mathcal{A}_j} X_{j_l}) \\ = &\hat{e}(\textstyle\prod_{v \in \mathcal{V}} h(t_j, W_{v,j})^{a_v} u_j^{\sum_{v \in \mathcal{V}} a_v m_v}, \prod_{U_{j_l} \in \mathcal{A}_j} X_{j_l}) \end{aligned}$$

We can get $\prod_{r=1}^{s} h(t_j, W_{i_r,j})^{a_r} = \prod_{v \in \mathcal{V}} h(t_j, W_{v,j})^{a_v}$. The probability that the above formula holds is q^{-1} because of h is hash oracle. It is negligible.

Based on Case 1 and Case 2, the forged group block tag pair can pass the dealer's checking with the probability no more than q^{-1}. It is negligible.

Lemma 1 states that an untrusted PCS cannot forge individual tag to cheat the proxy. Lemma 2 implies that the untrusted PCS cannot aggregate fake tags to cheat the dealer.

Theorem 3. *According to our proposed PPDP protocol with general access structure, if some queried block tag pairs are modified, PCS's response can only pass the proxy's CheckProof phase with negligible probability based on the assumption that the CDH problem is hard on $\mathcal{G}_1$.*

Proof. Suppose the stored blocks set is $\{m_1, m_2, \cdots, m_n\}$. We denote the challenger as $\mathcal{C}$ and the adversary as $\mathbb{A}$. Let the public parameters be $params = \{\mathcal{G}_1, \mathcal{G}_2, \hat{e}, f, \Omega, \pi, H, h, q\}$. Input (g, g^a, g^b), the goal of $\mathcal{C}$ is to compute the value g^{ab}. Let the client subset that can generate tag is $\mathcal{A}_j$. First, $\mathcal{C}$ picks random $z_{j_l} \in \mathcal{Z}_q^*, u_j \in \mathcal{G}_1$ and calculates $X_{j_l} = (g^a)^{z_{j_l}}$ for all $U_{j_l} \in \mathcal{A}_j$. u_j can be regarded as the public parameter of the access subset $\mathcal{A}_j$. Let X_{j_l} be the client U_{j_l}'s public key. The corresponding private key is unknown to $\mathcal{C}$. The challenger maintains three tables T_H, T_h, T which are initialized empty. PCS picks a random $y \in \mathcal{Z}_q^*$ and computes $Y = g^y$. Let (y, Y) be the PCS's private/public key pair. PCS picks a random $z \in \mathcal{Z}_q^*$ and computes $Z = g^z$. Let (z, Z) be the proxy's private/public key pair. Then, $\mathcal{C}$ answers all the queries that $\mathbb{A}$ makes.

H-Oracle, h-Oracle, Tag-Oracle are the same as the corresponding procedures in the Lemma 1.

We consider the challenge $chal = (c, k_1, k_2)$. Assume the forged aggregated block-tag pair $(\hat{m}, T)$ can pass the dealer's data integrity checking, *i.e.*,

$$\hat{e}(\hat{T}, g) = \hat{e}(\prod_{r=1}^{c} h(t_j, W_{i_r,j})^{a_r} u_j^{\hat{m}}, \prod_{U_{j_l} \in \mathcal{A}_j} X_{j_l}) \tag{2}$$

where $a_j = f_{k_2}(j)$ are random, $1 \leq j \leq c$.

According to Lemmas 1 and 2, we know that if some queried block-tag pairs are corrupted, the verification formula (2) holds with negligible probability. Thus, our propose multi-client PDP protocol is provably unforgeable in the random oracle model.

Theorem 4. *For the general access structure, the proposed PPDP protocol is* $(\frac{d}{n}, 1 - (\frac{n-d}{n})^c)$*-secure. The probability* P_R *of detecting the modification satisfies:*

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

where n *denotes the stored block-tag pair number,* d *denotes the modified block-tag pair number, and the challenge is* $chal = (c, k_1, k_2)$.

Proof. It is similar with the Ref. [5]. We omit it due to the page limits.

3.3 Performance Analysis

In this section, we analyze the performance of our proposed PPDP protocol in terms of computation and communication overheads.

Computation: In our proposed PPDP protocol, suppose there exist n message blocks and the tag generating client subset is $\mathcal{A}_j$ which comprises n_j clients. In the *TagGen* phase, the clients need to perform n_j n_j-linear map, n_j exponentiations on the group G_{n_j+1} and $2nn_j$ exponentiations on the group G_1. On the other hand, the proxy needs to perform 1 n_j-linear map, 1 exponentiations on the group G_{n_j+1}, $2nn_j$ bilinear pairings. In the *CheckTag* phase, PCS has to compute 1 n_{j_1}-linear map, 1 exponentiations on the group G_{n_j+1}, $2n$ bilinear pairings and n exponentiations on the group G_1. In the *GenProof* phase, PCS needs to perform c exponentiations on the group G_2. In the *CheckProof* phase, the proxy can perform 1 n_j-linear map (it can be pre-computed and stored in the *TagGen* phase), 2 bilinear pairings, and $c+1$ exponentiations on $\mathcal{G}_1$. Compared to the pairings and exponentiation, other operations, such as hashing, permutation, multiplication, *etc.*, are omitted since their costs are negligible.

Communication: The communication overhead mostly comes from the PPDP queries and responses. In PPDP query, the proxy needs to send $\log_2 c$ bits and 2 elements in $\mathcal{Z}_q^*$ to PCS. In the response, the PCS responds 1 element in $\mathcal{G}_1$ and 1 element in $\mathcal{Z}_q^*$ to the proxy. Thus, our PPDP protocol has low communication cost.

Notes: Our proposed PPDP protocol is a general remote data integrity checking method with the general access structure. The idea is motivated by the application requirements which has been given in the subsection 1.1. The existing PDP protocols can only be applied for single client. It is not enough because the multi-client PDP and proxy PDP are also indispensable in some application fields. Of course, single client PDP is only the special case of our protocol when the size of the valid subset is 1 and the proxy is omitted. In general access structure, the PPDP protocol is proposed for the first time. It can be used in many application fields.

4 Conclusion

In this paper, we proposes a PPDP protocol with general access structure. We give its concept, security model, formal security proof and performance analysis. It is shown that our PPDP protocol is provably secure and efficient. It can be used in the public clouds to ensure remote data integrity.

References

1. Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., Song, D.: Provable data possession at untrusted stores. In: Capitani, D., di Vimercati, S., Syverson, P. (eds.) CCS 2007, pp. 598–609. ACM, New York (2007)
2. Ateniese, G., Di Pietro, R., Mancini, L.V., Tsudik, G.: Scalable and efficient provable data possession. In: Liu, P., Molva, R. (eds.) SecureComm 2008, pp. 9:1–9:10. ACM, New York (2008)
3. Erway, C.C., Küpçü, A., Papamanthou, C., Tamassia, R.: Dynamic provable data possession. ACM Trans. Inf. Syst. Secur. **17**(4), 1–29 (2015). 15
4. Sebé, F., Domingo-Ferrer, J., Martinez-Balleste, A., Deswarte, Y., Quisquater, J.J.: Efficient remote data possession checking in critical information infrastructures. IEEE Trans. Knowl. Data Eng. **20**(8), 1034–1038 (2008)
5. Wang, H.: Proxy provable data possession in public clouds. IEEE Trans. Serv. Comput. **6**(4), 551–559 (2013)

6. Wang, H., Wu, Q., Qin, B., Domingo-Ferrer, J.: Identity-based remote data possession checking in public clouds. IET Inf. Secur. **8**(2), 114–121 (2014)
7. Wang, H.: Identity-based distributed provable data possession in multicloud storage. IEEE Trans. Serv. Comput. **8**(2), 328–340 (2015)
8. Curtmola, R., Khan, O., Burns, R., Ateniese, G.: MR-PDP: multiple-replica provable data possession. In: ICDCS 2008, pp. 411–420. IEEE Press (2008)
9. Barsoum, A.F., Hasan, M.A.: Provable possession and replication of data over cloud servers (2010). http://www.cacr.math.uwaterloo.ca/techreports/2010/cacr2010-32.pdf
10. Hao, Z., Yu, N.: A multiple-replica remote data possession checking protocol with public verifiability. In: ISDPE 2010, pp. 84–89. IEEE Press (2010)
11. Barsoum, A.F., Hasan, M.A.: On Verifying Dynamic Multiple Data Copies over Cloud Servers(2011). http://eprint.iacr.org/2011/447.pdf
12. Juels, A., Kaliski Jr., B.S.: PORs: Proofs of retrievability for large files. In: ACM CCS 2007, pp. 584–597. ACM, New York (2007)
13. Shacham, H., Waters, B.: Compact proofs of retrievability. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 90–107. Springer, Heidelberg (2008)
14. Wang, H.: Anonymous multi-receiver remote data retrieval for pay-TV in public clouds. IET Inf. Secur. **9**(2), 108–118 (2014)
15. Wang, H., Wu, Q., Qin, B., Domingo-Ferrer, J.: FRR: fair remote retrieval of outsourced private medical records in electronic health networks. J. Biomed. Inform. **50**, 226–233 (2014)
16. Bowers, K.D., Juels, A., Oprea, A.: Proofs of retrievability: theory and implementation. In: ACM CCSW 2009, pp. 43–54. ACM, New York (2009)
17. Zheng, Q., Xu, S.: Fair and dynamic proofs of retrievability. In: CODASPY 2011, pp. 237–248. ACM, New York (2011)
18. Dodis, Y., Vadhan, S., Wichs, D.: Proofs of retrievability via hardness amplification. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 109–127. Springer, Heidelberg (2009)
19. Zhu, Y., Wang, H., Hu, Z., Ahn, G.J., Hu, H.: Zero-knowledge proofs of retrievability. Sci. China Inf. Sci. **54**(8), 1608–1617 (2011)
20. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
21. Miyaji, A., Nakabayashi, M., Takano, S.: New explicit conditions of elliptic curve traces for FR-reduction. IEICE Trans. Fundam. Electron. commun. comput. sci. **84**(5), 1234–1243 (2001)
22. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)
23. Boneh, D., Silverberg, A.: Applications of multilinear forms to cryptography. Contemp. Math. **324**(1), 71–90 (2003)
24. Huang, M.D., Raskind, W.: A multilinear generalization of the tate pairing. Contemp. Math. **518**, 255–263 (2010)
25. Garg, S., Gentry, C., Halevi, S.: Candidate multilinear maps from ideal lattices. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 1–17. Springer, Heidelberg (2013)
26. Hohenberger, S., Sahai, A., Waters, B.: Full domain hash from (leveled) multilinear maps and identity-based aggregate signatures. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 494–512. Springer, Heidelberg (2013)

27. Freire, E.S.V., Hofheinz, D., Paterson, K.G., Striecks, C.: Programmable hash functions in the multilinear setting. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 513–530. Springer, Heidelberg (2013)
28. Coron, J.-S., Lepoint, T., Tibouchi, M.: Practical multilinear maps over the integers. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 476–493. Springer, Heidelberg (2013)
29. Bagherzandi, A., Jarecki, S.: Identity-based aggregate and multi-signature schemes based on RSA. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 480–498. Springer, Heidelberg (2010)
30. Kawauchi, K., Minato, H., Miyaji, A., Tada, M.: A multi-signature scheme with signers' intentions secure against active attacks. In: Kim, K. (ed.) ICISC 2001. LNCS, vol. 2288, pp. 175–196. Springer, Heidelberg (2002)
31. Kumanduri, R., Romero, C.: Number Theory with Computer Applications, pp. 479–508. Prentice Hall, New Jersey (1998)
32. Miller, V.S.: Use of elliptic curves in cryptography. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 417–426. Springer, Heidelberg (1986)
33. Rivest, R.L., Hellman, M.E., Anderson, J.C., Lyons, J.W.: Responses to NIST's proposal. Commun. ACM **35**(7), 41–54 (1992)