Computers & Security 92 (2020) 101741

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

A secure and scalable data integrity auditing scheme based on hyperledger fabric

Ning Lu[a,b], Yongxin Zhang[a], Wenbo Shi[a], Saru Kumari[c], Kim-Kwang Raymond Choo[d,*]

[a] College of Computer Science and Engineering, Northeastern University, Shenyang, China
[b] School of Computer Science and Technology, Xidian University, Xian, China
[c] Department of Mathematics, Chaudhary Charan Singh University, Meerut 250004, Uttar Pradesh, India
[d] Department of Information Systems, University of Texas at San Antonio, San Antonio, TX 78249, USA

A R T I C L E I N F O

Article history:
Received 15 June 2019
Revised 17 January 2020
Accepted 30 January 2020
Available online 31 January 2020

Keywords:
IoT
Data integrity
Hyperledger fabric
Security
Scalability
Smart society

A B S T R A C T

As our society becomes smarter and more interconnected, more data such as those generated by Internet of Things (IoT) devices are stored remotely. These devices and services are generally externally owned and operated (e.g., commercial cloud servers). Hence, there has been interest in verifying the integrity of outsourced data, such as those stored in the remote cloud server, for example using schemes involving a third-party auditor (TPA). However, existing solutions involving TPA do not generally consider credibility and centralization, and such solutions may not be easily scalable. Thus, in this paper, we propose an efficient decentralized data integrity auditing scheme based on Hyperledger Fabric (HF-Audit), a popular consortium blockchain. Specifically, we use Hyperledger Fabric as a communication platform, where TPA can be dynamically selected for each auditing task. In order to improve the scalability of TPA, we design an efficient auditing protocol for data integrity based on bilinear pairing and commitments. Also, to improve auditing efficiency, we design two TPA selection algorithms under complete and incomplete information. Finally, we prove the security of the proposed approach, and evaluate its performance to demonstrate the utility of our proposed approach.

1. Introduction

1.1. Background and motivation

In the foreseeable future, information generated by Internet of Things (IoT) devices is expected to increase significantly, as more devices around us (e.g., sensors and smart home devices), on us (e.g., wearable devices), and in us (e.g., embedded medical devices) become more tightly integrated with our society. Such data may be stored centrally in some servers, which are likely to be externally owned and operated. For example, the data may be stored on some commercial cloud servers, where data can be stored, analyzed, shared, and so on. This is also the focus of this paper, where we study cloud storage audit (i.e., to ensure the integrity of data stored in the cloud).

There are a number of cloud storage audit solutions, and one such straightforward solution is for the user to perform the auditing tasks (Ateniese et al., 2007; Juels and Kaliski Jr, 2007). However, this is not realistic particularly if it involves significant volume of data and/or the user is executing the solution using a lower-end computation device (e.g., smartphone) or from unreliable Internet connection. As shown in Fig. 1, we can also involve a third-party auditor (TPA) to assist in the public auditing. In other words, the TPA (rather than the user) performs the auditing tasks. However, such an approach is not without limitations, such as the following.

- Security. A TPA can never be fully trusted, and risks associated with involving a TPA include (1) potential privacy leaks: the TPA may disclose or sell user data, user identity, cloud service provider (CSP)'s identity, and the link between user and CSP; (2) collusion: if the TPA colludes with the relevant CSP, then the user's auditing task results may not be reliable; (3) cheating: the TPA may reply with a fabricated auditing task result without carrying out the computation; (4) framing: the TPA may reply with a fabricated auditing task result to frame the CSP; and (5) procrastination: the TPA may deliberately delay the computation until a sufficient number of tasks have been collected to perform a batch verification. This may have performance implications for the user.

* Corresponding author.
E-mail addresses: shiwb@neuq.edu.cn (W. Shi), raymond.choo@fulbrightmail.org (K.-K.R. Choo).

https://doi.org/10.1016/j.cose.2020.101741
0167-4048/© 2020 Elsevier Ltd. All rights reserved.

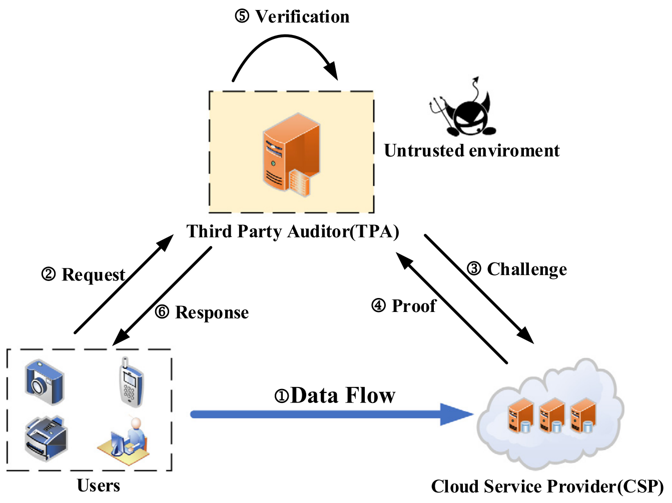


Fig. 1. Remote data integrity auditing with Third Party Auditor.

- Scalability. The TPA, a core system component, may experience (1) inefficiency (e.g. performance bottleneck): for example, a large number of users issue auditing requests to the TPA at the same time, and hence the TPA may not be able to respond to all requests in time; and (2) single point of failure: if the TPA is down, then the user's auditing task would not be able to be completed in time.

It is not surprising that a number of solutions have been proposed to address the above and other limitations relating to having TPA in cloud storage audit solutions (Fu et al., 2017; Wang et al., 2014; 2015; 2013a), but few have considered both centralization and credibility issues. Huang et al. (2014), for example, proposed a multi-TPA based remote data auditing scheme. In their approach, both CSP and user would collaboratively construct several redundant audit tasks in a linearly independent manner, and each redundant task would be assigned to a TPA. Once the TPA's auditing period is over, the user would specify a group of authorized TPAs to aggregate all the feedback from every TPA, and verify if the aggregation result matches the user's expectation. In other words, the scheme's decentralized system model avoids relying on a single TPA, and minimizes the risk of collusion between TPAs since audit tasks assigned to each TPA are linearly independent. Also, due to the verification equation required for confirmation of the auditing sub-result from each TPA, it can minimize the collusion risk between TPA and CSP, the cheating and the framing behaviors. As it utilizes both a time server and the receiver server to ensure that each TPA can complete the task within the specified time, it avoids the procrastination issue. However, it also suffers from the following disadvantages. First, it requires the user to directly communicate with each TPA, and this would leak the user's identity. Second, the calculation of expectation imposes extra processing overhead on the user. Thus, Hao et al. (2018) proposed a decentralized remote data auditing scheme based on blockchain, in which the audit task is delegated to a blockchain network. The latter comprises verification peers (essentially, TPAs). On one hand, this scheme leverages the distributed consensus mechanism based on Proof of Work (PoW) in blockchain to effectively prevent the collusion, cheating, framing and procrastination behaviors. On the other hand, this scheme does not require the users to perform any computing task, and thus reduce users' overheads. However, it also has the following drawbacks. First, to a certain extent, it can protect the user's identity privacy due to the use of pseudonym mechanism in blockchain. However, studies have also shown that such a pseudonym cannot effectively protect a user's privacy (Liao et al., 2016; Reid and Harrigan, 2013). Second, it requires every verification peer in the blockchain to perform the audit task once and calculate a random value for PoW; thus, increasing time to delivery and reducing auditing efficiency. Hence, this motivates us to design a secure and scalable remote data auditing scheme in an untrusted environment.

1.2. Proposed scheme

In this paper, we propose an efficient decentralized data integrity auditing scheme based on Hyperledger Fabric, hereafter referred to as HF-Audit. Our scheme uses hyperledger fabric, a popular Consortium chain, to establish two separate communication channels for User-TPA-CSP. These communication channels are the crediting channel and the auditing channel. The former is used to publish TPA's credit information (e.g., response time and price), and the latter is used to publish auditing information. During the auditing, a user divides the audit task into data-based subtask and label-based subtask, and recommends two groups of appropriate TPAs through the collection of their information published in the crediting channel. Then, employing the transaction identity specified by Fabric, we send the subtask requests to CSP via the auditing channel. If CSP finds the incoming messages marked with its own identity, it will generate a proof for each subtask and return them to the recommended TPAs via the auditing channel. Each TPA would compute the received proofs to generate the verification required, and publish these generated verification on the auditing channel. The user then determines whether the results of two subtasks are equal; thus, completing the verification.

In doing so, HF-Audit achieves the following security and scalability features.

1. Leveraging the identity mixer mechanism in Fabric, the publisher changes a new pseudo-name each time a message is published; thus, preventing a malicious TPA from accurately guessing the user's identity. Hence, the user's privacy is ensured.
2. Due to the open, non-tamper and traceable nature of Fabric, it can effectively prevent the occurrence of cheating, framing and procrastination.
3. The scheme minimizes the collusion risk between TPAs and between TPA and CSP, by dynamically changing the TPAs that perform the audit task.
4. The scheme only requires the user to perform a simple comparison operation; thus, minimizing the required overhead. Also, since Fabric adopts the lightweight KAFKA sort-based consensus mechanism, our scheme does not incur significant audit delays, and has higher efficiency.
5. The scheme delegates the audit task to multiple TPAs, which avoids the single point of failure limitation.

Hence, one can observe that HF-Audit can perform privacy-sensitive and computation-heavy auditing task in an untrusted network.

There are, however, a number of technical challenges underpinning the design of an efficient data integrity auditing protocol.

1. In order to improve the scalability of TPAs, both storage and bandwidth overheads associated with data auditing should be minimized. Thus, we adopt bilinear pairing-based integrity auditing protocol, in which the TPA does not require the storing and transferring of data label(s) during the auditing process. In addition, although both user and CSP can directly request for Fabric's administrators to trace the TPA's real identity, this straightforward method incurs high overhead for the administrators under heavy audit loads. In order to improve the scalability of Fabric, we design an asymmetric encryption based commitment mechanism to manage the TPAs' identities,

in which each TPA's identity is encrypted using the relevant public key to hide itself when submitting verification. Only when the CSP publishes the private key will TPA's identity be disclosed during the confirmation stage.

2. The second technical challenge is to design an efficient TPA selection algorithm. In order to ensure auditing efficiency, both speed and quality of TPA selection should be as high as possible. Thus, we first specify the indicators that affect the service quality of TPA, including price, response time, credit value and so on. Second, we formalize the TPA selection problem, and define their utility function and filter function. Third, we propose a TPA service selection algorithm under complete information, in which the user sets its own weights according to preferences, and employs Euclidean distance as the selection criteria. To further improve the applicability, we propose a TPA service selection algorithm under incomplete information, in which the user adopts Pearson Correlation Coefficient (PCC) to determine their neighbors with similar preferences, and further utilize their information predict its own missing performance parameters.

In other words, our key contributions in this paper are as follow:

- This is the first attempt to study a decentralized data integrity auditing system model based on Hyperledger Fabric in the untrusted network, which can improve the security and scalability of auditing scheme based on TPA.
- We design an auditing protocol for data integrity based on bilinear pairings and commitments, which can complete the audit task and TPA's identity traceability at a reduced cost.
- We formulate the TPA selection model and design two selection algorithms under complete and incomplete information, which can choose the TPAs that satisfy users at a fast speed.

1.3. Paper layout

The rest of this paper is organized as follows. Section 2 provides a brief overview of remote data integrity auditing with a TPA and Hyperledger Fabric. In Sections 3–5, we present our proposed HF-Audit scheme and the building blocks of the scheme, namely: our proposed data integrity auditing protocol and TPA selection algorithms. In Section 6, we demonstrate that our scheme achieves anonymity, public-auditability, traceability, security, data-preserving and collusion-resistance. In Section 7, we present two use cases to explain how our system works. Then in Section 8, we carry out extensive theoretical analysis and simulations to evaluate the computing overhead in our scheme, including the bilinear operation and TPA selection. The results show that our scheme is efficient and scalable. In Section 9, we discuss why embedding Hyperledger Fabric in the auditing process is viable. We then present the related literature and conclusion in the last two sections, respectively.

2. Background

In this section, we briefly introduce the remote data integrity auditing involving a TPA (see Section 2.1), and Hyperledger Fabric (see Section 2.2).

2.1. Conventional TPA based remote data integrity auditing

Remote data integrity implies that data will not be tampered with or discarded without authorization, and this is a basic data security requirement. Such a requirement also directly relates to the correctness of cloud storage services. A typical system model for remote data integrity auditing based on TPA generally contains the following three entities (see also Fig. 1):

- Users: Individuals or organizations who outsource their data to some remote cloud services, due to storage and computing resource and cost requirements.
- CSP: CSP is typically a commercial entity, which offers users on-demand network access to a large shared pool of storage and computing resources, usually at a cost.
- TPA: This entity is tasked with auditing users' outsourced data, upon request.

On the premise that the transmission of messages is over secure channels, a typical user-CSP-TPA data integrity auditing system works as follows:

1. Given a data block F_i, the user first computes the $F_i$âs verifiable tags ϕ_i with a hash function (e.g., SHA-256) and a secret key sk. Then, the user uploads F_i and ϕ_i to the CSP. Finally, the user deletes the file locally and reveals part of the parameters (e.g., public key pk).
2. The user sends an audit request to the TPA.
3. Upon accepting the audit request, the TPA sends a message that consists of the relevant authentication information (e.g., the user's identity signed using the user's secret key, and the TPA's identity signed using the TPA's secret key) and auditing information (e.g., the audited data block numbers, and their corresponding random numbers, which are used in homomorphic encryption to protect the user privacy of this block), to the CSP. The auditing information is also referred to as the Challenge in the remaining of this paper.
4. When the CSP receives the Challenge, it immediately checks the sender's signature and identity of user to verify its qualification. Then, the CSP uses both the Challenge and the original data blocks to produce the intermediate results, which are the Proof. Finally, the CSP returns this Proof to the TPA.
5. When the TPA receives the Proof, it verifies the CSP's identity. Once the CSP's identity is verified, the TPA uses this Proof to calculate the final result (i.e., Verification). The Verification value is usually expressed as 1 or 0, where the former implies that the data has not been tampered with and the latter means that the data integrity is compromised.
6. At last, the TPA sends the Verification to the user.

We will now use the following simplified example to explain how TPA works in a typical data integrity auditing scheme. Let us suppose that a user, Alice, wishes to upload her sensors' data to some CSP, say AWS S3. To ensure that the data is not modified by any entity, including the AWS, she engages the service of a trusted TPA, say some auditing organization (e.g., KPMG or Deloitte), to audit the data. In the following, we will only focus on explaining the auditing process, rather than the identification process.

1. Alice divides the data into 1000 blocks and generates tags for each block. She sends these data and tags to the AWS, before deleting them locally.
2. Whenever Alice wishes to check the integrity of these data, she sends an auditing request to the TPA.
3. Upon receiving the request, the TPA will generate the audited data block numbers (e.g., 3, 50, 90, 271, and 487) and their corresponding random numbers (a6d0d124d1097581e1e64596dfbb5f3fd6405add, etc.). Next, the TPA sends the block numbers and their corresponding random numbers to the AWS, with a request to return the proof of Alice's data.
4. After the AWS accepts the request (comprising the block numbers and their corresponding random numbers), it will extract the corresponding data blocks and tags according to the audited data block numbers. Next, the AWS generates a proof associated with these data blocks, tags and random numbers. Lastly, the AWS sends the proof to the TPA.

5. When the TPA receives the proof, it verifies the proof to obtain the final result (i.e., either 1 or 0), and then returns the final result to Alice.
6. Alice obtains the final result sent by the TPA, and is then able to determine the status of the data's integrity.

2.2. Hyperledger fabric

It is known that Blockchain is a peer-to-peer (P2P) distributed ledger, nodes can send transactions to the blockchain, and all nodes keep one consistent ledger with certain consensus algorithm (e.g., Proof of Work – PoW (Dwork and Naor, 1992), Proof of Stake – PoS (King and Nadal, 2012), and Practical Byzantine Fault Tolerance – PBFT (Castro et al., 1999)). The so-called ledger is a string of data blocks generated and chained cryptographically in a chronological manner, and each block can contain multiple transactions. Blockchain can be broadly categorized into public blockchain, private blockchain, and consortium blockchain, based on its access mechanism. In a public blockchain, all nodes are free to join or exit at will. However, in either private or consortium blockchain, nodes cannot access the blockchain until they have been authorized by the administrator. Consortium blockchain offers better decentralization, since the administrators comprise more than one organization (unlike a private blockchain). In this paper, we mainly focus on the consortium blockchain.

At the time of this research, (one of) the most popular consortium blockchain(s) is Hyperledger Fabric, which is a modular and extensible open-source system for deploying and operating blockchain (Androulaki et al., 2018). It has the following characteristics:

- Tamper-proof: Similar to public blockchain, once consensus is reached, the ledger will be maintained by all nodes. This means any change on a single node is invalid. Thus, it is challenging to modify the contents of historical ledger.
- Access permission: Hyperledger Fabric uses Public Key Infrastructure (PKI) to build the Membership Service Provider (MSP) module, which is then used to generate digital certificates to identify and manage the members' identities.
- Anonymity: In Hyperledger Fabric, each entity publishes a transaction with a new pseudonym, instead of using a constant pseudonym (like in public blockchain). This is known as "identity mixer" (Camenisch et al., 2010). The identity mixer is based on both zero knowledge proof and blind signature. Specifically, to achieve anonymity, each transaction is accompanied by a zero knowledge proof of the user, and others can only learn the validity but not the user's true identity. To achieve unlinkability, each zero knowledge proof differs between transactions even for the same user. Hence, no other entity can analyze these proofs to identify the user. Analogously, we may know each other but we have no idea who is doing what.
- Efficient processing: Hyperledger Fabric divides all nodes into three roles, namely: the endorsement nodes for executing and endorsing transactions, the ordering nodes for ordering and packaging transactions into blocks, and the normal nodes for publishing transactions to endorse nodes and receive new blocks from ordering nodes. Thus, this allows one to avoid a bottleneck situation during execution and the ordering for a single node. Hence, Hyperledger Fabric is more efficient, compared to the public blockchain, such as Bitcoin and Ethereum.
- Faster consensus algorithm: The consensus process in Hyperledger Fabric is faster than many public blockchains (e.g.,Bitcoin). In Bitcoin, for example, the consistency among nodes is maintained by PoW, which requires nodes to compute a block hash value for the accounting right. This process takes approximately 10 minutes on average, making the throughput very low. However, in Hyperledger Fabric, the consensus module is designed to be pluggable and supports consensus algorithms, such as PBFT and Raft. The latter uses election to replace complex computations in Bitcoin, and thus achieves significant savings in time (i.e., significantly faster than PoW).
- Channel: Hyperledger Fabric provides a channel mechanism for private communication and private data between members of a consortium. Each channel owns one ledger, and multiple channels means multiple ledgers. The data in a channel is completely isolated from the other channels.

3. Proposed scheme

In this section, we describe our proposed efficient decentralized data integrity auditing scheme based on Hyperledger Fabric (HF-Audit). The key concept is to respectively make use of Hyperledger Fabric and the decentralization mechanism to improve the security and scalability in the auditing system.

3.1. Assumptions and design goals

Since achieving secure and scalable data integrity auditing is complex, we will make a few assumptions in HF-Audit's design. Some of the assumptions are already supported by the current Internet infrastructure, while the remaining can be satisfied through existing research proposals. This allows us to limit the study scope, and at the same time, ensure that HF-Audit is fail-safe.

1. **Anti-counterfeiting.** Each entity can use the Elliptic Curve Digital Signature Algorithm (ECDSA), which ensures that its signature is not vulnerable in subexponential time. Thus, we assume the adversary is not capable of forging a signature without having access to the right secret key.
2. **Safe Fabric.** Since Hyperledger Fabric is a distributed consortium blockchain, failure of a few nodes will not affect the entire system. Thus, we assume Fabric has strong security and reliability.
3. **Resource-constrained user and TPA.** We assume the user owns neither sufficient storage resource to store and process significant amount of data, nor sufficient computation resource to implement the auditing task. And, each TPA does not have unlimited computing and storage resources.
4. **Untrusted TPA and CSP.** We assume the TPA may engage in some illicit activities (e.g., user privacy leakage) for its own benefits. And, the CSP may experience operational challenges such as outage, and/or modify or delete users' data intentionally or unintentionally.

As HF-Audit is designed to operate in an untrusted environment, it is designed to achieve the following goals. The first six goals are security-related issue, and the last two goals are scalability-related.

1. **Anonymity.** It must achieve two types of anonymity, namely: identity anonymity (i.e., the TPA cannot figure out who sends the task and who returns the proof), and connection anonymity (i.e., the TPA cannot associate the connection between the user and the CSP).
2. **Public-Auditability.** The auditing process should be public to the consortium, where each member of the consortium can review the auditing process.
3. **Traceability.** In the event that the protocol does not operate correctly, each entity can obtain the evidence relating to the process and determine who violated the rule.
4. **Regulation.** Once the consortium is under attack, it can quickly find the vulnerabilities and fix them.

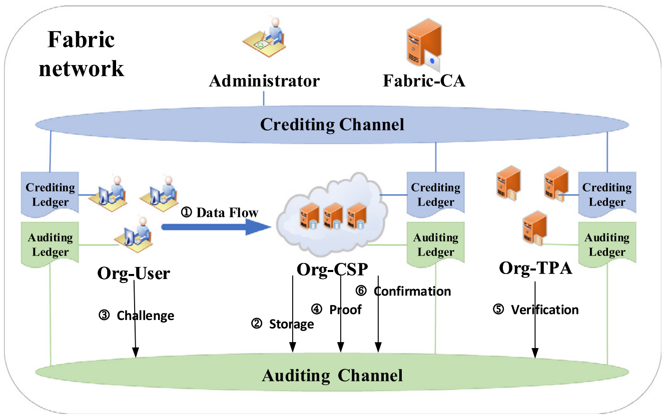


Fig. 2. System model.

5. **Data-Preserving.** The true data should be protected by some mechanism so as to prevent any TPA from obtaining the true data from auditing information.
6. **Collusion-resistance.** It is difficult for TPA to profit by colluding with CSP or other TPAs.
7. **Efficiency.** In order to reduce the auditing time and minimize the burden on resource-constrained entities' (including user and TPA), the processing overheads (including the computation overhead, bandwidth overhead and storage overhead) in the auditing operation should be minimal.
8. **Decentralization.** To avoid the single point of failure and performance bottleneck, an audit task should be performed on multiple TPAs concurrently, which can guarantee the quality of auditing service for users.

3.2. System model

In HF-Audit, Hyperledger Fabric is used as the communication platform, due to its tamper-proof, access permission, anonymity, efficient processing and private channel features. Next, we will introduce the TPA selection strategy, in which the users can dynamically select multiple TPAs for each auditing task, so as to solve the scalable issue in auditing system.

In addition to the three entities (i.e., users, CSP and TPA) described in Section 2.1, our model comprises the following four entities:

- **Fabric-CA** is the certificate authority (CA) of Hyperledger Fabric, which provides the following functions: registration of user information, and issuance and management of digital certificates.
- **Administrator** manages the Fabric network, whose functions include transaction auditing upon request (i.e., find the real identity of transaction sender), and responsibility for a member's joining and exiting.
- **Auditing channel** records the auditing information, such as the auditing request from users, the Storage, Proof and Confirmation published by the CSPs, and the Verification sent by the TPAs. Note that every node in this channel can verify the auditing result's correctness.
- **Crediting channel** records information about the credit of TPAs. Each CSP and TPA are obligated to join this channel, but it is optional for users. The administrator is the only peer who is qualified to post information about the TPA, if the evidence is conclusive.

There are five roles and two channels (ledger) in our proposed system model (see also Fig. 2). All the entities need to register with the Fabric-CA and join the network with the administrator's consent. Users upload their data (e.g., from IoT devices) and related tags to the CSP. The CSP generates a one-time public/private key pair and posts these tags and other public information to the auditing channel. During the auditing process, users obtain information about the TPA from the crediting channel, select the desired TPA for audit task that is divided into the data-based subtask and label-based subtask, and send Challenge to the auditing channel with identity assigned by the Fabric. These two subtasks are used to compute a particular value for the user to determine the integrity status of the data stored with the CSP. The label-based task is a series of computations, based on data labels stored in the blockchain. This task provides an indication of the integrity when the user uploads the data to the CSP. The data-based task is a computation of the original data group stored with the CSP, and hence implies current data integrity status. If both values are equal, the integrity would be assured; otherwise, the integrity is considered invalidated. Next, the CSP generates the Proof if the Challenge is marked with its own identity, and sends them to the corresponding TPAs in the auditing channel. The selected TPAs should calculate the Proof to generate the Verification when they receive the message, and send these Verification and their identity encrypted using the user's public key. Finally, the CSP issues a Confirmation with its one-time private key to disclose the identity of TPA, and a Verification via the auditing channel to complete the audit task. Any entity can determine whether the results of the two groups match, and if not identify the dishonest TPA.

The key benefits of our proposed system model can be summarized as follows:

(1) Security. Since Fabric adopts the identity mixer mechanism, the publisher obtains a new pseudonym for each transaction. This prevents any malicious TPA from tracing the user's identity, and protects the user's identity privacy. It can effectively prevent the TPA from conducting fraudulent activities, and carrying out framing and procrastination, due to the open, tamper-proof and traceable features of Fabric. In addition, it provides TPA selection algorithms to dynamically change the TPAs for each audit task, and this can prevent the collusion of TPA-TPA and the collusion of TPA-CSP.
(2) Scalability. Due to the lightweight KAFKA sort-based consensus algorithm, we reduce delays in Fabric. Consequently, this enhances the efficiency of auditing. In addition, it divides an audit task into two subtasks and then distributes them to multiple TPAs. This prevents single point of failure.

4. Proposed data integrity auditing protocol

In this section, we now describe the first building block for our scheme – our proposed data integrity auditing protocol. First, we will introduce the bilinear pairing, commitment and channel that will be used in our proposed protocol. A summary of notation used in our protocol is presented in Table 1.

4.1. Preliminaries

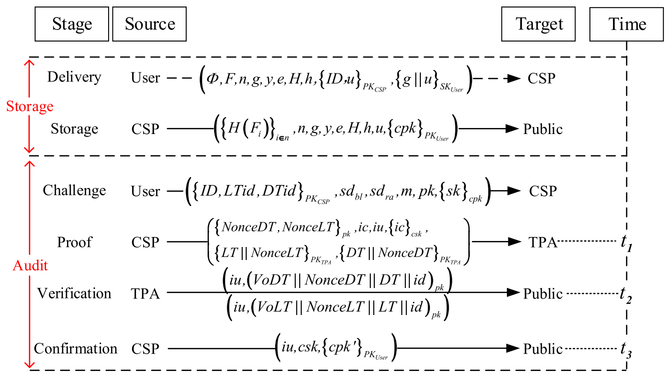
In our protocol, we adopt bilinear pairings to complete data integrity verification and use asymmetric encryption based commitment to manage TPAs' identities. The choice is due to the following: (1) a straightforward method for data integrity verification is based on hash function, but it requires both the users and TPA to store and transmit a large number of intermediate results, and this leads to higher storage and bandwidth overheads. For this, in order for users and TPA to avoid storing these information, we use bilinear pairings in our proposed protocol. (2) In order to minimize the requirements on the administrators, we use the commitment mechanism to manage TPAs' identities. In this way, once rules are

Table 1
Summary of notations.

Notations	Meanings
F	The user's file to be uploaded to the CSP
F_i	The ith file block of F, $F = \{F_i\}_{i\in n}$
ϕ_i	The verification tag of F_i
Φ	The set of verification tags, $\Phi = \{\phi_i\}_{i\in n}$
G_1, G_2, G_T	Multiplicative cyclic groups
p	The prime order of G_1 and G_2
g	The generator of G_2
e: $G_1 \times G_2 \to G_T$	The bilinear pairing
H: $\{0, 1\}^* \to G_1$	The secure hash function that map a string to a point in G_1
Z_q^*	The set of nonnegative integers less than p
$h : G_1 \to Z_q^*$	The secure hash function that maps a point in G_1 to a point in Z_q^*
PK_{CSP}/SK_{CSP}	The public/private key pair of CSP, this kind of key pair indicates the identity in the Internet. The key pair of users and TPA in the same form.
y/x	The public/private key pair generated by the user for one auditing task
pk/sk	The public/private key pair generated by the user for one auditing task, and sk can share with others at specific stage.
cpk/csk	The public/private key pair generated by the CSP for one auditing task, and csk can share with others at specific stage.
DT/LT	The proof of Data-based Task and Label-based Task
$DTid/LTid$	The id set of TPAs chosen for the data-based task and label-based task
$VoDT/VoLT$	The verification of data-based task and label-based task
ic/iu	The transaction id of Storage sent by the CSP and Challenge sent by the user
sd_{bl}/sd_{ra}	The seed to produce the audited data block number and its corresponding random number

violated, the violator can be exposed by his/her previous commitment. Although symmetric encryption, asymmetric encryption, and other cryptographic approaches can be used to implement a commitment, these approaches (with the exception of asymmetric encryption) either cannot hide the result or cannot trace the identities. Hence, we integrate the asymmetric encryption based commitment mechanism into our proposed protocol. In order to realize in one auditing task, the user needs to generate a pair of public/private key (i.e., pk/sk). The CSP also needs to generate a pair of public/private key (i.e., cpk/csk). Both key pairs pk/sk and cpk/csk are required in operations relating to the commitment.

(1) Bilinear pairing. We let G_1, G_2 and G_T be three cycle groups with the same order q. We then select a generator $g_1 \in G_1$, $g_2 \in G_2$ and $g_3 \in G_T$. A bilinear map is a map e: $G_1 \times G_2 \to G_T$ with the following conditions: 1) Bilinear: for $u \in G_1$, $v \in G_2$ and $\forall a, b \in Z_q^*$, $e(u^a, v^b) = e(u, v) = e(u, v)^{ab}$. 2) Non-degenerate: $\exists g_1 \in G_1$ and $\exists g_2 \in G_2$ such that $e(g_1, g_2) \neq 1$. 3) Computable: It is efficient for an algorithm to compute the map e.
(2) Commitment. The sender, say Alice, promises a sequence b to the receiver, Bob. In the first commitment stage, Alice promises Bob this sequence b, but Bob cannot know the information of b. In the second disclosure stage, Alice confirms to Bob what she previously promised in the commitment stage is indeed b, but Alice cannot cheat Bob (i.e., Alice cannot tamper with the value of b in the disclosure stage). The commitment mechanism has three features, namely: 1) Correctness: If both Alice and Bob implement the protocol honestly, Bob will correctly get the sequence b that Alice has promised during the disclosure stage. 2) Confidentiality: Bob cannot know any information of b before the disclosure stage. 3) Binding: Bob can only get a unique b in the disclosure stage. In other words, Alice cannot change the value of b after the commitment stage.
(3) Channel. As an important component of Hyperledger Fabric, each channel consists of five parts, namely: member (organization), anchor peer (the representative of organization, which is responsible for communicating with ordering nodes), ledger (belonging to this channel), chaincode (smart contract for this channel), and ordering nodes. Each node can take part in multiple channels and maintain multiple ledgers, while each ledger is isolated from other ledgers. In our context, we construct two channels in Hyperledger Fabric network: one for auditing and the other for crediting.



Stage
Source
Target
Time
Storage
Delivery
User
$\left(\Phi, F, n, g, y, e, H, h, \{ID, u\}_{PK_{CSP}}, \{g \| u\}_{SK_{User}}\right)$
CSP
Storage
CSP
$\left(\{H(F_i)\}_{i\in n}, n, g, y, e, H, h, u, \{cpk\}_{PK_{User}}\right)$
Public
Audit
Challenge
User
$\left(\{ID, LTid, DTid\}_{PK_{CSP}}, sd_{bl}, sd_{ra}, m, pk, \{sk\}_{cpk}\right)$
CSP
Proof
CSP
$\left(\{NonceDT, NonceLT\}_{pk}, ic, iu, \{ic\}_{csk}, \{LT \| NonceLT\}_{PK_{TPA}}, \{DT \| NonceDT\}_{PK_{TPA}}\right)$
TPA
t_1
Verification
TPA
$\left(iu, (VoDT \| NonceDT \| DT \| id)_{pk}\right)$
$\left(iu, (VoLT \| NonceLT \| LT \| id)_{pk}\right)$
Public
t_2
Confirmation
CSP
$\left(iu, csk, \{cpk'\}_{PK_{User}}\right)$
Public
t_3

Fig. 3. Proposed scheme, where dotted line denotes off-chain operation and solid line refers to on-chain operation.

4.2. Proposed protocol

There are four stages in our protocol, namely: enrollment, preparation, storage, and audit. The first two stages are part of the initialization process, and the latter two stages are related to the auditing process – see Fig. 3.

Enrollment Stage: At this stage, each role needs to enroll with the fabric-CA in the consortium, and join the channel when approved by the administrator. The identity ID of each role in the network is also sent to others.

Preparation Stage: The CSP broadcasts the necessary cryptographic parameters. The user generates his/her one-time public/private key pair for the auditing task based on these parameters. G_1, G_2 and G_T are multiplicative cyclic groups, p is the prime order of $G_1$and G_2, g is the generator of G_2, e: $G_1 \times G_2 \to G_T$ is the bilinear pairing, H: $\{0, 1\}^* \to G_1$ denotes the hash function that maps a string data to a point in G_1, and $h : G_1 \to Z_q^*$ indicates another hash function that maps a point in G_1 to a point in Z_q^*. The user generates a random secret key $x \in Z_q^*$ and computes the public key $y = g^x \in G_2$.

Storage stage: In this stage, the user generates its necessary parameters and delivers them to the CSP off-chain. Then, the CSP packages these parameters to a transaction and stores it to the ledger.

- **Delivery.** Firstly, the user divides its file F into n blocks $F = F_i(i \in n)$, and generates a random value $u \in G_1$ to compute the tag $\phi_i = (H(F_i) \cdot u^{(F_i)})^x$ for each block. Then, the tag set is computed as $\Phi = \phi_i (i \in n)$, and the user sends $\{\Phi, F, n, g, y, e, h, \{ID, u\}_{PK_{CSP}}, \{g||u\}_{SK_{User}}\}$ to the CSP, where *ID* is the true identity of user in the consortium.

- **Storage.** Once the above message is received, the CSP extracts the user's *ID* to determine whether this user has enrolled. If not, the task would terminates. Otherwise, the CSP checks the encrypted content to identify if it is legitimate, and computes $H(F_i)(i \in n)$ for each block F_i. In addition, a random key pair (*cpk, csk*) is generated, where *cpk* is the public key and *csk* is the secret key, and $\{cpk\}_{PK_{User}}$ is computed. Then, the invoke function to submit a transaction is executed (i.e., sends $\{H(F_i)(i \in n), n, g, y, e, H, h, u, \{cpk\}_{PK_{User}}\}$ to the ledger). Finally, a quintuple $SR = (ID, F, cpk, csk, ic)$ is recorded locally, where *ic* is the transaction ID generated by Fabric.

Audit stage: In this stage, the user makes an auditing request, and selects one or more TPAs to execute the auditing task. Here, we just focus on the auditing process, in which users and CSP need to respectively generate a Challenge and a Proof, and TPAs are responsible for verifying this proof. The TPA selection is described in Section 5. Note that all of these actions should be executed within a limited time.

- **Challenge.** As previously discussed, HF-Audit divides a auditing task into data-based and label-based subtasks. The user first generates a random key pair (*pk, sk*), where *pk* is the public key and *sk* is the secret key for each auditing task. Then, the user selects a group of TPAs whose id set is *LTid* to execute the label-based subtask *LT*. Meantime, the user selects another group of TPAs whose id set is *DTid* to execute the data-based subtask *DT*. Subsequently, the user generates two seeds sd_{bl} and sd_{ra}, in which seed sd_{bl} is used to produce the audited data block number, and seed sd_{ra} is used to produce its corresponding random number v_i. The user computes its commitment $\{sk\}_{cpk}$, and sends $\{\{ID, LTid, DTid\}_{PK_{CSP}}, sd_{bl}, sd_{ra}, m, pk, \{sk\}_{cpk}\}$ to the ledger through the invoke function, where *m* represents the number of data blocks to be audited.
- **Proof.** The CSP extracts the user's *ID* from $\{ID, LTid, DTid\}_{PK_{CSP}}$, and searches the record list to find a quintuple associated with this ID. If the search does not return any result, then the user has not submitted a transaction to the CSP and the task will be terminated. Otherwise, the CSP uses seed sd_{bl} to produce the audited data block numbers, which form a set *I*, as well as using seed sd_{ra} to generate a series of random numbers, which form a set $Chal = v_i, (i \in I)$. Then, Proofs *LT* and *DT* are constructed, where $LT = \prod_{i \in I} {\phi_i}^{v_i}$ and $DT = \sum_{i \in I} F_i v_i$, and both Proofs (*LT* and *DT*) and *Nonce* are encrypted with the TPA's public key, where *Nonce* is a random number generated by CSP for each TPA. Its details are $\{LT||NonceLT\}_{PK_{TPA}}$, $\{DT||NonceDT\}_{PK_{TPA}}$ and $\{NonceDT, NonceLT\}_{pk}$. Finally, the CSP executes the invoke function to submit a transaction (i.e., sends $\{\{NonceDT, NonceLT\}_{pk}, ic, iu, \{ic\}_{csk}, \{LT||NonceLT\}_{PK_{TPA}}, \{DT||NonceDT\}_{PK_{TPA}}\}$to the ledger before time t_1, where *ic* is the *ID* of transaction during storage stage, and *iu* is the *ID* of transaction during the Challenge stage).
- **Verification.** In order to confirm the auditing qualification, the TPA decrypts both $\{LT||NonceLT\}_{PK_{TPA}}$ and $\{DT||NonceDT\}_{PK_{TPA}}$ to obtain Proofs (*LT* and *DT*). If the decryption fails, it means this TPA does not need to perform the auditing task. Otherwise, this TPA is selected by the user. In this case, the selected TPA extracts the two transaction IDs *ic* and *iu*, and obtains the parameters including *e, g, y, u,* $H(F_i)$ and *pk*. Then, Verifications of label-based task $VoLT = e(LT, g)$ and Verifications of data-based task $VoDT = e(\prod_{i \in I} H(F_i)^{v_i} \cdot u^{DT}, y)$ are computed, and the public key *pk* is used to encrypt the content $\{VoDT||NonceDT||DT||id\}_{pk}$. Finally, the TPAs in group *LTid* send $\{iu, \{VoLT||NonceLT||LT||id\}_{pk}\}$, and the TPAs in group *DTid* send $\{io, \{VoDT||NonceDT||DT||id\}_{pk}\}$ to the ledger. Note that each TPA is supposed to complete the mission within time t_2.
- **Confirmation.** Once all selected TPAs have completed the auditing task, the CSP would first reveal *csk* and generate a new key pair (*cpk*′, *csk*′) for the next auditing task within time t_3. Then, $\{cpk'\}_{PK_{User}}$ is computed, and the invoke function is executed (i.e., sends $\{iu, csk, \{cpk'\}_{PK_{User}}\}$ to the ledger). Finally, each role in the Fabric can check Eq. (1) and obtains the auditing result. The left side of the equation is the bilinear pair result of the label-based task, and the right side of the equation is the bilinear pair result of the data-based task. If these two results equate, then the data integrity is assured; otherwise, it implies that the data integrity is not assured. When the latter occurs, one can then activate the process to locate the dishonest entity – see Section 6.

$$\begin{aligned} e(LT, g) &= e\left(\prod_{i \in I} {\phi_i}^{v_i}, g\right) \\ &= e\left(\prod_{i \in I} (H(F_i) \cdot u^{F_i})^{xv_i}, g\right) \\ &= e\left(\prod_{i \in I} (H(F_i) \cdot u^{F_i})^{v_i}, g^x\right) \\ &= e\left(\prod_{i \in I} H(F_i)^{v_i} \cdot u^{\sum_{i \in I} F_i v_i}, g^x\right) \\ &= e\left(\prod_{i \in I} H(F_i)^{v_i} \cdot u^{DT}, y\right) \end{aligned} \tag{1}$$

5. Proposed TPA selection algorithms

We will now present the second building block of our scheme, namely: two TPA selection algorithms, designed to recommend the appropriate TPAs for users at a short time cost. Under the condition that the indicators evaluating TPA service are collected completely, we adopt the weighted Euclidean distance to select the TPAs. Under the condition of incomplete indicators, we adopt the Pearson Correlation Coefficient(PCC) to select the TPAs. We will now introduce the formal definitions of TPA selection problem, and these two selection algorithms. A summary of notation used in TPA selection algorithms is presented in Table 2.

In order to formalize the TPA selection problem, we have to first define the TPA. The service quality of TPA is measured in terms of price, response time, availability, security, reliability and credit. The users can distinguish TPAs according to their own preference, which is the key selection basis. Clearly, the definition of TPA service attribute set can expand to $TSAS = \{tsa_1, tsa_2, ..., tsa_n\}$, where tsa_i denotes a TPA service attribute.

According to the user's tendency, these service attributes can be divided into positive attributes and negative attributes. Positive attributes are suitable for maximizing availability and reliability, and negative attributes can be used to minimize price and execution time. The calculation cost increases as the number of TPAs increases. To minimize the user's computation cost, one efficient method is to compress the state space (i.e., exclude TPAs that are not viable, and simplify the constraint condition). Thus, we define the TPA service filter function, which makes use of positive and negative attributes to compress the TPA candidate set *TCS* and further obtain the final TPA candidate set *TCS*′. We use an example to illustrate the filter process, as shown in Fig. 4. In addition, we

Table 2
Second set of notations (used in the proposed TPA selection algorithms).

Notations	Meanings
tsa	TPA service attribute
$TSAS$	The set of TPA service attributes,$TSAS = \{tsa_1, tsa_2, ..., tsa_n\}$
T	The set of all TPAs
TCS	The set of TPA candidates after the TPA filter function
TCS'	The set of final TPA candidates after the TPA selection algorithm under complete/incomplete information
Th_{pos}	The threshold of positive attributes
Th_{neg}	The threshold of negative attributes
w_i	The weight of ith attribute based on userâs preference, $w_1 + w_2+, ..., +w_n = 1$
W	The set of user's preference weights,$W = \{w_1, w_2, ..., w_n\}$
r_i	The user's desired value of ith attribute based on userâs preference
R	The set of user's requirements, $R = \{r_1, r_2, ..., r_n\}$
R_y/R_n	The set of user's determined and undetermined requirements, $R = R_y \cup R_n$, $R_y \cap R_n = \emptyset$

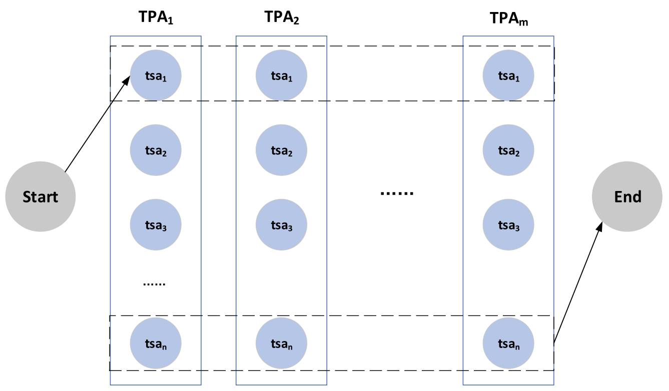


Fig. 4. An example of TPA service filter process. Suppose $TCS = \{TPA_1, TPA_2, TPA_m\}$ and each TPA's $TSAS = \{tsa_1, tsa_2, ..., tsa_n\}$. The dotted box denotes a stepwise filtering of TPA attribute tsa_i from top to bottom.

cannot directly evaluate the attributes due to the different range of allowed values. For this, we define the TPA service utility function.

Different users have different requirements when selecting the TPAs. For example, some users have a higher emphasis on price, while others focus on reliability. However, not all users know their requirements and preferences. Hence, we have two situations. In the first situation, we have information about the user's requirements and preferences, and hence we use the first proposed TPA service selection algorithm. This algorithm draws upon the user preferences, in the search for an appropriate TPA according to the weight of the different attributes. In the second situation where we have incomplete information, we use the second proposed TPA service selection algorithm based on neighboring users. Specifically, the selector makes use of the weight allocation of similar users to predict its own, and then chooses the appropriate TPAs according to its preferences.

- **TPA Service Attribute (TSA)** denotes a TPA service and the associated need(s) of the users. Each attribute can be categorized as positive attribute or negative attribute, according to the user. A higher value for the positive attribute is better, as it relates to availability, reliability, etc.; whilst a lower negative attribute value is better, as it relates to price, response time, etc.
- **TPA Service Attribute Set (TSAS)** contains the service attributes rendered by a TPA, and again the attributes can be either positive or negative – $TSAS = TSAS_{pos} \cup TSAS_{neg}$ and $TSAS_{pos} \cap TSAS_{neg} = \emptyset$. Suppose each TPA has n attributes, n_1 positive attributes and n_2 negative attributes, and $n_1 + n_2 = n$. In this way, we have $TSAS_{pos} = \{tsa_1, tsa_2, ...tsa_{n_1}\}$ and similarly, $TSAS_{neg} = \{tsa_{n_1+1}, tsa_{n_1+2}, ..., tsa_n\}$. For example, a TPA service attribute set $TSAS'$ can be measured in terms of price, response time, availability, security, reliability and credit. That is to say, $TSAS' = \{Pri, Time, Ava, Sec, Rel, Cre\}$, where Pri denotes the service price defined by TPA, $Time = Time_{exe} + Time_{delay}$ denotes the response time for executing the audit task, $Time_{exe}$ denotes the executing time, $Time_delay$ denotes the transmission delay, $Ava = \frac{Num_{rep}}{Num_{req}}$ denotes the availability, Num_{rep} denotes the number of responses, and Num_{req} denotes the number of requests, Sec denotes the security (e.g., the number of times that a TPA is hacked each year), $Rel = \frac{Num_{correct}}{Num_{total}}$ denotes the reliability, $Num_{correct}$ denotes the correct number, Num_{total} denotes the total number of responses, and Cre denotes the credit that is evaluated by users.
- **TPA** comprises its unique real-world identity (e.g., x.509 certificate) and its $TSAS$. The former is not the focus of our study, and we will simply denote it as TPA_{id}. The TPA can be viewed as a combination of them, that is $TPA = \{TPA_{id}, TSAS\}$.

The TPA service filter function filters TPA set T based on the user's requirements to obtain a set of TPA candidate set TCS that meets the user's expectations. Th_{pos} is the threshold set of positive attributes, $TCS = TFF(T, Th_{pos}, Th_{neg})$. The corresponding algorithm is described in Algorithm 1.

Algorithm 1 TPA filter function

```
 input: TPA set T, The threshold set of positive attributes $Th_{pos}$,The threshold set of negative attributes $Th_{neg}$
 output: TPA candidate set TCS
1: BEGIN
2:   int $flag_{pos} := 0$, $flag_{neg} := 0$
3:   foreach TPA $t$ in $T$
4:       foreach $t.tsa$ in $t.TSAS_{pos}$
5:           if $t.tsa < t.th_{pos}$ then
6:               $flag_{pos} = 1$ // Each positive attribute should be greater than the positive threshold
7:           end if
8:       end foreach
9:       foreach $t.tsa$ in $t.TSAS_{neg}$
10:          if $t.tsa > t.th_{neg}$ then
11:              $flag_{neg} = 1$ // Each negative attribute should be less than the positive threshold
12:          end if
13:      end foreach
14:      if $flag_{pos} == 0$ and $flag_{neg} == 0$ then
15:          add $t$ to $TCS$
16:      end if
17:  end foreach
18:END
```

The TPA service utility function is responsible for mapping each attribute in $TSAS$ to the range of [0,1]. For positive attributes, we use Eq. (2) to complete the mapping:

$$Normal(ts_{i,j}) = \frac{ts_{j\max} - ts_{i,j}}{ts_{j\max} - ts_{j\min}} \tag{2}$$

For negative attributes, we use Eq. (3) to complete the mapping:

$$Normal(ts_{i,j}) = \frac{ts_{ij} - ts_{j\min}}{ts_{j\max} - ts_{j\min}} \tag{3}$$

where $ts_{i,j}$ denotes *ith* TPA's *jth* attribute, $ts_{j\max}$ denotes the maximum value among all TPAs about *j*th attribute, and $ts_{j\min}$ is the minimum value.

5.1. TPA selection under complete information

Based on TPA service attribute set, we denote the user's preference weight as $W = \{w_1, w_2, ..., w_n\}$, where w_i represents the weight of attribute *i*, and $\sum_1^n w_i = 1$. In addition, we have already used the TPA filter function to obtain TPA set $|T| = m$.

We first define *m* TPA service as a matrix $Q_{m\times n}$, which indicates that there are m TPA candidates and each of them consists of *n* attributes.

$$Q_{m\times n} = \begin{pmatrix} q_{1,1} & \cdots & q_{1,n} \\ \cdots & \cdots & \cdots \\ q_{m,1} & \cdots & q_{m,n} \end{pmatrix} \tag{4}$$

Then, we define the user requirements as a matrix $R_{1\times n}$. Each element in the matrix denotes the TPA service attribute value that the user expects.

$$R = \begin{pmatrix} r_{1,1} & \cdots & r_{1,n} \end{pmatrix} \tag{5}$$

Now, we combine both $Q_{m\times n}$ and $R_{1\times n}$ into matrix $QR_{m+1\times n}$.

$$QR_{m+1\times n} = \begin{pmatrix} r_{1,1} & \cdots & r_{1,n} \\ q_{1,1} & \cdots & q_{1,n} \\ \cdots & \cdots & \cdots \\ q_{m,1} & \cdots & q_{m,n} \end{pmatrix} \tag{6}$$

Then, we use the TPA service utility function to calculate *QR* and obtain the standardized matrix *QR*′.

$$QR'_{m+1\times n} = \begin{pmatrix} r'_{1,1} & \cdots & r'_{1,n} \\ q'_{1,1} & \cdots & q'_{1,n} \\ \cdots & \cdots & \cdots \\ q'_{m,1} & \cdots & q'_{m,n} \end{pmatrix} \tag{7}$$

Finally, the TPA selection problem can be transformed into measuring the distance between two vectors. The closer the distance is, the more it conforms to the users' requirements. Considering that data has been standardized with the TPA service utility function, the weighted Euclidean Distance is used to compute the users' satisfaction (see Eq. (8)). The matching result needs to meet the users' requirements (i.e., *R*) and the user's preference (i.e., *w*). The result of each line is the matching result of each TPA for the user.

$$dis_w(R, q'_i) = \sqrt{\sum_{j\in n} (w_j \times (r'_{1,j} - q'_{i,j})^2)} \quad (i \in m) \tag{8}$$

The user preference based TPA selection under complete information is presented in Algorithm 2.

5.2. TPA selection under incomplete information

The user's requirements *R* can be divided into R_y and R_n, where $R_y \cup R_n = R$ and $R_y \cap R_n = \emptyset$. The former represents the requirements that the user has determined, while the latter represents the requirements that the user has not yet determined. We first look for similar users based on their determined requirements R_y,

Algorithm 2 TPA selection algorithm based on user preference weights under complete information

```
input: TPA candidate set TCS, user preference W, user requirement R
output: final TPA candidates set TCS′
1: BEGIN
2:   Construct matrix $Q_{m+1,n}$ with $TS$ and $R$
3:   Normalize matrix $Q'_{m+1,n}$ with equation (2) and (3)
4:   Separate $Q'_{m+1,n}$ into $Q'_{m,n}$ and $R'$
5:   Calculate Euclidean distance between $Q'_{m,n}$ and $R'$ with $W$
6:   Construct appropriate TPA set $TS'$ according to the distance
7:END
```

Algorithm 3 TPA selection algorithm based on neighboring user under incomplete information.

```
input: TPA candidate set TCS, user preference W, user determined requirement $R_y$, user undetermined requirement $R_n$, rest user requirement set UR
output: final TPA candidates set TS′
1: BEGIN
2:   Separate $UR$ into $UR_y$ and $UR_n$ according to $R_y$ and $R_n$
3:   Calculate neighboring user set $NUS$ with $UR_y$ and $R_y$, using equation(9) and $Top-K$
4:   Calculate undetermined requirement $R_n$ with $UR_n$, using equation(10)
5:   Calculate the appropriate TPA set $TS'$ based on the $Algorithm1$
6:END
```

then predict R_n based on similar users, and further obtain the appropriate TPA. The TPA selection under incomplete information is described in Algorithm 3.

Since Pearson Correlation Coefficient(PCC) has been widely used in recommendation system due to its lightweight and efficiency, we also adopt it to solve the similarity between users.

$$Sim(a, b) = \frac{\sum_{i\in n_y} ((r_{a,i} - \overline{r_{a,i}})(r_{b,i} - \overline{r_b}))}{\sqrt{\sum_{i\in n_y} (r_{a,i} - \overline{r_a})^2}\sqrt{\sum_{i\in n_y} (r_{b,i} - \overline{r_b})^2}} \quad (j \in n) \tag{9}$$

In the above equation, $r_{a,i}$ denotes the value of attribute *i* of user *a*, $\overline{r_{a,i}}$ denotes the mean value of attributes *i* of user *a*, $r_{b,i}$ denotes the value of attribute *i* of user *b*, $\overline{r_{b,i}}$ denotes the mean value of attribute *i* of user *b*, and n_y denotes the number of determined attributes in the set R_y. The larger the value of *Sim*(*a*, *b*) is, the higher the similarity between users will be.

After calculating the similarity, the user can adopt $Top-K$ algorithm to select *K* users with the highest similarity, and get the value of user requirements $r_i(r_i \in R_n)$ in the set of adjacent users U_k.

$$P(u, r_i) = \overline{r_i} + \frac{\sum_{u_j\in U_k} (Sim(u, u_j)(r_{u_j,i} - \overline{u_j}))}{\sum_{u_j\in U_k} Sim(u, u_j)} \quad (j \in n) \tag{10}$$

where $\overline{r_l}$ denotes the mean value of attribute *i* of the user, and $\overline{u_l}$ denotes the mean value of attribute *i* of its neighbors.

6. Security analysis

In this section, we analyze the following security features of our scheme HF-Audit, namely: privacy-preserving, public audibility, traceability, security, data-preserving and collusion-resistance.

Theorem 1 (Privacy-Preserving)**.** *For the purpose of protecting the user's privacy, HF-Audit is able to restrict the following behaviors of TPA: (1) obtain the identity of user who sends the mission, (2) obtain the identity of CSP who returns the proof, and (3) link the anonymous CSP with any number of users.*

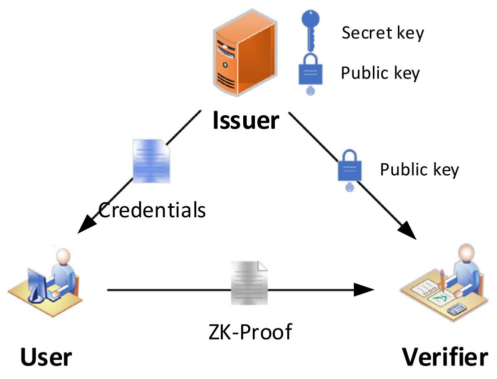


Fig. 5. Workflow of identity mixer.

Proof. HF-Audit is based on Hyperledger Fabric, which adopts Identity Mixer (Camenisch et al., 2010), a cryptographic protocol suite proposed by IBM, to guarantee privacy. There are three roles in an Identity Mixer workflow, namely: issuer, user and verifier – see also Fig. 5.

1. The issuer certifies a set of user's attributes and are issued in the form of a digital certificate, named as "credential" in the Hyperledger Fabric.
2. The user later generates a "zero-knowledge proof" of possession of the credential, and also selectively discloses only the attributes the user chooses to reveal.
3. The verifier uses the public key from the issuer to verify the "zero-knowledge proof". The latter reveals no additional information to the verifier.

Users can have multiple independent public keys for the same secret key, instead of each validator binding to a fixed single public key. This allows the user to use different public keys for each transaction. The credentials can be converted into a valid "zero-knowledge proof" for any public key of the user. These zero-knowledge proofs contain only a part of the attributes in the original certificate, and the converted zero-knowledge proofs can still be verified under the issuer's public key.

Identity Mixer technology is built from a blind signature scheme that supports multiple messages and efficient zero-knowledge proofs of signature possession.[2] The zero-knowledge proofs ensure that the user's attributes are not overexposed to the verifier (i.e., **anonymity**), and the blind signature scheme ensures that the user cannot be associated with the blind data (i.e., **unlinkability**). This particular implementation for Fabric uses a pairing-based signature scheme proposed by Camenisch and Lysyanskaya (2004) (see also Au et al., 2006). The ability to prove knowledge of a signature in a zero-knowledge proof is proven in Camenisch et al. (2016). The authors of Veeningen et al. (2014) used their framework to analyze and compare four identity management systems, including Identity Mixer. They showed that Identity Mixer satisfies privacy requirements, such as irrelevant attribute undetectability, property-attribute undetectability, session unlinkability, and anonymity revocation. □

Theorem 2 (Public-Auditability). *Even if the audit task is completed, any other entities in HF-Audit could re-execute this task to verify the auditing result.*

Proof. The public auditability consists of two parts: (1) any entity can verify the final result according to the results from two groups of TPAs, and (2) any entity can verify the results of the TPA. Now, we will present the respective proofs.

(1) In our scheme, verification result is divided into two parts. The first part LT is computed by a group of TPAs and the second part DT is calculated by another group. Any entity in HF-Audit is able to obtain the secret key csk when the process works properly, and decrypt sk with csk to obtain the auditing results $VoDT$ and $VoLT$ with sk. Then, one uses $VoLT = e(LT, g)$ and $VoDT = e(\prod_{i\in I} H(F_i)^{v_i} \cdot u^{DT}, y)$ to determine whether the auditing result is correct.

(2) For either group $DTid$ or group $LTid$, the Verification values computed by all TPAs of the group should be consistent. Otherwise, this miscalculation of TPA occurs. In order to locate the inaccurate TPA, any entity requires the calculation of a new Verification according to the Proof will be returned by TPA. Then, it will be further verified whether the TPA result is correct by comparing the old and new Verification values. □

Theorem 3 (Traceability). *In Hf-Audit, any entity who violates the rules would be exposed.*

Proof. As the ledger is maintained by all nodes through the consensus mechanism, the ledger's data is reliable. All interactive information are stored in the auditing ledger. When one entity violates the auditing process, the information in the auditing ledger can be used as evidence to prove its violation, and thus HF-Audit achieves traceability. For example, if CSP losts one data block F_j, $j \in n$, then $DT = \sum_{i\in I} F_i v_i$ is incorrect. We respectively use F_i' to replace F_i, and DT' to replace DT, and further obtain Eq. (11). Clearly, the left and right hand sides in Eq. (11) are not equal. This implies that either CSP or TPA is dishonest. The tracking process will then be activated. Specifically, the user first computes $VoDT$ and $VoLT$, and the incorrect value for any of these two implies that TPA is dishonest. If both values are right, then it implies that CSP is dishonest. The users also need to verify the proof LT. By comparing the old and new Proof LT, the user can find that LT is correct. In this way, the proof DT could be determined to be incorrect. As DT is calculated by the data block, data in CSP is corrupted.

$$\begin{aligned} e(LT, g) &= e\left(\prod_{i\in I} {\phi_i}^{v_i}, g\right) \\ &= e\left(\prod_{i\in I} (H(F_i)\cdot u^{F_i})^{xv_i}, g\right) \\ &= e\left(\prod_{i\in I} (H(F_i)\cdot u^{F_i})^{v_i}, g^x\right) \\ &= e\left(\prod_{i\in I} H(F_i)^{v_i}\cdot u^{\sum_{i\in I} F_i v_i}, g^x\right) \\ &= e\left(\prod_{i\in I} H(F_i)^{v_i}\cdot u^{DT}, y\right) \\ &\neq e\left(\prod_{i\in I} H(F_i)^{v_i}\cdot u^{\sum_{i\in I} F_i' v_i}, g^x\right) \\ &= e\left(\prod_{i\in I} H(F_i)^{v_i}\cdot u^{DT'}, y\right) \end{aligned} \tag{11}$$

□

Theorem 4 (Regulation). *If someone attempts to interfere with HF-Audit, its administrator can quickly intervene and take corrective action(s).*

Proof. To join HF-Audit, an entity needs to be verified using his/her real identity and be approved by the administrator. If any

[2] https://hyperledger-fabric.readthedocs.io/en/release-1.3/idemix.html.

Table 3
Security analysis: a comparative summary.

	Our scheme	Scheme (Huang et al., 2014)	Scheme (Yu et al., 2019)	Scheme (Fu et al., 2017)
Anonymity	Y	N	–	Y
Data-preserving	Y	Y	Y	Y
Public-Auditability	Y	N	Y	Y
Traceability	Y	Y	Y	Y
Decentralized	Y	N	Y	N
Collusion-resistance	Y	Y	–	–

entity in HF-Audit shows some malicious behavior, intentionally or unintentionally (e.g., compromised by an adversary), the administrator can trace his/her identity and take protective measures (e.g., temporarily cancel its authorization and expel it from HF-Audit). If the entity's misbehavior is determined to be due to compromise, the administrator can restore its authorization when the entity is restored to normal. □

Theorem 5 (Data-Preserving). *In HF-Audit, TPA cannot recover the true data from the auditing information.*

Proof. It is challenging for TPAs to recover user's data by computing LT or DT, where $LT = \prod_{i\in I} {\phi_i}^{v_i}$ and $\phi_i = (H(F_i) \cdot u^{(F_i)})^x$. The original data F_i is protected by H: $\{0, 1\}^* \to G_1$, u and the user's secret key x. $DT = \sum_{i\in I} F_i v_i$, where all of the F_i is blinded by v_i. Based on the hardness of Computational Diffie-Hellman (CDH) problem in G_1, it is challenging for anyone to compute F_i. □

Theorem 6 (Collusion-resistance). *Collusion can occur (1) between CSP and TPA; or (2) among TPAs. Collusion between the respective entities should be challenging in either setting.*

Proof. We will now present the proofs from two aspects, namely: high risk of detection, and low success rate of colluding.

(1) High risk of detection. Due to the crediting channel, TPA bears the risk of being reported. When the report is determined to be correct, TPA's credit value would be reduced, and this leads to a decrease in the probability of being selected by the user. Therefore, TPAs have a vested interest to be honest, and not colluding.
(2) Low success rate of colluding. Even if some entities attempt to collude, it is difficult to complete collusion. We assume that there are N TPAs in HF-Audit, among which D TPAs are in collusion. These D TPAs are malicious TPAs (hereafter referred to as M-TPAs), and the remaining $N - D$ TPAs are normal TPAs (i.e., N-TPAs). For each auditing task, the user selects n TPAs, among which k TPAs are M-TPAs (i.e., $k \in (0, min\{D, n\})$). To simplify this explanation, we assume that each TPA is selected with the same probability. At this point, it approximately satisfies the hypergeometric distribution:

$$P\{X = k\} = \frac{C_D^k \cdot C_{N-D}^{n-k}}{C_N^n} \tag{12}$$

$P\{X = k\}$ is a monotone decreasing function for k. When $k \geq 2$, we consider the collusion successful. The probability of collusion success R_c is calculated as:

$$R_c = \sum_{k\in\min\{n,D\}} \frac{C_D^k \cdot C_{N-D}^{n-k}}{C_N^n} \tag{13}$$

For example, when $N = 100$, $D = 10$, and $n = 4$, $R_c \approx 0.0488$. In fact, on the basis of our credit channel and TPA service selection algorithm, users tend to choose TPAs with a high credit value. This, consequently, reduces the incentive to collude and further reduces R_c. □

Table 3 presents a comparative summary of the security features between our scheme and those of Huang et al. (2014), Yu et al. (2019), and Fu et al. (2017). One can observe all the auditing schemes achieve both data-preserving and traceability features. Our scheme also protects the connection between user and CSP (i.e., collusion-resistance) and avoids TPA being the single point of failure.

7. Use cases

In this section, we will explain how the proposed system works.

7.1. Auditing process

In the conventional model, there are only three roles (i.e., users, CSP and TPA) involved in the auditing process. Auditing information is only exchanged among the user, CSP and TPA, so it is a private process. In our system, the exchange (or interaction) information is on the blockchain, and the process will be stored permanently and challenging to be tampered with. Thus, this is a trusted public process.

We will use an example to explain how our system works in the data integrity auditing. The roles are still the same as before, where we have a user Alice and a CSP (AWS S3). However, now we have ten TPAs, TPA_1, TPA_2...TPA_{10}. Again, we will only focus on explaining the auditing process, and not the identification process. In other words, we assume all the entities have already joined the consortium blockchain Hyperledger Fabric, and the administrator has also uploaded these ten TPAs' information to the crediting channel. For example, TPA_1's information is shown as follows (and for simplicity, *id* of the first TPA is labeled 1):

```
TPA
{
 id:1;
 price:150;
 response_time:260;
 availability:0.96;
 reliability:0.98;
 security:0.96;
 credit:98;
}
```

1. Alice divides the data into 1000 blocks and generates tags for each block. She sends these data and tags to the AWS. The content contains (Φ, F, 1000, g, y, e, H, h, $\{Alice, u\}_{PK_{AWS}}$, $\{g||u\}_{SK_{Alice}}$).
2. AWS computes the hash value of each block, which is then sent to the auditing channel. The transaction contains $(H(F_i)_{(i\in n)}, 1000, g, y, e, H, h, u, \{cpk\}_{PK_{Alice}})$. When Alice checks for correctness, she deletes them locally.
3. Alice wishes to check the integrity of these data. First, she chooses the desired TPAs based on the TPA selection algorithm with information from crediting channel. Based on

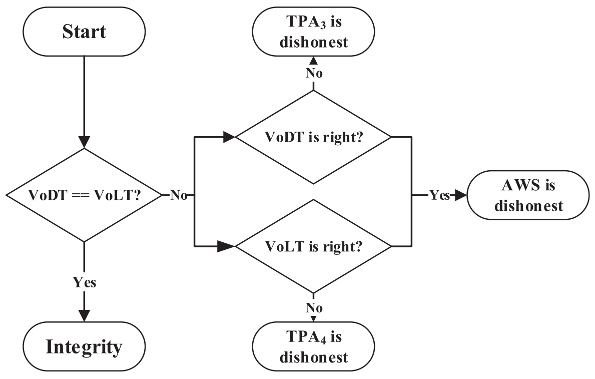


Fig. 6. Tracing process.

her preference, she chooses TPA_3 for the data-based task and TPA_4 for the label-based task. Next, the user packages the selection and seeds of block and random into a transaction (e.g., $8cc3d64002b33f1d8dc48c542185b271696adc2e$). Finally, Alice sends the auditing request transaction to the auditing channel to ask the AWS to send the proof to TPA_3 and TPA_4. The transaction contains $(seed_{block}, seed_{random}, \{AWS, TPA_4, TPA_3\}_{PK_{AWS}}, 300, pk, \{sk\}_{cpk})$.

4. When AWS receives the request, it uses the block's seed and randomly generates the audited data block number (3, 7, 50, 90, 271, 398, 486, 487, etc.) as well as its corresponding random numbers ($a6d0d124d1097581e1e64596dfbb5f3fd6405add$, etc.). Next, the AWS generates the proof of data-based task for TPA_3 and label-based task for TPA_4, respectively labeled as *DT* and *LT*. Finally, the AWS packages these proofs into a transaction, which is sent to the auditing channel. The transaction contains $(ic, iu, \{DT||NonceDT\}_{PK_{TPA_3}}, \{LT||NonceLT\}_{PK_{TPA_4}}, \{NonceDT||NonceLT\}_{pk}, ic_{csk})$.
5. TPA_3 receives the request, verifies the proof *DT* to obtain the verification *VoDT*, and packages the verification result into a transaction. TPA_3 sends the transaction to the auditing channel. The transaction contains $(iu, \{VoDT||NonceDT||DT||TPA_3\}_{pk})$. We remark that the work of TPA_4 is similar to that of TPA_3, and the transaction contains $(iu, \{VoLT||NonceLT||LT||TPA_4\}_{pk})$.
6. When all verification tasks conclude, the AWS should publish the private key *csk*, and give a new pair key for the next auditing task. The AWS packages these keys into the Confirmation transaction, containing $(iu, csk, \{cpk'\}_{PK_{Alice}})$. It will also be sent to the auditing channel.

7.2. Tracing process

Based on the open nature of blockchain, we design the protocol to realize public auditing, in the sense that everyone in the blockchain can go through the process to be a witness. We will use the below example to show how one can trace a dishonest role in our system. Let us suppose a reviewer Bob is tasked with reviewing the process involving Alice, in order to determine the perpetrator – see also Fig. 6.

1. From $(iu, csk, cpk'_{PK_{Alice}})$, Bob can get *csk* to obtain *sk* in sk_{cpk}.
2. With *sk*, Bob can decrypt both transactions $(iu, \{VoDT||NonceDT||DT||TPA_3\}_{pk})$ and $(iu, \{VoLT||NonceLT||LT||TPA_4\}_{pk})$ to obtain *VoDT* and *VoLT*. If *VoDT* does not equal *VoLT*, then Bob should check both *VoLT* and *VoDT*.
3. For *VoLT*, Bob can use LT to verify the result of $TPA_4, VoLT' = e(LT, g)$, and the other parameters can be obtained from the blockchain. If $VoLT' = VoLT$, then TPA_4 is honest; otherwise TPA_4 is deemed to be dishonest.

Table 4
Bilinear pairing computation cost of the proposed scheme.

	TagGen	Proof	Verification
User	$n(H+M+2E)$	0	0
CSP	0	$nE+(2n-1)M+(n-1)A$	0
TPAU	0	0	P
TPAO	0	0	$P+(n+1)E+nM$

4. For *VoDT*, Bob can use *DT* to verify the result of $TPA_3, VoDT' = e(\prod_{i\in I} H(F_i)^{v_i} \cdot u^{DT}, y)$, and the other parameters can be obtained from the blockchain. If $VoDT' = VoDT$, then TPA_3 is honest; otherwise TPA_3 is deemed to be dishonest.
5. If all TPAs are found to be honest, then either DT or LT is wrong and this implies a dishonest AWS.

8. Performance evaluation

In this section, we focus on the proof of scalability in HF-Audit. Specifically, we theoretically analyze the efficiency of HF-Audit, prior to using simulations to evaluate its efficiency.

8.1. Theoretical analysis

We will now mathematically evaluate the computational costs associated with the bilinear pairing, asymmetric encryption and decryption operations.

(1) Bilinear pairing operation

We use the generic definition to evaluate the computational cost of the user, CSP, TPALT and TPADT during auditing. Suppose *M* denotes the multiplication operation on group *G*, *E* is the exponentiation operation on *G*, *P* is the bilinear pairing operation, and *H* is the hash function mapping a string to a point on *G*. In addition, *A* denotes the extra operation on group Z_q^*. The computational overheads are mainly in the Tag Generation, Proof and Verification steps.

Let *n* denotes the number of blocks to be audited. In the Delivery stage, users need to generate tags for their data blocks, where the operation is $\phi_i = (H(F_i) \cdot u^{(F_i)})^x$. Thus, the user computes *n* hash, *n* multiplications and 2*n* exponentiation. In the Proof stage, the CSP should compute Proof for TPA, where the operations are $LT = \prod_{i\in I} {\phi_i}^{v_i}$ and $DT = \sum_{i\in I} F_i v_i$. For the former, the CSP computes *n* exponentiation and $(n-1)$ multiplications, and for the latter, the CSP computes *n* multiplications and $(n-1)$ additions. In total, the computation is *n* exponentiation, $(2n-1)$ multiplications and $(n-1)$ additions. In the Verification stage, TPAU is supposed to perform $VoLT = e(LT, g)$, so the operation is only one pairing. However, TPAO performs $VoDT = e(\prod_{i\in I} H(F_i)^{v_i} \cdot u^{DT}, y)$ including one pairing, $(n+1)$ exponentiation and *n* multiplications. The result is shown in Table 4.

(2) Asymmetric encryption and decryption operation

Asymmetric encryption is used to protect entities' information in the auditing ledger. We define these operation to evaluate the encryption and decryption cost of the user, CSP and TPA in the proposed protocol, where *GN* is used to represent the operation of generating a pair of public and secret keys, *EN* denotes the encryption operation using public key, *DE* is the decryption operation with secret key, *SI* denotes the signature operation with secret key, and *VS* denotes the verification operation using public key.

Let *u* be the number of TPA in group *U*, and *o* is the number of TPA in group *O*. In the Delivery stage, the user needs to prove and hides its identity *ID*, where the operations are $\{ID, u\}_{PK_{CSP}}$ and $\{g||u\}_{SK_{User}}$. Thus, the user needs one *EN* and one *SI*. In the Storage stage, the CSP checks the identity of user, generates a pair of keys and encrypts the new public key. In this way, there are one *DE*, one *VS* one *GN* and one *EN* for

Table 5
Asymmetric encryption and decryption computation cost of the proposed scheme.

	Delivery	Storage	Challenge	Proof	Verification	Confirmation	Total
User	$EN+SI$	0	$GN+2EN+DE$	0	0	$(u+o)DE$	$GN+3EN+(u+o+1)DE+SI$
CSP	0	$GN+EN+DE+VS$	0	$(u+o+1)EN+DE+SI$	0	$GN+EN$	$2GN+(u+o+3)EN+2DE+SI+VS$
TPA	0	0	0	0	$EN+(u+o)DE$	0	$EN+(u+o)DE$

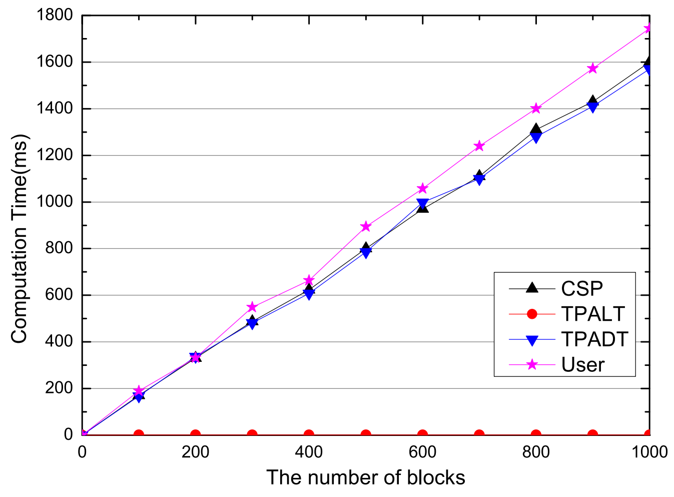


Fig. 7. Computation time of bilinear pairings operation for off-chain.

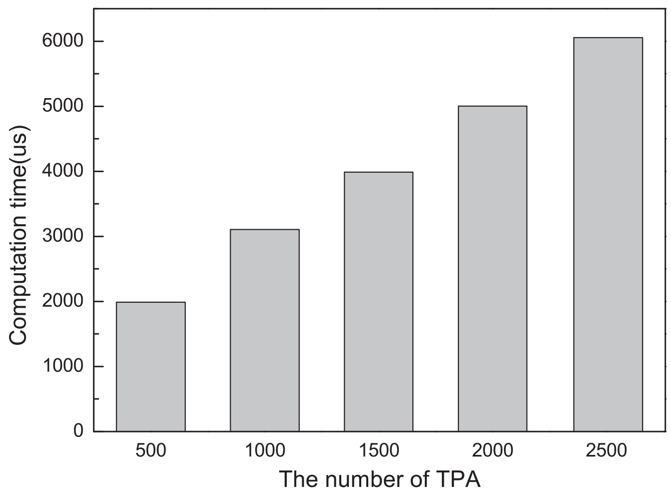


Fig. 8. Computation time with complete information.

CSP. In the Challenge stage, the user decrypts $\{cpk\}_{(PK_{User})}$, generates a pair of keys, computes $\{ID, LTid, DTid\}_{PK_{CSP}}$ and encrypts $\{sk\}_{cpk}$. The operation includes one GN, one DE and two EN. In the Proof stage, the CSP decrypts $\{ID, LTid, DTid\}_{PK_{CSP}}$, encrypts $\{LT||NonceLT\}_{PK_{TPA}}$ and $\{DT||NonceDT\}_{PK_{TPA}}$ for corresponding TPAs, encrypts $\{NonceDT, NonceLT\}_{pk}$ and signs ic. So the CSP needs one DE, (u+o+1) EN and one SI. In the Verification stage, the TPA decrypts $\{DT||NonceDT\}_{PK_{TPA}}$ and $\{LT||NonceLT\}_{PK_{TPA}}$ to obtain the task, and encrypts its result. In this way, the operations include $(u+o)$ DE at most and one EN. In the Confirmation stage, the CSP generates a new pair of keys and encrypts the public key. The user can verify all results from TPAs using the secret sk. In this stage, the CSP needs one GN and one EN, and the user needs $(u+o)$ DE. To sum up, the operations of user include one GN, $(u+o+1)$ DE, three EN and one SI. The operations of CSP are two GN, two DE, $(u+o+3)$ EN, one SI and one VS. The operations of TPA are $(u+o)$ DE and one EN. The result is shown in Table 5.

8.2. Experimental evaluation

We mainly measured the computation time of the entire auditing process, including bilinear pairings operations and TPA selection algorithm for users, for both off-chain and on-chain. In the adoption of Hyperledger Fabric, both consensus time and chain operation are factors to be considered. Therefore, we measured these computation time for each entity in Fabric. The experiment was performed on a laptop, which runs CentOS7 on an Intel i5-4210m CPU at 2.6GHz and 4GB RAM. Coding was written using the GO language and the bilinear pairings operation algorithms were implemented with the goPBC (Pairing-Based Cryptography) Library.[3]

(1) For off-chain: first, we tested the bilinear pairings operation computation times for different roles. From Fig. 7, we observe that the computation time of TPADT remains stable even when the number of blocks increases.

Second, for the TPA selection algorithm, dataset about the TPA service is required but one fails to obtain such a dataset. However, there are two real datasets for QoS that meet our requirement. QWS dataset[4] (Al-Masri and Mahmoud, 2007a; 2007b) includes 2507 real Web services, each consists of 9 QoS indicates in which three of them satisfy our need. The dataset2 of WS-DREAM[5] (Zhang et al., 2011) includes results from 142 users on 4500 Web services. To combine both datasets, we cut the latter to 142 users on 2507 Web services. The experiment focused on the computation time, so the cut had little influence on it. First, we tested the time for varying number of TPA with complete information using the QWS dataset. As shown in Fig. 8, the computation time is directly proportional to the number of TPAs. The time to measure 2507 TPAs is less than 6500us. Then, we determines the computation time for different number of TPA with incomplete information. From Fig. 9, we observe that the computation time is also directly proportional to the number of neighboring users. However, combining the two figures, we observe that the main computation time is due to the choosing of the undetermined factors.

(2) For on-chain: we combined the bilinear pairing operation and consensus process. Our system consists of four peers, namely: one CSP, one User, one TPALT, and one TPADT. These four peers were added to the same channel in Hyperledger Fabric v1.4. There was one single chaincode containing two functions, Query and Invoke. Query was used to get the data from the ledger and the Invoke was used to publish the information for the next task. We added a delay function in the process to simulate the computation time of each stage of auditing due to goPBCâs not working properly in chaincode.

Experimental results with different blocks to be audited are shown in Fig. 10. Here, the user just publishes the Challenge, which is not connected with the blocks. Hence, the cost is stable. The computation time of CSP is still close to TPADT. Fig. 11 shows the difference among stages, and Fig. 12 shows the difference among

[3] https://github.com/Nik-U/pbc.

[4] https://qwsdata.github.io/.

[5] http://wsdream.github.io/.

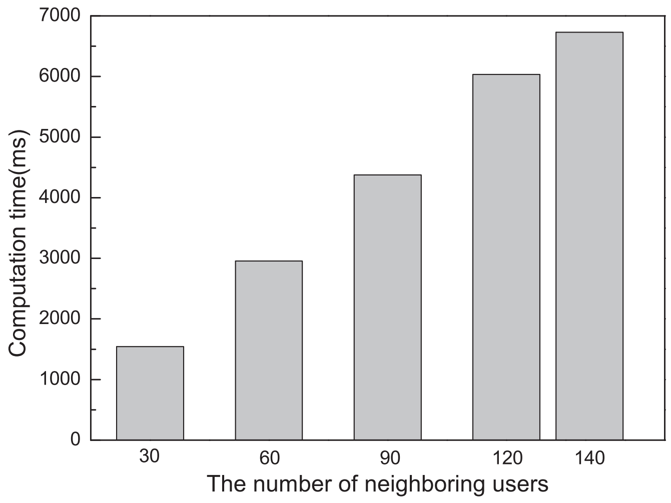


Fig. 9. Computation time with incomplete information.

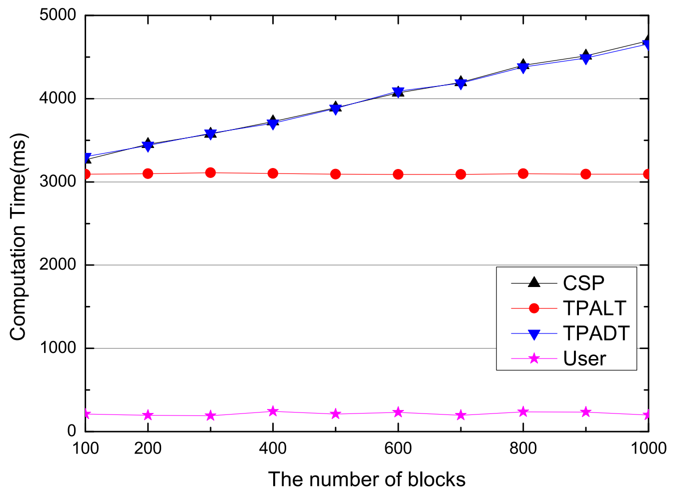


Fig. 10. Computation time of bilinear pairings operation for on chain.

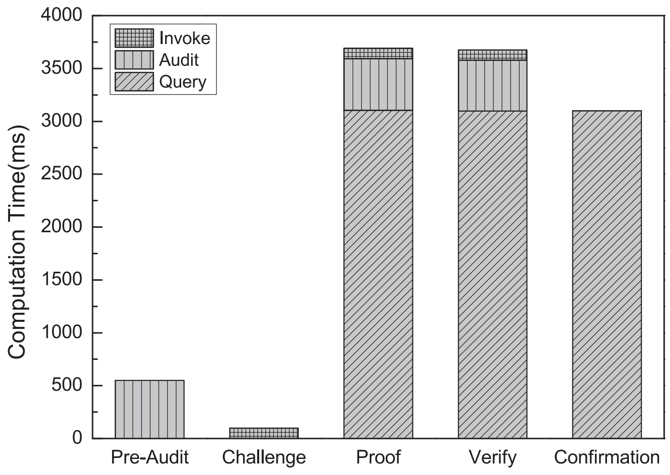


Fig. 11. Computation time in different processes where blocks = 300.

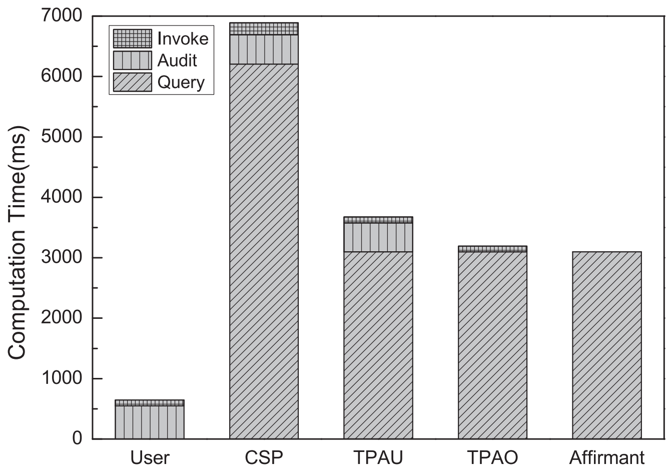


Fig. 12. Computation time in different roles where blocks = 300.

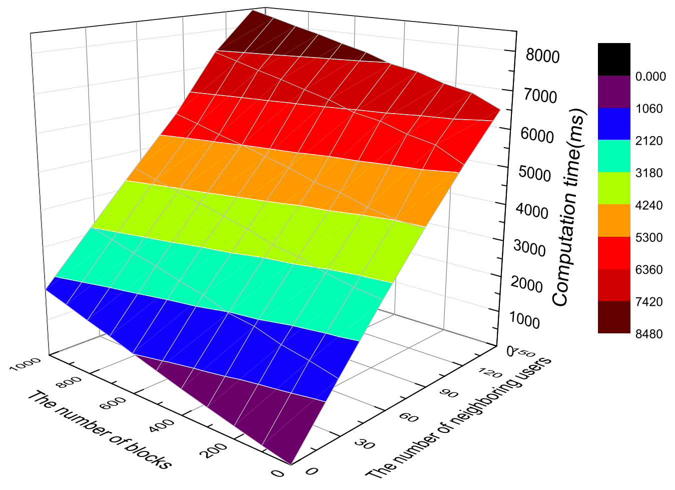


Fig. 13. Computation time with incomplete information in different blocks.

entities. In general, user is the affirmant, but the protocol allows anyone to be the affirmant. Thus, we list this role separately. From both Figs. 10 and 11, we observe that the consensus time is the key cost component for the auditing process, the function Invoke is the smallest (about 100ms), and the time overhead of Query is relatively large (about 3000ms). Unlike Bitcoin and Ethereum that use Difficulty to limit the consensus time of the blockchain, Fabric has the flexibility to set the consensus time. Its consensus components are pluggable, so the consortium is free to choose the more appropriate consensus algorithm to improve efficiency.

Then, we combined bilinear pairings operations and TPA selection algorithm for a user to measure the integrated time. The result shown in Fig. 13 is primarily influenced by the TPA selection and auditing processes. We observe that computation time increases with the number of neighboring users and the number of blocks to be audited.

9. Discussion

An ideal data integrity auditing scheme should directly use the endorsement node in Hyperledger Fabric as the TPA.

(1) Unfortunately, it is not feasible because the current program language Golang lacks support for bilinear pairs and relies on the local PBC library written in C.

(2) In addition, the current endorsement strategy does not meet the needs of auditing task. First, it is a large computational overhead for endorsement nodes to execution Verification function. Different endorsement nodes can provide different levels of computing power, and the corresponding speeds naturally differ. This will bring different service experiences to users. Second, the endorsement strategy uses principle and threshold

gate, which has limited flexibility. The combination of AND and OR in Organization cannot satisfy the users' choice.

(3) Finally, the endorsement strategy set in chaincode instantiation stage is difficult to change and it is designed for chaincode level, not user level. Users cannot customize their own strategy in a private way. However, a practical date integrity auditing scheme can use Hyperledger Fabric as a communication platform, which results in the following benefits: (1) it can provide strong anonymity and effectively protect the identity privacy of all entities. (2) With the help of administrators, it can trace the malicious role and ensure that the audit task is executed properly. (3)The ledger is maintained by all nodes through the consensus mechanism, so the ledger's data is extremely reliable.

10. Related literature

At the time of this research, a large number of data integrity auditing schemes have been proposed in the literature and they can be categorized into User-CSP based data integrity auditing schemes and User-CSP-TPA based data integrity auditing schemes.

(1) User-CSP based auditing. Ateniese et al. (2007) proposed the concept of provable data possession (PDP), designed to verify the integrity of remote data without downloading original data. Juels and Kaliski Jr (2007) proposed the proof of retrievability (PoR), which realizes data recovery while checking data integrity by using erasure codes. Sebé et al. (2008) divided a file into several blocks, which minimizes verification overhead. However, it cannot guarantee correctness. Researchers such as Li et al. (2018) and Jiang et al. (2016) reinforced the importance of designing solutions to mitigate privacy and security concerns in the cloud environment. Xu et al. (2019) and Yang et al. (2019) thened present their blockchain-based solutions. Suzuki and Murai (2017) also used blockchain as the information channel between users and CSP, so as to enhance auditable. However, the bitcoin test network used in their evaluation was vulnerable to malleability attacks (Decker and Wattenhofer, 2014). Nguyen et al. (2018) used Ethereum to perform the audit tasks, but it comes at a significant financial cost. Although the use of public chain achieves public auditability naturally, there are a number of shortcomings,such as high delay and weak anonymity (Tschorsch and Scheuermann, 2016). Reid and Harrigan (2013) and Liao et al. (2016) designed approaches to determine the real identity of users through association analysis. The public chain generally uses tokens as incentives, which can be attractive to attackers (Eyal and Sirer, 2018; Karame et al., 2012). Liu et al. (2017) proposed a blockchain-based IoT data audit framework, which uses smart contracts to replace TPA on the private chain. Yu et al. (2019) used private blockchain to record the auditing process, but overhead to the users is significant.

(2) User-CSP-TPA based auditing. To minimize user's overheads, Wang et al. (2009) first introduced the concept of Third Party Auditor (TPA) into the PDP scheme. However, a fully trusted TPA may not exist in practice, for example as demonstrated by Wang et al. (2013b). Wang et al. (2015, 2013a) demonstrated the potential of using proxy re-signature to protect users' identity privacy. In a separate work, Wang et al. (2014) used homomorphic verifiable ring signature to hide the user's identity, but the auditing cost increases linearly with the user group. Clearly, this approach does not scale. Huang et al. (2014) used multiple TPAs to avoid relying on a single TPA, but the approach does not prevent the user's identity from been leaked. While the schemes discussed so far (including that of Fu et al., 2017) achieve varying levels of security in User-CSP-TPA based auditing system, scalability is often overlooked.

Our work is fundamentally different from these early works, in the sense that we focus on both security and scalability in User-CSP-TPA based auditing systems. First, we make use of Hyperledger Fabric and dynamical TPA selection to achieve improved security. Second, we design an efficient auditing protocol based on bilinear pairing and commitment and the TPA selection algorithms to enhance scalability.

11. Conclusion

In order to address the security and scalability challenges in existing TPA-based data integrity auditing schemes, we proposed an efficient decentralized data integrity auditing scheme based on Hyperledger Fabric (i.e., HF-Audit). In HF-Audit, the Fabric network is used as a user-TPA-CSP communication platform. When combined with the TPA dynamic selection, we achieve improved security in the auditing system. In addition, we achieve improved scalability through the bilinear pairing and commitment based auditing protocol and the TPA selection algorithms under both complete and incomplete information settings. We also demonstrated the security and utility of the proposed approach.

Future work includes implementing a prototype of the proposed approach in a real-world cloud environment, which will allow us to evaluate its utility in practice.

Declaration of Competing Interest

No conflict of interest is declared.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Nos. 61601107 and U1708262); China Postdoctoral Science Foundation (No. 2019M653568); the Fundamental Research Funds for the Central Universities (No. N172304023). Yongxin Zhang is the co-first author of this work.

References

Al-Masri, E., Mahmoud, Q.H., 2007. Discovering the best web service. In: Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007. ACM, pp. 1257–1258. doi:10.1145/1242572.1242795.

Al-Masri, E., Mahmoud, Q.H., 2007. Qos-based discovery and ranking of web services. In: Proceedings of the 16th International Conference on Computer Communications and Networks, IEEE ICCCN 2007, Turtle Bay Resort, Honolulu, Hawaii, USA, August 13-16, 2007. IEEE, pp. 529–534. doi:10.1109/ICCCN.2007.4317873.

Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., et al., 2018. Hyperledger fabric: a distributed operating system for permissioned blockchains. In: Proceedings of the Thirteenth EuroSys Conference, EuroSys 2018, Porto, Portugal, April 23-26, 2018. ACM, pp. 30:1–30:15. doi:10.1145/3190508.3190538.

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 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007. ACM, pp. 598–609. doi:10.1145/1315245.1315318.

Au, M.H., Susilo, W., Mu, Y., 2006. Constant-size dynamic k-taa. In: Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings. Springer, pp. 111–125. doi:10.1007/978-3-540-28628-8_4.

Camenisch, J., Drijvers, M., Lehmann, A., 2016. Anonymous attestation using the strong diffie hellman assumption revisited. In: Trust and Trustworthy Computing - 9th International Conference, TRUST 2016, Vienna, Austria, August 29-30, 2016, Proceedings. Springer, pp. 1–20. doi:10.1007/978-3-319-45572-3_1.

Camenisch, J., Lysyanskaya, A., 2004. Signature schemes and anonymous credentials from bilinear maps. In: Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings. Springer, pp. 56–72. doi:10.1007/978-3-540-28628-8_4.

Camenisch, J., Mödersheim, S., Sommer, D., 2010. A formal model of identity mixer. In: Formal Methods for Industrial Critical Systems - 15th International Workshop, FMICS 2010, Antwerp, Belgium, September 20-21, 2010. Proceedings. Springer, pp. 198–214. doi:10.1007/978-3-642-15898-8_13.

Castro, M., Liskov, B., et al., 1999. Practical byzantine fault tolerance. In: Proceedings of the Third USENIX Symposium on Operating Systems Design and Implementation (OSDI), New Orleans, Louisiana, USA, February 22-25, 1999, 99, pp. 173–186.

Decker, C., Wattenhofer, R., 2014. Bitcoin transaction malleability and mtgox. In: Computer Security - ESORICS 2014 - 19th European Symposium on Research in Computer Security, Wroclaw, Poland, September 7-11, 2014. Proceedings, Part II. Springer, pp. 313–326. doi:10.1007/978-3-319-11212-1_18.

Dwork, C., Naor, M., 1992. Pricing via processing or combatting junk mail. In: Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings. Springer, pp. 139–147. doi:10.1007/3-540-48071-4_10.

Eyal, I., Sirer, E.G., 2018. Majority is not enough: Bitcoin mining is vulnerable. Commun. ACM 61 (7), 95–102. doi:10.1145/3212998.

Fu, A., Yu, S., Zhang, Y., Wang, H., Huang, C., 2017. Npp: a new privacy-aware public auditing scheme for cloud data sharing with group users. IEEE Trans. Big Data doi:10.1109/TBDATA.2017.2701347.

Hao, K., Xin, J., Wang, Z., Jiang, Z., Wang, G., 2018. Decentralized data integrity verification model in untrusted environment. In: Web and Big Data - Second International Joint Conference, APWeb-WAIM 2018, Macau, China, July 23-25, 2018, Proceedings, Part II. Springer, pp. 410–424. doi:10.1007/978-3-319-96893-3_31.

Huang, K., Xian, M., Fu, S., Liu, J., 2014. Securing the cloud storage audit service: defending against frame and collude attacks of third party auditor. IET Commun. 8 (12), 2106–2113. doi:10.1049/iet-com.2013.0898.

Jiang, Q., Ma, J., Wei, F., 2016. On the security of a privacy-aware authentication scheme for distributed mobile cloud computing services. IEEE Syst. J. 12 (2), 2039–2042. doi:10.1109/JSYST.2016.2574719.

Juels, A., Kaliski Jr, B.S., 2007. Pors: proofs of retrievability for large files. In: Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007. ACM, pp. 584–597. doi:10.1145/1315245.1315317.

Karame, G.O., Androulaki, E., Capkun, S., 2012. Double-spending fast payments in bitcoin. In: the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012. ACM, pp. 906–917. doi:10.1145/2382196.2382292.

King, S., Nadal, S., 2012. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake 19. Self-Published Paper, August.

Li, X., Peng, J., Niu, J., Wu, F., Liao, J., Choo, K.R., 2018. A robust and energy efficient authentication protocol for industrial internet of things.. IEEE Internet Things J. 5 (3), 1606–1615. doi:10.1109/JIOT.2017.2787800.

Liao, K., Zhao, Z., Doupé, A., Ahn, G.-J., 2016. Behind closed doors: measurement and analysis of cryptolocker ransoms in bitcoin. In: 2016 APWG Symposium on Electronic Crime Research, eCrime 2016, Toronto, ON, Canada, June 1-3, 2016. IEEE, pp. 1–13. doi:10.1109/ECRIME.2016.7487938.

Liu, B., Yu, X.L., Chen, S., Xu, X., Zhu, L., 2017. Blockchain based data integrity service framework for iot data. In: 2017 IEEE International Conference on Web Services, ICWS 2017, Honolulu, HI, USA, June 25-30, 2017. IEEE, pp. 468–475. doi:10.1109/ICWS.2017.54.

Nguyen, H.-L., Ignat, C.-L., Perrin, O., 2018. Trusternity: auditing transparent log server with blockchain. In: Companion of the The Web Conference 2018 on The Web Conference 2018, WWW 2018, Lyon, France, April 23-27, 2018. International World Wide Web Conferences Steering Committee, pp. 79–80. doi:10.1145/3184558.3186938.

Reid, F., Harrigan, M., 2013. An analysis of anonymity in the bitcoin system. In: Security and Privacy in Social Networks, New York, NY. Springer, pp. 197–223. doi:10.1007/978-1-4614-4139-_10.

Sebé, F., Domingo-Ferrer, J., Martinez-Balleste, A., Deswarte, Y., Quisquater, J.-J., 2008. Efficient remote data possession checking in critical information infrastructures. IEEE Trans. Knowl. Data Eng. 20 (8), 1034–1038. doi:10.1109/TKDE.2007.190647.

Suzuki, S., Murai, J., 2017. Blockchain as an audit-able communication channel. In: 41st IEEE Annual Computer Software and Applications Conference, COMPSAC 2017, Turin, Italy, July 4-8, 2017. Volume 2, 2. IEEE, pp. 516–522. doi:10.1109/COMPSAC.2017.72.

Tschorsch, F., Scheuermann, B., 2016. Bitcoin and beyond: a technical survey on decentralized digital currencies. IEEE Commun. Surv. Tut. 18 (3), 2084–2123. doi:10.1109/COMST.2016.2535718.

Veeningen, M., De Weger, B., Zannone, N., 2014. Data minimisation in communication protocols: a formal analysis framework and application to identity management. Int. J. InfSecur. 13 (6), 529–569. doi:10.1007/s10207-014-0235-z.

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. doi:10.1007/s11227-018-2527-y.

Wang, B., Li, B., Li, H., 2015. Panda: public auditing for shared data with efficient user revocation in the cloud. IEEE Trans. Serv. Comput. 8 (1), 92–106. doi:10.1109/TSC.2013.2295611.

Wang, B., Li, H., Li, M., 2013. Privacy-preserving public auditing for shared cloud data supporting group dynamics. In: Proceedings of IEEE International Conference on Communications, ICC 2013, Budapest, Hungary, June 9-13, 2013. IEEE, pp. 1946–1950. doi:10.1109/ICC.2013.6654808.

Wang, C., Chow, S.S., Wang, Q., Ren, K., Lou, W., 2013. Privacy-preserving public auditing for secure cloud storage. IEEE Trans. Comput. 62 (2), 362–375. doi:10.1109/TC.2011.245.

Wang, Q., Wang, C., Li, J., Ren, K., Lou, W., 2009. Enabling public verifiability and data dynamics for storage security in cloud computing. In: Computer Security - ESORICS 2009, 14th European Symposium on Research in Computer Security, Saint-Malo, France, September 21-23, 2009. Proceedings. Springer, pp. 355–370. doi:10.1007/978-3-642-04444-1_22.

Xu, J., Wang, S., Bhargava, B.K., Yang, F., 2019. A blockchain-enabled trustless crowd-intelligence ecosystem on mobile edge computing. IEEE Trans. Industr. Inf. 15 (6), 3538–3547. doi:10.1109/TII.2019.2896965.

Yang, Y., Lin, H., Liu, X., Guo, W., Zheng, X., Liu, Z., 2019. Blockchain-based verifiable multi-keyword ranked search on encrypted cloud with fair payment. IEEE Access. 7, 140818–140832. doi:10.1109/ACCESS.2019.2943356.

Yu, H., Yang, Z., Sinnott, R.O., 2019. Decentralized big data auditing for smart city environments leveraging blockchain technology. IEEE Access. 7, 6288–6296. doi:10.1109/ACCESS.2018.2888940.

Zhang, Y., Zheng, Z., Lyu, M.R., 2011. Wspred: a time-aware personalized qos prediction framework for web services. In: IEEE 22nd International Symposium on Software Reliability Engineering, ISSRE 2011, Hiroshima, Japan, November 29 - December 2, 2011, pp. 210–219. doi:10.1109/ISSRE.2011.17.

Ning Lu received the M.S. degree from the Chongqing University of Posts and Telecommunications, Chongqing, China, in 2009 and the Ph.D. from State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2013. He is now an associate professor in Northeastern University. His current research interests include artificial intelligence security, data security and privacy protection, Denial of Service attack defense.

Yongxin Zhang is a M.S. candidate in the School of computer and Communication Engineering, Northeastern University, Shenyang, China. His research interest is blockchain technology.

Wenbo Shi received the M.S. degree from the Inha University, Incheon, South Korea, in 2007 and the Ph.D. degree from the Inha University, Incheon, South Korea, in 2010. Currently he is a professor at Northeastern University at Qinhuangdao. His research interests include cryptographic protocol, cloud computing security, artificial intelligence security, data security and privacy protection, Denial of Service attack defense.

Saru Kumari is currently an Assistant Professor with the Department of Mathematics, Ch. Charan Singh University, Meerut, Uttar Pradesh, India. She received her Ph.D. degree in Mathematics in 2012 from CCS University, Meerut, UP, India. She has published more than 141 research papers in reputed International journals and conferences, including 122 publications in SCI-Indexed Journals. She is a reviewer of more than 50 reputed Journals including SCI-Indexed Journals of IEEE, Elsevier, Springer, Wiley etc. Her current research interests include information security and applied cryptography.

Kim-Kwang Raymond Choo currently holds the Cloud Technology Endowed Professorship at UTSA. In 2016, he was named the Cybersecurity Educator of the Year - APAC, and in 2015 he and his team won the Digital Forensics Research Challenge organized by Germany's University of Erlangen-Nuremberg. He is the recipient of the 2019 IEEE TCSC Award for Excellence in Scalable Computing (Middle Career Researcher), the 2018 UTSA College of Business Col. Jean Piccione and Lt. Col. Philip Piccione Endowed Research Award for Tenured Faculty, British Computer Society's 2019 Wilkes Award (Runner-up), 2019 EURASIP JWCN, IEEE TrustCom 2018 and ESORICS 2015 Best Paper Awards, Korea Information Processing Society's JIPS Survey Paper Award (Gold) 2019, Inscrypt 2019 Best Student Paper Award, 2014 Highly Commended Award by the Australia New Zealand Policing Advisory Agency, Fulbright Scholarship in 2009, 2008 Australia Day Achievement Medallion, and British Computer Society's Wilkes Award in 2008.