Pers Ubiquit Comput (2018) 22:45–53
https://doi.org/10.1007/s00779-017-1090-5

ORIGINAL ARTICLE

File changes with security proof stored in cloud service systems

Cheng-Yu Yang[1] · Cheng-Ta Huang[2] · Ya-Ping Wang[3] · Yen-Wen Chen[1] · Shiuh-Jeng Wang[4]

Received: 3 August 2017 / Accepted: 26 October 2017 / Published online: 30 December 2017

Abstract This paper proposes a new scheme in cloud service applied to smart home systems based on the technology of the Internet of Things (IoT), and the key technologies include sensing technology and cloud computing ability. The IoT refers to the network of objects, devices, machines, and other physical systems with computing and communication capabilities. On the smart home paradigm, the data collected from sensors can be sensitive information and that security breaches can have devastating economic and social impact. This paper proposes a platform to prevent collusion between users and the cloud service provider (CSP). To protect the privacy of the checked data, the leakage of personnel information in the protocol for the proof must be considered. The proposed protocol can verify users who modify shared files, so that a cloud-storage practice is considered safe. In addition, the proposed method preserves data privacy and minimizes computational cost by applying the bilinearity property of bilinear pairings.

Keywords IoT security · Cryptography · Digital signature · Smart home

✉ Shiuh-Jeng Wang
sjwang@mail.cpu.edu.tw

1 Department of Communication Engineering, National Central University, Taoyuan, Taiwan

2 Department of Information Management, Oriental Institute of Technology, New Taipei, Taiwan

3 Foresight and Innovation Section of Information Management Office, National Police Agency, Ministry of the Interior, Taipei, Taiwan

4 Department of Information Management, Central Police University, Taoyuan, Taiwan

1 Introduction

In recent years, the requirements of safe, comfortable, and convenient for modern house are increasing. The research and design of smart home will satisfy aspire of people's life style [1, 2]. As the development of the Cloud computing and Internet of Things, the applications of smart home are continually improved. The IoT-Cloud paradigm enables a new breed of services; there are huge changes on the concept of smart environments including smart homes, smart buildings, and smart grids.

The new services create meaningful information from the cloud-based analysis of IoT-based data, due to the unlimited resources of the Clouds. On the IoT and Clouds platform, the big data collected from many sensors can be calculated to generate useful and sensitive information for the end users. In the new business models of smart home, paradigm security breaches can have devastating economic and social impact, and cybercrime is increasing in this context [3].

Although users may be under criminal investigation, evidence can be removed from cloud storage by CSPs. Even if the infrastructure provided by a CSP is more robust and reliable than that of personal computing devices, users still face internal and external threats to their security and privacy, including hardware failure, software errors, hackers, and malicious insiders. Moreover, a CSP may discard data that are rarely accessed and reuse the same storage space for other customers.

A considerable amount of data is now stored in the cloud. In addition to providing a secure channel within which to exchange data and download and upload files, determining how cloud systems can further improve security warrants investigation. From this perspective, CSPs must design a mechanism for civil and criminal investigations. However, in cloud systems, user data is generally under the physical control of CSPs. When user data in the cloud is deleted, the remaining small fractions of data are difficult to use in court. Therefore, identifying users who maliciously modify or delete data is essential in cloud data auditing.

To achieve security, several cloud-storage auditing protocols that can verify the integrity of the data stored over a cloud have been proposed [4–20]. Ateniese et al. formally defined protocols for Provable Data Possession (PDP) and presented two PDP schemes based on RSA algorithm in [4]. The server responses queried blocks and tags to the client; then, the client checks whether the server possesses the file. Shacham and Waters [5] improved the security of the original PDP using the data fragmentation concept. Erway et al. improved dynamicity of the original PDP schemes in [6]. Then, Ateniese et al. implemented a model for building public-key homomorphic linear authenticators (HLAs) thus reduce the communication and computation overhead in [7–9]. Yang and Jia improved security in [10] and the authors implemented an efficient data auditing scheme [11].

In [12, 13], Zhu et al. proposed schemes that support the batch auditing for multiple clouds. Zhu et al. developed the scheme minimizes computation and communication costs in [14]. Sookhak et al. design a new data structure to decrease the computation overhead in remote data auditing scheme [15]. Whereas in [16, 17], Yu et al. and Liu et al. have proposed key management systems for cloud storage to reduce the key management cost and private key exposure risk. In [18], Yang et al. proposed an identity tracing protocol that enables the group manager to trace the identity of members when dispute occurs. In [19, 20], the authors provide a privacy-preserving public auditing mechanism for shared data. The common properties of these protocols are as follows: (1) integrity (auditing of data in the cloud computing framework) and (2) efficiency (verification of data with minimum computational cost).

As described herein, our proposed method provides a mechanism to audit the following illegal behavior by malicious users or CSP insiders. CSPs and users cannot privately modify the electronically stored information even through collusion. Our approach provides a more efficient dynamic auditing than do the schemes in [11, 15] and addresses some of the security holes in [12]. Specifically, we propose a public data auditing method based on the bilinear arithmetic of elliptic curves. This method checks data possession in cloud storage at a computational cost lower than those of homomorphic cryptosystems. The two key features of our approach are that it verifies the integrity of the data against collusion and outperforms other state-of-the-art data auditing methods.

The rest of this paper is organized as follows. Section II describes the system model and defines the proposed protocol. Section III describes the properties of the framework, including dynamic operation. The security characteristics are described in Section IV, and Section V presents the results of performance tests of the proposed protocol. Finally, Section VI concludes the paper.

2 Preliminaries and definitions

This section describes the system model and defines the proposed protocol. Cloud-storage architecture includes three roles: that of the CSP, users, and third-party auditor (TPA). The TPA provides public audit services that enable users to review the integrity of their outsourced data.

2.1 Preliminaries

Bilinear map Let G_1, G_2, and G_T be multiplicative cyclic groups of prime order p. Let g_1 and g_2 be the generators of G_1 and G_2, respectively. A bilinear map is a map $e : G_1 \times G_2 \rightarrow G_T$ such that for all $u \in G_1$, $v \in G_2$, and $a, b \in Z_P$, $e\,(u^a, v^b) = e(u, v)^{ab}$.

This bilinearity implies that for any $u_1, u_2 \in G_1$ and $v \in G_2$, $e\ (u_1 \cdot u_2,\ v) = e\ (u_1,\ v) \cdot e\ (u_2,\ v)$ [19]. By using bilinearity, a homomorphic linear file block signature can be combined into one value. In our scheme, this kind of homomorphic property is based on mathematic homomorphism.

2.2 Definition of the system model

After owners create data in smart home systems, the CSP stores the data and maintains access for users. The TPA then performs data storage auditing for both the owners and CSP. The proposed protocol is shown in Fig. 1.

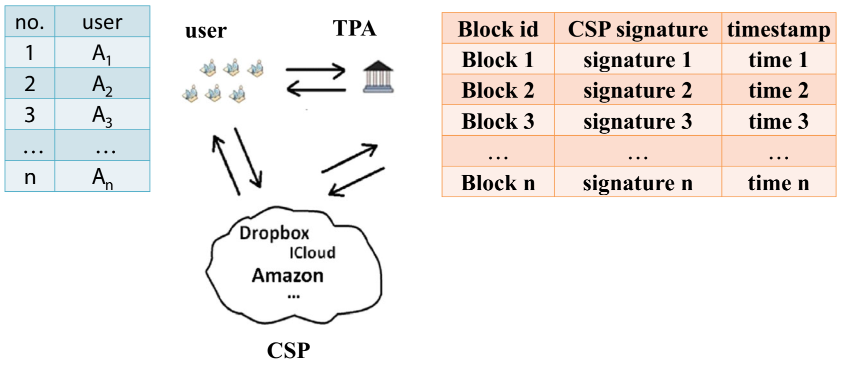
3 Proposed protocol

We consider that users belong to the same group in smart home systems if they share data between them. Notably, the number of users in a group must be more than two. The proposed protocol is detailed in the following subsections.

3.1 Construction

The proposed protocol contains six algorithms, namely, Key.Gen, Setup, Acknowledge, Challenge, Proof, and Verify. For cloud user A, Key.Gen generates a random key pair $(\alpha,\ \beta)$. The secret key is α, and the public key

Fig. 1 Architecture of the proposed protocol



is β. For a file F, setup establishes public parameters that are declared to the public through the TPA; the file F is then processed with file block signatures $T = \{\sigma_i'\}_{1 \le i \le n}$ and transferred to the CSP. The setup of the proposed scheme is shown in Table 1.

The CSP runs Acknowledge to recognize the client data and sends an acknowledgment message to the TPA. The TPA uses Challenge to select some random data blocks $Q = \{(i, v_i)_{i \in [1,\ n]}\}$, where the values of i are distinct. Finally, the TPA sends challenge set Q to the prover. Proof receives the blocks of file F, the set of block signatures T, and the challenge set Q from the auditor as its input, and returns a proof of possession P for the challenged blocks in F. The outputs of the auditing results are then verified (i.e., whether the possession has been proved). Table 2 provides a list of the notations used in the proposed scheme, and the main procedure is detailed in Table 3.

3.2 Dynamic protocol support

This section describes the support for dynamic data operations, including modification, insertion, and deletion. Dynamic data updates are supported through a data structure called the list table (LT). The TPA is responsible for this data structure. The LT prevents the CSP from using a previous version of the stored data rather than the updated one following the verification phase. An LT contains two fields: the original index O_i and the version number V_i. The index of each block in the LT is the actual position of the outsourced data block. We introduce this new data structure to decrease the computational overhead. First, we group all n data blocks into k^x groups. Then, a block is inserted after i or a specific block i is deleted; the verifier moves less than $\frac{n}{k^x}$ blocks and incurs a computation overhead of $\frac{n}{k^x}$. For example, if $x = 2$, the n data blocks are grouped into k^2 groups, as shown in Fig. 2. The procedures for the modification, insertion, and deletion of data are explained as follows:

3.2.1 Data modification

The user executes the modification algorithm as follows:

1) The user modifies the block of the file from m_i to m_i^* and updates $V_i^* = V_i + 1$.
2) The user generates a new block signature $(\sigma_i')^*$ for the modified data block m_i^* as follows:

$$(\sigma_i')^* = H(W_i^*) \cdot \left(\prod_{j=1}^{s} u_j^{m_{i,j}^*} \right)^{\alpha^{(l)}} , \tag{5}$$

where $W_i^* = (FID \,\|i\| O_i \,\| V_i^* \,\|\, t_i^*)$.

3) The user sends a modification request message to the CSP, which includes $((\sigma_i')^*, m_i^*)$.
4) Upon receiving the modification request message, the CSP replaces block m_i with m_i^* and updates the block signature to $(\sigma_i')^*$.
5) The CSP generates the acknowledgment message S_i^* using the secret key SK_{CSP} as follows:

$$S_i^* = (W_i)^{SK_{CSP}} . \tag{6}$$

Then, the CSP transfers S_i^* to the TPA. Upon receiving S_i^*, the TPA saves it in the signature table.

6) The user sends the modification request message $W_i^* = (FID \,\|i\| O_i \,\| V_i^* \,\|\, t_i^*), H(W_i^*)$ to the TPA, and the TPA updates the LT. Then, the CSP sends the modification request message $W_i^* = (FID \,\|i\| O_i^* \| V_i^* \,\|\, t_i^*)\ , H(W_i^*)$ to the TPA, and the TPA further updates the LT.

3.2.2 Data insertion

The user runs the data insertion algorithm through the following steps:

Table 1 Proposed protocol for the setup phase

Let G_1, G_2, and G_T be multiplicative cyclic groups of prime order p, and let e: $G_1 \times G_2 \rightarrow G_T$ be a bilinear map. Let g_1 and g_2 be generators of G_1 and G_2, respectively. Let H: $\{0,1\}^* \rightarrow G_1$ be a secure map-to-point hash function that uniformly maps strings to G_1.

Key.Gen

For a cloud client user A$_1$, choose a random $\alpha^{(1)} \in Z_P$ to generate a random key pair $(\alpha^{(1)}, \beta^{(1)})$ and compute $\beta^{(1)} \leftarrow g_2^{\alpha} \in G_2$. The secret key is $\alpha^{(1)}$ and the public key is $\beta^{(1)}$. For another cloud client user A$_2$ in the same group, the secret key is $\alpha^{(2)}$ and the public key is $\beta^{(2)}$. For user A$_l$, the secret key is $\alpha^{(l)}$ and the public key is $\beta^{(l)}$.

Setup

A file F is divided by applying the data fragmentation technique. This technique reduces number of data block signatures, and splits F into n blocks, each of which are s sectors long: $\{m_{ij}\}_{\substack{1\le i\le n\\ 1\le j\le s}}$. Choose s random values $x_1, x_2, \cdots, x_s$ from some sufficiently large domain Z_P and compute $u_j = g_1^{x_j} \in G_1$, for all $j \in [1, s]$, such that $u = (u_1, u_2, \cdots, u_s)$. The file block signature σ_i', for $1 \le i \le n$, is computed as follows:

$$\sigma_i' = H(W_i)\cdot\left(\prod_{j=1}^{s} u_j^{m_{i,\ j}}\right)^{\alpha^{(l)}} \quad (1)$$

After processing, the file F becomes $\{m_{ij}\}_{\substack{1\le i\le n\\ 1\le j\le s}}$. Together with the signatures $\{\sigma_i'\}_{1\le i\le n}$, the data block identifier $W_i = (FID \,\|i\|O_i \,\|V_i \,\|\, t_i)$, where FID is the file identifier, O_i is the original index of data block i, V_i is the current version of data block i, and t_i is the timestamp of the data block i and the secret key $\alpha^{(l)}$. The cloud client user is provided with the secret key $\alpha^{(l)}$ and a message $H(W_i)$.

The public parameters are given by $pub = (\beta^{(l)}, \{H(W_i), W_i\}_{1\le i\le n}, u, g_2)$, and the secret parameters are $(\alpha^{(l)}, x_1, x_2, \cdots, x_s)$; notably, the user saves the secret parameters.

Finally, public parameters are transferred to the TPA and the processed file F with file block signatures $T = \{\sigma_i'\}_{1\le i\le n}$ and data block identifier $\{W_i\}_{1\le i\le n}$ is transferred to the CSP.

1) The user inserts a new data block m_i^* and sets the initial value for the version V_i^* and the original index $O_i^* = n+1$ of the data block m_i^*.

Table 2 Notations

Symbol	Definition
a	Secret key
β	Public key
n	The number of blocks in a file
s	The number of sectors in a block
F	The processed file with $n \times s$ sectors, $F = \{m_{ij}\}_{\substack{1\le i\le n\\ 1\le j\le s}}$
T	The set of file block signatures
Q	The set of index-coefficient pairs
P	The response for the challenge

Table 3 Proposed protocol for the main procedures

Acknowledge

For the CSP, the secret key is SK_{CSP} and the public key is PK_{CSP}. The CSP receives the processed file F with file block signatures T and the data block identifier $\{W_i\}_{1\le i\le n}$; it generates S_i using the secret key SK_{CSP} as follows:

$$S_i = (W_i)^{SK_{CSP}} \quad (2)$$

The CSP then transfers $\delta = \{S_i\}_{1\le i\le n}$ to the TPA as the CSP acknowledgment message for the processed file F with file block signatures T and the data block identifier $\{W_i\}_{1\le i\le n}$. Upon receiving the acknowledgment message, the TPA saves it in the signature table as shown in Table 4.

Challenge

The TPA selects some random data blocks (i.e., subsets $[1, i]$ of $[1, n]$) to construct the challenge set Q. For each i, a random element $v_i \in Z_p^*$ is generated. Let Q be the set $\{(i, v_i)_{i\in[1,\ n]}\}$ for distinct values of i. Finally, the TPA sends challenge Q to the prover.

Proof

The CSP parses the processed file F as $\{m_{ij}\}_{\substack{1\le i\le n\\ 1\le j\le s}}$, along with $T = \{\sigma_i'\}_{1\le i\le n}$, which comes from the user. The CSP also parses the message sent by the TPA as Q. Then it computes $\mu' = \{\mu_j'\}_{1\le j\le s}$ as follows:

$$\mu_j' = \sum_{(i,v_i)\in Q} v_i\cdot m_{i,j}.$$

To ensure security, the CSP chooses s random values $r_j \in Z_p$. In addition, $\rho \in Z_p$ is a random value and $R = (r_1, r_2, \cdots, r_s)$. Notably, R and ρ are blind factors designed to prevent data leakage attacks. Then, the CSP computes $\mu = \{\mu_j\}_{1\le j\le s}$, γ, and τ.

$$\mu_j = r_j + \mu_j'. \quad (3)$$

$$\gamma = \prod_{j=1}^{s} e\left(u_j^{-r_j}, \beta^{(l)}\right).$$

$$\tau = \prod_{j=1}^{s} e\left(u_j^{\mu_j}, \beta^{(l)}\right).$$

Next, σ and ψ are computed as follows:

$$\sigma_i = \rho\cdot\sigma_i'.$$

$$\sigma = \prod_{(i,v_i)\in Q} \sigma_i^{v_i} = \prod_{(i,v_i)\in Q} \left\{\rho\cdot\sigma_i'\right\}^{v_i}.$$

$$\psi = e\left(\prod_{(i,v_i)\in Q} \rho^{v_i}, g_2\right).$$

Finally, the CSP sends the response $P = (\tau, \gamma, \sigma, \psi)$ to the verifier.

Verify

The TPA also parses the CSP's response P to obtain τ, γ, σ, and ψ and checks whether the following equation holds:

$$e\left(\prod_{(i,v_i)\in Q} \sigma, g_2\right) = \psi\cdot e\left(\prod_{(i,v_i)\in Q} H(W_i)^{v_i}, g_2\right)\cdot\gamma\cdot\tau. \quad (4)$$

If (4) holds, the algorithm outputs 1; otherwise, it outputs 0. The TPA checks the acknowledgment message $\delta = \{S_i\}_{1\le i\le n}$ in the signature table to determine whether the processed file F was created by the user A. If so, the algorithm accepts it; otherwise, it rejects it.

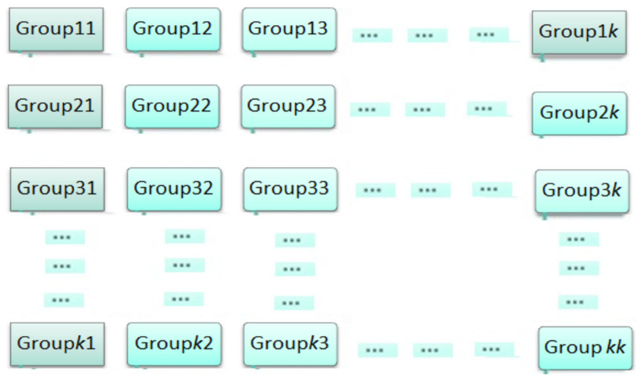


Fig. 2 Division of n data blocks into k^2 groups

2) The user generates a block signature $(\sigma_i')^*$ for the new data block m_i^*; then by (5),

$$(\sigma_i')^* = H(W_i^*)\cdot\left(\prod_{j=1}^{s} u_j^{m_{i,j}^*}\right)^{\alpha^{(l)}},$$

where $W_i^* = (FID\,\|i\|O_i^*\|V_i^*\,\|\,t_i^*)$.

3) The user sends an insert request message to the CSP, which includes $(\sigma_i^*, m_i^*, W_i^*)$. Upon receiving the insert request message, the CSP inserts m_i^* and the block signature σ_i^*.
4) The CSP generates the acknowledgment message S_i^* using its secret key SK_{CSP} as follows:

$$S_i^* = (W_i)^{SK_{CSP}}.$$

The CSP then transfers S_i to the TPA. Upon receiving the acknowledgment message S_i^*, the TPA saves it in the signature table.

5) The user sends the insert request message $W_i^* = (FID\,\|i\|O_i^*\|V_i^*\,\|\,t_i^*)$, $H(W_i^*)$ to the TPA; the TPA inserts a new row in the LT after the ith block and shifts the subsequent blocks down one position.

Table 4 Signature table

Block *id*	CSP signature	Timestamp
Block 1	Signature 1	Time 1
Block 2	Signature 2	Time 2
Block 3	Signature 3	Time 3
…	…	…
Block n	Signature n	Time n

6) The TPA receives the insert request message $W_i^* = (FID\,\|i\|O_i^*\|V_i^*\,\|\,t_i^*)$ and constructs a new row in the LT.

3.2.3 Data deletion

The user runs the data deletion algorithm by performing the following steps:

1) The user deletes a data block m_i and updates $V_i^* = V_i$ as well as the original index of data block $O_i^* = O_i$.
2) The user generates a data block identifier $W_i^* = (FID\,\|i\|O_i^*\|V_i^*\,\|\,t_i^*)$ for the data block m_i.
3) The user sends the delete request message to the CSP, which includes (σ_i, m_i). Upon receiving the delete request message, the CSP deletes m_i and the block signature σ_i.
4) The CSP generates the acknowledgment message S_i^* by using its secret key SK_{CSP} as follows:

$$S_i^* = (W_i)^{SK_{CSP}}.$$

Then, the CSP transfers S_i to the TPA. Upon receiving the acknowledgment message S_i^*, the TPA saves it in the signature table.

5) The user sends the delete request message $W_i^* = (FID\,\|i\|O_i^*\|V_i^*\,\|\,t_i^*)$,$H(W_i^*)$ to the TPA, and the TPA saves all of the message requests in the LT.

4 Security analysis

In this section, we first validate the correctness of the proposed protocol. To assess the security of our proposed scheme, we identify where the data may have been tampered with or lost; then, we describe how to prevent such collusion.

4.1 Correctness

The correctness of our protocol is illustrated in the following theorem:

Theorem 1 *In our proposed protocol, the TPA can determine whether chosen data blocks are correctly stored.*

Proof. The verification of (4) in Section III can be rewritten in detail as follows:

$$e\left(\prod_{(i,v_i)\in Q}\sigma, g_2\right) = \psi\cdot e\left(\prod_{(i,v_i)\in Q} H(W_i)^{v_i}, g_2\right)\cdot\gamma\cdot\tau.$$

Because σ can be expressed as

$$\sigma = \left(\prod_{(i,v_i)\in Q} \rho^{v_i}\right)\cdot \prod_{(i,v_i)\in Q} \left\{ H(W_i)\cdot\left(\prod_{j=1}^{s} u_j^{m_{i,j}}\right)^{\alpha^{(l)}}\right\}^{v_i}$$

$$= \prod_{(i,v_i)\in Q} \left\{ \rho^{v_i}\cdot H(W_i)^{v_i}\cdot\left(\prod_{j=1}^{s} u_j^{\mu_j'}\right)^{\alpha^{(l)}}\right\}.$$

So, the equation of LEFT as below,

$$LEFT = e\left(\prod_{(i,v_i)\in Q} \sigma_i^{v_i}, g_2\right)$$

$$= e\left(\prod_{(i,v_i)\in Q} \left\{ \rho^{v_i}\cdot H(W_i)^{v_i}\cdot\left(\prod_{j=1}^{s} u_j^{\mu_j'}\right)^{\alpha^{(l)}}\right\}, g_2\right).$$

The right-hand side of (4) is the prover's response τ, γ, ψ and the challenge set (i, v_i); by using (3), this can be expressed as follows:

$$RIGHT = e\left(\prod_{(i,v_i)\in Q} \rho^{v_i}, g_2\right)\cdot e\left(\prod_{(i,v_i)\in Q} H(W_i)^{v_i}, g_2\right)\cdot \prod_{j=1}^{s} e\left(u_j^{-r_j}, \beta^{(l)}\right)\cdot \prod_{j=1}^{s} e\left(u_j^{\mu_j}, \beta^{(l)}\right)$$

$$= e\left(\prod_{(i,v_i)\in Q} \rho^{v_i}\cdot H(W_i)^{v_i}, g_2\right)\cdot \prod_{j=1}^{s} e\left(u_j^{\mu_j - r_j}, \beta^{(l)}\right)$$

$$= e\left(\prod_{(i,v_i)\in Q} \rho^{v_i}\cdot H(W_i)^{v_i}, g_2\right)\cdot \prod_{j=1}^{s} e\left(u_j^{\mu_j'}, \beta^{(l)}\right).$$

Then, we can show that the right- and left-hand sides are equal:

$$RIGHT = e\left(\prod_{i\in Q} \rho^{v_i}\cdot H(W_i)^{v_i}, g_2\right)\cdot \prod_{j=1}^{s} e\left(u_j^{\mu_j'}, g_2^{\alpha^{(l)}}\right)$$

$$= e\left(\prod_{(i,v_i)\in Q} \rho^{v_i}\cdot H(W_i)^{v_i}\cdot\left\{\prod_{j=1}^{s} u_j^{\mu_j'}\right\}^{\alpha^{(l)}}, g_2\right).$$

4.2 Security model

In this section, we discuss how the proposed protocol provides the security properties introduced in section III. The proof of deletion may be bypassed through collusion between the user and CSP. Therefore, no traces of deleted files will remain. For example, Alice sends Bob some files to satisfy a contract, but the files might be deleted by the CSP. In this scenario, Bob is honest, but Alice is malicious and colludes with the CSP to deceive Bob for financial gain. Later, Alice can present the proof of the deletion of the file to accuse Bob of evidence tampering.

Therefore, addressing collusion between the user and CSP in cloud-storage auditing is a crucial problem. In proposed protocol, TPA saves a relative proof that is signed by CSP and thus can prove whether the message was modified or deleted. This is similar to monitoring a car in a public parking lot: if the car is damaged, the owner can identify the perpetrator.

Theorem 2 *If one is the original owner of a file F in proposed smart home systems, and then the block signature cannot be forged by other users.*

Proof. In the proposed protocol, the request message and data block identifiers are created by the user by using the secret key $\alpha^{(l)}$ and the public key $\beta^{(l)}$. The shared file is divided into a number of blocks, each of which is signed by the user. When a user modifies the shared file, they sign the new block using their secret key. From (1) and **Theorem** 1 we prove that the block signature cannot be forged by other users.

$$\sigma_i' = H(W_i)\cdot\left(\prod_{j=1}^{s} u_j^{m_{i,\ j}}\right)^{\alpha^{(l)}}.$$

Meanwhile, for the CSP, the secret key is SK_{CSP} and the public key is PK_{CSP}; the acknowledgment message of the CSP, $S_i = (W_i)^{SK_{CSP}}$, is proved as follows:

$$e(S_i, g_2) = e\left((W_i)^{SK_{CSP}}, g_2\right)$$

$$= e\left((W_i), g_2^{SK_{CSP}}\right)$$

$$= e(W_i, PK_{CSP}).$$

By utilizing the Boneh–Lynn–Shacham (BLS) signature technique [21, 22], other users cannot compute a valid signature without the secret key of the CSP. Thus, users cannot deny possession files that they created.

Theorem 3 If a user performs modifications, deletions, or insertions to a file in proposed smart home systems, then the block signatures of these modifications, deletions, or insertions cannot be forged.

Proof. In the proposed protocol, the request message and data block identifiers are created by the user by using the secret key $\alpha^{(l)}$ and the public key $\beta^{(l)}$. From (5) and **Theorem** 1, we prove that the block signature cannot be forged.

$$\left(\sigma_i^{'}\right)^* = H\left(W_i^*\right)\cdot\left(\prod_{j=1}^{s} u_j^{m_{i,j}^*}\right)^{\alpha^{(l)}},$$

For the CSP, the secret key is SK_{CSP} and the public key is PK_{CSP}; the acknowledgment message of the CSP, $S_i^* = \left(W_i^*\right)^{SK_{CSP}}$, is proved as follows:

$$\begin{aligned} e\left(S_i^*, g_2\right) &= e\left(\left(W_i^*\right)^{SK_{CSP}}, g_2\right) \\ &= e\left(\left(W_i^*\right), g_2^{SK_{CSP}}\right) \\ &= e\left(W_i^*\right), PK_{CSP}\right). \end{aligned}$$

Similar to the proof of **Theorem** 2, by utilizing the BLS signature technique [21, 22], other users cannot compute a valid signature without the secret key of the CSP. Thus, the CSP and these users cannot deny modifications, deletions, or insertions that they make to files.

However, in the scheme in [11], the CSP and these users can deny modifications to files. We take a group with two users as an example (Table 5). In this scenario, Bob is honest but Alice is malicious and colludes with the CSP to modify a shared file in order to deceive Bob for financial gain. Without proof of data modification held by a TPA, the CSP can collude with Alice and delete file blocks that were originally created or modified by Bob. Then, Alice can accuse Bob of tampering with evidence. On the other hand, with proof of data modification held by a TPA, the CSP cannot deny the proof that is signed by CSP. Thus, the proposed method provides a mechanism to audit the illegal behavior by malicious users or CSP insiders.

Theorem 4 Our protocol cannot reveal content to the TPA when an attack happens in [12].

Proof. The parameters are the same as those presented in Section III. Let $\mu_j^{'} = \sum_{(i,v_i)\in Q} v_i \cdot m_{i,j}$ and $\sigma^{'} = \prod_{i\in Q}(\sigma_i)^{v_i}$. The CSP chooses random r_j and ρ; then,

$$\mu_j = r_j + \mu_j^{'} = r_j + \sum_{(i,vi)\in Q} v_i \cdot m_{i,j}$$

$$\sigma_i = \rho\cdot\sigma_i^{'}.$$

To prevent from data leakage attack, the CSP needs to blind both μ_j and σ_i. Then, the CSP computes $\mu = \left\{\mu_j\right\}_{1\le j\le s}$, γ, and τ using R and ρ. As a result, μ, γ, and τ are unknown for TPA, as the response proof of storage correctness to the TPA. Therefore, the verifier cannot obtain $\left\{m_{ij}\right\}_{\substack{1\le i\le n \\ 1\le j\le s}}$ from $\mu = \left\{\mu_j\right\}_{1\le j\le s}$; similarly, the verifier cannot obtain the block signature $\sigma_i^{'}$. Because of the random masking of r_j and ρ, the verifier cannot obtain information about $\mu_j^{'}$ and $\sigma_i^{'}$.

To protect the privacy of the checked data, the leakage of personnel information in the protocol for the proof of data possession must be considered. Without background information, the proposed scheme provides security superior to the data auditing scheme proposed by Yang and Jia [11].

Theorem 5 The proposed protocol can resist the replay attack.

Proof. In proposed protocol, the TPA picks some random data blocks, i.e., subset $[1, i]$ of the set $[1, n]$, to construct the challenge set Q. And the TPA generates a random element $v_i \in Z_p^*$ for each i. Thus, there are different challenge sets Q in different challenge-response phase. The CSP cannot use the previous proof to pass the verification without retrieving the challenged data blocks and data tags.

Besides, the hash value $H(W_i)^{v_i}$ is unknown for CSP, the adversary can not forge the hash value. As result, the malicious CSP cannot launch the replay attack based on the same values.

5 Performance analysis

We compare the computational cost of the proposed scheme with those of [11, 15]. The experiments were performed on a Linux system with an Intel Core i5-2435M CPU at 2.40 GHz and with 2 GB RAM. We used the pairing-based cryptography library version 0.5.12 [23] to simulate our auditing scheme and the computational costs of dynamic data updates were simulated with MySQL Version 14.14 Distrib. 5.5.52. We also used a MNT d159 elliptic curve [23], with a base field size of 159 b and embedding degree of 6. The MNT d159 curve has a 160-b group order, which means that p is a 160-b long prime.

Table 5 Group with two users

Number	User
1	Alice
2	Bob

Table 6 Comparison of the dynamic data update operations of remote data auditing protocols

Computation cost	Scheme		
	Yang et al. (2012) [11]	Sookhak et al. (2015) [15]	Our scheme
Modification	$O\ (t)$	$O\ (t)$	$O\ (t)$
Insertion	$O\ (n)$	$O\ (\frac{n}{k})$	$O\ (\frac{n}{k^x})$
Deletion	$O\ (n)$	$O\ (\frac{n}{k})$	$O\ (\frac{n}{k^x})$

n is the total number of data blocks in the file; t is the number of challenged data blocks in the auditing query; k is the number of groups; x is an integer ($x \geq 2$)

The file used in the simulations of the proposed and existing auditing schemes had 125,000 data blocks; the total size was less than 1 GB and the size of each block was 8 kB (Fig. 3). The computational cost incurred by finding the position of each block is negligible, because the computation is shorter than those of the data update. Moreover, the positions can easily be precomputed before data update operations; thus, the computation overhead due to finding the positions of data block is negligible.

The mathematical model of the proposed scheme is supported by empirical results for file sizes from 1 to 5 GB, where the number of update requests for insertion or deletion is 50, and the number of groups k is equal to 10 (Fig. 3). To insert a block after i or delete a specific block i in the method proposed by Yang et al., the computational overhead is n. The computational overhead of the method proposed by Sookhak et al. is $\frac{n}{k}$. In our method, the maximum computational overhead of the verifier (when $x = 2$) is only $\frac{n}{k^2}$.

We compare the computational complexity of our method with that proposed by Sookhak et al. for dynamic data update operations (Table 6). The complexity of our method is $O(\frac{n}{k^x})$; when $x = 2$ and the CSP inserts a block after i or deletes block i, the verifier moves fewer than $\frac{n}{k^2}$ computational overhead of $\frac{n}{k^2}$.

In the method proposed by Yang et al., to insert a block after i or delete block i, the verifier must move $n - i$ blocks in the data structure. Therefore, the computational overhead of their method for inserting and deleting operations is $O\ (n)$. Similarly, for these operations in the method proposed by Sookhak et al., the verifier must move $\frac{n}{k} - i$ blocks, which incurs a computational overhead of $\frac{n}{k}$. The complexity of their method is $O\ (\frac{n}{k})$; thus, our method is faster than that of Sookhak et al.

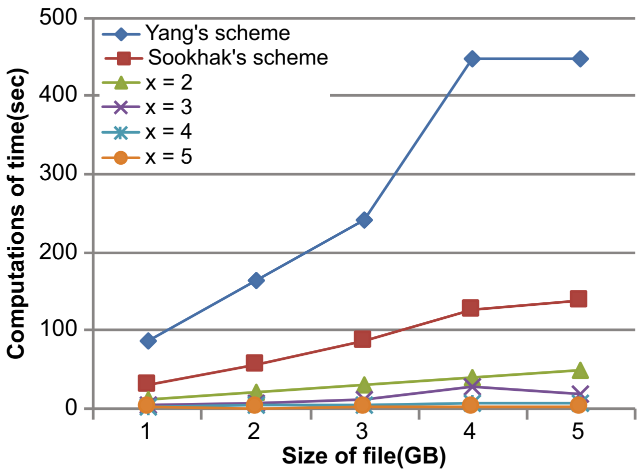


Fig. 3 Comparison of the computational costs of update operations for file sizes from 1 to 5 GB

6 Conclusions

In this study, we developed an efficient and secure collusion-resistant protocol in smart home systems. The contributions of this paper are summarized as follows:

1) First, we designed a suitable framework for collusion resistance and formalized the definition of the public auditing scheme for the TPA, which supports the public audit ability of remote data.
2) Additionally, our scheme is more efficient than other state-of-the-art data auditing methods. In particular, we introduced a new data structure that decreases the computational overhead in dynamic data update operations.

Overall, we proposed an efficient protocol for proving data possession in cloud service applied to smart home systems. The proposed method minimizes the computational cost incurred through the application of bilinear pairings.

Funding information This research was partially supported by the Ministry of Science and Technology of the Republic of China under the Grant MOST 106-2221-E-015-001-.

References

1. Chen EY, Pei Y, Chen S, Tian Y, Kotcher R, Tague P (2014) Oauth demystified for mobile application developers. In: Proceedings of the 2014 ACM SIGSAC conference on computer and communications security, pp. 892–903
2. Chin E, Felt AP, Greenwood K, Wagner D (2011) Analyzing inter-application communication in Android. In: Proceedings of the 9th international conference on mobile systems, applications, and services, pp. 239–252
3. Denning T, Kohno T, Levy HM (2013) Computer security and the modern home. Commun ACM 56(1):94–103
4. Ateniese G, Burns R, Curtmola R, Herring J, Kissner L, Peterson Z, Song D (2007) Provable data possession at untrusted stores. In: Proceedings of the 14th ACM conference on computer and communications security, pp. 598–609
5. Shacham H, Waters B (2008) Compact proofs of retrievability. In: Proceedings of the 14th international conference theory and application of cryptology and information security: advances in cryptology, pp. 90–107
6. Erway C, Kupcu A, Papamanthou C, Tamassia R (2009) Dynamic provable data possession. In: Proceedings of the 15th ACM conference on computer and communication security, pp. 213–222
7. Ateniese G, Pietro RD, Mancini LV, Tsudik G (2008) Scalable and efficient provable data possession. In: Proceedings of the 4th international conference on security and privacy in communication netowrks, article 9, pp. 1–10
8. Ateniese G, Kamara S, Katz J (2009) Proofs of storage from homomorphic identification protocols. In: Proceedings of the 15th international conference theory and application of cryptology and information security: advances in cryptology, pp. 319–333
9. Ateniese G, Burns R, Curtmola R, Herring J, Khan O, Kissner L, Peterson Z, Song D (2011) Remote data checking using provable data possession. ACM Trans Inf Syst Secur 14(1):1–34
10. Wang Q, Wang C, Ren K, Lou W, Li J (2011) Enabling public auditability and data dynamics for storage security in cloud computing. IEEE Trans Parallel Distrib Syst 22(5):847–859
11. Yang K, Jia X (2012) An efficient and secure dynamic auditing protocol for data storage in cloud computing. IEEE Trans Parallel Distrib Syst 24(9):1717–1726
12. Zhu Y, Wang S, Hu H, Ahn GJ, Ma D (2012) Secure collaborative integrity verification for hybrid cloud environment. Int J Coop Inf Syst 21(3):165–197
13. Zhu Y, Hu H, Ahn GJ, Yu M (2012) Cooperative provable data possession for integrity verification in multi-cloud storage. IEEE Trans Parallel Distrib Syst 23(12):2231–2244
14. Zhu Y, Ahn GJ, Hu H, Yau SS, An HG, CJ H (2013) Dynamic audit services for outsourced storages in clouds. IEEE Trans Serv Comput 6(2):227–238
15. Sookhak M, Gani A, Khan MK, Buyya R (2017) Dynamic remote data auditing for securing big data storage in cloud computing. Inf Sci 380:101–116
16. Yu J, Ren K, Wang C, Varadharajan V (2015) Enabling cloud storage auditing with key-exposure resistance. IEEE Trans Inf Forensics Secur 10(6):1167–1179
17. Liu X, Deng RH, Choo KKR, Weng J (2016) An efficient privacy-preserving outsourced calculation toolkit with multiple keys. IEEE Trans Inf Forensics Secur 11(11):2401–2414
18. Yang G, Yu J, Shen W, Su Q, Fu Z, Hao R (2016) Enabling public auditing for shared data in cloud storage supporting identity privacy and traceability. J Syst Software 113:130–139
19. 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
20. Yuan J, Yu S (2015) Public integrity auditing for dynamic data sharing with multiuser modification. IEEE Trans Inform Forensics Secur 10(8):1717–1726
21. Boneh D, Lynn B, Gentry C, Shacham H (2003) Aggregate and veriably encrypted signatures from bilinear maps. In: Proceedings of the 22nd international conference on theory and applications of cryptographic techniques, pp. 416–432
22. Boneh D, Lynn B, Shacham H (2004) Short signatures from the weil pairing. J Cryptol 17(4):297–319
23. Lynn B (2013) PBC library–the pairing-based cryptography library. http://crypto.stanford.edu/pbc/. Accessed 12 Sep. 2017