An efficient and secure data auditing scheme based on fog-to-cloud computing for Internet of things scenarios

Jun-Feng Tian¹,² and Hao-Ning Wang¹,²

Abstract
With the widespread use of fog-to-cloud computing–based Internet of things devices, how to ensure the integrity of the data uploaded to the cloud has become one of the most important security issues. This article proposes an efficient and secure data auditing scheme based on fog-to-cloud computing for Internet of things scenarios, which can better meet performance and security requirements. The proposed scheme realizes data sharing under the condition of protecting privacy by encrypting sensitive information. Using the private key separation method, the private key is divided into two parts using identity information generation and random selection which are, respectively, held by the user and the fog center. Then, using the two-time signature method, the Internet of things and fog computing center use two parts of the private key to generate the original signature and final signature in two separate times. Since the fog computing center only has a part of the private key generated using the identity information, the security of the system will not be damaged due to the leakage of part of the private key held by the fog center, and the fog center significantly participates in the signature generation process, which significantly reduces the computation and communication overhead of the Internet of things device. Security analysis and performance evaluation show that the proposed scheme is safe and efficient.

Keywords
Data integrity, data sharing, Internet of things, fog-to-cloud computing

Date received: 10 December 2019; accepted: 10 March 2020
Handling Editor: Yan Huang

Introduction
With the introduction and widely used of Internet of things (IoT) devices in our life and industrial production, Cisco first proposed fog computing in 2012¹ is also widely considered as a new paradigm that can effectively support for distributed, delay sensitive, and quality of service (QoS)-aware. As an intermediate device between the IoT device and the cloud, the fog computing center contains multiple fog nodes. The fog node has basic computing power, storage capabilities, and network resources to meet the data preprocessing and transmission requirements.² Therefore, the fog computing is an effective and practical solution from the perspective of delay constraints and resource constraints. It is estimated that the data currently generated by IoT devices have exceeded half of the total

¹School of Cyberspace Security and Computer Institute, Hebei University, Baoding, China
²Hebei Key Laboratory of High Confidence Information Systems, Hebei University, Baoding, China

Corresponding author:
Hao-Ning Wang, School of Cyberspace Security and Computer Institute, Hebei University, Baoding 071000, Hebei, China.
Email: 479601516@qq.com

Creative Commons CC BY: This article is distributed under the terms of the Creative Commons Attribution 4.0 License (https://creativecommons.org/licenses/by/4.0/) which permits any use, reproduction and distribution of the work without further permission provided the original work is attributed as specified on the SAGE and Open Access pages (https://us.sagepub.com/en-us/nam/open-access-at-sage).
In 2019, the data generated by IoT devices will reach 500 ZB, which will be preprocessed by fog and then outsourced to the cloud for further analysis and storage. As an emerging technology, fog-to-cloud computing still faces many security challenges. One of the most concerned issues is how to ensure the integrity and correctness of important data uploaded to the cloud. Like traditional cloud storage, the cloud service provider (CSP) also tries to hide the fact that data are corrupted because of its own benefits or reputation, and even deletes some data that users rarely access to save storage space. To this end, it is important to design an efficient and secure cloud data auditing scheme. At present, many cloud data auditing schemes have been proposed. However, these schemes cannot be directly applied to IoT devices based on fog-to-cloud computing. The reasons are as follows: (1) the signature generation overhead in traditional auditing scheme for cloud storage is too high, and it is difficult to apply to IoT devices with low computing power and (2) the traditional auditing scheme for cloud storage does not involve fog nodes, but the fog node plays a very important role in the fog-to-cloud computing, which can help IoT devices to efficiently process and transmit data. There are many successful applications for IoT based on fog-to-cloud computing. One of the most common situations is users use sensors to collect the required data, such as related environmental data. Of course, users can also set multiple sensors to collect different areas or different types of data and transmit them to the fog node. Each fog node is a small service device provided by a service provider that can simply process and analyze data. Multiple fog nodes form a fog computing center and a management node is set up to manage all fog nodes. Finally, all data are outsourced to the cloud for cost-effective storage and other future uses. In order to ensure the integrity of the data, the sensor should create verifiable metadata before sending the collected data to the fog node; for the received data, the fog node must first verify its correctness, and then perform local analysis and processing, and for the processed data, fog nodes should create new verifiable metadata. Finally, all data and its verifiable metadata are transferred to the cloud for long-term storage. The data owner or any other authorized party should be able to verify the integrity of the data anytime, anywhere, that is, remotely audit the correctness of the data. It is the very purpose of this work to design a secure and efficient auditing scheme for data storage based on fog-to-cloud computing for IoT scenarios.

Compared with the existing scheme, the proposed scheme reduces the user’s computation overhead through the two-time signature and private key separation method, so that the proposed scheme can be better used for IoT devices with low computing capabilities. At the same time, the proposed scheme will verify the system parameters and the file tag during the signature generation and the proof generation to ensure the security of the scheme.

Our contributions

In this article, we proposed an efficient and secure data auditing scheme based on fog-to-cloud computing for IoT scenarios (DAFCI). Contributions can be summarized as follows:

1. We use the method of separating the private key, which divides the private key into two parts, and part of the private key sk' was randomly selected and held by the IoT device. The other part of the private key sk'' is generated by the private key generator (PKG) based on the user’s identity information. This part of the private key will be held by fog nodes and used to generate the final signature. This method improves the security of the system.

2. We then propose the two-time signature method, which divided the signature process into two phases: original signature and final signature. The first is that the sensor uses sk to generate and send the signature set to the fog node. The fog nodes use sk'' to generate final signature based on the original signature. This method improves the security of the system.

3. We formally prove the security of the proposed scheme, and evaluate its performance by theoretical analysis and experimental comparisons with other schemes. The results show that the proposed scheme can effectively achieve secure auditing in fog-to-cloud computing, and outperforms the others in computation complexity and energy consumption.

Organization

The rest of this article is organized as follows. In section “Related work,” we review the related work on cloud data auditing, especially the public auditing schemes. In section “Symbols and preliminary knowledge,” we present notations and preliminaries. In section “System model and security model,” the system model and security model are presented. In section “The proposed scheme,” we introduced the proposed scheme. In section “security analysis” and section “performance evaluation,” the security analysis and the performance evaluation are given, respectively. Finally, in section “Conclusion,” we conclude our article.

Related work

With the increasing popularity and importance of IoT, the security and privacy issues of IoT have attracted
Many fruitful and relevant studies have been carried out, such as how to implement secure communication between IoT nodes, how to preprocess service privacy while providing users with secure data services in IoT applications, and how to ensure the integrity of the message moves the entire IoT infrastructure.

In most applications, the sensor device collects the data and generates verifiable metadata, and then uploads the metadata to the fog node. After receiving the metadata, the fog node will first verify the metadata. If the result is true, the metadata will be further processed, then upload it and the data to the cloud for long-term storage. Data owners can verify at anytime to ensure the correctness and integrity of the data stored in the cloud, that is, the data owner can perform remote integrity auditing of the data. The auditing model is divided into private auditing and public auditing. The verification operation of the private auditing is only carried out between the CSP and the user, without the intervention of a third party, and the public auditing in which the verification operation is customarily done by an authorized third-party auditor (TPA).

Public auditing is generally considered to be a more effective approach because it provides more convincing results while significantly reducing the user’s computation and communication overhead. At present, data integrity auditing based on IoT devices has received more and more attention. Aazam et al. believed that IoT devices use the cloud to process and store data. There are still some problems that must be solved, because the cloud is not trusted, so users cannot ensure that the cloud correctly stores the data uploaded by IoT. For data stored on the cloud server, the provable data possession (PDP) scheme can effectively verify the integrity of the cloud data and achieve blockless verification, that is, without downloading the original data. Integrity verification ensures storage security, but the computational and communication overhead of this scheme is too large to be applied to IoT devices. Xu et al. proposed a distributed fully homomorphic encryption-based Merkle Tree (FHMT) scheme, which effectively solves the cloud credibility problem using blockchain technology, but their scheme does not implement public auditing. Yu et al. proposed an identity-based private key generation auditing scheme, which reduces the overhead of certificate management, but the overhead of signature generation and auditing proof verification is still large to the IoT device. Zhu et al. proposed a short signature-based data auditing scheme, which reduced the computational overhead of signature generation. However, their proposed scheme is not based on fog-to-cloud computing. The communication and computation overhead of IoT devices is still very high. Tian et al. reduced the overhead of signature generation by introducing fog nodes, but in their scheme, fog nodes are considered to be credible, but in reality, the fog node used by most users is not a trustworthy entity, the same as the cloud. Huang et al. proposed the Efficient Versatile Auditing Scheme for IoT-Based Datamarket in Jointcloud to solve the single-point-of-failure (SPoF) of cloud server using the blockchain to record the data flow, but it still does not solve the problem that the signature generation overhead of the cloud auditing scheme is too high, so it cannot be directly applied to low computing IoT devices.

Li and Yu proposed a cloud auditing scheme based on threshold secret sharing, but compared to this, the two-time signature method has two advantages as follows:

1. One of the characteristics of threshold secret sharing scheme is that when the number of people who want to unlock the secret reaches a predetermined value, it can be decrypted even without other parts of private key. But this method is difficult to apply directly to the IoT scenarios, because if the private key is assigned to sensors, when the parts of private key held by some sensors are leaked, the adversary can forge a signature based on some of the private keys to access sensitive data.

2. The above-mentioned schemes and other existing schemes cannot well meet the requirements of low computing overhead and high security of IoT devices, so we try to propose a cloud auditing scheme that can better meet the requirements of the IoT environment.

In the secret sharing scheme with the threshold, parts of private key are different. However, in the proposed scheme, the part of private key held by all sensors in the same system is the same, which enables even different types of sensors can generate the same signature, ensuring the consistency of the signature and facilitating verification.

In the proposed scheme, the fog-to-cloud computing is introduced, but the fog is designed as an untrusted entity. The method of private key separation and two-time signature avoids the situation that the private key is directly leaked by the fog node, and the security is destroyed due to the leakage of the private key, improving efficiency and security.

Symbols and preliminary knowledge
Notions
The symbols and corresponding descriptions appearing in the DAFCI scheme can be seen in Table 1.
1. **Bilinear map**: let $G_1$, $G_2$ be two multiplicative cyclic groups of same large prime order $p$, and $g$ be a generator of $G_1$. Bilinear map is a map $e : G_1 \times G_1 \rightarrow G_T$ with the following properties:

2. (a) **Bilinearity**: for all $g_1, g_2 \in G_1$ and $a, b \in \mathbb{Z}_p^*$,

   \[ e(g_1^a, g_2^b) = e(g_1, g_2)^{ab} \]

   (b) **Computability**: there exists an efficiently computable algorithm for computing map $e$.

   (c) **Non-degeneracy**: $e(g, g) \neq 1$.

3. **Computational Diffie–Hellman (CDH) problem**: for unknown $x, y \in \mathbb{Z}_p^*$, given $g, g^x$ as input, outputs $g^{xy} \in G_1$. The CDH assumption in $G_1$ holds if it is computationally infeasible to solve the CDH problem in $G_1$.

4. **Discrete logarithm (DL) problem**: for unknown $x \in \mathbb{Z}_p^*$, given $g, g^x$ as input, outputs $x$. The DL assumption in $G_1$ holds if it is computationally infeasible to solve the DL problem in $G_1$.

**System model and security model**

**System model**

The system model of the DAFCI scheme is shown in Figure 1. The model mainly includes five different entities: CSP, sensor, PKG, fog computing center (FCC), and TPA, as follows:
**CSP**: CSP provides enormous data storage and data processing service to the users.

**Sensor**: the sensor device is responsible for collecting data, then hiding the sensitive information and generating the original signature, and uploading it to FCC. A system can contain multiple sensor devices. Multiple sensors belonging to the same system can be of different types, collecting different data separately, but they have the same private key.

**FCC**: the FCC consists of several fog nodes, each of which is responsible for generating a final signature for the sensors in the area to reduce the computational overhead of the sensor.

**PKG**: PKG is responsible for generating part of the private key $sk''$ for sensors and system public parameters. The sensor will check $sk''$ generated by PKG.

**TPA**: TPA is a third-party auditor. It helps sensors audit the integrity of the data stored in the cloud.

After the sensor collects the data, the sensitive information is blinded. Since the proportion of this part of the data is small, the process overhead is very low. The original signature is then generated based on a portion of the private key and sent to the FCC. After receiving the original signature, the FCC uses its holds partial private key to generate the final signature and upload it to the CSP.

When the sensor wants to verify the cloud data, it sends an audit request to the TPA. After receiving the request, the TPA selects challenge blocks according to the block number contained in the instruction and initiates an audit challenge to the CSP. The CSP then sends the data auditing proof back to the TPA. Finally, the TPA will verify the proof and inform the result to the sensor.

**Design goals**

In order to effectively ensure the secure storage of sensor data in the cloud, the DAFCI is designed to achieve the following goals.

**The detectability**

- The scheme should have a high enough probability to detect damage of the cloud data. When the proportion of damaged blocks is $p$, the probability of finding a damaged block during the audit is at $\delta$ least. This probability is called the probability of detection. At the same time, this scheme is called $(p, \delta)$ detectable.

**The correctness**. The correctness includes the correctness of the private key and the correctness of the auditing:

- The correctness of the private key: verify the partial private key $sk''$ generated by PKG to ensure its correctness.
- The correctness of auditing: TPA verifies the auditing proof generated by CSP to ensure the integrity of sensor data.

**The soundness**

- The soundness requires that any false proof pass verification probability is negligible. In other words, only the CSP that correctly stores the data can generate the correct auditing proof.

**Definition**

**Definition 1.** The DAFCI scheme consists of six algorithms: Setup, Extract, SigGenU, SigGenF, ProofGen, and ProofVerify. These algorithms are described as follows:

- **Setup**: this algorithm is a setup algorithm executed by PKG. It takes as input a security parameter $k$ and outputs the system public parameters $pp$.
- **Extract**: the algorithm consists of two parts, one part is the extraction algorithm run by PKG. It takes as input the system public parameters $pp$ and the user’s identity $ID$. It outputs the sensor’s partial private key $sk''$. The sensor will verify this part of the private key and accept it if $sk''$ can pass the verification. Another part of this algorithm is performed by the sensor. Sensor selects $x$ as another part of the private key $sk'$.  
- **SigGenS**: this algorithm is run by the sensor. It takes as input the signature private key $ssk$, the original file $F$, the partial private key $sk'$ of sensor, and the file identifier name. It outputs blind file $F^*$, corresponding original signature set $\Phi$ and file tag $\tau$.
- **SigGenF**: the algorithm is run by the FCC. It takes as input the sensor part private key $sk''$ generated by PKG, the file name name, and the first signature set $\Phi$, and it outputs the final signature set $\sigma$.
- **ProofVerify**: this algorithm is a verification algorithm executed by the TPA. It takes as input the auditing challenge $chal$, the system public parameters $pp$, and the auditing proof $P$. TPA will verify and inform the sensor auditing results based on the above information.

**Security model**

In the DAFCI scheme, the PKG may be untrusted because the PKG only generates a partial private key
The model of the DAFCI scheme is implemented through nature, and it is not completely trusted. The security used to help the sensor generate the corresponding signature, and it is not completely trusted. The security model of the DAFCI scheme is implemented through the following adversaries and challengers. The game is introduced as follows:

- **Setup phase:** the challenger C runs the Setup algorithm to get the private key sk and public parameters pp, and then sends the public parameters pp to the adversary A.
- **Query phase:** at this stage, adversary A will make the following two queries to the challenger C.
- **Extract queries:** the adversary A sends the identity information ID to challenger C. The challenger executes the Extract algorithm to get the private key skID, and then sends it to the adversary A.
- **SigGen queries:** the adversary asks for the corresponding signature of file F. The challenger executes the SigGen algorithm to calculate the signature corresponding to file F and sends it to adversary A.
- **Challenge phase:** at this stage, the adversary will act as a prover and the challenger will act as a verifier. The challenger C sends the challenge block \( \text{chal} = \{ i, v_i \}_{i \in I} \) to the adversary A, where \( I = \{ \omega_1, \omega_2, \ldots, \omega_c \}, (\omega_j \in [1, n], j \in [1, c]) \). The challenger asks the adversary to provide the data holding proof P of the corresponding data block \( \{ m_{a_1}, m_{a_2}, \ldots, m_{a_c} \} \).
- **ProofGen phase:** after receiving the challenge of challenger C, the adversary A generates a corresponding proof P of the challenge block and sends it to the challenger C. If it is proved that P can pass the verification with a non-negligible probability, it can be considered that the adversary A has won the game.

For the above security model, it is necessary to prove that if the adversary cannot have all the challenge blocks required by the challenger, it cannot generate the correct proof P, which means that the proof of the adversary’s forgery cannot be verified. The purpose of the adversary A is to correctly generate the corresponding signature of the challenge chal without holding the data blocks.

**Definition 2.** We consider a remote data integrity auditing scheme is secure if the following condition holds:

whenever the adversary A in the aforementioned game is able to generate a valid proof P to pass the challenge of the challenger C with a non-negligible probability, there is a knowledge extractor that can capture the challenged data block, but the probability can be neglected.

**Definition 3.** If only the data owner can view the sensitive information in the file, the CSP and other shared sensors can only view the non-sensitive information, then the sensitive information in the file is safe.

**The proposed scheme**

**An overview**

The sensor has lower computing power and shorter battery life. In order to minimize the energy consumption of the sensor device, the DAFCI scheme uses the blockless verification technology to enable the sensor to complete the integrity verification without downloading the blocks stored in the cloud. The private key separation method and the two-time signature method are used to further reduce the computation overhead of the sensor. The private key separation method refers to dividing the private key sk into two parts to calculate and generate, that is, \( sk = (sk’, sk’’) \). The two-time signature method means that the initial signature \( \Phi = (\sigma_i)_{i \in [1, n]} \) is generated by the sensor and uploaded to the FCC, and then the FCC generates the final signature \( \Phi’ = \{ \sigma_i’ \}_{i \in [1, n]} \).

In the DAFCI scheme, PKG uses the identity information received from the sensor to generate the partial private key \( sk’’ \). The sensor can verify the correctness of this part of the private key. The other part of the private key \( sk’ \) is selected by the sensor, and no additional verification is required. Before the sensor uploads data to the cloud, the file needs to be blinded. Other shared sensors can only access non-sensitive information, not any content that involves sensitive information. The sensor can recover the blinded file when needed. The signature will be used to verify the integrity of the data, which is split into two generations. After the file is blinded, the sensor device generates the original signature. After the original signature is generated, the FCC performs a two-time signature to generate a final signature set. Finally, the fog node sends the signature set and the blind file to the CSP and sends the file tag to the TPA. When the sensor wants to check the integrity of the data stored in the cloud, it sends an audit request to the TPA. After receiving the request, the TPA challenges the CSP and asks the CSP to return the corresponding auditing proof. The TPA verifies the auditing proof and determines whether the CSP stores the sensor’s data correctly, and sends the auditing result to the sensor. The specific details are described in the following section.
In addition, the proposed scheme can be further applied with edge-cloud computing. A large amount of data are collected by the sensors set in the system and analyzed and processed by the edge end close to the user, while being scalable. Users get real-time insights and experiences with responsive and context-aware applications.

**Description of the proposed scheme**

In the DAFCI scheme, the original file $F$ is divided into $n$ blocks $F = \{m_1, m_2, \ldots, m_n\}_{i \in \mathbb{N}}$, and $m_i \in \mathbb{Z}_p^*$ denotes the $i$–th block of file $F$. Assuming that the length of the sensor identity information $ID$ is $l$ bits, it can be described as $\{ID_1, ID_2, \ldots, ID_l\} \in \{0, 1\}^l$. Usually, sensitive information only accounts for a small part of whole files, and only needs to be blinded, so the blinding overhead is small. After the blind file $F^* = \{m'_1, m'_2, \ldots, m'_n\}_{i \in \mathbb{N}}$ is formed, the sensor uses the private key $sk' = x$ to generate the original signature $\sigma_i = H(m'_i \cdot x)$, the private key $sk'$ is selected and saved by the sensor device controlled by the sensor, not known by other entities, and then the original signature set $\Phi = (\sigma_i)_{i \in \mathbb{N}}$ is sent to the FCC. The FCC generates a final signature $\sigma'_i = \sigma'_i \cdot \mu^\prod_{j=1}^{i-1} \mu^{ID_j}$ according to a part of the private key generated by PKG, $sk'' = \prod_{j=1}^{l-1} \mu^{ID_j}$, and generates the final signature set $\Phi' = (\sigma'_i)_{i \in \{1, n\}}$. The two-time signature method is shown in Figure 2.

Since the DAFCI scheme divides the private key into two parts, which are generated by PKG and sensor, respectively, the FCC only has the part calculated by PKG, so even if the FCC leaks part of the private key it owns, it is still difficult to forge the correct signature.

The following is a detailed description of the DAFCI program:

1. **Algorithm Setup($^1k^1$)**
   - The PKG chooses two multiplicative cyclic groups $G_1$ and $G_2$ of a same prime order $p$, a generator $g$ of $G_1$, a bilinear pairing $e: G_1 \times G_1 \rightarrow G_2$, and a pseudorandom function $f: \mathbb{Z}_p^* \times \mathbb{Z}_p^* \rightarrow \mathbb{Z}_p^*$.
   - The PKG randomly chooses an element $x \in \mathbb{Z}_p^*$, then chooses elements $\mu', \mu \in G_1$, and a cryptographic hash function $H: \{0, 1^* \rightarrow G_1\}$.
   - The PKG publishes the system parameters $pp = (G_1, G_2, p, e, g, \mu', \mu, H, f)$.

2. **Algorithm Extract(pp, ID)**
   - The sensor sends the identity information $ID = \{ID_1, ID_2, \ldots, ID_l\} \in \{0, 1\}^l$ to PKG. After receiving the identity information $ID$, the PKG calculates partial private key $sk'' = \prod_{j=1}^{l-1} \mu^{ID_j}$ and partial public key $pk'' = pq^{sk''}$, and then sends $sk''$ and $pk''$ to the sensor.
   - After the sensor receives $sk''$ and $pk''$, it will verify the correctness of the received private key by checking the following equation holds or not
     \[
     e\left(\mu', \prod_{j=1}^{l-1} \mu^{ID_j}\right) = e(\mu', pk'')
     \] (1)
   - If the above equation does not hold, the sensor refuses to accept it; if the equation (1) holds, the sensor will calculate another partial private key $sk' = x \cdot p$, that is, the private key $sk = (sk', sk'')$ and the public key $pk = (pk', pk'')$. The above process is illustrated in Figure 3.

3. **Algorithm SigGenU(F, name, sk')**
   - The sensor will first blind the file, and the sensitive information blind method here draws on the scheme of Shen et al. The sensor randomly

---

![Figure 2. The process of two-time signature.](image-url)

---
selects a seed $k_1 \in Z_n^*$ as the input private key of the random number generation function $f$. The sensor uses the seed $k_1$ to generate a blinder $\alpha_i = f_k(i, name)(i \in K_1)$ and then uses $\alpha_i$ to encrypt sensitive information in the file. $name \in Z_n$ is a randomly selected random number that identifies the file.

- In order to protect sensitive information, the sensor should blind data blocks corresponding to the sensitive information of the original file $F$ before sending it to the FCC. The index of these data blocks is in set $K_1$. The sensor computes the blinded data block $m_i' = m_i + \alpha_i$ for each data block $m_i \in Z_n(i \in K_1)$ of the original file $F$. The blinded file is $F' = \{m_1', m_2', ..., m_n'\}$, where $i \in [1, n]$ and $i \not\in K_1$, otherwise $m_i \in Z_n(i \in K_1)$.

- After generating the blind file $F' = \{m_1', m_2', ..., m_n'\}$, the sensor then uses a hash function $H$ to compute its hash value $H(m_i')$, then uses the hash value and part of the private key $x$ generates an original signature $\sigma_i$. The calculation process is as follows: $\sigma_i = H(m_i') \cdot x$. The original signature set is $\Phi = \{\sigma_i\}_{i=1, n}$.

- The sensor computes $\tau_0 = name \parallel \mu'$ and then generates the corresponding file tag $SSig_kssk(\tau_0)$, where $SSig$ is the signature of the file tag $\tau_0$ using the signing private key $sk$. Finally, the sensor sends $\{\Phi, \tau_0, F', sk', pk''\}$ to FCC for further computing.

4. Algorithm $SigGenF(sk', \Phi)$

- The FCC first checks whether the signature $SSig_kssk(\tau_0)$ is valid. If it is a valid signature, the FCC parses the file tag $\tau_0$ to obtain the file name and verification value $\mu'$, and then does the next step.

The FCC further verifies the correctness of the original signature $\sigma_i(i \in [1, n])$ by verifying whether the following equation holds

$$e(\sigma_i, P) = e\left(\prod_{i=1}^{c} H(m_i'), pk'\right)$$

(2)

- If the above equation holds, then FCC considers that the original signature $\sigma_i$ is correct and can proceed the following steps.

- The FCC is responsible for computing the final signature $\sigma_i' = \sigma_i \cdot \mu' \prod_{i=1}^{c} m_i^p$ for the sensor, and the final signature set is $\Phi' = \{\sigma_i'\}_{i=1, n}$, then sends $\{F', \Phi'\}$ to the CSP, and sends the file tag $\tau$ to the TPA.

5. Algorithm $ProofGen(F', \Phi', \text{chal})$

- The sensor or the TPA is authorized to challenge the CSP from time to time. The TPA checks whether the file tag $\tau$ is valid. If it is invalid, the TPA will not perform the auditing request; if it is valid, the TPA obtains the file identity $name$ and the verification parameter $\mu$ by parsing the file tag $\tau_0$, and then generates the auditing challenge $chal$. The challenge generation process is as follows:

  (a) TPA randomly selects $c$ elements from the data block index set to form set $I$. Set $I \in [1, n]$.

  (b) For each element in the set $I$ choose a random number $v_i \in Z_n^*$.

  (c) Send the auditing challenge $chal = \{i, v_i\}_{i \in I}$ to the CSP.

  (d) After receiving the auditing challenge from the TPA, the CSP generates a proof of data possession for this
proof as follows: computes the aggregated signature \( \sigma = \prod_{i=1}^{c} \sigma_i \) and sends it to the TPA as the challenge proof \( P \).

6. Algorithm ProofVerify(chal, pp, P)

- The TPA will verify the correctness of the auditing proof as follows

\[
e(\sigma, g) = \left( e\left( \prod_{i=1}^{c} H(m_i^t), pk' \right) \cdot e(\mu', pk'^n) \right)^{\sum_{i=1}^{v_i}} \tag{3}
\]

- If equation (3) holds, the data stored in the CSP are intact, otherwise, it is not.

Security analysis

In this section, we prove the DAFCI is secure in terms of correctness, soundness, and detectability.

Theorem 1 (the correctness)

1. Private key correctness: when PKG sends a correct partial private key to the sensor, this partial private key can pass the verification of the sensor.
2. Original signature correctness: when the FCC receives the original signature, it will verify it to ensure that the original signature is correct.
3. Auditing correctness: if the CSP correctly stores the data uploaded by the sensor, then it can pass the TPA verification.

Proof

1. If the partial private key given by PKG is correct, then it can be verified by the Extract algorithm equation (1). Based on the properties of the bilinear pairing equation, the following is a proof that starts from the left equation and finally introduces the right equation

\[
e\left( \prod_{i=1}^{t} \mu_i, P \right) = e\left( \mu', P \prod_{i=1}^{t} \mu_i \right)
\]

2. If the original signature \( \sigma_i \) given by the sensor is correct, it can pass the verification by the FCC. Based on the properties of the bilinear pairing equation, equation (2) can be proved correct by deducing the left-hand side from the right-hand side

\[
e(\sigma_i, P) = e\left( \prod_{i=1}^{c} H(m_i^t, pk' \right) \]

3. If the original signature \( \sigma_i \) given by the sensor is correct, it can pass the verification by the FCC. Based on the properties of bilinear pairing, equation (2) can be proved correct by deducing the left-hand side from the right-hand side

\[
e(\sigma, P) = e\left( \prod_{i=1}^{c} H(m_i^t, \mu_{i-1} \mu_i \right) , P \)
\]

\[
e\left( \prod_{i=1}^{c} H(m_i^t), x \cdot P \right) \cdot e\left( \mu', P \prod_{i=1}^{c} \mu_i \right)^{\sum_{i=1}^{v_i}}
\]

\[
e\left( \prod_{i=1}^{c} H(m_i^t), x \cdot p \right)^{\sum_{i=1}^{v_i}} \cdot e\left( \mu', P \prod_{i=1}^{c} \mu_i \right)^{\sum_{i=1}^{v_i}}
\]

\[
e\left( \prod_{i=1}^{c} H(m_i^t), x \cdot p \right)^{\sum_{i=1}^{v_i}} \cdot e(\mu', pk'^n) \sum_{i=1}^{v_i}
\]

Theorem 2 (the soundness)

1. Privacy of private key: in our scheme, the FCC is considered to be an incompletely trusted entity, and it may leak the partial private key. The solution needs to consider that in the case that some of the private keys held by the FCC are leaked, it can still be guaranteed that the private keys held only by the sensor cannot be known through other information calculations.
2. Anti-replace attack: during the TPA verification process, replace attacks would never be effective. In other words, if the CSP does not store data blocks or signature correctly, it is impossible for the CSP to succeed in replacing such a data block with another block to respond to the challenge to maintain this reputation.
3. Anti-reply attack: during the TPA verification process, reply attack would never be effective. In other words, a malicious CSP could not pass the verification using the signature generated by previous auditing.
Proof

1. The private key consists of two parts, namely, $sk''$ generated by PKG based on the identity information provided by the sensor and $sk'$ selected by the sensor. $sk''$ was sent to the FCC for the second signature, but since the first signature was performed on the sensor device, even if the FCC leaked part of the private key it holds, it was still impossible to obtain $sk$.

2. The TPA selects some random blocks for auditing, and sends some challenge blocks $chal = \{i, v_i\}_{i\in[1, 4]}$ to the CSP. Assume that the TPA is currently under replay attack, that is, the CSP tries to use the previous block signature information and replaces part of the correct signature to pass verification, and then TPA verifies the received proof $\sigma^-$

$$e(\sigma^-, P) = e(\sigma, P)$$

(4)

If equation (4) holds, the replacement attack is successful. The received proof is $\sigma^- = \prod_{i\in[1, 4]} \sigma_i^{v_i} \cdot \sigma_j^{v_j}$. From the left-hand side of equation (4)

$$e(\sigma^-, P) = e\left(\prod_{i\in[1, 4]} \prod_{j=1}^{n_i} \sigma_i^{v_i}, P\right) \cdot e\left(\sigma_j^{v_j}, P\right)$$

In order for equation (4) to be true, there is

$$e\left(\sigma_j^{v_j}, P\right) = e\left(\sigma_i^{v_i}, P\right)$$

(5)

If equation (5) holds, there are $\sigma_j^{v_j} = \sigma_i^{v_i}$ and $\sigma_j = \sigma_j^{v_j} \cdot \mu \prod_{i=1}^{n_i} \mu^{v_i} = H(m_j^{v_j}) \cdot \mu \prod_{i=1}^{n_i} \mu^{v_i}$.

Due to the privacy of the private key, even if $sk''$ is leaked, the CSP cannot forge $sk'$, nor can it forge the correct signature. In other words, replay attacks in DAFCI are impossible to succeed.

3. The replay attack game is similar to the replacement attacks mentioned above. The sensor also authorizes TPA to auditing. After receiving the auditing request, the TPA randomly selects several data blocks in the indicated file and generates the challenge $chal = \{i, v_i\}_{i\in[1, 4]}$. In this scenario, the CSP attempts to use a forged signature to pass verification. The proof of anti-replay attacks is similar to the proof of anti-replacement attacks, which is not repeated here.

Theorem 3 (the detectability). Assume that the original file $F$ is divided into $n$ blocks and encrypted to generate a blind file $F^*$ then stored in the cloud. Among them, $k$ blocks are modified or deleted by the CSP and the number of challenge blocks is $c$. At this time, the detectability of the DAFCI scheme is

$$\left(\frac{k}{n}, 1 - \left(\frac{n-k}{n}\right)^c\right)$$

$X$ is the intersection of the damaged block and the challenge block. Then, there are

$$P_x = P\{X \geq 1\} = 1 - P\{X = 0\} = 1 - \frac{n-k}{n} \times \frac{n-1-k}{n-1} \times \cdots \frac{n-c+1-k}{n-c+1}$$

From the above equation, we can know that

$$1 - \left(\frac{n-k}{n}\right)^c \leq P_x \leq 1 - \left(\frac{n-c+1-k}{n-c+1}\right)^c$$

because $n-i-1-k/n-i-1 \leq n-i-k/n-i$, so the probability that the DAFCI scheme detects a damaged block is at least $1 - (n-k/n)^c$. Suppose the proportion of damaged blocks is 1%, in other words when $n-k/n = 0.99$, assuming that the challenge block is 300, the detection probability can reach 95%, and assuming that the challenge block is 400, the detection probability can reach 99%.

Theorem 4 (sensitive information hiding). Sensitive information cannot be accessed by any entity except the data owner, including FCC, TPA, CSP, and shared sensors.

Proof. This part borrows from the encryption scheme of Shen et al., but considering the performance of the sensor device, it is lightweight. Before the sensor has hashed the file block, the file has been blinded first, so under the random oracle model, only the sensor can access sensitive information.

Performance evaluation

In this section, the functional comparison of the DAFCI scheme and some other related schemes will be performed first, and then the computation overhead will be compared. Then, the communication overhead and computational complexity of the DAFCI scheme will be discussed. Finally, the specific performance of the DAFCI scheme is demonstrated through experiments.

Functionality comparison

As shown in Table 2, the functionality of the DAFCI scheme and other related schemes is...
Table 2. Functionality comparison with existing related schemes.

| Schemes          | Public | Certificate management | Data | Sensitive information | Lightweight device |
|------------------|--------|------------------------|------|-----------------------|--------------------|
|                  | Verifiability | Simplification | Sharing | Hiding | Support |
| Shacham et al.30 | Yes    | No                     | No    | No                    | No                 |
| Tian et al.25    | Yes    | Yes                    | No    | No                    | Yes                |
| Li et al.31      | Yes    | Yes                    | No    | No                    | No                 |
| Wang29           | Yes    | No                     | Yes   | Yes                   | No                 |
| Shen et al.29    | Yes    | Yes                    | Yes   | Yes                   | No                 |
| Ours             | Yes    | Yes                    | Yes   | Yes                   | Yes                |

Table 3. The computation overhead of our scheme and related schemes.

| Schemes          | TagGenS       | TagGenF       | ProofGen      | Verify            |
|------------------|---------------|---------------|---------------|-------------------|
| Tian’s scheme A  | $n \cdot (H + \text{Mul} + 3c)$ | $(2c + H) + m$ | $c \cdot \text{Mul} + c \cdot e$ | $(c + 1) \cdot H$ |
| Tian’s scheme B  | $n \cdot (H + \text{Mul} + 2c)$ | $m \cdot (H + \text{Mul} + 2c)$ | $c \cdot \text{Mul} + c \cdot e$ | $2c + P + (c + 1) \cdot H$ |
| Shen’s scheme    | $n \cdot (H + 2\text{Mul} + 2c)$ | $4c + (2 \cdot c + 1)$ | $c \cdot \text{Mul} + c \cdot e$ | $(c + 1) \cdot \text{Mul} + 2c + 3P$ |
| Purposed scheme  | $n \cdot (H + \text{Mul})$ | $2n \cdot (\text{Mul} + c)$ | $c \cdot \text{Mul} + c \cdot e$ | $(c + 1) \cdot \text{Mul} + 2c + 3P$ |

Compared. As can be seen from the table, the DAFCI solution is the only one that meets all the following properties: public verification, certificateless management, data sharing, sensitive information hiding, and suitable for lightweight device supporting.

Performance analysis and comparison

First define $H$ for a hash operation, $\text{Mul}$ for a multiplication operation, $\text{Add}$ for an addition operation, $P$ for a bilinear pairing operation, and $e$ for an exponential operation. $n$ is the number of file blocks, $c$ is the number of challenge blocks, and $l$ is the length of the identity information. $d$ is the number of file blocks containing sensitive information. $|n|$ represents the size of the set $[1, n]$, $|p|$ represents the number of elements in $Z_p^*$, and $|q|$ represents the number of elements in $G_1$.

1. Computation overhead comparison: computation overhead refers to the cost of corresponding computing tasks performed by the corresponding entity, including signature generation, proof generation, and proof verification. Table 3 compares the computation overhead of the DAFCI scheme with the scheme of Shen et al.29 and the scheme of Tian et al.25 In section “System model and security model,” we introduced how to encrypt sensitive information. Its computational overhead is $d \cdot \text{Add}$. As mentioned earlier, due to $d \ll n$, this part of the overhead is small, so it is not reflected in the table. The computation overhead of the sensor to generate the original signature is $n \cdot (H + \text{Mul})$. The original signature and related parameters are then sent to the FCC, and the overhead of the FCC to generate the final signature is $n \cdot (2\text{Mul} + 2c)$. The final signature is then sent to the CSP. During the challenge process, the overhead of generating the auditing proof is $c \cdot \text{Mul} + c \cdot e$, and the overhead of TPA verification after receiving the proof is $(c + 1) \cdot \text{Mul} + 2c + 3P$.

As shown in Table 3, compared with the related schemes, the computation overhead of the DAFCI scheme is lower. On the sensor side, that is, the overhead reduction is more obvious. Compared with the scheme that also introduces FCC, the computation overhead of the fog node is slightly higher, but it is not obvious. The generated overhead is consistent with scheme B of Tian et al.25 but smaller than scheme A of Tian et al. And scheme of Shen et al.’s29 TPA verification overhead is less than related schemes.

2. Communication overhead comparison: according to the description of section “The proposed scheme,” we can know that the communication overhead of the proposed scheme is mainly from the integrity auditing phase, as shown in Table 4. Communication overhead refers to the
overhead incurred when transferring files or related parameters between different entities. Reducing communication overhead can increase solution stability and practicability. Therefore, the increase in communication overhead caused by the introduction of the FCC is negligible compared with the auditing process. Therefore, only the overhead of the auditing process needs to be considered. In this part, the TPA sends a challenge block \( chal = \{i, vi\}_{i \in F} \) to the cloud. The size of the challenge block is \( c \cdot (|n| + |p|) \) bits. The CSP generates a certificate \( P = \{\sigma\} \) with size \( |q| \) bits. Therefore, in an audit task, the communication overhead is \( c \cdot (|n| + |p|) + |q| \).

3. **Computation complexity**: we analyze the computation complexity of the different entities in different phases in Table 5. The number of challenge blocks \( c \), the total number of file blocks \( n \), and the number of blocks containing sensitive information \( d \) determine the corresponding time complexity of each entity of the DAFCI solution. From Table 5, we can see that the time complexity of the file blinding process is \( O(d) \), the time complexity of the initial label generation is \( O(n) \), and the time complexity of the final label generation is \( O(d) \). The challenge generation and the time complexity of proof generation and proof verification is \( O(c) \).

**Performance evaluation**

In this section, we evaluate the performance of the proposed scheme by several experiments. The experiment was implemented using the Ubuntu 16.04 operating system with an Intel Core i5 3.0 GHz processor and an 8 GB memory. The programmer is written in C program, and it uses the library functions in the pairing-based cryptography (PBC) library to simulate the cryptographic operations, where the benchmark threshold is 512 bits, the size of the element is \( |p| = 160 \text{ bits in } Z_p^* \), the size of the file is 20 MB, and the length of user identity to be 160 bits.

The experimental results are the averages of the 10 experiments.

1. **Performance of different process**: to effectively evaluate the performance in different processes,
we set the number of data blocks to be 100 and the number of blinded data blocks to be 5 in our experiment. As shown in Figure 4, private key generation spends nearly 0.26 s. The time consumed by the original signature generation is 0.312 s. The time of the final signature generation is 2.4 s. The time of signature verification and that of sensitive information blindness, respectively, are 0.22 and 0.031 s. So we can conclude that in these processes, the final signature generation spends the longest time.

2. **Performance of signature generation**: to evaluate the performance of signature generation and signature verification, we generate the signatures for different numbers of blocks from 0 to 1000 increased by an interval of 10 in our experiment. As shown in Figure 5, the time cost of the original signature generation, the final signature generation, and the signature generation with DAFCI all linearly increases with the number of the data blocks. The time of original signature generation ranges from 0.312 to 3.12 s. The time of final signature generation ranges from 2.4 to 24.0 s. The time of signature generation without DAFCI ranges from 2.712 to 27.1 s.

3. **Performance of auditing**: Figures 6 and 7 show the TPA and CSP auditing overheads for different numbers of challenge blocks, respectively. Considering that the detection probability has reached 99% when the number of challenge blocks is 400, only the case where the challenge blocks are from 50 to 500 is shown in the figure. As shown in Figures 6 and 7, it can be seen that the computation overhead of the challenge generation and proof verification increases linearly with the number of challenge data blocks. The computation overhead of the challenge generation increases from 0.097 to 0.486 s as the challenge block increases. The overhead of the proof verification increased from 0.5 to 11 s. It can be concluded that compared with the challenge verification overhead, the challenge overhead is lower and grows slowly. In summary, it can be concluded that as the challenge block increases, the computation overhead of both TPA and CSP will increase.

**Conclusion**
This article proposed an efficient and secure public cloud data auditing scheme based on IoT scenarios. Through the encryption of sensitive information, the separation of private key method, and the two-time signature method, data sharing under privacy protection
is realized, which reduce the computation and communication overhead of IoT devices, while ensured security. The security analysis shows that the DAFCI scheme is safe under the random oracle model. Performance analysis shows that compared with traditional cloud data auditing schemes and other schemes using fog-to-cloud computing, the DAFCI scheme is more efficient and has certain advantages, which can be better applied to low power IoT devices. However, considering the rapid development and widespread application of IoT technology, further reducing the overhead of computation and communication under the condition of ensuring the security of the solution will still be the focus of work in the future for a long time.

Declaration of conflicting interests
The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding
The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported, in part, by the National Natural Science Foundation of China (grant no. 61802106) and the Natural Science Foundation of Hebei Province (grant no. F2016201244).

ORCID iD
Hao-Ning Wang https://orcid.org/0000-0002-4105-8394

References
1. Bonomi F, Milito R, Zhu J, et al. Fog computing and its role in the Internet of things. In: Proceedings of the 1st Edition of the MCC workshop on mobile cloud computing, Helsinki, 13–17 August 2012, pp.13–16. New York: ACM.
2. Adat V and Gupta BB. Security in Internet of Things: issues, challenges, taxonomy, and architecture. Telecommun Syst 2018; 67(3): 423–441.
3. Cai H, Xu B, Jiang L, et al. IoT-based big data storage systems in cloud computing: perspectives and challenges. IEEE Internet Things J 2017; 4(1): 75–87.
4. Roman R, Lopez J and Mammo M. Mobile edge computing, fog et al.: a survey and analysis of security threats and challenges. Future Gener Comput Syst 2018; 78: 680–698.
5. Wang C, Ren K, Lou W, et al. Toward publicly audit-able secure cloud data storage services. IEEE Netw 2010; 24: 19–24.
6. He D, Kumar N, Zeadally S, et al. Certificateless provable data possession scheme for cloud-based smart grid data management systems. IEEE Trans Ind Informat 2018; 14(3): 1232–1241.
7. Wang HQ. Identity-based distributed provable data possession in multycloud storage. IEEE Trans Servic Comput 2015; 8(2): 328–340.
8. Shen W, Yu J, Xia H, et al. Light-weight and privacy-preserving secure cloud auditing scheme for group users via the third party medium. J Netw Comput Appl 2017; 82: 56–64.
9. Li J, Yao W, Han J, et al. User collusion avoidance CP-ABE with efficient attribute revocation for cloud storage. IEEE Syst J 2018; 12(2): 1767–1777.
10. Wang B, Li B and Li H. Panda: public auditing for shared data with efficient user revocation in the cloud. IEEE Trans Serv Comput 2015; 8(1): 92–106.
11. Luo Y, Xu M, Fu S, et al. Efficient integrity auditing for shared data in the cloud with secure user revocation. In: IEEE Trustcom/BigDataSE/ISPAA, Helsinki, 20–22 August 2015, pp.434–442. New York: IEEE.
12. Yu Y, Au MH, Ateniese G, et al. Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage. IEEE Trans Inf Forensics Security 2017; 12(4): 767–778.
13. Shen W, Yang G, Yu J, et al. Remote data possession checking with privacy-preserving authenticators for cloud storage. Future Gener Comput Syst 2017; 76: 136–145.
14. Singh J, Pasquier T, Bacon J, et al. Twenty security considerations for cloud-supported Internet of Things. IEEE Internet Things J 2016; 3: 269–284.
15. Esposito C, Ficco M, Castiglione A, et al. Distributed group key management for event notification confidentiality among sensors. IEEE Trans Dependable Secure Comput. Epub ahead of print 30 January 2018. DOI: 10.1109/TDSC.2018.2799227.
16. Belguith S, Kaaniche N, Laurent M, et al. PHOABE: securely outsourcing multi-authority attribute based encryption with policy hidden for cloud assisted IoT. Comput Network 2018; 133: 141–156.
17. Esposito C, Castiglione A, Palmieri F, et al. Integrity for an event notification within the Industrial Internet of Things by using group signatures. IEEE Trans Ind Infor 2018; 14: 3669–3678.
18. Juels A and Kaliski BS Jr. PoRs: proofs of retrievability for large files. In: Proceedings of the 14th ACM conference on Computer and communications security, Alexandria, VI, 29 October–2 November 2007, pp.584–597. New York: ACM.
19. Wang Q, Wang C, Ren K, et al. Enabling public audit-ability and data dynamics for storage security in cloud computing. IEEE Trans Parallel Distr Syst 2011; 22: 847–859.
20. Aazam M, Khan I, Alsaffar AA, et al. Cloud of things: integrating Internet of Things and cloud computing and the issues involved. In: Proceedings of the 11th international Bhurban conference on applied sciences and technology (IBCAST), Islamabad, Pakistan, 14–18 January 2014, pp.414–419. New York: IEEE.
21. Ateniese G, Burns R, Curtmola R, et al. Provable data possession at untrusted stores. In: Proceedings of the 14th...
22. Xu J, Wei L, Wu W, et al. Privacy-preserving data integrity verification by using lightweight streaming authenticated data structures for healthcare cyber-physical system. *Future Gener Comput Syst* 2018; 108: 1287–1296.
23. Yu J, Hao R, Xia H, et al. Intrusion-resilient identity-based signatures: concrete scheme in the standard model and generic construction. *Inf Sci* 2018; 442–443: 158–172.
24. Zhu H, Yuan Y, Chen Y, et al. A secure and efficient data integrity verification scheme for cloud-IoT based on short signature. *IEEE Access* 2019; 7: 90036–90044.
25. Tian H, Nan F, Chang CC, et al. Privacy-preserving public auditing for secure data storage in fog-to-cloud computing. *J Netw Comput Appl* 2019; 59–69: 127.
26. Huang K, Zhang X, Mu Y, et al. EVA: efficient versatile auditing scheme for IoT-based datamarket in jointcloud. *IEEE Internet Things J* 2020; 7(2): 882–892.
27. Wang C, Chow SS, Wang Q, et al. Privacy-preserving public auditing for secure cloud storage. *IEEE Transactions on Computers* 2013; 62: 362–375.
28. Shen J, Shen J, Chen X, et al. An efficient public auditing protocol with novel dynamic structure for cloud data. *IEEE Trans Inf Forensics Security* 2017; 12(10): 2402–2415.
29. Shen W, Qin J, Hao R, et al. Enabling identity-based integrity auditing and data sharing with sensitive information hiding. *IEEE Trans Inf Forensics Security* 2019; 14(2): 1556–6013.
30. Shacham H and Waters B. Compact proofs of retrievability. *J Cryptol* 2013; 26(3): 442–483.
31. Li Y, Yu Y, Min G, et al. Fuzzy identity-based data integrity auditing for reliable cloud storage systems. *IEEE T Depend Secure Comput* 2019; 16: 72–83.
32. Wang H. Proxy provable data possession in public clouds. *IEEE Trans Serv Comput* 2013; 6(4): 551–559.