


RESEARCH

Open Access

A threshold hybrid encryption method for integrity audit without trusted center



Yange Chen^{1,2}, Hequn Liu³, Baocang Wang^{1,2*} , Baljinnyam Sonompil⁴, Yuan Ping² and Zhili Zhang²

Abstract

Cloud storage with sharing services is increasingly popular among data owners. However, it is difficult for the users to know if the cloud server providers (CSPs) indeed protect their data. To verify data integrity and preserve data and key privacy in the group, this paper proposes a new threshold hybrid encryption for integrity auditing method without trusted center. The proposed method is developed based on the Advanced Encryption Standard (AES) and the Elliptic Curve Cryptography (ECC) with Shamir secret sharing. In this way, the key can be distributed and managed without trusted center, preserving the privacy of the key of the AES and users' private key. Besides, we design and implement a novel integrity auditing and re-signature method which verifies the data integrity and solves the collusion question of the cloud and the revoked users. Security analysis and performance evaluation demonstrate that the proposed scheme realizes the correctness, security, and efficiency with a low communication and computation cost.

Keywords: Sharing services, Integrity auditing, Cloud storage, Threshold hybrid encryption, Multiple managers

Introduction

With the emergence and widespread application of information technology in artificial intelligence (AI), Internet of things (IoT), and mobile internet [1, 2], data has achieved the explosive growth, especially in the industrial Internet of things (IIoT) applications [3]. Facing the burden of data storage, an increasing number of individuals and organizations has chosen to store their data in the cloud, which accelerates the rapid development of cloud storage in cloud computing [4, 5]. On the upside, the cloud storage can save the local space, freeing a lot of local computing power; on the downside, the data outsourced to the cloud may face many risks, including but not limited to data loss, privacy leak, and security attacks [6, 7].

In fact, the outsourced data are not controlled by the user. To save the storage space, the cloud may remove the rarely used or highly repeated data, which breaks the integrity of the user data in the cloud. It is impossible

for most users, with limited auditing power, to know if their data in the cloud are still complete. To solve the problem, the user can entrust a third party to audit the integrity of his/her data by checking the accuracy, validity, and consistency of the data and pinpointing the incomplete or missing entries in the data storage. The integrity auditing is particularly important for group users of cloud storage with sharing services, as any misbehaving user in the group may endanger the data security of other group members. After all, users in the same group can share data with each other, and access and modify the shared data [8–10].

The integrity auditing mechanisms [11–13] have been proposed continuously. Verifying the integrity of shared data in the group based on public key infrastructure (PKI) faces severe security risks and a high computation cost [14, 15], most of them do not preserve the data privacy in the cloud environment, and do not mention safe management of the key. What is worse, the previous studies have concentrated on data security and data integrity in the cloud, failing to tackle the data security in the upload or download process, while the data are easily stolen by

*Correspondence: bcwang79@aliyun.com

¹State Key Laboratory of Integrated Service Networks, Xidian University, Taibai South Road, Xi'an, China

²School of Information Engineering, Xuchang University, Bayi Road, Xuchang, China

Full list of author information is available at the end of the article

hacking attacks. To mitigate the risk and protect the privacy, the data and key should be encrypted. Therefore, it is important to develop an efficient and secure encryption scheme to preserve the data and key privacy, which supports the integrity auditing of the data shared between multiple parties.

The data shared by multiple parties can be applied to various applications, especially in collaborative scenarios. For instance, deep learning has provided an effective solution to collaborative learning and parallel computing in federal learning. However, many deep learning schemes cannot preserve privacy due to the lack of cryptographic tools [16]. Few schemes [17, 18] combining encryption technique fail to tackle the collusion between the CSP and users.

To address the aforementioned challenge, this paper designs an integrity auditing method based on a threshold hybrid encryption method without trusted center, which supports public auditing of data integrity with Shamir secret sharing. The main contributions of this paper are as follows:

- (1) We introduce a novel threshold hybrid encryption scheme that the user data are encrypted by the Advanced Encryption Standard (AES), and the key seed of the AES is encrypted by the Elliptic Curve Cryptography (ECC), promoting the encryption efficiency and protecting data and key privacy.
- (2) We adopt Shamir secret sharing employing multiple managers in the group to generate and dispense secret without trusted center, which facilitates the distribution and management of the keys. In this way, the multi-managers scenario is transformed into the multi-proxy scenario, which reduces the probability of malicious managers and makes the entire mechanism reliable and secure.
- (3) Our method supports public auditing with a trusted third-party auditor (TPA) and re-signature with the revoked users' data by the cloud with the help of a manager. Security analysis and performance evaluation indicate that our method achieves the validity and security, verifies the encryption effectiveness, shortens the re-signature time, and reduces the communication and computation cost.

Literature review

Much research has been done to audit data integrity at home and abroad. For instance, Ateniese et al. [19] proposed a public auditing protocol for static data integrity under the challenge-proof-verify mechanism, eliminating the need to retrieve the entire data. To support dynamic data, Ateniese et al. [20] designed a scalable public auditing strategy based on symmetric encryption. However, this strategy can only respond to a limited

number of auditing requests, failing to fully support the integrity auditing of dynamic data. Neither of the above two schemes considers data privacy. Wang et al. [21] introduced a TPA to realize privacy-preserving integrity auditing of the dynamic data in the cloud.

To ensure secure data sharing among users, Guo et al. [22] presented a key-aggregate authentication cryptosystem to share dynamic data in the cloud. Shen et al. [23] prepared a data integrity auditing scheme that shared the data based on identity, without exposing sensitive information. Chi et al. [24] designed a novel cloud storage encryption scheme that allowed the CSP to protect user privacy with convincing fake secrets. Yu et al. [25] came up with a paradigm named strong key-exposure resilient auditing for secure cloud storage.

Harn [26] developed a threshold signature mechanism that enabled the group members to produce public and private keys of the group, however, it could not resist the collusion attack. Wang [27] put forward the identity-based distributed provable data possession (ID-DPDP) scheme which achieved public auditing of the data stored in multiple clouds. Since then, the threshold signature has attracted a lot of attention in the research of data integrity auditing [28, 29].

Recent years saw some threshold encryption schemes that encrypted data with dispersed encryption rights by secret sharing [30]. The existing threshold encryption schemes [31, 32] take a trusted center to produce and distribute the secret shares of all managers, which is an authoritative member of the system and can recover the secret without the aid of other members. Nonetheless, the trusted center might lead to authority deception and nullify the meaning of secret [33]. To solve the problem, Jie et al. [34] presented a threshold encryption method without trusted center, which encrypted and decrypted the privacy data through the collaboration between multiple players. Nevertheless, Jie's method, solely relying on the ECC, cannot efficiently encrypt a large amount of data and does not apply to the cloud environment. Based on the TPA and the AES, Shimbire et al. [35] proposed enhancing distributed data storage security scheme for cloud computing utilizing the file distribution and SHA-1 technique, however, it did not consider signature and key security. Based on the state of the art, we compare common functions for auditing schemes in Table 1. These schemes are named according to the literature name or using the original naming method.

In machine learning, Sangaiah et al. [37] proposed a method for conserving position confidentiality using machine learning techniques by merging decision trees and k -nearest neighbor. Meanwhile, Sangaiah et al. [38] presented an energy-aware green adversary model for its use in the smart industrial environment through achieving position and information confidentiality in

Table 1 The comparison of various auditing schemes

Scheme	Privacy preservation	Shared data	Group management	Verification	Revocation	Without trusted center
PAPNDS [4]	✓	✗	✗	✓	✓	✓
Panda [8]	✓	✓	✓	✓	✓	✓
PDP [19]	✗	✗	✗	✗	✗	✓
IIADSSIH [23]	✓	✓	✗	✗	✗	✓
CoRPA [31]	✓	✓	✓	✓	✓	✗
EURI [32]	✓	✓	✓	✓	✓	✗
Oruta [36]	✓	✓	✓	✓	✓	✓
Ours	✓	✓	✓	✓	✓	✓

cyber-physical security. Both of the above two schemes protected the position or information confidentiality well in non-cryptographic mechanism. In a privacy-preserving multi-user collaborative deep learning model, Phong et al. [17] presented a privacy-preserving deep learning model via additively homomorphic encryption through federal learning. However, the model was still weak against the collusion of the server and trainers. Subsequently, Phong et al. [18] proposed a privacy-preserving deep learning model via weight transmission which protected the input privacy of each participant through symmetric encryption, without considering the collusion and key security.

To solve the problem of data and key privacy, resist the hacking and collusion attack, and ensure signature security, this paper proposes a threshold hybrid encryption method for integrity auditing without trusted center. The method employs the AES and the ECC to promote encryption efficiency and protect data and key privacy. To manage and distribute the keys, we use Shamir secret sharing with multiple managers in a group. Using the challenge-response game for the cooperation of the TPA and the CSP with a re-signature mechanism, we realize the integrity auditing and data dynamic update with a low communication and computation cost. The method is a universal encryption construction for cloud storage with sharing services and collaborative learning of deep learning, which promotes data sharing in the group and enhances the encryption efficiency and key security.

Organization

The remainder of this paper is organized as follows: “Preliminaries” section introduces some preliminaries; “System model and threat model” section establishes the system model and threat model; “Construction of the scheme” section details the construction of our method; “Correctness and security analysis” section analyzes the correctness and security of our method; Performance analysis is demonstrated in “Performance analysis”; “Conclusions” sections wraps up this paper with conclusions.

Preliminaries

The main notations used in the description of our scheme are shown in Table 2.

Bilinear pairing and discrete logarithm (DL) problem

Bilinear pairing has been widely employed in cryptography and utilizes in data auditing at present. The detailed description is followed.

Definition 1 (Bilinear Pairing) *Let \mathbb{G}_1 and \mathbb{G}_2 be two multiplicative cyclic groups with a prime order p , g be a generator of \mathbb{G}_1 . A bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is a map function satisfying the three properties below:*

- (1) *Bilinearity: for $\forall u, v \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_p^*$, there is $e(u^a, v^b) = e(u, v)^{ab}$.*
- (2) *Computability: for $\forall u, v \in \mathbb{G}_1$, $e(u, v)$ can be computed efficiently.*
- (3) *Non-degeneracy: $\exists u, v \in \mathbb{G}_1$, there is $e(u, v) \neq 1$.*

Definition 2 (Discrete Logarithm (DL) problem) *Let \mathbb{G}_1 be a multiplicative cyclic group, g be a generator of \mathbb{G}_1 . Let*

Table 2 Notation

Notation	Meaning
$\mathbb{G}_1, \mathbb{G}_2$	Two multiplicative cyclic groups with a prime order p
g, ω	Generators of \mathbb{G}_1
e	A bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$
\mathbb{Z}_p	Residue class ring modulo p
\mathbb{Z}_p^*	Multiplicative group of invertible integers modulo p
\mathbb{F}_η	A finite field with η prime elements
$E(\mathbb{F}_\eta)$	An elliptic curve on the finite field \mathbb{F}_η
Q	The generator of the elliptic curve $E(\mathbb{F}_\eta)$
N	The number of the managers $\{P_1, P_2, \dots, P_N\}$
\mathbb{F}_q	A finite field with a prime number $q > N$
t	The preset threshold value of Shamir secret sharing
$H(\cdot)$	A hash function: $\{0, 1\}^* \rightarrow \mathbb{G}_1$

unknown element $a \in \mathbb{Z}_p^*$, given the value of $g, g^a \in \mathbb{G}_1$ as input, the DL problem is to output a .

Definition 3 (Discrete Logarithm (DL) assumption) For any probabilistic polynomial time (PPT) adversary \mathcal{A}_{DL} , the advantage for \mathcal{A}_{DL} to solve the DL problem in \mathbb{G}_1 is negligible, which is defined as

$$Pr \left[\mathcal{A}_{DL}(g, g^a) = a : a \xleftarrow{R} \mathbb{Z}_p^* \right] \leq \epsilon$$

Thereinto, a negligible value is denoted as ϵ in the above definitions.

Shamir secret sharing

Shamir secret sharing [39] assumes that a dealer holds a secret s and shares among N users by giving each user a share, and the secret s can be recovered from any t users. In other words, the original secret cannot be restored unless the number of users involved in decryption exceeds the threshold value t . This secret preservation strategy is also known as the (t, N) threshold method.

The advantage of this method applied in our scheme lies in the decentralized authority of group managers [40]. In this method, each manager can encrypt the user data, but cannot decrypt the ciphertext alone. To recover the plaintext data, there must be no less than t managers involved in decryption. More importantly, the failure of any manager in decryption does not affect the recovery of the plaintext. This property leads to the robustness of our system.

System model and threat model

System model

As shown in Fig. 1, our system model contains three entities: the CSP, the TPA, and a user group with multiple managers.

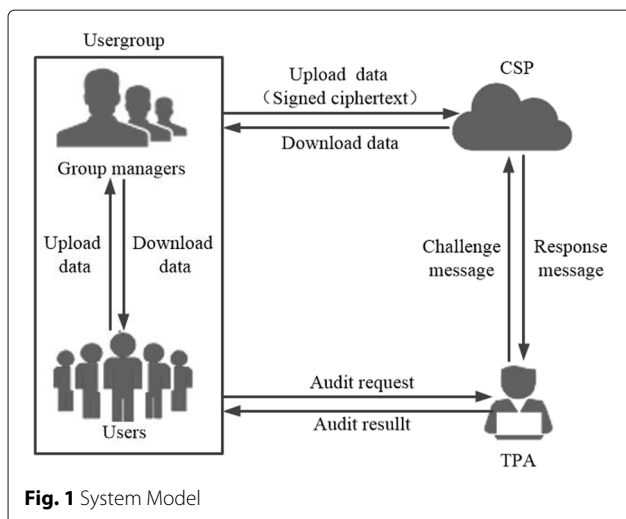


Fig. 1 System Model

CSP

The CSP stores user data in the cloud platform with sharing services, works with the TPA to validate data integrity, and checks the legality of signature users.

TPA

With the professional knowledge of data auditing, the TPA executes public auditing with the CSP through the challenge-response game. During the game, the TPA launches challenges to the CSP, and the CSP responses the proof to prove the integrity of the data challenged by the TPA. Then, the TPA calculates the auditing results to return to the manager or the user.

User group with multiple managers

The user group contains both managers and users. The group managers can encrypt user data, upload them to the CSP, and decrypt the data before returning them to users. The encryption and decryption are conducted by multiple managers through Shamir secret sharing. To ensure correct decryption, the number of managers participating in the process must surpass the threshold value t . Any group user can upload his/her data to the CSP through a manager and download the shared data from the CSP. Each user can also access, modify, and delete the data shared by other users.

Threat model

As mentioned before, the data outsourced to the cloud may face many risks, including but not limited to data loss, privacy leakage, and security attacks. This paper mainly focuses on data leakage, authoritative attack, key exposure, and data integrity verification. Data integrity was mentioned earlier and will not be described here.

Data leakage

Data privacy is the key to the outsourcing storage and integrity auditing of sensitive data. Nonetheless, the privacy-preserving techniques for cloud storage, such as data encryption or differential privacy, cannot prevent data leakage in the outsourcing storage process. For example, when a user is revoked from the user group, its privilege to access, modify, and delete other users' data are not immediately revoked, sowing the risk of data leakage.

Authoritative attack

If there is only one manager in the group, the manager will possess virtually unlimited authority and easily acquire all the data before encryption, causing the authoritative attack of the trusted center. To prevent the attack, the data are encrypted by the hybrid threshold encryption method with the secret sharing and recovered by not less than t managers in our scheme.

Key exposure

Key exposure is one serious security problem for cloud storage, which will cause the data more insecurity. Many methods have been developed to solve this problem, such as the paradigm called strong key-exposure resilient auditing for secure cloud storage [25]. Here, the key exposure is guarded against by encrypting the AES key with the ECC.

Construction of the scheme

In general, our threshold hybrid encryption method consists of five phases: key generation, data encryption, integrity auditing, data decryption, and user revocation & re-signature.

Key generation

Let \mathbb{F}_η be a finite field with η elements, $E(\mathbb{F}_\eta)$ be an elliptic curve on the finite field, Q be the generator of the curve. Both the elliptic curve and the generator are public. Let \mathbb{F}_q be a finite field containing the domain of possible secrets with $q > N$, q be a prime number, and N be the number of the managers $\{P_1, P_2, \dots, P_N\}$ with Shamir secret sharing, P_i be the i th manager with the identity number ID_i , and t be the preset threshold value of Shamir secret sharing.

In our method, each manager can generate its own secret share and acquire the identity number and shares of other managers through interaction. Then, manager P_i computes the public parameter, which is executed in the elliptic curve $E(\mathbb{F}_\eta)$ in the following steps:

Step 1: Manager P_i ($1 \leq i \leq N$) randomly selects an integer $d_i \in \mathbb{Z}_p$ as the private key and then calculates its public key y as:

$$y = d_i Q \quad (1)$$

Step 2: Manager P_i sets up the polynomial $f_i(x)$ of degree $t-1$ with f_{iz} as the coefficient:

$$f_i(x) = d_i + f_{i1}x + \dots + f_{i(t-1)}x^{t-1} \quad (2)$$

where $f_{iz} \in \mathbb{F}_q$, $z = 1, 2, \dots, t-1$.

Step 3: Manager P_i calculates the secret share $f_i(ID_j)$ of group manager P_j ($j \neq i$) according to their identity number ID_j .

Step 4: Manager P_i calculates the parameter Y_{ij} for manager P_j :

$$Y_{ij} = f_i(ID_j) Q \quad (3)$$

Step 5: Meanwhile, manager P_i computes the validation parameter: $\alpha_{iz} = f_{iz}Q$, and then sends the public parameter: $pp = \{y, Y_{ij}, \alpha_{iz}\}$ to the other managers. Manager P_i keeps its secret share $f_i(ID_i)$ and public parameter Y_{ii} .

Upon receiving the public parameters from the other $t-1$ managers, manager P_j can check the validity of $f_i(ID_j)$ in the elliptic curve $E(\mathbb{F}_\eta)$ by:

$$f_i(ID_j) Q = y + \sum_{z=1}^{t-1} (ID_j)^z \alpha_{iz} \quad (4)$$

Data encryption

In our hybrid encryption method, managers encrypt user data by the AES and encrypt the key of the AES through the ECC with the message non-embeddable method [41]. Then, the data are signed with users' private key before uploading. The efficient encryption process facilitates key distribution and management and protects the key of the AES, which provides an effective encryption solution to outsourced data. The details of the encryption process are given in Fig. 2.

Let ω be a generator of multiplicative cycle group \mathbb{G}_1 , $H(\cdot)$ be a hash function: $\{0, 1\}^* \rightarrow \mathbb{G}_1$ with a bilinear pairing, and U_A be the user planning to upload its data M to the cloud.

First, user U_A generates the private key sk_A and the public key pk_A satisfying $pk_A = g^{sk_A}$, and then sends plaintext M and the private key sk_A to manager P_i in the group. After receiving plaintext M and the private key sk_A , manager P_i verifies whether U_A is a legitimate user with the right to upload the data by:

$$e(f_{up}, pk_A) = e(f_{up}^{sk_A}, g) \quad (5)$$

where f_{up} is the uploaded file. If the above equality holds and U_A is in the user list, U_A must be a legitimate user who can upload the data. Then, manager P_i will receive data M ; otherwise, manager P_i will delete data M . Next, manager P_i will encrypt data M in the following steps:

Step 1: Manager P_i selects a random number S ($S \in \mathbb{Z}_p$) and sets up an AES key seed sk_{AES} ;

Step 2: Manager P_i encrypts data M by the AES with sk_{AES} , yielding the ciphertext C_M of data M :

$$C_M = Enc(sk_{AES}, M) \quad (6)$$

where $Enc(\cdot)$ represents the encryption function.

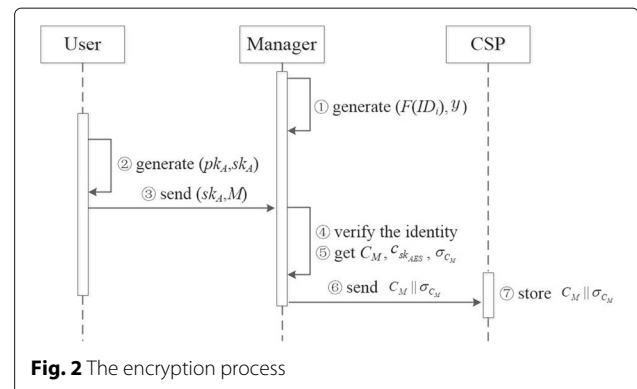


Fig. 2 The encryption process

Step 3: Manager P_i encrypts AES key sk_{AES} by the ECC, yielding the ciphertext $c_{sk_{AES}}$ of sk_{AES} :

$$C_1 = (x_1, y_1) = SQ \tag{7}$$

$$C_2 = (x_2, y_2) = Sy \tag{8}$$

$$c_1 = x_2 \cdot sk_{AES} \tag{9}$$

$$c_{sk_{AES}} = (C_1, c_1) \tag{10}$$

where (x_1, y_1) is the coordinates of C_1 , (x_2, y_2) is the coordinates of C_2 , and x_2 is the X coordinate of C_2 .

Then, manager P_i will send ciphertext $c_{sk_{AES}}$ of sk_{AES} to each manager in the group, without saving the random number S and random key seeds sk_{AES} . Upon receiving ciphertext C_M , the manager will sign the encrypted data C_M using the private key of U_A as:

$$\sigma_{C_M} = \left(H(id_M) \omega^{C_M} \right)^{sk_A} \tag{11}$$

where id_M is the identity of ciphertext C_M .

Finally, manager P_i will send the ciphertext of data M with its signature $C_M || \sigma_{C_M}$ to the cloud.

Integrity auditing

Before downloading the data, the data integrity should be verified by the auditing method. A user only needs to submit an auditing application to the TPA who will check the data integrity in the CSP through the challenge-response game and return the results to the user. To lower communication and computation cost, our method assumes that the user entrusts the TPA to perform integrity auditing through interaction with the CSP. Our integrity auditing plan is as follows, and the process is in Fig. 3.

Firstly, the user U_A requests auditing a certain file, and the TPA generates the challenge message through the following steps:

Step 1: The TPA randomly extracts a ciphertext subset $L = \{C_{M_1}, C_{M_2}, \dots, C_{M_o}\}$ to serve as the numbers of data to be audited. Thereinto the set L are divided into N_{U_i} blocks $L = \{l_1, l_2, \dots, l_{N_{U_i}}\}$, where l_i is the i th data blocks

containing o_i elements. Then, the following equation can be derived:

$$o = \sum_{i=1}^{N_{U_i}} o_i \tag{12}$$

where data blocks number $i \in \{1, 2, \dots, N_{U_i}\}$ is denoted as $[1, N_{U_i}]$.

Step 2: The TPA randomly selects a value v_n from \mathbb{Z}_p^* for $n \in l_i$ and sends $msg_{chal} = \{(i, v_n)\}_{i \in [1, N_{U_i}]}$ to the cloud as challenge messages.

Upon receiving the challenge messages, the cloud returns the proof of the outsourced data through the following steps:

Step 1: The cloud computes the tag proof β_i and the data proof γ_i of each part l_i in the following formula:

$$\beta_i = \prod_{n \in l_i} \sigma_n^{v_n} \tag{13}$$

$$\gamma_i = \sum_{n \in l_i} C_{M_n} v_n \tag{14}$$

where σ_n is the signature of data block l_i satisfying

$$\sigma_n = \left(H(id_n) \omega^{C_{M_n}} \right)^{sk_A} \tag{15}$$

and id_n is the identity of $n \in l_i$.

Step 2: Then, the CSP sets $\beta = \{\beta_1, \beta_2, \dots, \beta_{N_{U_i}}\}$ and $\gamma = \{\gamma_1, \gamma_2, \dots, \gamma_{N_{U_i}}\}$, and returns the proof $msg_{re} = \{\beta, \gamma, id_n\}$ to the TPA to see if:

$$e \left(\prod_{i=1}^{N_{U_i}} \beta_i, g \right) = \prod_{i=1}^{N_{U_i}} e \left(\prod_{n \in l_i} H(id_n)^{v_n} \omega^{\gamma_i}, pk_A \right) \tag{16}$$

If the equality holds, the TPA will output 1; otherwise, the TPA will output 0.

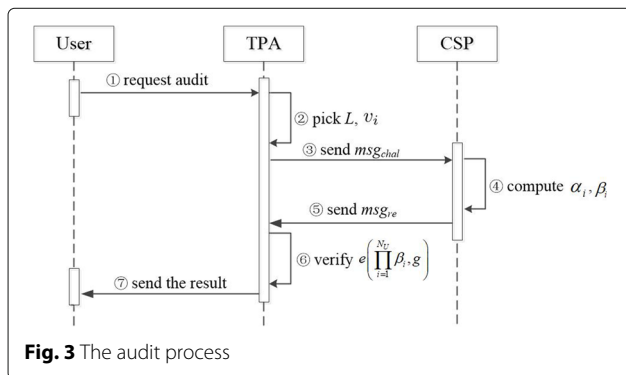
Step 3: Finally, the TPA sends the results to the user U_A who wants to be informed.

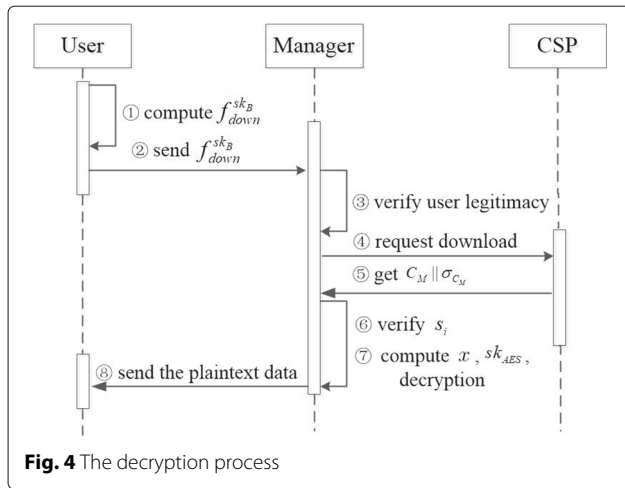
Data decryption

To obtain the shared data in the CSP, a user must download the data from the cloud storage. First, the user needs to file a download application to a manager. Then, the manager will verify if the user is a legitimate user with the privilege to download the data. If the user is rightful, the manager downloads the shared data. Afterward at least t managers will jointly decrypt the encrypted data, and then one of managers sends the data to the user. The verification is implemented in the following steps in Fig. 4.

Let U_B be the user who wants to download data M from the cloud. The user needs to prepare the downloaded auditing file $f_{down}^{sk_B}$ first and sends it to any manager as the downloaded application. Upon receiving $f_{down}^{sk_B}$, the manager will execute the following formula:

$$e(f_{down}, pk_B) = e(f_{down}^{sk_B}, g) \tag{17}$$





where f_{down} denotes the downloaded file, sk_B and pk_B are the private key and the public key of user U_B , respectively. If the above equality holds and U_B is in the user list, U_B must be a legitimate user with the privilege to download the data. Then, manager P_i will file the download application to the CSP and obtain $C_M || \sigma_{C_M}$. The $C_M || \sigma_{C_M}$ will be split into the ciphertext C_M and signature σ_{C_M} by the manager, and C_M will be decrypted by t managers.

Let $W = \{P_1, P_2, \dots, P_t\}$ be t managers involved in data decryption. The details on the decryption process are given as follows:

Step 1: Upon receiving the ciphertext, manager P_i in W calculates its decryption factor s_i from manager P_j by its secret share with lagrange interpolation polynomial:

$$s_i = C_1 f_i(ID_j) \prod_{1 \leq k \leq t, k \neq j} \frac{-ID_{i_k}}{ID_{i_j} - ID_{i_k}} \quad (18)$$

where ID_{i_k} and ID_{i_j} is the identity number of k th and j th managers in manager P_i , and $k \in \{1, 2, \dots, t\}$.

Step 2: The manager P_j validates his/her decryption factor s_i as follows.

$$s_i Q = C_1 Y_{i_j} \prod_{1 \leq k \leq t, k \neq j} \frac{-ID_{i_k}}{ID_{i_j} - ID_{i_k}} \quad (19)$$

If the above formula holds, the decryption factor s_i must be true; otherwise, the decryption factor s_i must be false, and the application will be rejected.

Step 3: After manager P_i having received t decryption factors, manager P_i can calculate (x_2, y_2) by the ECC decryption algorithm:

$$\sum_{j=1}^t s_{ij} = (x_2, y_2) \quad (20)$$

The correctness of the formula is verified in the following formula (28).

Step 4: Then, manager P_i obtains the x_2 , and calculates the AES key seed sk_{AES} by:

$$sk_{AES} = x_2^{-1} c_1 \quad (21)$$

After getting the AES key seed sk_{AES} , manager P_i decrypts to obtain the plaintext M and sends the data to the user U_B .

User revocation and re-signature

The user revocation mechanism in data sharing services with the cloud has been studied in many papers [42–45]. For instance, Wang et al. [8] presented a proxy re-signature scheme that the user whose registration expires would be revoked by managers from the group, and lost the right to share, acquire or update data; all the data signed by the revoked user should be resigned by another legitimate user in the group, such that the integrity auditing of the revoked users' data could proceed normally.

Since the data of the revoked user are stored in the cloud, its signatures can be directly recomputed by selecting a legitimate user in the group, who downloads the data signed by the revoked user to the local space. Next, the legitimate user should verify the correctness of the data, re-sign the data with its private key, and send the re-signed data back to the cloud. However, the direct download and re-signature method consumes lots of time and computing power and incurs a high communication cost, especially when a huge amount of data need to be signed and the users in the group change frequently.

It is obvious that if the cloud can obtain the private key of each user, and then the arduous task of re-signature can be completed quickly by the cloud itself. This approach eliminates the need to download data to the local space, saving the re-signature time. Nevertheless, the cloud is not completely reliable, making it dangerous to outsource the users' private keys to the cloud. This calls for a fast and efficient re-signature method for legitimate users which can eliminate downloading the process of the cloud data. Namely, a user is revoked from the user group, the cloud as the re-signature proxy will convert a signature of the revoked user into a signature of the legitimate user on the same block, using the proxy re-signature technology, without learning any private key or data.

In view of the above, the proxy re-signature technique proposed by Wang et al. [8] is as follows. When user U_A is revoked, the cloud will cooperate with another legitimate user U_C to re-sign the data signed by the revoked user U_A . The details of the re-signature process without considering the collusion are provided in Fig. 5.

Step 1: The CSP generates a random number r and sends it to the revoked user U_A .

Step 2: User U_A calculates the temporary data r/sk_A by the random number r and the private key sk_A , and then sends the data to the picked user used for re-signature.

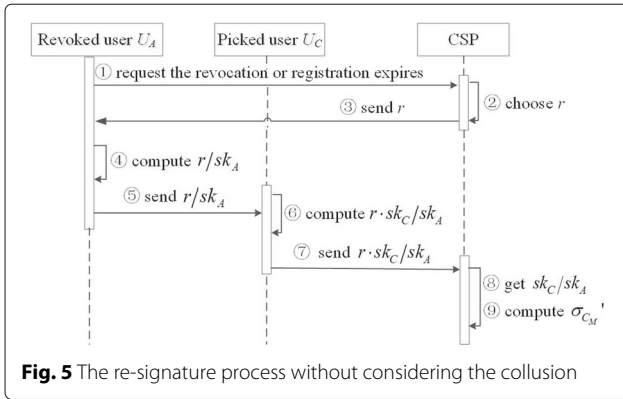


Fig. 5 The re-signature process without considering the collusion

Step 3: After receiving r/sk_A , user U_C uses its private key sk_C to calculate $r \cdot sk_C/sk_A$ and sends it to the CSP.

Step 4: After receiving $r \cdot sk_C/sk_A$, the cloud obtains sk_C/sk_A by dividing the random number r and then calculates the new signature:

$$\begin{aligned} \sigma_{C_M}' &= \left(H(id_M) \omega^{C_M} \right)^{sk_A \cdot (sk_C/sk_A)} \\ &= \left(H(id_M) \omega^{C_M} \right)^{sk_C} \end{aligned} \quad (22)$$

Through these steps, the signature of user U_A is changed to the new signature of user U_C .

The data are re-signed without being downloaded from the CSP, thus reducing the communication and computation cost and improving system efficiency. Nonetheless, the above process fails to consider the collusion between the CSP and the revoked users. If a revoked user reveals the sk_A to the CSP, the latter can calculate the sk_C by sk_C/sk_A , posing a threat to the data security of the user U_C .

To avoid the collusion attack between the CSP and the revoked users, we propose a new re-signature method with the manager participating in the user re-signature process. The re-signature mechanism of avoiding the collusion is followed in Fig. 6.

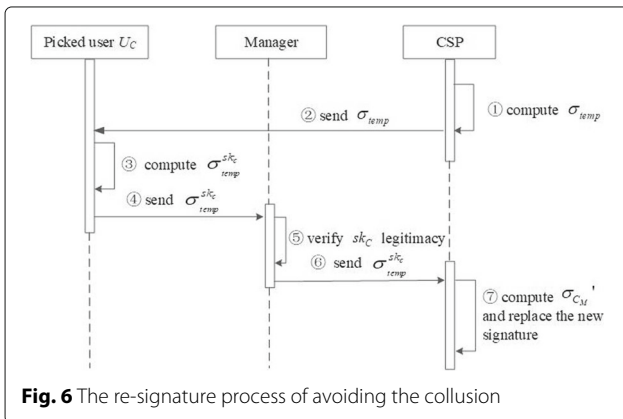


Fig. 6 The re-signature process of avoiding the collusion

Step 1: The CSP computes the temporary signature based on the information of the revoked user U_A and a random number r generated by the CSP.

$$\sigma_{temp} = \left(H(id_M) \omega^{C_M} \right)^r \quad (23)$$

Step 2: The CSP sends the temporary signature σ_{temp} to the user U_C , who receives the temporary signature and computes $\sigma_{temp}^{sk_C}$ using the private key sk_C .

$$\sigma_{temp}^{sk_C} = \left(H(id_M) \omega^{C_M} \right)^{r \cdot sk_C} \quad (24)$$

Then sends $\sigma_{temp}^{sk_C}$ to manager P_i .

Step 3: Manager P_i verifies the following equation:

$$e(\sigma_{temp}, pk_C) = e(\sigma_{temp}^{sk_C}, g) \quad (25)$$

If the equation holds, manager sends $\sigma_{temp}^{sk_C}$ to the CSP.

Step 4: The CSP receives $\sigma_{temp}^{sk_C}$ and computes the new signature in the following formula:

$$\sigma_{C_M}' = \left(\sigma_{temp}^{sk_C} \right)^{1/r} = \left(H(id_M) \omega^{C_M} \right)^{sk_C} \quad (26)$$

Then, the CSP replaces σ_{C_M} with σ_{C_M}' and places the later after the data C_M .

In summary, the CSP does not obtain any information about the private key of any user, which enhances the system security and avoids the collusion attack.

Correctness and security analysis

Decryption correctness

This subsection mainly analyzes the correctness of the data decryption after downloading.

Conclusion 1 *The AES key seed sk_{AES} can be decrypted with t decryption factors.*

Proof Carrying the features of Shamir secret sharing:

$$f_i(0) = d_i = \sum_{j=1}^t f_j(ID_j) \prod_{1 \leq k \leq t, k \neq j} \frac{-ID_{i_k}}{ID_{i_j} - ID_{i_k}} \quad (27)$$

We have the sum of decryption factors as: □

$$\begin{aligned}
 & \sum_{j=1}^t s_{ij} \\
 = & \sum_{j=1}^t C_1 f_i(ID_j) \prod_{1 \leq k \leq t, k \neq j} \frac{-ID_{i_k}}{ID_{ij} - ID_{i_k}} \\
 = & C_1 \sum_{j=1}^t f_i(ID_j) \prod_{1 \leq k \leq t, k \neq j} \frac{-ID_{i_k}}{ID_{ij} - ID_{i_k}} \\
 = & C_1 f_i(0) = SQ \cdot f_i(0) \\
 = & SQ \cdot d_i = Sy \\
 = & (x_2, y_2)
 \end{aligned} \tag{28}$$

Thus, extracting the x_2 , the AES key seed can be derived as the formula (21).

Integrity auditing correctness

The audit Eq. (16) can be derived as follows:

$$\begin{aligned}
 & e\left(\prod_{i=1}^{N_U} \beta_i, g\right) \\
 = & e\left(\prod_{i=1}^{N_U} \left(\prod_{n \in l_i} \sigma_n^{v_n}\right), g\right) \\
 = & e\left(\prod_{i=1}^{N_U} \left(\prod_{n \in l_i} \left(H(id_n) \omega^{C_{M_n}}\right)^{sk_A v_n}\right), g\right) \\
 = & e\left(\prod_{i=1}^{N_U} \left(\prod_{n \in l_i} \left(H(id_n) \omega^{C_{M_n}}\right)^{v_n}\right), g^{sk_A}\right) \\
 = & e\left(\prod_{i=1}^{N_U} \left(\left(\prod_{n \in l_i} H(id_n)^{v_n}\right) \omega^{\sum_{n \in l_i} C_{M_n} v_n}\right), pk_A\right) \\
 = & \prod_{i=1}^{N_U} e\left(\prod_{n \in l_i} H(id_n)^{v_n} \omega^{y_i}, pk_A\right)
 \end{aligned} \tag{29}$$

Encryption/Decryption security

The private key $f_i(0)$ of the system is safe

The private key $f_i(0)$ of the system is created implicitly during the generation of the system public key y . When t managers hand out their secret shares, $f_i(0)$ can be calculated:

$$f_i(0) = \sum_{j=1}^t f_i(ID_j) \prod_{1 \leq k \leq t, k \neq j} \frac{-ID_{i_k}}{ID_{ij} - ID_{i_k}} \tag{30}$$

Both attackers cannot get $f_i(0)$ from the public key of the system. Because of the DL problem is very difficult to solve on the elliptic curve, it is not feasible for attackers to get the private key $f_i(0)$ of managers from the public key of the system $y = f_i(0) Q = d_i Q$.

Similarly, the attackers cannot deduce the secret share $f_i(ID_j)$ of each manager even if they have obtained the public information $Y_{ij} = f_i(ID_j) Q \bmod q$. In other words, the attackers cannot derive $f_i(0)$ from the secret share $f_i(ID_j)$.

The false interaction information between managers can be found

- (1) During the public key generation, manager P_i can verify $f_i(ID_j)$ sent by manager P_j

Proof Since $f_i(ID_j) = d_i + f_{i1}ID_j + \dots + f_{i(t-1)}ID_j^{t-1}$, the following can be derived:

$$\begin{aligned}
 & f_i(ID_j)Q \\
 = & \left(d_i + f_{i1}ID_j + \dots + f_{i(t-1)}ID_j^{t-1}\right)Q \\
 = & d_iQ + f_{i1}ID_jQ + \dots + f_{i(t-1)}ID_j^{t-1}Q
 \end{aligned} \tag{31}$$

Since $\alpha_{iz} = f_{iz}Q$, we have:

$$\begin{aligned}
 & f_i(ID_j)Q \\
 = & d_iQ + \alpha_{i1}ID_j + \dots + \alpha_{i(t-1)}ID_j^{t-1} \\
 = & d_iQ + \sum_{z=1}^{t-1} (ID_j)^z \alpha_{iz} \\
 = & y + \sum_{z=1}^{t-1} (ID_j)^z \alpha_{iz}
 \end{aligned} \tag{32}$$

therefore, the formula (4) is correct and can be verified. □

- (2) During data decryption, no manager is cheated.

Proof Owing to the formula (18), we gain the following:

$$s_{ij}Q = C_1 f_i(ID_j) \prod_{1 \leq k \leq t, k \neq j} \frac{-ID_{i_k}}{ID_{ij} - ID_{i_k}} Q \tag{33}$$

Due to the formula (3), we can get the formula (19). As shown in formula (19), each manager in the set W can verify the decryption factor s_{ij} with the public information, and any false decryption factor will be identified because the formula (19) does not hold, ensuring that no manager is cheated. □

- (3) It is not feasible for the attacker to acquire the ciphertext $c_{sk_{AES}} = (C_1, c_1)$

As mentioned before, the DL problem is very difficult to solve on the elliptic curve. Thus, it is impossible to calculate the random number S from the formula $C_1 = SQ$. The lack of the random number fails the calculation of x_2 from $(x_2, y_2) = Sy$. Therefore, even if the attacker can

intercept the ciphertext $c_{sk_{AES}} = (C_1, c_1)$, it is impossible for him/her to obtain the AES key seed $sk_{AES} = x_2^{-1}c_1$. If a user in the group as an attacker obtains the $sk_{AES} = x_2^{-1}c_1$ and the plaintext M , it is impossible to send the forged ciphertext because of the ciphertext signature and verification.

Performance analysis

In this section, we compare the communication and computation cost with the state of the art using numerical results. Further, we implemented the performances analysis through simulations.

Communication cost

As mentioned in “Construction of the scheme” section, the communication cost of our scheme is analyzed and compared in the following five phases. In the key generation phase, manager P_i sends the public parameter to other group managers, producing a communication cost of $O(1)$. In the encryption phase, manager P_i sends the ciphertext $c_{sk_{AES}}$ of sk_{AES} to each manager or sends the ciphertext data with their signatures to the cloud, which brings in $O(1)$ communication cost. In the integrity auditing phase, the TPA launches a challenge to the CSP, generating $O(N_U)$ communication cost with the number of challenging blocks N_U . Then, the CSP returns the proof to the TPA, which costs $O(1)$ communication cost. In the decryption phase, the communication cost between user and manager or manager and the CSP costs $O(1)$. In the user revocation and re-signature phase, it costs $O(1)$ communication cost to re-signature. To sum up, the total communication cost is $O(N_U)$.

Finally, we also compare the communication cost with several the prior compared work. Note that R is the number of users, $|m|$ is the size of an element of \mathbb{Z}_q , $|h|$ is the bit number of a block, K is the number of the elements. As shown in Table 3, the communication cost of our scheme is the same as [4], and is less than [8, 31], and [36], which realizes high performance and is more efficient than the schemes of [8, 31], and [36].

Computation cost

Succinctly, we define modular exponentiation as Exp_{G_1} , point multiplication as Mul_{G_1} , and pairing operation as

Pair. Compared with Exp_{G_1} , Mul_{G_1} , and Pair, the computation cost of hash operation is ignored [4].

In the key generation phase, the computation cost is $(t + 3)\text{Mul}_{G_1} + (t - 2)\text{Exp}_{G_1}$, which includes the secret sharing and the public key generation. The computation cost of the encryption phase is $\text{Pair} + 4\text{Mul}_{G_1} + 2\text{Exp}_{G_1}$. In the integrity auditing process, the computation cost is $N_U\text{Exp}_{G_1} + (N_U - 1)\text{Mul}_{G_1}$ for the tag proof generation, and $N_U\text{Mul}_{G_1}$ for the data proof generation. After the cloud returns the proof to the TPA, whose verified cost is $(3N_U - 2)\text{Mul}_{G_1} + N_U\text{Pair}$. For the decryption phase, the cost is $(t^2 + 2t + 2)\text{Mul}_{G_1}$. In the user revocation and re-signature stage, the cost of the CSP is $2\text{Mul}_{G_1} + 4\text{Exp}_{G_1}$, and manager's cost is 2Pair .

Table 4 compares the computation cost of our scheme with the contrastive schemes. It can be seen that our scheme is more pair operation and less $2(N_U - 1)\text{Exp}_{G_1} + (N_U - 4)\text{Mul}_{G_1}$ in the pre-processing, which is obviously underlying the scheme [4] for N_U relatively large case. In the proof generation, ours is the same as [4] and is better than [36]. Because [36] is more $(K + (R - 1)N_U)\text{Exp}_{G_1} + ((R - 2)N_U + 1)\text{Mul}_{G_1} + N_U K \text{Mul}_{\mathbb{Z}_p} + K \text{Hash}_{\mathbb{Z}_p}$ computation cost than ours.

Experiment analysis

To evaluate the efficiency of our scheme, our experiments are implemented in Intel(R) Core(TM) i3-8100 CPU @ 3.60GHz, RAM 4GB with Win10 Operation System, utilizing the Eclipse platform with Java programming language and Java Pairing Based Cryptography (JPBC) to realize the cryptography operations for the AES, the ECC, and Shamir secret sharing, and using the SQLite lightweight database to simulate the cloud storage with the security level of 80 bits. In the following, we compare the communication cost of our scheme with [8, 31], and [36]. We simulate the running time of each phase including the key generation phase with the number of managers increasing and the cloud proof phase and the TPA verify phase with the number of blocks increasing for our scheme. Finally, we compare the re-signature time of our scheme with [8].

In the communication cost comparison process, Figs. 7 and 8 compare the communication cost of our scheme with contrastive schemes [8, 31], and [36] following the changes of the number of challenging blocks and the number of users, respectively. Thereinto, the number of challenging blocks is $\{10, 20, 30, \dots, 100\}$ separately, the bit number of a block $|h|$ is 100 with K as 1, and $|m|$ is 160 bit. When analyzing the number of users on the influence of communication cost, we change the number of users for $\{10, 20, 30, \dots, 100\}$, taking $|N_U|$ as 10, $|h|$ as 100, $|m|$ as 160, and K as 100. As shown in Fig. 7, as the number of challenged blocks grows, the communication cost increases

Table 3 Communication cost comparison

Scheme	Communication cost
Ref. [4]	$N_U h $
Ref. [8]	$2R m + N_U h $
Ref.[31]	$(3 + 2N_U) m + N_U h $
Ref. [36]	$(2K + R) m + N_U h $
Ours	$N_U h $

Table 4 Computation cost comparison

Scheme	KeyGen	File Pre-processing	ProofGen	Verifying	Re-signing
Ref. [4]	1Exp	$2N_U \text{Exp}_{G_1} + N_U \text{Mul}_{G_1}$	$N_U \text{Exp}_{G_1} + (2N_U - 1) \text{Mul}_{G_1}$	-	-
Ref. [8]	-	-	-	$(N_U + R) \text{Exp}_{G_1} + (N_U + 2R) \text{Mul}_{G_1} + (R + 1) \text{Pair} + \text{Hash}_{G_1} + R \text{Mul}_{G_2} + N_U \text{Hash}_{G_1}$	-
Ref. [31]	$1 \text{Hash}_{G_1} + 1 \text{Exp}_{G_1}$	1Mul_{Z_q}	$N_U \text{Mul}_{Z_q} + N_U \text{Exp}_{G_1} + N_U \text{Mul}_{G_1} + 1 \text{Exp}_{G_2} + 1 \text{Mul}_{G_2} + 1 \text{Pair}$	$2 \text{Exp}_{G_2} + 1 \text{Pair} + N_U \text{Exp}_{G_1} + N_U \text{Mul}_{G_1}$	$2 \text{Exp}_{G_1} + 1 \text{Hash}_{G_1} + 1 \text{Mul}_{G_1} + 2 \text{Pair}$
Ref. [36]	-	-	$(K + RN_U) \text{Exp}_{G_1} + RN_U \text{Mul}_{G_1} + N_U K \text{Mul}_{Z_p} + K \text{Hash}_{Z_p}$	$(2K + N_U) \text{Exp}_{G_1} + N_U \text{Mul}_{G_1} + N_U \text{Hash}_{G_1} + (R + 2) \text{Pair}$	-
Ours	$(t+3) \text{Mul}_{G_1} + (t-2) \text{Exp}_{G_1}$	$\text{Pair} + 4 \text{Mul}_{G_1} + 2 \text{Exp}_{G_1}$	$N_U \text{Exp}_{G_1} + (2N_U - 1) \text{Mul}_{G_1}$	$(3N_U - 2) \text{Mul}_{G_1} + N_U \text{Pair}$	$2 \text{Mul}_{G_1} + 4 \text{Exp}_{G_1} + 2 \text{Pair}$

continually, and our scheme is smaller communication cost than other comparative schemes. From Fig. 8, it can be seen that our scheme and [31] remain unchanged, but [8, 36] significantly change with the number of users increasing.

In Fig. 9, we plot the running time of each phase for our scheme. In the key generation phase, we choose the sampling value {10, 20, 30, ..., 100} of the number N of the managers and set the threshold value $t = 0.8 * N$ to experiment. Focusing on the integrity auditing phase, we set 1024 bytes as a block and the block numbers are {50, 100, 150, ..., 500}. The cloud proof process in formula (13), (14), (15) and the TPA verify process in formula (16) are experimented, respectively. Thereinto, we plot the key

generation time for the blue line, which are controlled by the blue x axis (the top x axis) with the increase of the number of managers and the blue y axis (the right y axis) with the increase of time. We plot the cloud proof time and the TPA verify time as the increase of the number of blocks with the bottom x axis and the left y axis. As shown in Fig. 9, as the number of managers grows, the running time increases continually for the key generation phase, and as the number of blocks increases, the running time also augments constantly for the cloud proof phase and the TPA verify phase. But it can be seen that the phase running time is short obviously, owing to its unit is microseconds, so our method is run quickly and more efficient.

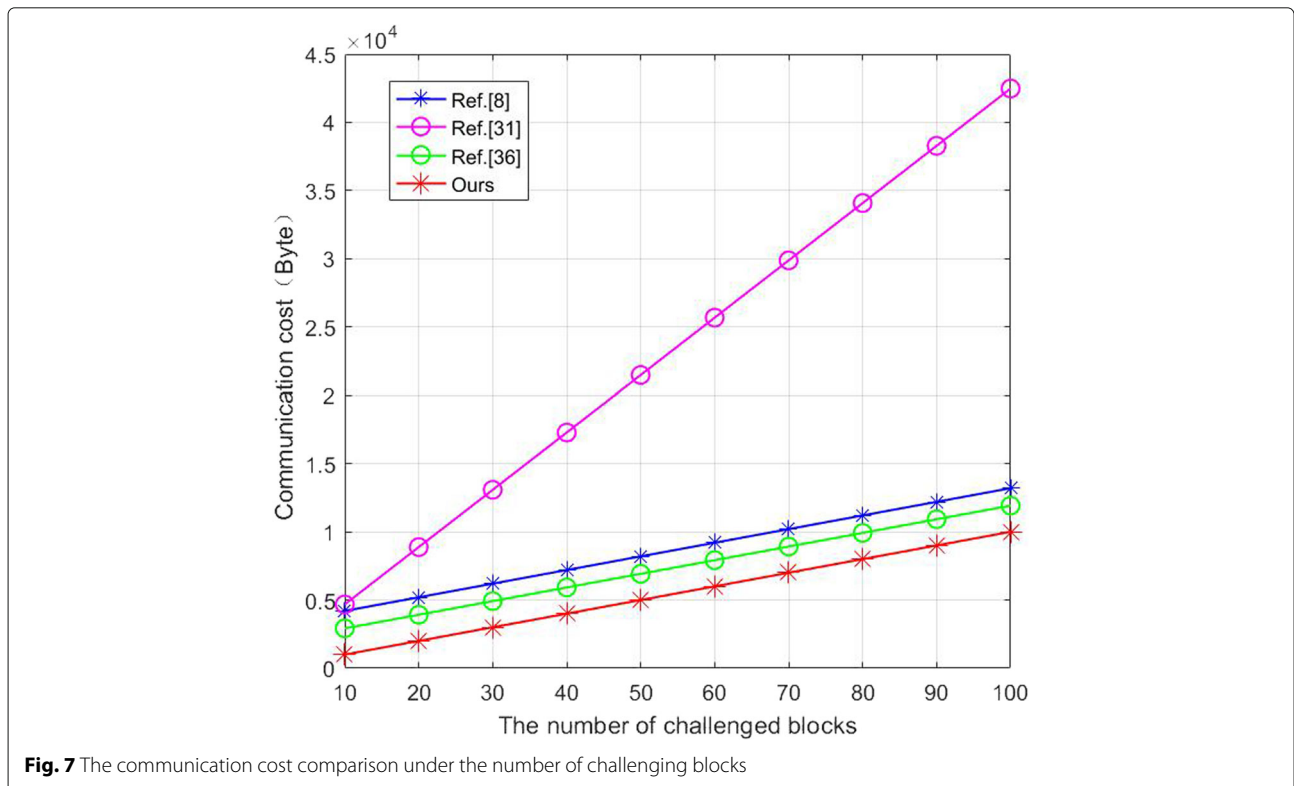


Fig. 7 The communication cost comparison under the number of challenging blocks

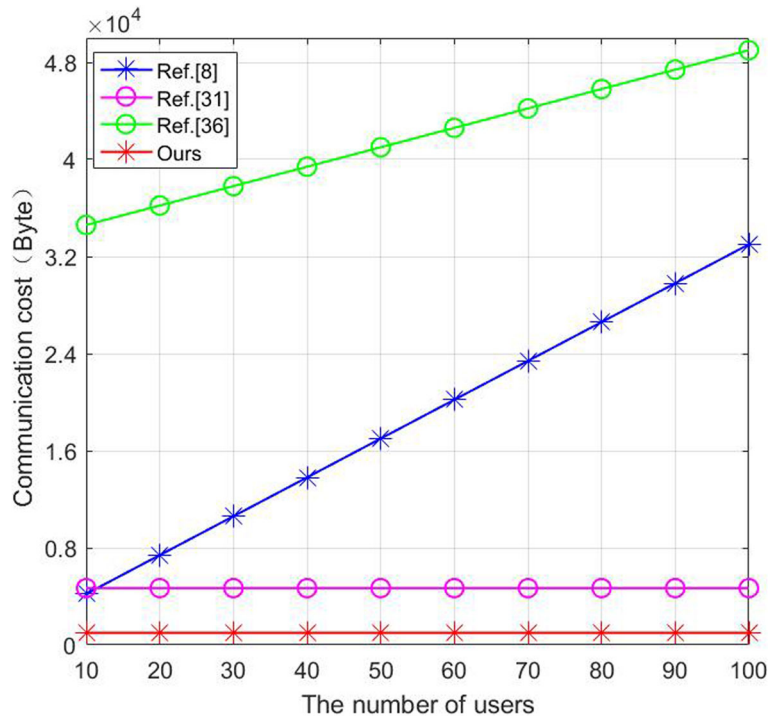


Fig. 8 The communication cost comparison under the number of users

In the re-signature process, we also adopt the sampled method to obtain the re-signature time of the block numbers {0, 50, 100, ..., 350}. As shown in Fig. 10, as the number of re-signature blocks grows, the re-signature time increases constantly for our scheme and comparative scheme [8]. But it is obvious that our scheme consumes less than [8] with linear growth of re-signature blocks. When the number of the re-signature reaches 350, our scheme saves more than 20 seconds.

Conclusions

This paper combines the AES and the ECC with Shamir secret sharing into a novel hybrid encryption method without trusted center, which is suitable for integrity auditing of cloud computing and collaborative learning of machine learning. The hybrid approach not only facilitates key distribution and management but also improves the encryption speed and efficiency of shared data. By our method, the uploaded data are encrypted to avoid

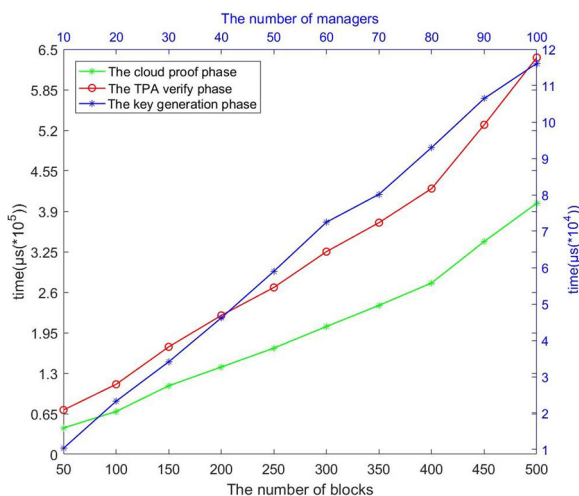


Fig. 9 The phase running time of our scheme

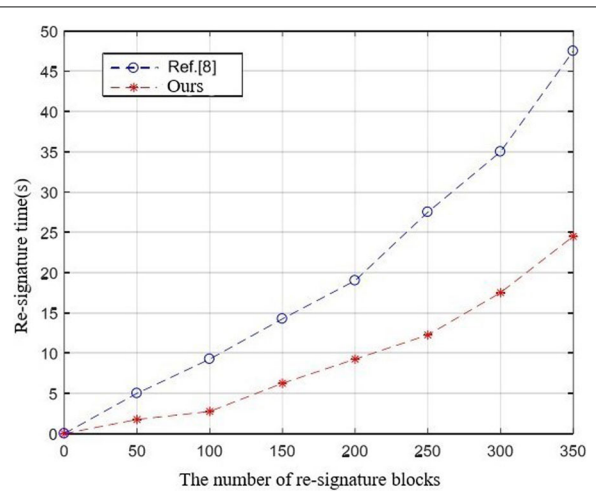


Fig. 10 The re-signature time with the number of re-signature blocks

privacy leakage and security attacks, the downloaded data are audited to keep the data integrity, and the re-signature is against the revoked user from learning any private key or data information. In addition, even if a manager fails to participate in decryption, the other managers can work together to restore the data when the number of participating managers exceeds a preset threshold. This feature ensures the high robustness of the system. The correctness and security of our method are verified through detailed analysis. Moreover, we evaluate the performance and efficiency of our method, and the results show that our scheme is correct, security, and efficient.

Abbreviations

CSPs: Cloud server providers; AES: Advanced encryption standard; ECC: Elliptic curve cryptography; AI: Artificial intelligence; IoT: Internet of things; IIoT: Industrial Internet of things; PKI: Public key infrastructure; TPA: The third-party auditor; ID-DPDP: Identity-based distributed provable data possession; DL: Discrete logarithm; JPBC: Java pairing based cryptography

Acknowledgements

The authors would like to thank the editor and the anonymous reviewers who have helped to improve the paper.

Authors' contributions

All authors contributed to the study conception and design. Conceptualization was performed by Baocang Wang, Hequn Liu, and Yange Chen. The first draft of the manuscript was written by Yange Chen and Hequn Liu. Language was revised by Baljinnayam Sonompil, and all authors commented on previous versions of the manuscript. All author(s) read and approved the final manuscript.

Funding

This research was funded by the National Natural Science Foundation of China under Grant Nos. U19B2021, 61972457, the National Cryptography Development Fund under Grant No. MMJJ20180111, Science & Technology Plan Projects of Henan Province Nos. 212102210084, 192102210295, Key Research and Development Program of Shaanxi under Grant No. 2020ZDLGY08-04, the Innovation Scientists and Technicians Troop Construction Projects of Henan Province.

Availability of data and materials

Data sharing not applicable to this paper as no datasets were generated or analyzed during the current study.

Competing interests

The authors declare that they have no competing interests.

Author details

¹State Key Laboratory of Integrated Service Networks, Xidian University, Taibai South Road, Xi'an, China. ²School of Information Engineering, Xuchang University, Bayi Road, Xuchang, China. ³Chongqing College of Electronic Engineering, university chengdong road, Chongqing, China. ⁴Key Laboratory of Intelligent Perception and Image understanding of Ministry of Education, International Research Center for Intelligent Perception and Computation, Xidian University, Taibai South Road, Xi'an, China.

Received: 1 December 2019 Accepted: 15 December 2020

Published online: 11 January 2021

References

1. Wu D, Au MH, Yan J, Wang H, Wu D, Wang R, et al (2017) Social attribute aware incentive mechanisms for video distribution in device-to-device communications. *IEEE Trans Multimed* 19(8):1908–1920
2. Wu D, Liu Q, Wang H, Wu D, Wang R (2017) Socially aware energy efficient mobile edge collaboration for video distribution. *IEEE Trans Multimed* 19(10):2197–2209. <http://dx.doi.org/10.1109/TMM.2017.2733300>
3. Sangaiah AK, Hosseinabadi AAR, Sadeghilalimi M, Zhang W (2019) Energy consumption in point-coverage wireless sensor networks via bat algorithm. *IEEE Access*:1–1. <http://dx.doi.org/10.1109/ACCESS.2019.2952644>
4. Shen J, Shen J, Chen X, Huang X, Susilo W (2017) An efficient public auditing protocol with novel dynamic structure for cloud data. *IEEE Trans Inf Forensic Secur* 12(10):2402–2415. <http://dx.doi.org/10.1109/TIFS.2017.2705620>
5. Green M (2013) The threat in the cloud. *IEEE Sec Priv* 11(1):86–89. <http://dx.doi.org/10.1109/MSP.2013.20>
6. Fernandes DA, Soares LF, Gomes JV, Freire MM, Inácio PR (2014) Security issues in cloud environments: a survey. *Int J Inf Sec* 13(2):113–170. <http://dx.doi.org/10.1007/s10207-013-0208-7>
7. Dudin E, Smetanin YG (2011) A review of cloud computing. *Sci Tech Inf Process* 38(4):280–284. <http://dx.doi.org/10.3103/S0147688211040083>
8. Wang B, Li B, Li H (2013) Panda: Public auditing for shared data with efficient user revocation in the cloud. *IEEE Trans Serv Comput* 8(1):92–106. <http://dx.doi.org/10.1109/TSC.2013.2295611>
9. Li J, Yan H, Zhang Y (2018) Certificateless public integrity checking of group shared data on cloud storage. *IEEE Trans Serv Comput*:1–12. <http://dx.doi.org/10.1109/TSC.2018.2789893>
10. Wang XA, Liu Y, Sangaiah AK, Zhang J (2019) Improved publicly verifiable group sum evaluation over outsourced data streams in IoT setting. *Computing* 101(7):773–790. <https://doi.org/10.1007/s00607-018-0641-6>
11. Kim D, Kwon H, Hahn C, Hur J (2016) Privacy-preserving public auditing for educational multimedia data in cloud computing. *Multimed Tools Appl* 75(21):13077–13091. <http://dx.doi.org/10.1007/s11042-015-2594-5>
12. Zhang Y, Xu C, Li H, Liang X (2016) Cryptographic public verification of data integrity for cloud storage systems. *IEEE Cloud Comput* 3(5):44–52. <http://dx.doi.org/10.1109/MCC.2016.94>
13. Zhang J, Wang B, He D, et al (2019) Improved secure fuzzy auditing protocol for cloud data storage. *Soft Comput* 23(10):3411–3422. <https://doi.org/10.1007/s00500-017-3000-1>
14. Fu A, Shui Y, Zhang Y, Wang H, Huang C (2017) NPP: a new privacy-aware public auditing scheme for cloud data sharing with group users. *IEEE Trans Big Data* 99:1–1. <http://dx.doi.org/10.1109/TBDATA.2017.2701347>
15. Shen W, Yu J, Xia H, Zhang H, Lu X, Hao R (2017) Light-weight and privacy-preserving secure cloud auditing scheme for group users via the third party medium. *J Netw Comput Appl* 82:56–64. <http://dx.doi.org/10.1016/j.jnca.2017.01.015>
16. Rodrigues JJPC, Wang X, et al (2018) Guest editorial Special Issue on integrated computing: computational intelligence paradigms and Internet of Things for industrial applications. *IEEE Internet of Things J* 5(3):1572–1574. <http://dx.doi.org/10.1109/JIOT.2018.2838958>
17. Phong LT, Aono Y, Hayashi T, Wang L, Moriai S (2018) Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Trans Inf Forensic Secur* 13(5):1333–1345. <http://dx.doi.org/10.1109/TIFS.2017.2787987>
18. Phong LT, Phuong TT (2019) Privacy-preserving deep learning via weight transmission. *IEEE Trans Inf Forensics Secur* 14(11):3003–3015. <http://dx.doi.org/10.1109/TIFS.2019.2911169>
19. Ateniese G, Burns R, Curtmola R, Herring J, Kissner L, Peterson Z, et al (2007) Provable data possession at untrusted stores. In: *Proceedings of the 14th ACM conference on Computer and communications security*. ACM, pp 598–609. <http://dx.doi.org/10.1145/1315245.1315318>
20. Ateniese G, Di Pietro R, 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*. ACM, pp 1–9. <http://dx.doi.org/10.1145/1460877.1460889>
21. Wang Q, Wang C, Li J, Ren K, Lou W (2009) Enabling public verifiability and data dynamics for storage security in cloud computing. In: *European symposium on research in computer security*. Springer, Berlin, pp 355–370. http://dx.doi.org/10.1007/978-3-642-04444-1_22
22. Guo C, Luo N, Bhuiyan MZA, Jie Y, Chen Y, Feng B, et al (2018) Key-aggregate authentication cryptosystem for data sharing in dynamic cloud storage. *Futur Gener Comput Syst* 84:190–199. <http://dx.doi.org/10.1016/j.future.2017.07.038>
23. Shen W, Qin J, Yu J, Hao R, Hu J (2019) Enabling identity-based integrity auditing and data sharing with sensitive information hiding for secure cloud storage. *IEEE Trans Inf Forensic Secur* 14(2):331–346. <http://dx.doi.org/10.1109/TIFS.2018.2850312>

24. Chi PW, Lei CL (2015) Audit-free cloud, Storage via deniable attribute-based encryption. *IEEE Trans Cloud Comput* 6(2):414–427. <http://dx.doi.org/10.1109/TCC.2015.2424882>
25. Yu J, Wang H (2017) Strong key-exposure resilient auditing for secure cloud storage. *IEEE Trans Inf Forensic Secur* 12(8):1931–1940. <http://dx.doi.org/10.1109/TIFS.2017.2695449>
26. Ham L (1994) Group-oriented (t, n) threshold digital signature scheme and digital multisignature. *IEE Proc Comput Digit Techniques* 141(5):307–313. <http://dx.doi.org/10.1049/ip-cdt:19941293>
27. Wang H (2014) Identity-based distributed provable data possession in multicloud storage. *IEEE Trans Serv Comput* 8(2):328–340. <http://dx.doi.org/10.1109/TSC.2014.1>
28. Nagar P, Sethia D (2017) Group authorization using threshold signatures for medical procedures. In: 2017 9th International Conference on Communication Systems and Networks (COMSNETS). IEEE. pp 492–497. <http://dx.doi.org/10.1109/COMSNETS.2017.7945441>
29. Ham L, Wang F (2016) Threshold signature scheme without using polynomial interpolation. *IJ Netw Secur* 18(4):710–717
30. Shen J, Zheng WY, Wang J, Zheng YH, Sun XM, Lee SY (2013) An efficient verifiably encrypted signature from weil pairing. *J Internet Technol* 14(6):947–952
31. Rabaninejad R, Ahmadian AM, Asaar M, Aref M (2019) A lightweight auditing service for shared data with secure user revocation in cloud storage. *IEEE Trans Serv Comput*:1–1. <http://dx.doi.org/10.1109/TSC.2019.2919627>
32. Zhang Y, Yu J, Hao R, Wang C, Ren K (2020) Enabling efficient user revocation in identity-based cloud storage auditing for shared big data. *IEEE Trans Dependable Secure Comput* 17(3):608–619
33. Martin KM (2005) Dynamic access policies for unconditionally secure secret sharing schemes. In: IEEE Information Theory Workshop on Theory and Practice in Information-Theoretic Security. IEEE. pp 61–66. <http://dx.doi.org/10.1109/ITWTP1.2005.1543958>
34. Jie Y, Yu L, Li-yun C, Wei N (2016) A SM2 elliptic curve threshold signature scheme without a trusted center. *KSII Trans Internet Inf Syst* 10(2):897–913. <http://dx.doi.org/10.3837/tiis.2016.02.025>
35. Shimbire N, Deshpande P (2015) Enhancing Distributed Data Storage security for cloud computing using TPA and AES algorithm. In: 2015 International Conference on Computing Communication Control and Automation. IEEE. pp 35–39. <http://dx.doi.org/10.1109/ICCUBEA.2015.16>
36. Wang B, Li B, Li H (2014) Oruta: privacy-preserving public auditing for shared data in the cloud. *IEEE Trans Cloud Comput* 2(1):43–56. <http://dx.doi.org/10.1109/TCC.2014.2299807>
37. Sangaiah AK, Medhane DV, Han T, Hossain MS, Muhammad G (2019) Enforcing position-based confidentiality with machine learning paradigm through mobile edge computing in real-time industrial informatics. *IEEE Trans Ind Inform* 15(7):4189–4196. <http://dx.doi.org/10.1109/TII.2019.2898174>
38. Sangaiah AK, Medhane DV, Bian G, Ghoneim A, Alrashoud M, Hossain MS (2019) Energy-aware green adversary model for Cyber physical security in industrial system. *IEEE Trans Ind Inform*:1–1. <http://dx.doi.org/10.1109/TII.2019.2953289>
39. Shamir A (1979) How to share a secret. *Commun ACM* 22(11):612–613. <http://dx.doi.org/10.1145/359168.359176>
40. Wu D, Si S, Wu S, Wang R (2017) Dynamic trust relationships aware data privacy protection in mobile crowd-sensing. *IEEE Internet Things J*:10. <http://dx.doi.org/10.1109/JIOT.2017.2768073>
41. Zhu Y, Zhang Y (2006) Elliptic curve public key cryptosystem guidance. Science Press, Beijing. p 246
42. Jiang T, Chen X, Ma J (2015) Public integrity auditing for shared dynamic cloud data with group user revocation. *IEEE Trans Comput* 65(8):2363–2373. <http://dx.doi.org/10.1109/TC.2015.2389955>
43. Yuan J, Yu S (2015) Public integrity auditing for dynamic data sharing with multiuser modification. *IEEE Trans Inf Forensic Secur* 10(8):1717–1726. <http://dx.doi.org/10.1109/TIFS.2015.2423264>
44. Liu CW, Hsien WF, Yang CC, Hwang MS (2016) A survey of public auditing for shared data storage with user revocation in cloud computing. *IJ Netw Secur* 18(4):650–666
45. Wang XA, Weng J, Ma J, Yang X (2019) Cryptanalysis of a public authentication protocol for outsourced databases with multi-user modification. *Inf Sci* 488:13–18. <https://doi.org/10.1016/j.ins.2019.03.002>

Publisher's Note

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

Submit your manuscript to a SpringerOpen® journal and benefit from:

- Convenient online submission
- Rigorous peer review
- Open access: articles freely available online
- High visibility within the field
- Retaining the copyright to your article

Submit your next manuscript at ► springeropen.com
