Journal of Cloud Computing: Advances, Systems and Applications

RESEARCH Open Access

An efficient and secure data sharing scheme for mobile devices in cloud computing

Xiuqing Lu[1,2], Zhenkuan Pan[1*] and Hequn Xian[1]

Abstract

With the development of big data and cloud computing, more and more enterprises prefer to store their data in cloud and share the data among their authorized employees efficiently and securely. So far, many different data sharing schemes in different fields have been proposed. However, sharing sensitive data in cloud still faces some challenges such as achieving data privacy and lightweight operations at resource constrained mobile terminals. Furthermore, most data sharing schemes have no integrity verification mechanism, which would result in wrong computation results for users. To solve the problems, we propose an efficient and secure data sharing scheme for mobile devices in cloud computing. Firstly, the scheme guarantees security and authorized access of shared sensitive data. Secondly, the scheme realizes efficient integrity verification before users share the data to avoid incorrect computation. Finally, the scheme achieves lightweight operations of mobile terminals on both data owner and data requester sides.

Keywords: Big data, Security, Integrity, Cloud computing

Introduction

With the rapid development of information technology and Internet of Things (IoT), enterprises generate more and more big data, which needs to be stored and processed efficiently and securely. Cloud computing is a developed storage platform and has many advantages including low cost and scalability [1–3]. Therefore, many enterprises and individuals are apt to outsource their data to cloud for storage and sharing with authorized data requesters. For example, in a cloud based health information system, patients upload their health information to cloud for sharing with medical experts to diagnose diseases. Similarly, the manager of an enterprise not only want to store the big data in cloud, but also want to share the data among their authorized employees wherever needed. Outsourcing data for sharing in cloud not only saves local storage space, but also greatly reduces the cost of enterprises in software purchase and hardware maintenance [4–6]. Although people take advantages of this new technology and service, their concerns about data security arises as well. Security problem in cloud is the most critical issue because of the valuable information that data owners share. Cloud providers should address privacy and security issues as a matter of high and urgent priority [7–9]. One of the prominent security concerns in data sharing is data privacy. In addition, terminals of users are usually resource-constrained mobile devices with small storage space and low processing speed. Therefore, it is essential to propose an efficient and secure data sharing scheme for mobile devices in cloud computing.

Main contributions

We propose a lightweight and secure sensitive data sharing scheme for mobile devices in cloud computing. The main contributions of the paper are as follows.

* Correspondence: panzhenkuan@126.com
[1]College of Computer Science and Technology, Qingdao University, Qingdao 266071, China
Full list of author information is available at the end of the article

1) We design an efficient integrity mechanism based on algebraic signature for sensitive data before data sharing.
2) We guarantee privacy preserving of sensitive data in sharing process and access authorization control for data requesters.
3) We achieve lightweight computation operations on data owner and data requester sides with mobile devices.

Organization

The rest of paper is organized as follows. Section II introduces the related works in secure data sharing. Section III describes architecture and security requirements. Section IV presents the definitions and preliminaries. In section V, we describe the implementations of the efficient data sharing scheme. We analyze security in section VI and evaluate the scheme performance in Section VII. Finally, we conclude this paper in Section VIII.

The related works

With more and more sensitive information sharing among enterprise employees, preserving data integrity and guaranteeing access control for only authorized users to access the data has become the core security problem. Therefore, security problems of data sharing in cloud mainly focus on access control and data integrity [10–14].

At present, data sharing schemes mainly employ access control mechanism to achieve authorized access. Akl and Taylor [15] proposed use of cryptography to realize access control in hierarchical structures. As data owner and the data center are not in the same trusted domain in cloud storage system, access control schemes employing Attributed Based Encryption (ABE) [16] are put forward. ABE commonly comes in Key Policy ABE (KP-ABE) and Cipher text Policy ABE (CP-ABE). KP-ABE uses attributes to describe the encryption data and builds policies into user's key [17]. CP-ABE uses attributes to describe user's credentials and the user encrypting the data determines a policy on who can decrypt the data [18]. Lewko [19] proposed the first unbounded KP-ABE scheme. Waters [20] firstly put forward a fully expressed CP-ABE scheme in the standard model. Cheung and Newport [21] proposed another CP-ABE scheme and proved its security in the standard model. To ensure data security in smart health, Zhang and Zhen [22] realize the fine-grained access control, cipher text-policy attribute based encryption (CP-ABE). To achieve privacy and authorized access, many other schemes [23–33] are proposed in diverse fields and applications.

To verify integrity of data in cloud, many integrity verification schemes [26, 34–42] have been proposed in recent years. Ateniese [34] proposed the first public auditing scheme, which allows any public verifier to check the data integrity. Later, to prove the integrity of dynamic data, Ateniese [26] proposed another scheme based on the symmetric key provable data possession (PDP) scheme. To support dynamic operation of data, Erway et al. [35] proposed a dynamic provable data possession (DPDP) scheme by introducing an authenticated skip list. Later many auditing schemes are proposed by using the authenticated data structure to support dynamic data update. Zhu et al. [36] introduced an index-hash table (IHT) for dynamic verification. Yang et al. [37] proposed another authenticated data structure called index table (ITable) to store the abstract information of blocks. Tian [38] proposed a data structure named Dynamic-Hash-Table (DHT). Wang et al. [39] and Liu et al. [40] respectively proposed dynamic public auditing schemes based on Merkle Hash Tree (MHT). The two schemes can achieve both public verification and dynamic data operations. However, block signatures are generated by users, which would incur multitude computation and communication overhead on the user side. In 2018, Gan [41] proposed an auditing scheme on algebraic signatures that can achieve lower computation and communication costs. The properties of algebraic signature allows cloud to return a sum of the selected blocks in the proof instead of the original data file, which saves the bandwidth between the cloud and the verifier and make the algebraic signature suited for cloud computing [42]. Nowadays, more and more schemes [23, 43–52] on cloud computing security are put forward to achieve great advantages.

Architecture and security requirement

System model

The system model consists of four entities named Key Generator Center (*KGC*), Data Owner (*DO*), Cloud Servers (*CS*) and Data Requester (*DR*) as depicted in Fig. 1.

Key generation center (KGC)

It is responsible for generating public parameters and master key for the system and issuing private key for other entities.

Data owner (DO)

It is responsible to generate and encrypt the shared data, define access structures, and divide encrypted data into blocks.

Cloud servers (CS)

Cloud servers comes in Cloud Storage Servers (*CSS*) and Cloud Manage Servers (*CMS*) based on their roles. CSS is responsible for storing shared data, block tags and

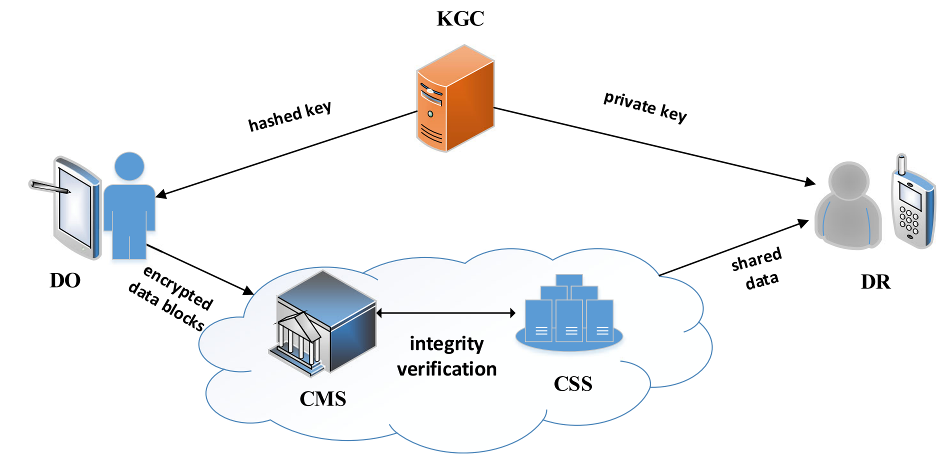

Fig. 1 System Architecture

supply the data integrity proof. To save computation and communication costs of mobile terminals of *DO* and *DR*, *CMS* is employed to manipulate complex computations including generating algebraic signatures of blocks, verifying data integrity of shared data and computing the intermediate data for encryption and decryption.

Data Requester (DR): It is responsible to download and decrypt the shared data for utilization. In the scheme, only the authorized *DR* is able to download shared data from *CSS* and decrypt the data.

In our secure data sharing scheme for mobile terminal devices, DO has large sensitive data to share with legitimate DR. Before sharing, DO encrypts the data with his private key and outsources the data to CSS. If a DR wants to access the data, he must register his identity to KGC and obtain his private key for decryption. To achieve authorized access, only legitimate DRs with correct attributes can download and utilize the shared data. To ensure cloud data intact and decrease computation burden of requesters, CMS helps DR to verify the integrity of data before sharing. Only when data is undamaged, DR downloads and decrypts shared data with his private key.

Security requirement

In the scheme, we suppose *CSS* and *CMS* are both semi-trusted. *CSS* is responsible to store data and block tags for data sharing. However, once data is corrupt or lost, it might launch forge attack or replace attack for economic reasons. Similarly, *CMS* is curious about the content of sensitive data, so the data should preserve secret to *CMS*. In the scheme, we assume *KGC* is a fully trusted authority and can honestly generate private key for the system and other entity. Therefore, the following security requirements of the scheme should be satisfied.

Data confidentiality

The shared data must keep confidential to *CSS*, *CMS* and any unauthorized *DR*s for privacy and security. Any disclosure of shared data is undoubtedly harmful to enterprise benefits. Consequently, it is important to ensure the confidentiality of shared data.

Data integrity

The data should keep intact before shared by DR. It means that the data is undamaged in an unauthorized manner during storage and sharing process.

Authorized access

To achieve authorization, only *DR* with correct attributes can access shared data stored in *CSS*.

User revocation

The membership of *DR* must be revoked to stop his access to shared data when he leaves the organization. To achieve security of the scheme, user revocation should be required in the data sharing scheme.

Design goals

The data sharing scheme for mobile devices is designed to achieve data privacy preservation, data security and lightweight operations.

Privacy preservation

The scheme should satisfy data privacy during data sharing process. As sensitive data is encrypted by data owner before outsourcing to cloud and only authorized data requesters can access the encrypted data, the shared data is private to CSS, CMS and any unauthorized DRs.

Data security

The scheme should achieve sensitive data security during the whole sharing process. The security requirement can be guaranteed by data confidentiality, data integrity, authorized access and user revocation in the scheme.

Lightweight operations

The scheme should decrease computation operations of *DO* and *DR* for efficiency. In our scheme, *CMS* is responsible to divide encrypted data into blocks and computes block tags. Furtherly, when *DR* wants to access shared data, *CMS* compute intermediate data of decryption to less *DR*'s computation burden.

Definitions and preliminaries

Denifitions

1) Discrete Logarithm (DL) Assumption. Suppose $\boldsymbol{g}$ is a generator of multiplicative cyclic group $\mathbb{G}$ with prime order $\boldsymbol{q}$. On input $\boldsymbol{y} \in \mathbb{G}$, there does not exist probabilistic polynomial time algorithm that outputs a value $\boldsymbol{x} \in \boldsymbol{Z}_q^*$ such that $\boldsymbol{g}^{\boldsymbol{x}} = \boldsymbol{y}$ with non-negligible probability.
2) Computational Diffie-Hellman (CDH) Assumption. Suppose $\boldsymbol{g}$ is a generator of multiplicative cyclic group $\mathbb{G}$ with prime order $\boldsymbol{q}$. On input $\boldsymbol{g}^{\boldsymbol{x}}, \boldsymbol{g}^{\boldsymbol{y}} \in \mathbb{G}$, there does not exist probabilistic polynomial time algorithm that outputs $\boldsymbol{g}^{\boldsymbol{xy}} \in \mathbb{G}$ with non-negligible probability.
3) Access Structure. Suppose $\boldsymbol{P} = \{\boldsymbol{P}_1, \boldsymbol{P}_2, \ldots, \boldsymbol{P}_n\}$ is a set of parties. A collection of $\boldsymbol{W} \subseteq \boldsymbol{2}^{\boldsymbol{P}}$ is monotone if $\forall \boldsymbol{B}, \boldsymbol{C} : \boldsymbol{B} \in \boldsymbol{W}$ and $\boldsymbol{B} \subseteq \boldsymbol{C}$ then $\boldsymbol{C} \in \boldsymbol{W}$. An access structure is the collection $\boldsymbol{W}$ with non-empty subsets of $\boldsymbol{P}$, i.e., $\boldsymbol{W} \subseteq \boldsymbol{2}^{\boldsymbol{P}} \backslash \{\varnothing\}$. The sets in $\boldsymbol{W}$ are named as authorized sets, and the sets not in $\boldsymbol{W}$ are named as the unauthorized sets.

Preliminaries

1. Linear secret-sharing schemes (LSSS). LSSS is a share-generating matrix $\boldsymbol{A}$ with rows labeled by attributes. Assume $\boldsymbol{S} \in \boldsymbol{A}$ is an authorized set and $\boldsymbol{I}$ is defined as $\boldsymbol{I} = \{\boldsymbol{i} \mid \boldsymbol{\rho}(\boldsymbol{i}) \in \boldsymbol{S}\}$. Then there exists constants $\boldsymbol{\omega}_i \in \boldsymbol{Z}_q$ satisfying $\sum_{i \in I} \boldsymbol{\omega}_i \boldsymbol{\lambda}_i = \boldsymbol{s}$ where $\boldsymbol{\lambda}_i$ is valid share of secret share $\boldsymbol{s}$. suppose $\boldsymbol{A}_i$ is the $\boldsymbol{i}^{\boldsymbol{th}}$ row of $\boldsymbol{A}$, the equation $\sum_{i \in I} \boldsymbol{\omega}_i \boldsymbol{A}_i = (\boldsymbol{1}, \boldsymbol{0}, \ldots, \boldsymbol{0})$ also satisfies.
2. Algebraic signature. The algebraic signature of a file block composed of strings $\boldsymbol{s}_0, \boldsymbol{s}_1, \ldots, \boldsymbol{s}_{n-1}$ is defined as $\boldsymbol{sig}_g(\boldsymbol{s}_0, \boldsymbol{s}_1, \ldots, \boldsymbol{s}_{n-1}) = \sum_{i=0}^{n-1} \boldsymbol{s}_i \cdot \boldsymbol{g}^i$. The algebraic signature has the following two properties: i) the algebraic signature of a combination of file $\boldsymbol{F}_1$ and file $\boldsymbol{F}_2$ is defined as $\boldsymbol{sig}_g(\boldsymbol{F}_1 \oplus \boldsymbol{F}_2) = \boldsymbol{sig}_g(\boldsymbol{F}_1) \oplus \boldsymbol{sig}_g(\boldsymbol{F}_2)$. Ii) the algebraic signature of a combination of $\boldsymbol{n}$ blocks in file $\boldsymbol{F}$ is equal to the combination of algebraic signatures of each block named $\boldsymbol{m}_i \in \mathbb{G}$, which is described as $\sum_{i=1}^{n} \boldsymbol{sig}_g(\boldsymbol{m}_i) = \boldsymbol{sig}_g \sum_{i=1}^{n} \boldsymbol{m}_i$.
3. XOR-homomorphic function. A XOR-homomorphic function $\boldsymbol{h}$ is a pseudo-random function that can ensure data privacy. Its properties is as follows. For any inputs $\boldsymbol{x}, \boldsymbol{y}$, there exists $\boldsymbol{h}(\boldsymbol{x} \oplus \boldsymbol{y}) = \boldsymbol{h}(\boldsymbol{x}) \oplus \boldsymbol{h}(\boldsymbol{y})$.
4. Bilinear maps. Suppose $\mathbb{G}_1, \mathbb{G}_2$ are two multiplicative groups with same large prime order $\boldsymbol{q}$, and $\boldsymbol{g}$ is a generator of $\mathbb{G}_1$. A bilinear map $\boldsymbol{e}$ is a map function $\boldsymbol{e}: \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_1$ with the following properties: i) computability. $\forall \boldsymbol{u}, \boldsymbol{v} \in \mathbb{G}_1$, an efficient algorithm exists to compute $\boldsymbol{e}(\boldsymbol{u}, \boldsymbol{v})$. Ii) Binearity. $\forall \boldsymbol{a}, \boldsymbol{b} \in \boldsymbol{Z}_q, \exists\, \boldsymbol{e}(\boldsymbol{u}^{\boldsymbol{a}}, \boldsymbol{v}^{\boldsymbol{b}}) = \boldsymbol{e}(\boldsymbol{u}, \boldsymbol{v})^{\boldsymbol{ab}}$. Iii) nondegeneracy. $\boldsymbol{e}[\boldsymbol{g}, \boldsymbol{g}] \neq \boldsymbol{1}$. Iv) security. It is hard to compute discrete logarithm (DL) in $\mathbb{G}_1$.

Notations

Table 1 describes the main notations in the scheme.

Scheme implementations

In this section, we present the efficient and secure data sharing scheme for mobile devices in cloud computing in detail. We divide the sharing scheme into four phases named initial phase, data processing phase, integrity verification phase and data sharing phase.

Table 1 Important notations

Notation	Meaning
λ	Security parameter
A	Attribute universe
$\mathbb{G}_1, \mathbb{G}_2$	Multiplicative groups
q	Prime order of group
sig_g	Algebra signature
PK, MK	Public and master keys
S	Attribute of DR
SK	Private key of DR
F, F′	Plain text of file and encrypted file
T_i	Block tag
DP, TP	Data proof and tag proof
AS	Access policy
m_i	Data block
Uid	Identity of user
Usk, Upk	Private and public key of user

Initial phase

This phase consists of three algorithms named ***Para-Setup,KeyGen,IdReg***. Algorithm ***ParaSetup*** is mainly responsible to generate system parameters before data sharing. Algorithm ***KeyGen*** is mainly used to obtain the private key for DR to decrypt the cipher-text of shared data. Algorithm ***IdReg*** is responsible for registering DR's identity information in a table for checking the validity of DR. Figure 2 describes the data flow of the phase.

1) $ParaSetup(\lambda, A) \to (PK, MK)$:It is run by KGC. Given system security parameter λ, KGC constructs the bilinear map group system $\Theta = (\mathbb{G}_1, \mathbb{G}_2, q, e)$ where $\mathbb{G}_1, \mathbb{G}_2$ are multiplicative groups with prime order q, and e is a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. Suppose A is the attribute universe A whose attribute number is $|A|$. KGC picks random $\alpha, \beta, s_1, s_2, \ldots, s_{|A|} \in Z_q^*$ and computes $\theta = e(g, g)^\alpha$, $\sigma = g^\beta$. Then KGC defines heteromorphic function $h : \mathbb{G}_1 \to \mathbb{G}_1$ and algebraic signature $sig_g(m_i) = m_i \cdot g^i$, where $m_i \in \mathbb{G}_1$ and g is a generator of $\mathbb{G}_1$. Next, KGC selects three secure hash function $H_1 : \{0,1\}^* \to \mathbb{G}_1, H_2 : \mathbb{G}_1 \to \{0,1\}^{len}, H_3 : \{0,1\}^* \to Z_q^*$. KGC publishes public parameters $PK = (e, g, H_1, H_2, H_3, h, sig_g, \theta, \sigma, f_i = g^{s_i},_{1 \le i \le |A|})$ and keeps master key $MK = (\alpha, \beta)$ secretly.

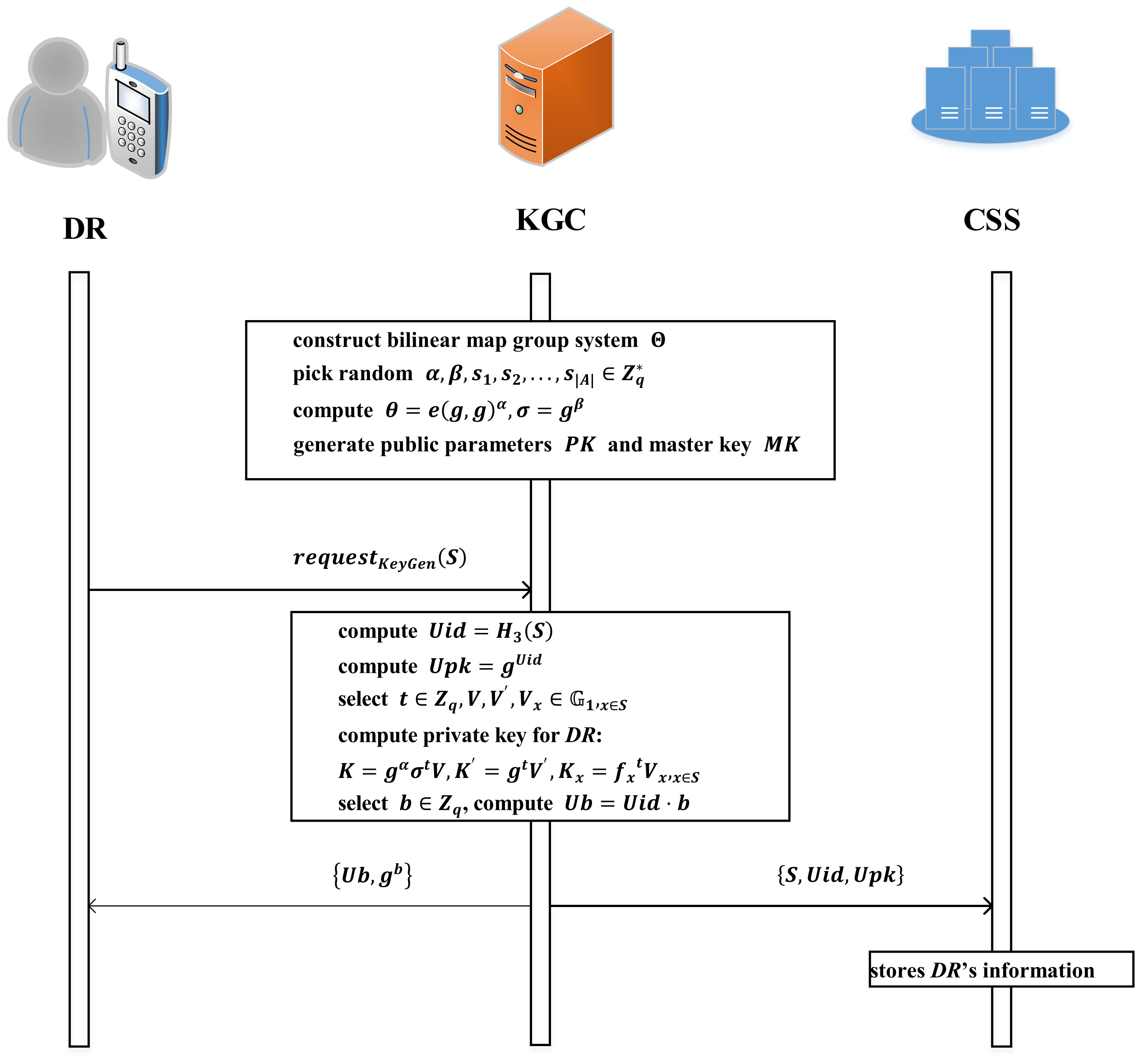
DR
KGC
CSS
construct bilinear map group system Θ
pick random $\alpha, \beta, s_1, s_2, \ldots, s_{|A|} \in Z_q^*$
compute $\theta = e(g,g)^\alpha, \sigma = g^\beta$
generate public parameters PK and master key MK
$request_{KeyGen}(S)$
compute $Uid = H_3(S)$
compute $Upk = g^{Uid}$
select $t \in Z_q, V, V', V_x \in \mathbb{G}_{1, x \in S}$
compute private key for DR:
$K = g^\alpha \sigma^t V, K' = g^t V', K_x = f_x{}^t V_{x, x \in S}$
select $b \in Z_q$, compute $Ub = Uid \cdot b$
$\{Ub, g^b\}$
$\{S, Uid, Upk\}$
stores DR's information

Fig. 2 Dataflow of Initial Phase

2) $\boldsymbol{KeyGen}(\boldsymbol{MK}, \boldsymbol{S}) \rightarrow (\boldsymbol{Usk})$**:** It is run by *KGC*. Before *DR* with attribute $\boldsymbol{S}$ share the data, he should get the private key to decrypt the cipher text of shared data. *DR* first sends key generation request $\boldsymbol{request_{KeyGen}}(\boldsymbol{S})$ to *KGC*. After receiving the request, *KGC* computes $\boldsymbol{Uid} = \boldsymbol{H_3}(\boldsymbol{S})$ as the *DR* identity and computes $\boldsymbol{Upk} = \boldsymbol{g^{Uid}}$ as *DR*'s public key. Next *KGC* selects $t \in Z_q, V, V', V_x \in \mathbb{G}_{1}, x \in S$ and computes the private key $\boldsymbol{Usk}$ for *DR* as follows: $K = g^{\alpha}\sigma^{t}V, K' = g^{t}V', K_x = f_x{}^{t}V_x, x \in S$. *KGC* randomly selects $b \in Z_q$ and computes $\boldsymbol{Ub} = \boldsymbol{Uid} \cdot \boldsymbol{b}$ and sends $\{\boldsymbol{Ub}, \boldsymbol{g^b}\}$ to *CMS* for later integrity verification . Then, *KGC* sends $\{\boldsymbol{Uid}, \boldsymbol{Usk}\}$ to *DR* and $\{\boldsymbol{S}, \boldsymbol{Uid}, \boldsymbol{Upk}\}$ to *CSS*.
3) $\boldsymbol{IdReg}(\boldsymbol{S}, \boldsymbol{Uid}) \rightarrow \boldsymbol{DRTable}$. It is run by *CSS*. To verify the validity of *DR* before transferring the shared data, *CSS* stores *DR*'s information including identity $\boldsymbol{Uid}$, attribute set $\boldsymbol{S}$ and public key $\boldsymbol{Upk}$ in a table named $\boldsymbol{DRTable}$. If identity of DR is valid, *CSS* transfers shared data to him. Otherwise, *CSS* refuses the download request of *DR*.

Data processing phase

The phase includes two algorithms named ***DataEnc***, ***TagGen*** and ***InterEnc***. To achieve confidentiality, algorithm ***DataEnc*** encrypts shared data and divides the data into blocks. To ensure the data is intact, algorithm ***TagGen*** generates block tag for each block. To decrease DO's computation burden, algorithm ***InterEnc*** helps DO computer the intermediate value for encryption. Fig. 3 describes the data flow of the phase.

1) $DataEnc(F) \rightarrow (F')$: It is run by *DO*. To ensure only authorized *DR* have access to shared data, *DO*

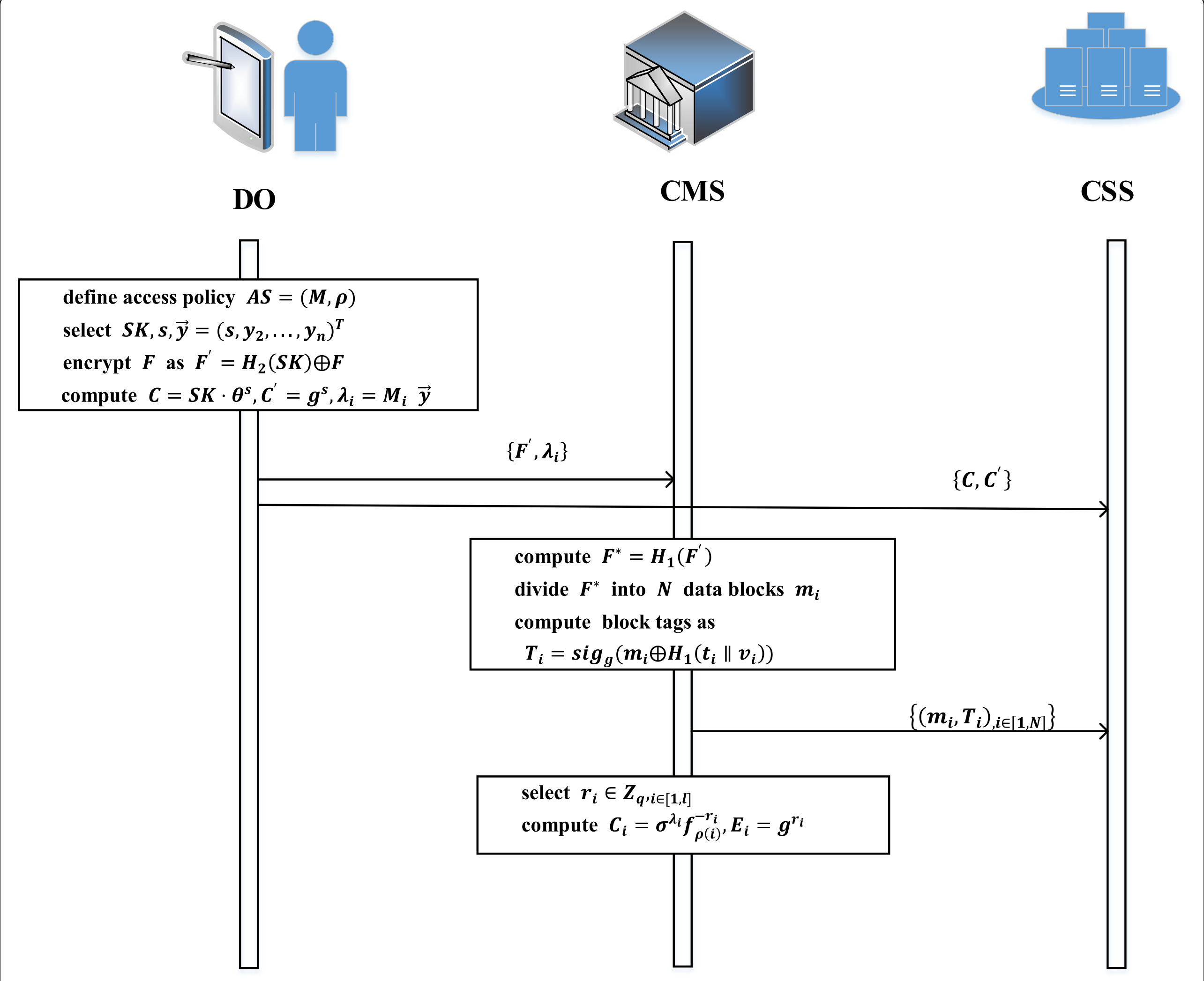
DO
CMS
CSS
define access policy $AS = (M, \rho)$
select $SK, s, \vec{y} = (s, y_2, \dots, y_n)^T$
encrypt F as $F' = H_2(SK) \oplus F$
compute $C = SK \cdot \theta^s, C' = g^s, \lambda_i = M_i \vec{y}$
$\{F', \lambda_i\}$
$\{C, C'\}$
compute $F^* = H_1(F')$
divide F^* into N data blocks m_i
compute block tags as
$T_i = sig_g(m_i \oplus H_1(t_i \parallel v_i))$
$\{(m_i, T_i)_{,i \in [1,N]}\}$
select $r_i \in Z_{q, i \in [1,l]}$
compute $C_i = \sigma^{\lambda_i} f_{\rho(i)}^{-r_i}, E_i = g^{r_i}$

Fig. 3 Data flow of data processing phase

adefines access policy $AS = (M, \rho)$, where M is a $l \times n$ matrix and ρ is the function that maps each row of the matrix to an associated attribute. That is, for $i \in [1, l]$, the value $\rho(i)$ is the attribute associated with row i. Suppose the shared data is $F \in \{0,1\}^{len}$, where len is the max length of F and the identity is Fid. DO selects key $SK \in \mathbb{G}_1, s \in Z_q$ and a random column vector $\overrightarrow{y} = (s, y_2, \ldots, y_n)^T \in Z_q^n$. Next, he encrypts F as $F' = H_2(SK) \quad F$ and computes $C = SK \cdot \theta^s, C' = g^s, \lambda_i = M_i \overrightarrow{y}, _{1 \le i \le l}$. Finally, DO sends $\{F', \lambda_i\}$ to CMS and $\{C, C'\}$ sends to CSS.

2) $TagGen(F') \rightarrow (T_i)$: It is run by CMS. CMS computes $F^* = H_1(F')$ and divides F^* into N data blocks named m_i. Suppose t_i, v_i represent the recent timestamp and version of each block m_i. CMS computes block tags as follows

$$T_i = sig_g(m_i \oplus H_1(t_i \| v_i)) \tag{1}$$

and sends $\{(m_i, T_i)_{, i \in [1,N]}\}$ to CSS.

3) $InterEnc(M) \rightarrow (C_i, E_i)$. It is run by CMS. To decrease the computation burden of mobile devices at DO side, CMS helps to compute the intermediate encryption data. He selects $r_i \in Z_{q}, _{i \in [1,l]}$ and computes $C_i = \sigma^{\lambda_i} f_{\rho(i)}^{-r_i}, E_i = g^{r_i}$. For later integrity verification, CMS stores the intermediate data C_i, , E_i locally.

Integrity verification phase

When DR wants to access shared data, he first sends integrity request to CMS for verifying whether the

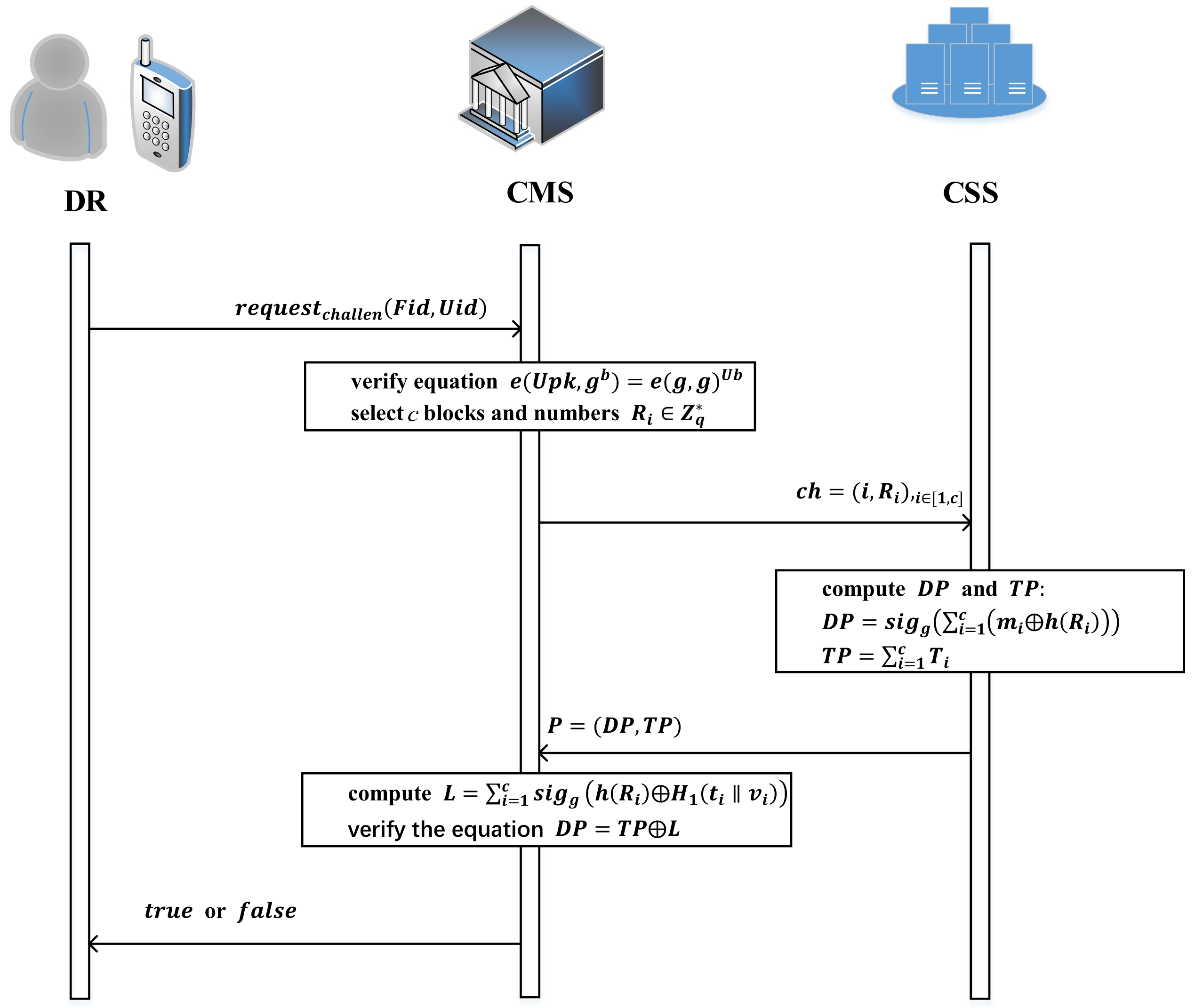
DR
CMS
CSS
$request_{challen}(Fid, Uid)$
verify equation $e(Upk, g^b) = e(g,g)^{Ub}$
select c blocks and numbers $R_i \in Z_q^*$
$ch = (i, R_i)_{, i \in [1,c]}$
compute DP and TP:
$DP = sig_g(\sum_{i=1}^{c}(m_i \oplus h(R_i)))$
$TP = \sum_{i=1}^{c} T_i$
$P = (DP, TP)$
compute $L = \sum_{i=1}^{c} sig_g(h(R_i) \oplus H_1(t_i \| v_i))$
verify the equation $DP = TP \oplus L$
$true$ or $false$

Fig. 4 Integrity Verification Phase

data is intact. Then *CMS* generates integrity challenge ***ch*** and sends it to *CSS*. After getting the challenge, *CSS* computes data proof ***P*** and sends it to *CMS* to verify the integrity. This phase consists of the following three algorithms and Fig. 4 describes the data flow of the phase.

1) *ChalGen*(*Fid*) → (*ch*): Before downloading shared data, *DR* first sends request $request_{challen}(Fid, Uid)$ to *CMS* for integrity verification. After receiving the request, *CMS* verifies the validity of *DR*'s identity with equation $e(Upk, g^b) = e(g, g)^{Ub}$. If the equation does not hold, *CMS* rejects the request. Otherwise, *CMS* randomly selects *c* blocks and corresponding random numbers $R_i \in Z_q^*$. Then *CMS* sends challenge $ch = (i, R_i)_{,i \in [1, c]}$ to *CSS*.
2) *ProfGen*(T_i, F') → (*P*): After receiving *ch* from *CMS*, *CSS* computes data proof and tag proof

$$DP = sig_g\left(\sum_{i=1}^{c}(m_i \oplus h(R_i))\right) \tag{2}$$

$$TP = \sum_{i=1}^{c} T_i. \tag{3}$$

Then he sends proof *P* = (*DP*, *TP*) to *CMS*.

3) *IntegityVer*(*P*) → (*true*, *false*).After getting the proof *P* , *CMS* computes $L = \sum_{i=1}^{c} sig_g(h(R_i) \oplus H_1(t_i \| v_i))$ and verifies whether the following equation holds.

$$DP = TP \oplus L \tag{4}$$

If the equation holds, it indicates that the data is intact and outputs *true*, Otherwise, *CMS* outputs *false*.

Data sharing phase

If the shared data is intact, DR downloads and decrypts ***F'***. This phase consists of the following four algorithms and Fig. 5 describes the data flow of the phase.

1) *InvalidyCheck*(*S*, *Uid*) → (*true*, *false*). It is run by *CSS*. Before downloading shared data F', *DR* sends

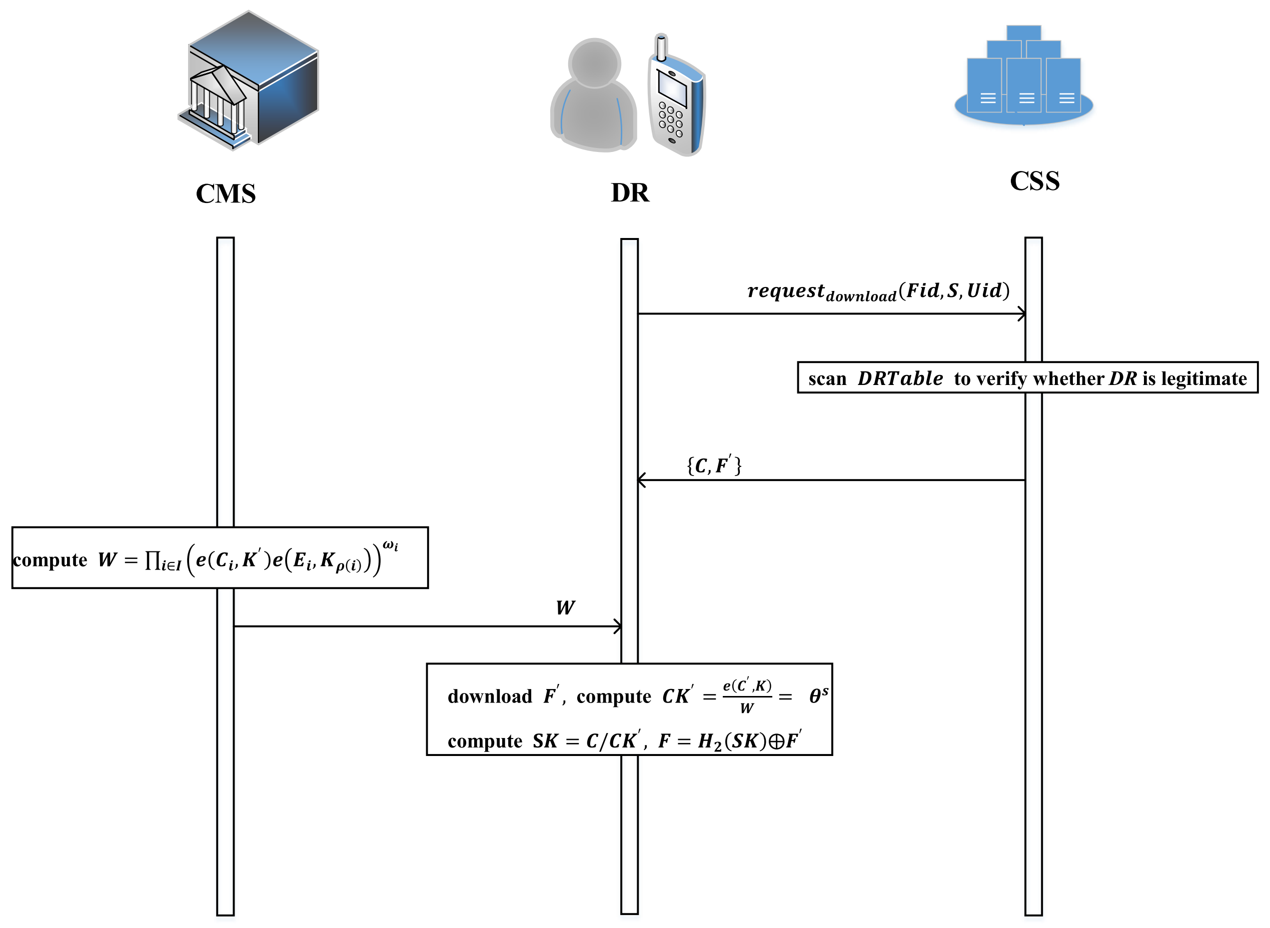
CMS
DR
CSS
$request_{download}(Fid, S, Uid)$
scan $DRTable$ to verify whether DR is legitimate
$\{C, F'\}$
compute $W = \prod_{i \in I}\left(e(C_i, K')e(E_i, K_{\rho(i)})\right)^{\omega_i}$
W
download F', compute $CK' = \frac{e(C', K)}{W} = \theta^s$
compute $SK = C/CK'$, $F = H_2(SK) \oplus F'$

Fig. 5 Data Flow of Sharing Phase

request $request_{download}(Fid, S, Uid)$ to *CSS*. Then *CSS* scans *DRTable* to verify whether *DR* is legitimate. If *DR* is in *DRTable*, *CSS* transfers $\{C, F'\}$ to *DR*. Otherwise, *CSS* refuses the request.

2) $InterDec(\ C_i, E_i, K', K_{\rho(i)}) \rightarrow (W)$. If the shared data is intact, *CMS* computes the intermediate value for *DR* to decrease the computation burden of *DR* side. Let $=\{i : \rho(i) \in S\} \subseteq [1, l]$, where S is the attribute of *DR*. Suppose $\omega_i \in Z_{q},_{i\ \in\ I}$ is constant and satisfies the equation $\sum_{i\ \in\ I}\omega_i A_i = (1, 0, \ldots, 0)$. *CMS* computes $W = \prod_{i\in I}(e(C_i, K')e(E_i, K_{\rho(i)}))^{\omega_i}$ and sends W to *DR* securely.

3) $Decryption(\ F') \rightarrow (F)$: If *DR* is legitimate, he downloads F' and computes the follows.

$$CK' = \frac{e(C', K)}{W} = \theta^s \tag{5}$$

Then *DR* computes $SK = C/CK'$, $F = H_2(SK) \oplus F'$ and recovers the plaintext of shared data .

4) ***Revocation***$(\ \boldsymbol{Uid}) \rightarrow (\boldsymbol{DRTable})$. *DR* can be revoked after leaving the organization. After revocation of *DR*, *DSS* deletes *DR*'s information from ***DRTable*** and *DR* cannot download the shared file later.

Security analysis

In this section, we analyze the security of the scheme, including correctness, unforgeability and privacy.

Theorem 1. Authorized *DR* can correctly verify the integrity of the data stored in *CSS*.

Proof. Theorem 1 can be proved by verifying the correctness of eq. (4). The proof is as follows.

$$\begin{aligned}
&\boldsymbol{TP}\oplus\boldsymbol{L}\\
&=\sum\nolimits_{i=1}^{c}\boldsymbol{T}_i\oplus\sum\nolimits_{i=1}^{c}\boldsymbol{sig}_g(\boldsymbol{h}(\boldsymbol{R}_i)\oplus\boldsymbol{H}_1(\boldsymbol{t}_i\|\boldsymbol{v}_i))\\
&=\sum\nolimits_{i=1}^{c}\boldsymbol{sig}_g(\boldsymbol{m}_i\oplus\boldsymbol{H}_1(\boldsymbol{t}_i\|\boldsymbol{v}_i))\oplus\boldsymbol{sig}_g(\boldsymbol{h}(\boldsymbol{R}_i)\oplus\boldsymbol{H}_1(\boldsymbol{t}_i\|\boldsymbol{v}_i))\\
&=\sum\nolimits_{i=1}^{c}\boldsymbol{sig}_g(\boldsymbol{m}_i\oplus\boldsymbol{h}(\boldsymbol{R}_i))\\
&=\boldsymbol{sig}_g\left(\sum\nolimits_{i=1}^{c}(\boldsymbol{m}_i\oplus\boldsymbol{h}(\boldsymbol{R}_i))\right)\\
&=\boldsymbol{DP}
\end{aligned}$$

From the proof of eq. (4), *DR* can verify whether the data is undamaged stored in *CSS*.

Theorem 2. Authorized *DR* can correctly recover the shared data if he owns the legal attributes.

Proof. Theorem 2 can be proved by verifying the correctness of eq. (5). The proof is as follows.

$$\begin{aligned}
\mathrm{CK}' &= \frac{\mathrm{e}(\mathrm{C}', \mathrm{K})}{\mathrm{W}}\\
&= \frac{\mathrm{e}(\mathrm{C}', \mathrm{K})}{\prod_{\mathrm{i}\in\mathrm{I}}\left(\mathrm{e}(\mathrm{C_i}, \mathrm{K}')\mathrm{e}(\mathrm{E_i}, \mathrm{K}_{\rho(\mathrm{i})})\right)^{\omega_\mathrm{i}}}\\
&= \frac{\mathrm{e}(\mathrm{g^s}, \mathrm{g}^\alpha\sigma^\mathrm{t}\mathrm{V})}{\prod_{\mathrm{i}\in\mathrm{I}}\left(\mathrm{e}(\mathrm{C_i}, \mathrm{g^t}\mathrm{V}')\mathrm{e}(\mathrm{E_i}, \mathrm{K}_{\rho(\mathrm{i})})\right)^{\omega_\mathrm{i}}}\\
&= \frac{\mathrm{e}(\mathrm{g^s}, \mathrm{g}^\alpha\sigma^\mathrm{t}\mathrm{V})}{\prod_{\mathrm{i}\in\mathrm{I}}\left(\mathrm{e}\left(\sigma^{\lambda_\mathrm{i}}\mathrm{f}_{\rho(\mathrm{i})}^{-\mathrm{r_i}}, \mathrm{g^t}\mathrm{V}\right)\mathrm{e}\left(\mathrm{g}^{\mathrm{r_i}}, \mathrm{f}_{\rho(\mathrm{i})}{}^{\mathrm{t}}\mathrm{V}_{\rho(\mathrm{i})}\right)\right)^{\omega_\mathrm{i}}}\\
&= \mathrm{e}(\mathrm{g}, \mathrm{g})^{\alpha\mathrm{s}}\\
&= \theta^\mathrm{s}
\end{aligned}$$

Then *DR* computes $\boldsymbol{SK} = \boldsymbol{C}/\boldsymbol{CK}'$,$\boldsymbol{F} = \boldsymbol{H}_2(\boldsymbol{SK}) \oplus \boldsymbol{F}'$ to recover the plaintext of shared data.

Theorem 3. It is computationally infeasible for *CSS*, *CMS* and unauthorized *DR* to get the plaintext of health data in the scheme.

Proof. In data processing phase, *DO* encrypts file $\boldsymbol{F}$ to $\boldsymbol{F}'$ with $\boldsymbol{F}' = \boldsymbol{H}_2(\boldsymbol{SK}) \oplus \boldsymbol{F}$, where $\boldsymbol{SK}$ is only secret to *DO*. Therefore, the file is confidential to both *CSS* and *CMS*. The confidentiality guarantee depends on the security of hash function $\boldsymbol{H}_2$. As $\boldsymbol{H}_2$ is a secure one-way hash function, the data is private to *CSS* and *CMS*. In data sharing phase, *CSS* sends $\{\boldsymbol{C}, \boldsymbol{F}'\}$ to *DR*, where $\boldsymbol{C} = \boldsymbol{SK}\cdot\boldsymbol{\theta}^s$ and $\boldsymbol{F}'$ is the cipher text of shared data. *CMS* computes the intermediate value for *DR* to decrypt the shared data $\boldsymbol{F}'$ only if *DR's* attributes satisfy the access structure. Therefore, any unauthorized *DR* cannot get any information on the sensitive data.

Theorem 4. It is computationally impossible for *CSS* to forge an integrity proof to pass the public verification, if XOR-homomorphic function is secure.

Proof. We can prove the theorem with the following games. In the games, we suppose the adversary is the party who forge an integrity proof to pass the public verification.

Game 1 is the challenge game. The challenger generates public-private key pair (PK, MK) and provides PK to the adversary. The adversary is able to interact with the challenger and query some data blocks. Then the

Table 2 Computation overhead of DO and DR

Entity	Computation overhead
Do	$(1+l)Mul + 2Exp + Hash + Xor$
DR	$Pair + Xor + 2Div$

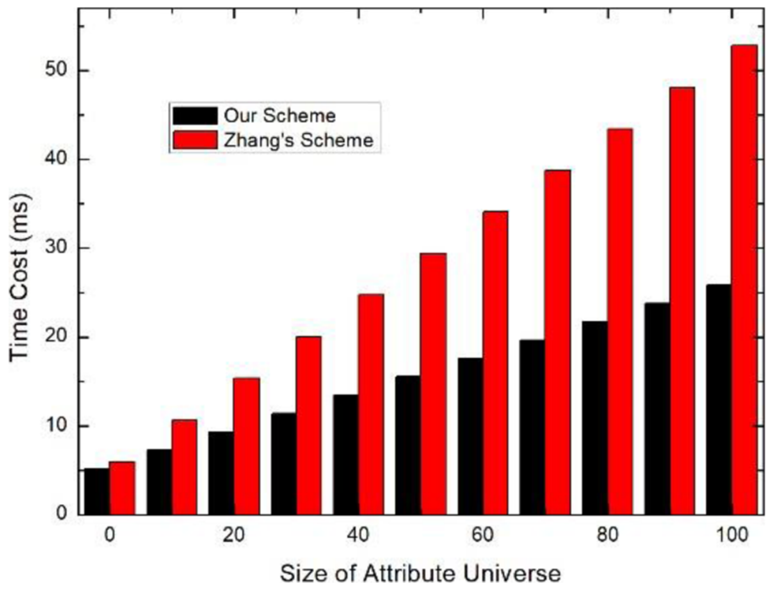

Fig. 6 Computation Time of DO with Different Size of Attribute Universe

challenger computes corresponding block tags and returns the tags to the adversary. When challenger launches challenge to the adversary, he can respond to the challenger with data proof and tag proof.

Game 2 is another challenge game in which the challenger keeps all the tags ever issued as part of the queries. If the challenger detects the aggregated block tags *TP* is not equal to $TP = \sum_{i=1}^{c} T_i$, he declares the game fails.

Game 3 is the same as game 2 with one difference that the challenger keeps all response sequences to the adversary's queries. Suppose the challenger sends $ch = (i, R_i)$ to the adversary, the adversary's reply to the query is $P = (DP, TP)$ where $T = \sum_{i=1}^{c} T_i$. In the scheme, *P* is the correct proof and equation $DP = TP \oplus L$ holds. Suppose the

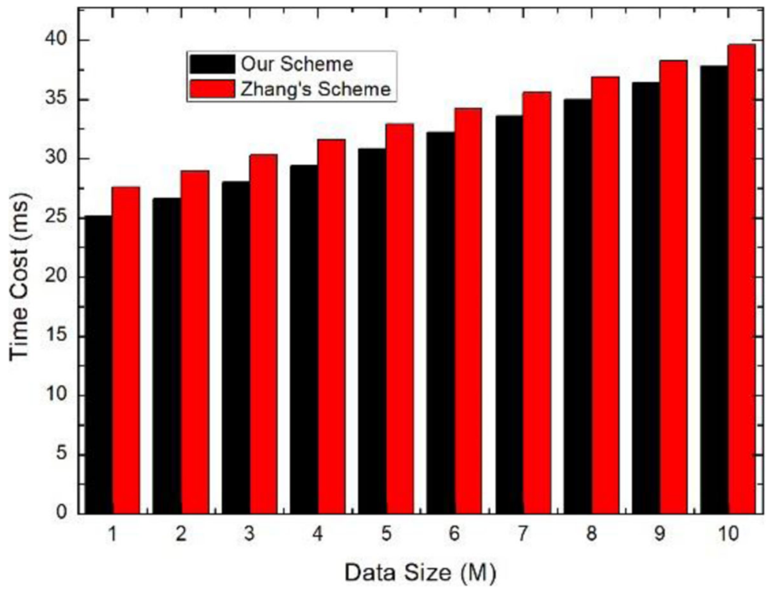

Fig. 7 Computation Time of DO with Different Size of Shared Data

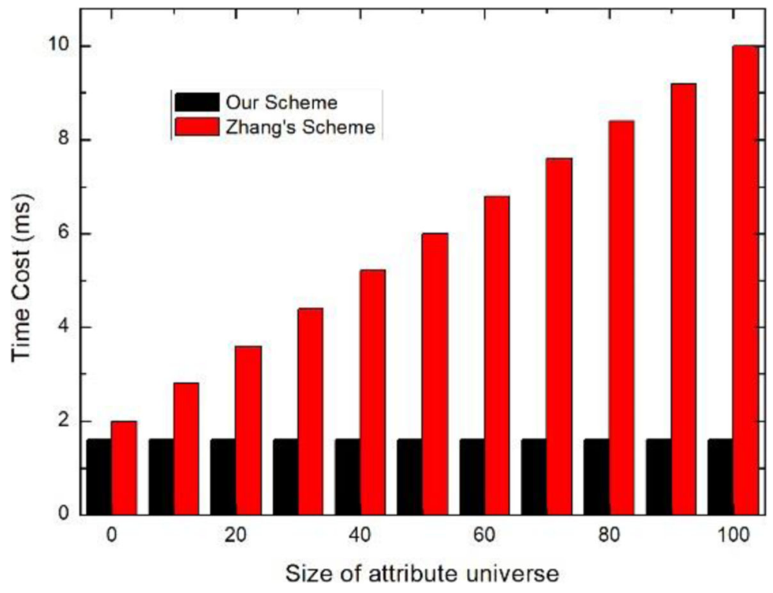

Fig. 8 Decryption Time of DU with Different Size of Attribute Universe

adversary's proof is $P^{'} = (DP^{'}, TP^{'})$, where $TP^{'} = \sum_{i=1}^{c} T_i^{'}$, then the equation $DP^{'} = TP^{'} \oplus L$ also holds. We define $\triangle DP = DP^{'} \oplus DP$, $\triangle TP = TP^{'} \oplus TP$.We make the *XOR* operation on the above two verification equations and get $\triangle DP = \triangle DP$. The above equation holds with the probability is $\frac{1}{q}$. The probability can be negligible.

Performance evaluation

In this section, we evaluate the computation costs of *DO* and *DR* in the scheme and compare it with scheme [22].

Performance analysis of mobile terminals

To analyze computation overhead of the scheme, we define the following notations to denote the corresponding operations: Let ***Pair*** denote a paring operation, ***Hash*** denote a hash operation and ***Exp*** denote an exponentiation operation. Similarly, ***Mul*** represents a multiplication operation. ***Xor*** and ***Sig*** respectively denote *XOR* and algebraic signature operation of the scheme.

1) Computation overhead of *DO* side

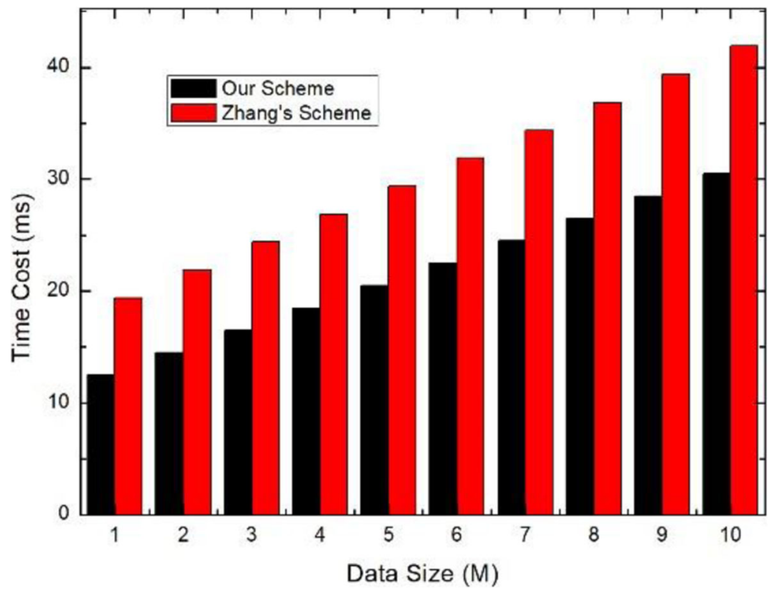

Fig. 9 Decryption Time of DR with Different Size of Shared Data

The computation overhead of *DO* is mainly in encryption phase. *DO* first computes $\boldsymbol{F'=H_2(K)\oplus F, C=K\cdot\theta^s}$, $\boldsymbol{C'=g^s, \lambda_i=M_i\,\vec{y}}_{,1\le i\le l}$. Therefore, the computation overhead of *DO* side is $\boldsymbol{(1+l)Mul+2Exp+Hash+Xor}$.

2) Computation overhead of *DR* side

The computation overhead of *DR* is mainly in data sharing phase. *DR* downloads $\boldsymbol{F'}$ and computes $\boldsymbol{CK'=\frac{e(C',K)}{V}=\theta^s}$. Then *DR* gets $\boldsymbol{K=C/CK'}$ and recovers the plaintext $\boldsymbol{F=H_2(K)\oplus F'}$. Therefore, the computation overhead of DR side is $\boldsymbol{Pair+Xor+2Div}$.

Table. 2 illustrates the computation overhead of mobile terminals on both sides.

Experimental results

We simulate our scheme with the Pairing Based Cryptography (PBC) library of version 0.5.14. We compare the computation time of *DO* and *DU* with scheme [22] by utilizing an MNT d159 curve with 160-bit group order. All the experiment results represent the average of 20 trials.

1) Computation time of *DO* in processing phase

The computation time of *DO* mainly generates in algorithm ***DataEnc*** of data processing phase. We first test the relation between *DO*'s computation time and the size of attribute universe ***A*** as described in Fig. 6. We can see that when size of attribute universe ***A*** varies from 1 to 100, the computation time of *DO* increases slowly and costs less time than Zhang's. Then we test the relation between *DO*'s computation time and the size of shared data as described in Fig. 7. From Fig. 7, we can conclude that *DO*'s computation time in our scheme is lower than that of Zhang's scheme when shared data varies from 1 M to 10 M.

2) Computation time of *DR* in data sharing phase

In data sharing phase, we first test the relationship between *DR*'s computation time and the size of attribute universe ***A*** as described in Fig. 8. Because *CMS* help *DR* to computes the intermediate data of decryption, *DR*'s computation time is constant when the size of attribute universe ***A*** increases. We also test the relationship between *DR*'s computation time and the size of shared data as described in Fig. 9. We can see that with the size of shared data growing, the computation time of *DR* increases slowly. From Figs. 8 and 9, we can conclude that *DR*'s computation time in our scheme is less than that in Zhang's scheme.

Conclusion

In this paper, we propose an efficient and secure data sharing scheme for mobile devices. The scheme guarantees security and authorized access of shared sensitive data. Furtherly, the scheme realizes efficient integrity verification before *DR* shares the data to avoid incorrect computation. Finally, the scheme achieves lightweight operations of mobile terminals on both *DO* and *DR* sides.

Abbreviations

KGC: Key Generator Center; DO: Data Owner; CS: Cloud Servers; DR: Data Requester; CSP: Cloud Storage Servers; CMS: Cloud Manage Servers; KP-ABE: Key Policy Attribute Based Encryption; CP-ABE: Cipher Text Policy Attribute Based Encryption

Acknowledgments

We sincerely thank the Reviewers and the Editor for their valuable suggestions.

Authors' contributions

Xiuqing Lu, Zhenkuan Pan, Hequn Xian designed the study and write the paper. Hequn Xian performed the simulations. Xiuqing Lu, Zhenkuan Pan wrote the paper. All authors reviewed and edited the manuscript. All authors read and approved the final manuscript.

Funding

This work is supported in part by Natural Science Foundation of China under Grant No. 61772294 and Grant No. 61602214, in part by Natural Science Foundation of Shandong Province under Grant No. ZR2019LZH002.

Availability of data and materials

The data used to support the findings of this study is available from the corresponding author upon request.

Competing interests

The authors declare that they have no competing interests.

Author details

[1]College of Computer Science and Technology, Qingdao University, Qingdao 266071, China. [2]Department of management science and Engineering, College of Business, Qingdao University, Qingdao 266071, China.

Received: 16 June 2020 Accepted: 6 October 2020
Published online: 23 October 2020

References

1. Farahat IS, Tolba AS (2018) A secure real-time internet of medical smart things (IOMST). Comput Electrical Eng 72:455–467
2. Rahmani AM, Gia TN, Negash KB (2018) Exploiting smart e-Health gateways at the edge of healthcare Internet-of-Things: A fog computing approach. Futur Gener Comput Syst 78:641–658
3. Zhang Y, Qiu M, Tsai C, Hassan M, Alamri A (2017) Health-CPS: healthcare cyber-physical system assisted by cloud and big data. IEEE Syst J 11:88–95
4. Ghazvini A, Shukur Z (2013) Security challenges and success factors of electronic healthcare system. Proc Technol 11:212–219
5. Guan Z, Lv Z, Du X et al (2019) Achieving data utility-privacy tradeoff in internet of medical things: a machine learning approach. Futur Gener Comput Syst 98:60–68
6. Elhoseny M, Abdelaziz A (2018) A hybrid model of internet of things and cloud computing to manage big data in health services applications. Futur Gener Comput Syst 86:1383–1394
7. Han K, Li Q, Deng Z (2016) Security and efficiency data sharing scheme for cloud storage. Chaos Solitons Fractals 86:107–116
8. Zhang L, Zhang H, Jia Y (2020) Blockchain-based two-party fair contract signing scheme. Inf Sci 535:142–155
9. Lu X, Cheng X (2020) A secure and lightweight data sharing scheme for internet of medical things. IEEE Access 8:5022–5030. https://doi.org/10.1109/ACCESS.2019.2962729

10. Bahga A, Madisetti VK (2013) A cloud-based approach for interoperable electronic health records (EHRs). IEEE J Biomed Health Inf 17(5):894–906
11. Tawalbeh LA, Mehmood R, Benkhlifa E, Song H (2016) Mobile cloud computing model and big data analysis for healthcare applications. IEEE Access 4:6171–6180
12. Nguyen DC, Pathirana PN, Ding M, Seneviratne A (2019) Blockchain for Secure EHRs Sharing of Mobile Cloud Based E-Health Systems. IEEE Access 7:66792–66806. https://doi.org/10.1109/ACCESS.2019.2917555
13. Chang V (2017) Towards data analysis for weather cloud computing. Knowl Based Syst 127:29–45
14. Gao F, Sunyaev A et al (2019) Context matters: a review of the determinant factors in the decision to adopt cloud computing in healthcare. Int J Inf Manag 48:120–138
15. Akl SG, Taylor PD (1983) Cryptographic solution to a problem of access control in a hierarchy. ACM Trans Comput Syst 1:239–248
16. Goyal V, Pandy O, Sahai A, Waters B (2006) Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM conference on Computer and communications security, p 89
17. Hur J, Noh CD (2011) Attribute-based access control with efficient revocation in data outsourcing systems. IEEE Trans Parallel Distributed Syst 22:1214–1221
18. Bethencourt J, Sahai A, Waters B (2007) Ciphertext-policy attribute based encryption. In: Proceedings of the IEEE symposium on security privacy, p 321
19. Lewko A, Waters B (2011) Decentralizing attribute-based encryption. In: Proceedings of the 30th Annual international conference on Theory and applications of cryptographic techniques: advances in cryptolog, p 568
20. Waters B (2011) Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. In: Proceedings of the International Workshop on Public Key Cryptography, pp 53–70
21. Cheung L, Newport C (2007) Provably secure ciphertext policy ABE. In: Proc. ACM Conf. Comput. Commun. Security (CCS), pp 456–465
22. Zhang Y, Zhen D, Deng R (2018) Security and privacy in smart health: efficient policy-hiding attribute-based access control. IEEE Internet Things J 5:2130–2145
23. Yang M, Zhang T (2018) Efficient Privacy-Preserving Access Control Scheme in Electronic Health Records System. Sensors 18:3520–3545.
24. Phuong T, Yang G, Susilo W (2016) Hidden ciphertext policy attribute-based encryption under standard assumptions. IEEE Trans Inf Forensic Secur 11: 35–45
25. Zhang Y, Chen F, Li J (2017) Ensuring attribute privacy protection and fast decryption for outsourced data security in mobile cloud computing. Inf Sci 379:42–61
26. Ateniese G, PietroR D, Mancini LV, Tsudik G (2008) Scalable and efficient provable data possession. In: Proceedings of the 4th international conference on Security and privacy in communication networks, pp 1–10
27. Yang K (2017) An efficient and fine-grained big data access control scheme with privacy-preserving policy. IEEE Internet Things Journal 4:563–571
28. Wang G, Liu Q, Wu J Hierarchical attribute-based encryption and scalable user revocation for sharing data in cloud servers. Comput Secur 30:320–331
29. Guo Z, Li M, Fan X (2013) Attribute-based ring signcryption scheme. Secur Commun Netw 6:790–796
30. Wei J, Hu X, Liu W (2014) Traceable attribute-based signcryption. Secur Commun Netw 7:2302–2317
31. Liu J, Huang X, Liu JK (2015) Secure sharing of personal health records in cloud computing: Ciphertext-policy attribute-based signcryption. Futur Gener Comput Syst 52:67–76
32. Hu C, Li W (2017) A secure and verifiable access control scheme for big data storage in clouds. IEEE Trans Big Data 4:341–355
33. Cai H, Xu B, Jiang L (2017) IoT-based big data storage systems in cloud computing: perspectives and challenges. IEEE Internet Things 4:75–87
34. Ateniese G, Burns R, Curtmola R (2007) Provable data possession at untrusted stores. ACM Conf Comput Commun Secur 14:598–609
35. Erway C, Papamanthou C, Tamassia R (2009) Dynamic provable data possession. ACM Conf Comput Commun Secur 17:213–222
36. Zhu Y, Ahn GJ, Hu H, Yau SS (2013) Dynamic audit services for outsourced storages in clouds. IEEE Trans Serv Comput 6:227–238
37. Yang K, Jia X (2013) An efficient and secure dynamic auditing protocol for data storage in cloud computing. IEEE Trans Parallel Distributed Syst 24: 1717–1726
38. Tian H, Chen Y, Chang CC (2017) Dynamic-hash-table based public auditing for secure cloud storage. IEEE Trans Serv Comput 10:701–714
39. Wang Q, Wang C, Ren K, Lou W (2011) Enabling public auditability and data dynamics for storage security in cloud computing. IEEE Trans Parallel Distributed Syst 22:847–859
40. Liu C, Chen J, Yang LT, Zhang X (2014) Authorized public auditing of dynamic big data storage on cloud with efficient verifiable fine-grained updates. IEEE Trans Parallel Distributed Syst 25:2234–2244
41. Gan Q, Wang X, Fang X (2018) Efficient and secure auditing scheme for outsourced big data with dynamicity in cloud. Inf Sci 61:93–107
42. Luo Y, Fu S, Xu M (2014) Enable data dynamics for algebraic signatures based remote data possession checking in the cloud storage. China Commun 11:114–124
43. Bhaskaran K, Ilfrich P, Liffman D (2018) Double-blind consent-driven data sharing on blockchain. In: Cloud Engineering (IC2E), IEEE International Conference on IEEE, pp 385–439
44. Liu CW, Hsien WF, Yang CC (2016) A survey of public auditing for shared data storage with DU revocation in cloud computing. Int J Netw Secur 18: 650–666
45. Shamir A (1985) Identity-Based Cryptosystems and Signature Schemes. In: Blakley GR, Chaum D (eds) Advances in Cryptology. CRYPTO 1984. Lecture Notes in Computer Science, vol 196. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-39568-7_5
46. Yoon E-J, Choi Y, Kim C (2013) New ID-based proxy signature scheme with message recovery. In: Grid and Pervasive Computing (Lecture Notes in Computer Science), vol 7861. Springer-Verlag, Berlin, pp 945–951
47. Yu J, Hao R (2019) Comments on SEPDP: secure and efficient privacy preserving provable data possession in cloud storage. IEEE Trans Serv Comput. https://doi.org/10.1109/TSC.2019.2912379
48. Lu XQ, Pan ZK, Xian HQ (2020) An integrity verification scheme of cloud storage for internet-of-things mobile terminal devices. Comput Secur 92. https://doi.org/10.1016/j.cose.2019.101686
49. Y. Zhang, J. Yu, R. Hao, C, Wang, K. Ren, "Enabling Efficient User Revocation in Identity-based Cloud Storage Auditing for Shared Big Data," IEEE Trans Dependable Secure Comput, vol. 17, pp. 608–619, 2020
50. Shen W, Su Y, Hao R (2020) Lightweight cloud storage auditing with Deduplication supporting strong privacy protection. IEEE Access 8:44359–44372
51. Zhao P, Yu J, Zhang H (2020) How to securely outsource finding the min-cut of undirected edge-weighted graphs. IEEE Trans Inf Forensic Secur 15: 315–328
52. Zhang H, Jia Y, Cheng L (2020) Practical and secure outsourcing algorithms for solving quadratic congruences in internet of things. IEEE Internet Things J 7:2968–2981

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.