TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 04/14 pp28–43
DOI: 10.26599/TST.2019.9010025
Volume 25, Number 1, February 2020

A Novel Transparent and Auditable Fog-Assisted Cloud Storage with Compensation Mechanism

Donghyun Kim, Junggab Son, Daehee Seo, Yeojin Kim, Hyobin Kim, and Jung Taek Seo*

Abstract: This paper introduces a new fog-assisted cloud storage which can achieve much higher throughput compared to the traditional cloud-only storage architecture by reducing the traffics toward the cloud storage. The fog-storage service providers are transparency to end-users and therefore, no modification on the end-user devices is necessary. This new system is featured with (1) a stronger audit scheme which is naturally coupled with the proposed architecture and does not suffer from the replay attack and (2) a transparent and efficient compensation mechanism for the fog-storage service providers. We provide rigorous theoretical analysis on the correctness and soundness of the proposed system. To the best of our knowledge, this is the first paper to discuss about a storage data audit scheme for fog-assisted cloud storage as well as the compensation mechanism for the service providers of the fog-storage service providers.

Key words: fog computing; cloud computing; network storage; data audit; merkle hash tree; integer factorization; payment; transparency

1 Introduction

It is widely believed that the concept of Internet-of-Things (IoT) will play an important role in shaping our daily lives in the near future. IoT devices such as smartwatchs and smartphones are highly resource-limited due to the constraints on the size, weight, and/or cost, and thus heavily rely on cloud service providers. However, as the number of IoT device are rapidly growing, it is expected that in the following years, the demand from numerous end-user IoT devices toward the cloud service providers will exponentially grow, exacerbate the already high communication delay between end-user devices and the cloud service providers, and eventually drop Quality-of-Experience (QoE) of the IoT applications to an unacceptable level.

The notion of fog computing has been coined by Cisco in 2012 to challenge the limit of the state-of-the-art cloud computing paradigm. The core idea of fog computing is to migrate the functionalities of central cloud to network edge closer to end-users so that users' QoE can be improved and the demands toward cloud can be reduced. In many recent reports, fog computing environment is envisioned as a spatio-temporal network of end-user IoT devices, regional fog service providers, and back-end global cloud service provider. However, despite many notable efforts such as OpenFog Reference Architecture for fog computing by Open Fog Consortium, there is no firm consensus on how fog computing environment should be shaped as of today. Meanwhile, it is noteworthy that Software Defined Network (SDN) technologies are frequently

• Donghyun Kim, Junggab Son, Daehee Seo, and Yeojin Kim are with Department of Computer Science, Kennesaw State University, Marietta, GA 30060, USA. E-mail: {donghyun.kim, json4, dseo1}@kennesaw.edu; ykim89@students.kennesaw.edu.
• Hyobin Kim and Jung Taek Seo are with Department of Information Security Engineering, Soonchunhyang University, Asan 31538, South Korea. E-mail: gyqls1234@naver.com; seojt@sch.ac.kr.
• A part of this paper has been presented at 2018 IEEE International Conference on Communications (ICC 2018)[1] and 2018 IEEE International Workshop on Big Data Security and Services[2].
* To whom correspondence should be addressed.
Manuscript received: 2019-05-06; revised: 2019-05-16; accepted: 2019-05-28

incorporated into fog computing environment to realize various fog computing applications with transparency[3].

1.1 Data audit for fog-cloud storage service

Before the emergence of fog computing paradigm, various cloud services are utilized to overcome the limit of the resource constraint IoT devices. Cloud storage is one of the main services provided by the Cloud Service Provider (CSP) to allow end-users to overcome the storage capacity limit of their devices and/or share important data among geographically distributed clients. Data storage audit is an essential service for the users of network attached storage such as cloud storage and it aims to ensure the users that their data on the cloud is sound, i.e., data is well-stored in the storage and its content is exactly what the owner of the data expects. In addition, in case of any data loss or unauthorized alteration, the service will help to identify the responsible entity. As a result, data storage audit is a critical service accompanied by a commercial network storage service requiring higher level of reliability.

Recently, the potential of fog computing paradigm to implement various network storage services has been investigated in the literatures[4–6]. In this fog-node-front cloud-end storage model, or *fog-cloud storage* in short, the data storage demands from users are absorbed by local fog storage nodes if possible and the rest is accommodated by the central cloud. As fog-cloud storage is a network storage, it is desirable to provide a data storage audit service for it. However, it is not straightforward to adopt existing data storage audit schemes designed for central cloud storage to fog-cloud storage due to the following salient difference among them: In the traditional central cloud storage setting, a user explicitly deals with each CSP. Under the circumstance, as soon as the CSP receives data from a user and issues a undeniable receipt of the data to the user, the CSP will be responsible for keeping the data in its original form. On the other hand, in case of fog-cloud storage environment, the data from the user is stored among multiple fog nodes as well as the central cloud, whose operators can be independent from each other.

Furthermore, we believe that in order to minimize the modification on end-user devices, it is ideal to make the distribution of user data stored among the various network storages (provided by fog operators and cloud operator) and the generation of the receipt to be transparent to the user. This means that a data storage audit scheme for fog-cloud storage should be able to keep the track of the liability of each fog/cloud service provider without a user intervention. Clearly, this is a unique issue of fog-cloud storage environment and the existing data storage audit scheme for cloud storage cannot effectively handle.

To the best of our knowledge, however, no effort has been made so far to introduce a data storage audit scheme for fog-cloud storage service in the literature.

1.2 Compensation for fog-cloud storage service

Meanwhile, a fog-cloud storage framework with transparency and auditability cannot be used in practice if there is no practical compensation mechanism for the fog and cloud storage service providers. That is, unlike similar paradigms such as edge computing, in which the edge devices in the closed region (e.g., subnet) form a kind of a union and help each other without any explicit compensation, fog-cloud storage paradigm requires explicit compensation to each fog service operator for the services provided. As a result, there is a need of a new compensation framework for the fog-cloud storage architecture.

However, it is highly challenging to design a proper compensation mechanism for this architecture while maintaining the transparency of fog-storage service providers. That is, it is necessary to make each fog storage operator to be invisible from users, otherwise it will impose significant overheads to the users, e.g., keep the track of possibly temporal fog service providers. This also means that each user does not know which fog storage maintains which data of his/hers. One straightforward solution to address this issue is that the user provides the compensation to the cloud operator, which is the only visible entity to the user in this fog-cloud storage system. Then, the cloud operator distributes the compensation to each fog service provider proportional to their contributions to maintain the user's data. However, such approach may suffer from a critical issue that when the payment from the cloud to each fog operator is not delivered on time for some reason, the fog storage service may be suspended and the user, who paid all the required fee for the compensation, may suffer from undesired consequences, e.g., the data is not accessible to the user.

To the best of our knowledge, there is no work discussing about a payment mechanism for fog cloud storage system in the literature so far.

1.3 Summary of contributions

In order to address the issues discussed so far, this paper first introduces a new fog-cloud storage framework with transparency and auditability. On the top of the framework, we construct a transparent and efficient compensation mechanism for the fog-cloud storage service providers. The summary of our main contributions is as below.

(1) **New fog-cloud storage architecture with efficiency and transparency:** Our construction of the framework consists of three main components: end-user IoT devices, SDN-compatible fog routers with regional fog storages, and global/central cloud storages (See Fig. 1).

In our framework, an end-user IoT device submits an upload request with a set of data blocks, to the fog-cloud storage. Each upload request including data blocks is being routed from the end-user IoT device toward the central cloud storage throughout network routers, some of which are SDN-compatible fog routers. Each of these SDN-compatible fog router serves as a clusterhead of the regional fog storages. For each request that it receives, depending on available storage space, an SDN-compatible fog router may accommodate some or all of the data blocks in the upload request and forward the request with the remaining (unaccommodated so far by and fog storages) data block to the next hop SDN-compatible fog router. The request will eventually arrive at the cloud storage and any remaining data blocks will be stored at the cloud storage.

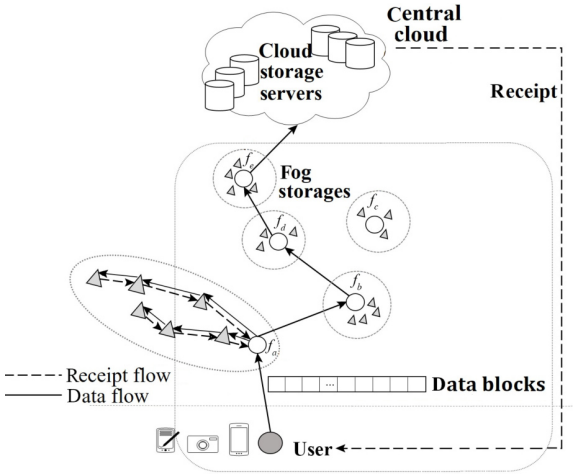


Fig. 1 Fog-storage environment is composed of five SDN-compatible fog routers with regional fog storages F=$\{f_a, f_b, f_c, f_d, f_e\}$ and a central cloud. Only four of them provide storage service to the user u_1, so the routing path from u_1 to *cloud* becomes $\mathcal{P}_1$=$u_1 \rightarrow f_a^1 \rightarrow f_b^1 \rightarrow f_d^1 \rightarrow f_e^1 \rightarrow$ *cloud* if f_c passes forward the data blocks to the next available fog router f_d without storing any subset of M, where M represents a data message. Suppose that the user's data storage request is managed by only SDN-compatible fog routers, and the central cloud does not store any subset of M. Then the central cloud will only keep the receipt and prove that the data integrity.

There are several benefits of this framework. Most of all, given that the capacity of regional fog storages is sufficient, our framework is capable of reducing (a) the amount of data blocks toward the central cloud storage greatly, (b) the heavy traffics on the bottleneck routers nearby the cloud storage, and consequently (c) from the end-IoT device point of view, the time between submitting the request and obtaining the confirmation from the central cloud.

Second, in our framework, users do not need to know the existence of fog infrastructure, but only need to know the central cloud storage, and therefore the proposed storage framework provides a flawless transparency to the user.

(2) **Efficient collection of receipt of data from fog-cloud for user assurance:** In the proposed framework, while the upload request with data blocks of a user is being forwarded from an end-user IoT device to the back-end cloud storage, the head of each fog storage cluster (an SDN-compatible fog router) on the routing path between them can either decide to store some of the data blocks inside fog storages within its own cluster. If so, the cluster head issues a receipt for each of the data blocks, and adds this receipts to the upload request, and forwards the upload request which now includes the rest of data blocks and receipts toward the next hop. Otherwise, the whole upload request will be forwarded to the next hop without any modification.

When the upload request arrives at the central cloud, the request may or may not contain any data blocks. It is important to notice that there exists a receipt from a fog clusterhead for each missing data block in the upload request. At this point, any data block left in the upload request will be stored at the central cloud. Then, the central cloud will issue an accumulated receipt based on all the receipts in the upload request as well as any receipt that it generates and send the accumulated receipt to the user. After the cryptographically constructed accumulated receipt is obtained, the user can easily verify its validity, and may delete its local copy if desired without worrying

any data loss.

(3) **Efficient verification of data with accountability for lost data:** After the initial verification of accumulated receipt, a user may choose to delete its local copy of the data blocks. Then, the user may check the soundness of the data on the fog-cloud later. Generally speaking, without the original data, it is difficult for the user to check the soundness of the data in the fog-cloud storage against the replay attack of a fog storage or cloud storage. This is because each storage may keep the receipt which was generated during the initial setup and be able to reuse it. On the other hand, the user is disadvantaged that as it does not have the original data, it is difficult to check the freshness of the receipt from the storages.

To resolve this issue, we introduce a new scheme to regenerate a fresh challenge based on the initial receipt of the data blocks without the actual copy by exploiting integer factorization. As a result, each storage will be able to generate a valid response with respect to the fresh challenge only if the data blocks with it are sound. At the end of the verification process, the user will receive a single response from the cloud and the transparency will be still maintained. In case that the response is corrupted and the data block is not sound, our framework will reveal which storages are responsible for the loss along with undeniable proofs. As a result, our scheme is proper to the fog-cloud storage environment which may be operated by multiple independent authority.

(4) **A lightweight and transparent compensation mechanism for the proposed fog-cloud storage framework:** We also introduce an efficient compensation mechanism for our fog-cloud storage service providers in our framework while maintaining transparency. The main merits of the proposed compensation mechanism are (a) *Low computation overhead.* The proposed compensation scheme is tightly coupled with our fog-cloud storage framework and the operations at each end-user device are highly light-weighted, e.g., cryptographic hash function and symmetric encryption; (b) *Low communication overhead.* In the proposed scheme, each user only needs to upload one message to send the payment to the central cloud operator and receive another message (the receipt of the payment) from the central cloud operator; and (c) *Transparency.* Our payment scheme does not require the user to have any knowledge over the underlying fog storage infrastructure, and each user only needs to know the central cloud operator.

1.4 Organization of this paper

The rest of this paper is organized as follows. Section 2 discusses related work. We introduce the details of our fog-cloud storage architecture and the audit scheme in Section 3. Section 4 discusses about our new compensation mechanism for fog-cloud storage service providers. Finally, we conclude this paper in Section 5.

2 Related Work

Over years, the concept of cloud computing has attracted much attentions and many efforts are made to make it more useful and practical[7–11]. Fog computing is an emerging network technology to overcome the limit of popular cloud computing technology and attracts much interest from both academia and industry very recently. As the fog computing technology is immature, only a handful of them discuss about the architecture of fog system[12–15] and relevant security issues[7, 16, 17]. As the research on the fog computing is still in its early stage, most of papers in fog computing architecture are struggling to define the detailed structure and relevant performance metrics based on Cisco's original proposal for fog computing structure[18], which has a three-tiered structure of a peer-to-peer network of IoT devices, an intermediate fog layer between IoT devices and cloud servers, and an established cloud structure. In addition to the discussion on the details of fog computing architecture, the authors in Refs. [13, 15] offered a new workload allocation mechanism which offers tradeoff among power consumption, service latency, CO_2 emission, etc. On the other hand, the authors in Refs. [7, 16, 17] have conduced a survey on the security concerns and challenges in fog computing. Alrawais et al.[16] introduced their case study on certification revocation issue in IoT using fog computing environment. However, none of these work has discussed about the significance of data audit issue in fog-cloud storage environment. To the best of our knowledge, data audit issue has never been discussed in the context of fog-cloud storage while some relevant results have been reported in the context of cloud storage.

So far, various cloud storage auditing approaches have been discussed in Refs. [19–26]. However, these existing audit schemes have several problems, e.g., Third-Party Auditor (TPA) may leak the data or the number of audit is limited. In Ref. [20], Yang and

Jia provided a security analysis on those existing audit schemes in terms of storage overhead and communication cost. Their report shows that these schemes are suffering from a common significant issue that the amount of metadata to sustain the schemes will grow unmanageably as the systems operate over time. At the same time, all of these schemes require frequent communications among users and TPA, therefore suffer from large communication overhead. The more recent works on cloud storage auditing[19, 22, 24, 26] have utilized the famous Merkle Hash Tree (MHT) to introduce a superior audit scheme for the cloud storage which incurs much less metadata related storage overhead and communication overhead than existing ones.

Recently, Liu et al.[19, 24] proposed a multi-replica MHT for each data block in which the cloud storage operator is notified if any of replica fails. In this way, the communication overhead for verification is alleviated and user's data is more fault-resilient from storage service flaw. Especially, the Multi-Replica Dynamic Public Auditing (MuR-DPA) scheme in Ref. [19] complements the problem that user should store and generate each different MHT for every replica so that it brings severe communication overheads. However, it is not straightforward to apply these audit schemes for cloud storage to fog-cloud storage directly due to the unique nature of fog-cloud storage such as multiple operating parties. Most importantly, all of the existing MHT-based audit schemes suffer from a replay attack; once a storage operator computed a hash value, it can be reused to prove the soundness of the data even though the data is altered or lost.

3 A New Fog-Cloud Storage Framework with Transparency and Auditability

3.1 Proposed fog-cloud storage system model

The representative network architecture for the fog-cloud data storage is illustrated in Fig. 1. There are three main network entities:

(1) End-user IoT devices $U = \{u_1, u_2, \ldots, u_{|U|}\}$,

(2) SDN-compatible fog routers $F = \{f_a, f_b, \ldots, f_{|F|}\}$, and

(3) The global cloud storage *cloud*.

Furthermore, each SDN-compatible fog router $f_i \in F$ serves as a clusterhead of a set of regional fog storages $FS(f_i)$. In this fog-cloud storage framework, we assume the gateway router of each user submitting a data message M is the first SDN-compatible fog router, f_a, which is typically a gateway router of the subnet with the user. Suppose that $\mathcal{P}_\ell = u_\ell \rightarrow f_a^\ell \rightarrow f_b^\ell \rightarrow \cdots \rightarrow f_{|F|}^\ell \rightarrow cloud$ is the routing path from a user u_ℓ to the central cloud when all SDN-incompatible fog routers are ignored. In order to simplify our discussion, we assume that M consists of k message blocks with uniform length, i.e., $M = \{m_1, m_2, \ldots, m_k\}$, for some positive integer k. After M is submitted and moves over the path $\mathcal{P}_\ell$, M will be absorbed by some fog storage clusters and/or the central cloud storage. Suppose that $M(x)$ is a subset of message blocks in M stored by a storage $x \in F \bigcup \{cloud\}$. Then, $\forall x, y \in F \bigcup \{cloud\}$ such that $x \neq y$, we have $M(x) \bigcap M(y) = \varnothing$. For each $m_i \in M$, $\exists x \in F \bigcup \{cloud\}$ such that $m_i \in M(x)$.

This paper assumes that each fog storage cluster has a limited storage capacity and the central cloud storage has a unlimited capacity. This means that while M moves over $\mathcal{P}_\ell$, M will be completely absorbed by fog storage clusters, and the central cloud may not store any subset of M. Even in such case, an empty storage request (otherwise non-empty) will be forwarded the next-hop toward the central cloud. Once cloud receives the request, it will go to user.

3.2 Proposed fog-cloud storage audit scheme

3.2.1 Preliminaries

In fog-cloud storage processing, we introduce two auditing modes:

(1) *Initial audit mode*, when a user stores data at the first time, he/she can verify data integrity and decides to delete original data file based on the verification, and

(2) *Post-initial audit mode*, when user retrieves user's data after deleting the original data, he/she can verify data integrity.

An MHT is similar to the complete binary tree. However, it is constructed by hashing paired data based on a one-way cryptographic hash function. The leaf nodes have hash value of the data blocks from user, and non-leaf nodes have the hash value of its child nodes. The root value of the MHT can be computed through hash values and signatures from Fog-cloud Storage Service Provider (FSSP). In order to support the verification of data uploads, this authenticated data structure is utilized in the initial audit mode, and Fig. 2 shows the example of the MHT based on the 15 data blocks.

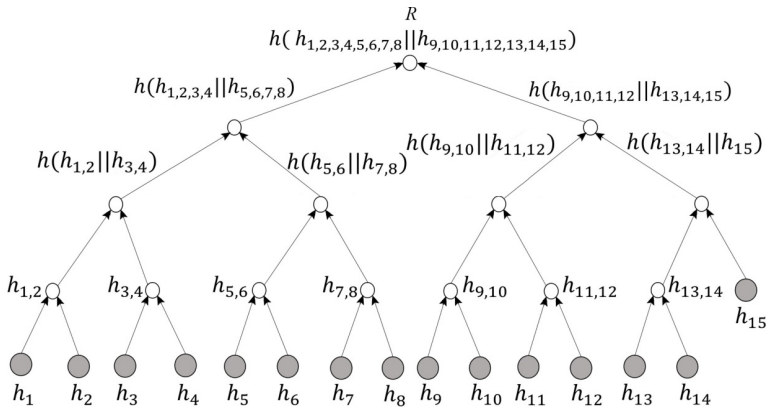


Fig. 2 An example of MHT. $h_i = h(m_i)$, $h_{i,i+1} = h(h_i||h_{i+1})$, and $h(h_{i,i+1}||h_{i+2,i+3})$=$h(h(h_i||\ h_{i+1})||h(h_{i+2}||h_{i+3}))$, where h is a hash function that satisfies h: $\{0,1\}^* \rightarrow \{0,1\}^n$.

3.2.2 Design goal

Our fog-cloud storage auditing design goals can be summarized as follows.

Fog-cloud storage auditability. User audits if the data stored correctly through two modes, initial audit mode and post-initial audit mode. At the first time user uploads the data file to the fog-cloud storage, an MHT is constructed according to the number of data blocks. Each SDN-compatible fog router, which is on the routing path $\mathcal{P}_\ell$, absorbs data blocks from a user. Every time that the FSSP stores user's data block, the hash function operates for each node in MHT. Then user can verify whether the FSSPs store user's data correctly or not by comparing the root value R of the MHT between computed value of R and proved value of R by *cloud*. Based on the initial audit verification, user can delete original data file. Without the original data block, user can verify the data integrity through the post-initial audit mode. Based on the integer factorization, user challenges the central cloud with modular equations. Central cloud computes the linear congruences with the user's challenge k coprimes and responses the results to the user. Then, user creates another linear congruences for the verification equations with the challenge k coprimes and the cloud's responses. Finally, user compares those equations. If they are correct, then the verification of data integrity is proved.

Fog-cloud storage accountability. In this fog-cloud storage architecture, the auditing scheme does not only verify the data integrity but also allows to trace all the participating SDN-compatible fog routers. If the auditing scheme is aware of any fault in storage processing, it records the rogue fog cluster based on the receipt tuple which is generated by each SDN-compatible fog router. Those discovered rogue fog clusters are prohibited to participate in storage service again later on.

3.2.3 Our construction

Unlike the cloud storage environment, fog-cloud storage environment composed of much more storage service providers. Thus, in the fog-cloud architecture, it requires that each fog cluster to be verified for data integrity respectively. To maintain not only the efficiency of using fog-cloud storage over using conventional central cloud storage service but also preserving privacy and secure storage service, we show how the existing cloud audit scheme should be improved to apply properly to the fog-cloud storage environment. The procedure of our audit scheme is as follows.

Initial audit mode. Figure 3 shows the procedure of initial audit mode for $\mathcal{P}_1 = u_1 \rightarrow f_a^1 \rightarrow f_b^1 \rightarrow f_c^1 \rightarrow f_d^1 \rightarrow cloud$. In our MHT structure, each time the FSSP provides storage service, a $Receipt_{f_i}$ is generated. For example, in Fig. 3, the receipt will be updated total four times by the SDN-compatible fog routers f_a, f_b, f_c, and f_d. The $Receipt_{f_i}$ is composed of 3-tuple of $(\sigma_i, Sh_i, M_{rem})$ where σ_i represents data blocks which are stored by f_i in the current fog storage cluster, Sh_i is a Signing hash value of σ_i, and M_{rem} is a set of the leftover data blocks.

The signing hash, Sh_i in the receipt tuple, will be generated by following Algorithm 1. $GenSh(T.current, sk_i)$ generates Sh_i which is corresponding to the hash values of the stored data blocks by f_i. With two input values, *T.current* as a tree on the current node *current* and sk_i as a secret key of f_i, the function will return boolean value to

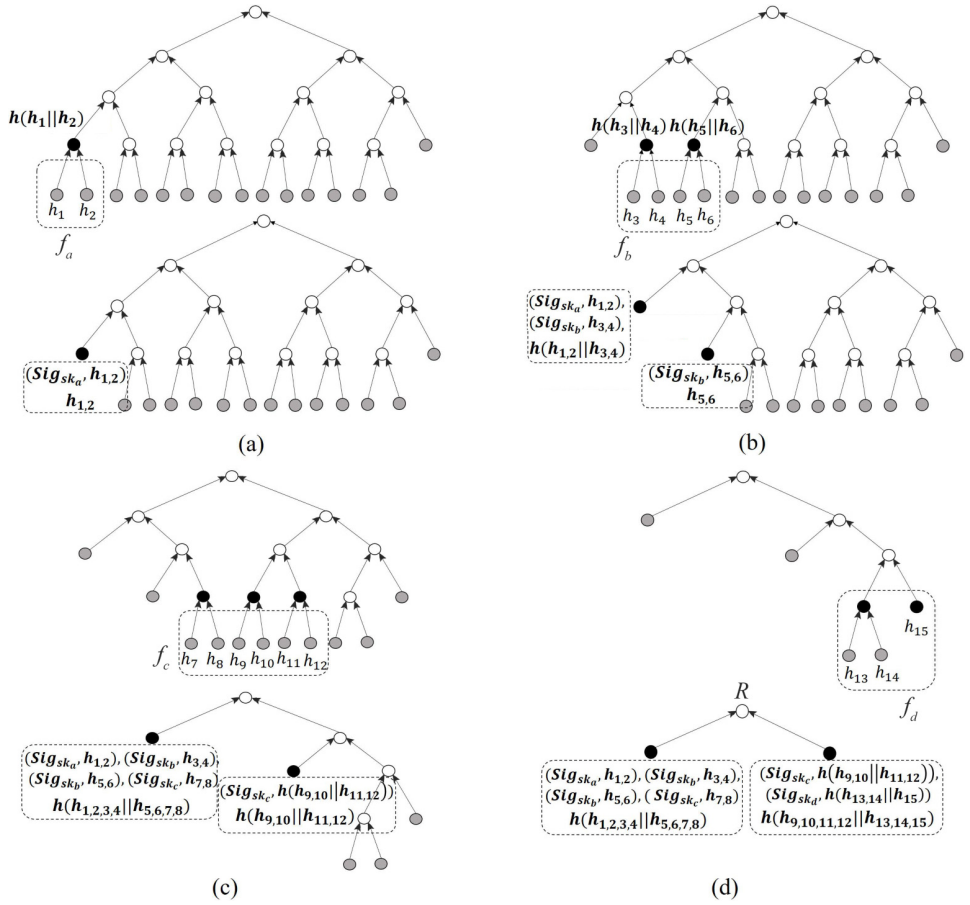


Fig. 3 Reconstruct *R*. (a) Re-constructed MHT by f_a, (b) re-constructed MHT by f_b, (c) re-constructed MHT by f_c, and (d) re-constructed MHT by f_d.

generate Signing hash for the subtree which is taken by each SDN-compatible fog router. Since the hash value can be computed only when the child nodes belong to the same parent node, if fog router absorbs n leaf nodes and partial nodes belong to the other parent node, Algorithm 1 will generate a Sig_{sk_i} for the child nodes which have same parent, and will return $False$ for the Signing hash from the subtree.

Based on the original MHT in Fig. 2, suppose that f_a, f_b, f_c, and f_d store 2, 4, 6, and 3 data blocks in M, respectively. Figure 3a shows how the first SDN-compatible fog router f_a stores two data blocks m_1 and m_2, and generates a $Receipt_{f_a}$. The *GenSh* generates Sig_{sk_a} which is corresponding to the hash values h_1 and h_2. Since two leaf nodes m_1 and m_2 have same parent node, the *GenSh* returns *True* from the first if statement, and Sh_a is generated by f_a. The first receipt tuple is generated as $(\sigma_a, Sh_a, M_{rem})$ where $\sigma_a = \{m_1, m_2\}$, $Sh_a = (Sig_{sk_a}, h_{1,2})$, and $M_{rem} = \{m_3, m_4, \ldots, m_{15}\}$. The MHT is reconstructed by f_a as illustrated in the bottom MHT of Fig. 3a. Next, second SDN-compatible fog router f_b stores four data blocks m_3, m_4, m_5, and m_6. However, m_3, m_4, and m_5, m_6 belong to two different parent nodes. Thus, this time the *GenSh* returns *False* from the first *if* statement, and next *retValL* will be *True* and *retValR* will be *False*. First two hash values h_3 and h_4, from m_3 and m_4, will generate $h(h_3||h_4)$, then with this hash value its parent node will have new hash $h_{3,4}$ with Sig_{sk_b}. Last two hash values h_5 and h_6, from m_5 and m_6, will generate Sh_b. The new updated receipt tuple $Receipt_{f_b} = (\sigma_b, (Sh_{a,b}, Sh_b), M_{rem})$, where

$$\sigma_b = \{m_3, m_4, m_5, m_6\},$$
$$Sh_{a,b} = (((Sig_{sk_a}, h_{1,2}), (Sig_{sk_b}, h_{3,4})), h(h_{1,2}||h_{3,4})),$$
$$Sh_b = (Sig_{sk_b}, h_{5,6}),$$
$$M_{rem} = \{m_7, m_8, \ldots, m_{15}\}.$$

The MHT is reconstructed by f_b as illustrated in the bottom MHT of Fig. 3b. Third SDN-compatible fog router f_c stores six data blocks $m_7, m_8, \ldots, m_{12}$, and generates a $Receipt_{f_c}$. The *GenSh* generates $Sh_{a,b,c}$ and Sh_c for the MHT by f_c. The new updated receipt tuple

Algorithm 1 Generation of Sh_i in receipt tuple

GenSh($T.current, sk_i$);
Input: a hash tree T of the current node *current*, and a private key sk_i of the fog router f_i
Output: *True* or *False*
begin
  if *(*current == leaf*) && (*current == owned*)* **then**
    return *True*;
  else if *(*current == ownedByOthers*)* **then**
    return *False*;
  retValL= *GenSh*($T.current(left), sk_i$);
  retValR= *GenSh*($T.current(right), sk_i$);
  if *(*retValL == True*) && (*retValR == True*)* **then**
    return *True*;
  else if *(*retValL == True*) && (*retValR == False*)* **then**
    generate a Sig_{sk_j} for left subtree with *current*(*left*) as the root;
    return *False*;
  else if *(*retValL == False*) && (*retValR == True*)* **then**
    generate a Sig_{sk_j} for right subtree with *current*(*right*) as the root;
    return *False*;
  else
    return *False*;
end

is $(\sigma_c, (Sh_{a,b,c}, Sh_c), M_{rem})$ where

$$\sigma_c = \{m_7, m_8, \ldots, m_{12}\},$$
$$Sh_{a,b,c} = (((Sh_{a,b}), (Sig_{sk_b}, h_{5,6}), (Sig_{sk_c}, h_{7,8})), h(h_{1,2,3,4}||h_{5,6,7,8})),$$
$$Sh_c = (Sig_{sk_c}, h(h_{9,10}||h_{11,12})),$$
$$M_{rem} = \{m_{13}, m_{14}, m_{15}\}.$$

The MHT is reconstructed by f_c as illustrated in the bottom MHT of Fig. 3c. The last SDN-compatible fog router f_d stores the remaining three data blocks m_{13}, m_{14}, and m_{15}. The new updated receipt tuple, $Receipt_{f_d}$, is $(\sigma_d, (Sh_{a,b,c}, Sh_{c,d}), M_{rem})$ where $\sigma_d = \{m_{13}, m_{14}, m_{15}\}$, $Sh_{a,b,c}$ is same as the previous receipt tuple, $Sh_{c,d} = (((Sh_c), (Sig_{sk_d}, h(h_{13,14,15}))), h(h_{9,10,11,12}||h_{13,14,15}))$, and $M_{rem} = \varnothing$. The MHT is reconstructed by f_d as illustrated in the bottom MHT of Fig. 3d.

The *cloud* will have final receipt from previous fog routers. User can verify if his/her data stored correctly without any changes by comparing the computed root of MHT and proof of R which is provided by the cloud. If the value is not same, the data is not soundness, then the central cloud detects the accountability of participated fog storage cluster by audit all the Signing hash in the receipt tuple.

Post-initial audit mode. The initial audit mode could efficiently and effectively provide recepits for users' data. However, after the initail step, the FSSP could bypass the audit process by simply storing all the hash values of users' data. To address this problem, we propose the post-initial audit mode. Although this auditing process will be described in a user's point of view, it can be periodically performed by a TPA[22] to lessen burden of users and to prevent users' misbehaviors.

After user verifies the data integrity through the initial audit mode, user decides whether deleting the local data file or not. Without the original data file, user still must verify the data integrity and central cloud should not be able to deceive user by replay attack. For this issue, the post-initial audit mode provides that the user can verify by computing reasonable number of linear congruence equations selectively. Before user deletes the original data, k linear congruence equations are generated as follows: $m_1 \bmod p_1 = x_1$, $m_2 \bmod p_2 = x_2, \ldots, m_k \bmod p_k = x_k$. We assume that the p_is are RSA moduli with safe primes[27].

In these equations, $m_1, m_2, \ldots, m_k$ represent the original data blocks user has stored, and $p_1, p_2, \ldots, p_k$ represent the pairwise coprimes which are used to compute a remainder for each data block. To improve the storage efficiency of users, p_i can be replaced to an index of the prime table. Now, the user is able to delete original data file after keep the values of $x_1, x_2, \ldots, x_k$, the corresponding indices of the prime table.

Now without the original data, user can audit for his/her data soundness through four steps: (1) Challenge, (2) Compute, (3) Response, and (4) Verify (See Fig. 4). In order to challenge the central cloud, user chooses three or four data blocks' indices as the challenge parameters, and then generates $\alpha = (p_i \times p_j \times \cdots \times p_n) \cdot c$, where c is a randomly generated prime

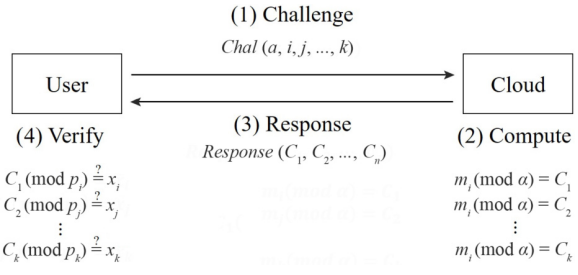


Fig. 4 Challenge and response communication in post-initial audit mode.

number used to hide the block primes, and $i, j, \ldots, n$ indicate the indices of data blocks. Then the user challenges the central cloud with $Chal(\alpha, i, j, \ldots, n)$. Since the value of α becomes larger exponentially as user chooses more challenge primes, three or four number of challenge parameters are preferred to be selected for a single challenge. This minor weakness can easily be overcomed by repeating the auditing process or sending multiple challenges.

Suppose that the user challenges with four message blocks, m_1, m_3, m_4, and m_5. Upon receiving the challenge $Chal(\alpha, 1, 3, 4, 5)$ from a verifier, the prover who should retain the original data including m_1, m_3, m_4, and m_5 will compute as following equations: $m_1 \bmod \alpha = C_1$, $m_3 \bmod \alpha = C_2$, $m_4 \bmod \alpha = C_3$, and $m_5 \bmod \alpha = C_4$. Next, the prover sends the computation results $Response(C_1, C_2, C_3, C_4)$ to the verifier.

Now, user can verify the data integrity through comparing following verification equations: $C_1 \bmod p_1 \stackrel{?}{=} x_1$, $C_2 \bmod p_3 \stackrel{?}{=} x_3$, $C_3 \bmod p_4 \stackrel{?}{=} x_4$, and $C_4 \bmod p_5 \stackrel{?}{=} x_5$. If these verification equations have same solution to x_1, x_3, x_4, and x_5, then it is proved that user's data is possessed correctly in the fog-cloud storage. If any of those verification equations have incorrect solution, the verification is failed. In this case, user should be able to demand the accountability of participated fog storage cluster.

Fog-cloud storage accountability. Based on the result from two audit modes, the fog-cloud storage scheme provides an accountability to manage the rogue FSSP. Accountability in the fog-cloud storage environment is necessary due to lots of fog participants provide the storage service. In both initial and post-initial audit modes, if the soundness of data is broken, the central cloud is not possible to create the right response for the verification proof.

For the *initial audit mode*, since the user obtains a final receipt from the central cloud, R can be computed with $Sh_i = (Sig_{sk}, h_i)$ and user verifies the Sig_{sk} for each data block. Then, user compares R with cloud's response. User will trust the data stored correctly if the verifications passed. Otherwise, the user makes a request for finding data blocks that have errors, and the FSSP investigates the user's data depending on the *Receipts* received from nodes. The user sends the data block and performs the initial audit mode again. Once the verification is proved through the initial audit service, deleting the user's original data is safe.

For the *post-initial audit mode*, during the user challenges the central cloud, in order to make modular equations, more than one $p \in P$ are used as moduli corresponding to the indices of data block which is selected by user. User will trust that the retrieved data is genuine if all the selective modular equations have correct answers, otherwise user detects the index number of the data blocks from the modular equation which had incorrect answer. For example, suppose that the user challenges central cloud with three challenge parameters, $Chal(\alpha, 2, 5, 6)$, and one of the verification equations is not correct such as $C_1 \bmod p_2 = x_2$, $C_2 \bmod p_5 = x_5$, and $C_3 \bmod p_6 \neq x_6$, then the user can detect which data block makes to fail the audit verification. In this situation, user reports the data block, m_6, to the central cloud. Through this report, the central cloud can track which fog cluster stores m_6, and the SDN-compatible fog router in the detected fog cluster will hold the ultimately accountable to the data integrity violation.

3.3 Security analysis of the proposed audit schemes

This section provides security analyses of the proposed fog-cloud storage audit scheme. Specifically, we prove that the FSSP cannot generate a correct proof which will pass the auditing process without possession of original data. The security of our auditing scheme is based on one-wayness of a cryptographic hash function, unforgeability of a digital signature scheme, and difficulty of integer factorization.

Theorem 1 Suppose the proposed scheme is developed with the one-way hash function and the unforgeable digital signature scheme. Then, it is computationally impossible for the FSSP to generate a valid proof without absorbing whole original data in polynomial time, which can pass the initial audit mode.

Proof As the root of MHT is computed based on all of the data blocks, there are two ways for the FSSP to pass the initial audit mode: (1) it computes a receipt with the actual data blocks or (2) it uses unjust way to compute the root of MHT without data blocks, where (2) can be covered by the following cases: (a) **Dishonesty of fog nodes**. A node may pretend to absorb more data blocks than it actually did, which brings more profit to the node. However, this case can be easily detected by FSSP as all SDN-compatible fog routers must generate a signature for their absorption. If the node cannot generate a forged signature, it also

cannot repudiate the generated signature. In addition, the dishonesty will not pass the post-initial audit mode without actual data blocks. (b) **Dishonesty of FSSP.** The FSSP may try to generate a correct root without original data blocks upon data loss or other unintended errors. All that the FSSP can do for this case is guessing the root depending on partial information. Although the success probability is extremely low, the FSSP might pick a correct value for the root. Unfortunately, it is computationally infeasible to pass the post-initial audit mode, and thus, the FSSP will fail to pass latter auditing. We will provide security analysis for the post-initial audit mode in *Theorem* 2. ■

Definition 1 (Integer factorization problem[28]) Suppose there exists a polynomial-time algorithm, named *GenModulus*, which takes 1^k as an input and outputs (N, p, q) such that $N = pq$, p and q are k-bit prime, and k is a security parameter. Then, we can say integer factorization is hard relative to the *GenModulus* if for all probabilistic polynomial time algorithm $\mathcal{A}$, the following probability is negligible:

$$Pr[GenModulus(1^k) \rightarrow (N, p, q); \\ \mathcal{A}(N) \rightarrow (p, q) : N = pq].$$

Theorem 2 If there exists an oracle that can break the post-initial audit mode, then it can also break RSA public key cryptosystem.

Proof If a user generates a challenge with only two primes (one is a data block prime and the other is randomly chosen prime), the security of the proposed scheme depends on *Definition* 1. However, since the proposed scheme assumed to use up to four primes for a single challenge, we prove that it is still secure. Let $\mathcal{O}$ be an oracle that can break the post-initial audit mode. It takes a large composite number N as an input and outputs four primes p_1, p_2, p_3, and p_4. We will show that $\mathcal{O}$ can also break RSA system. Suppose a user A's private/public key pair was generated based on sufficiently large number $N_A, N_A = q_1 \cdot q_2$, and another user B's private/public key pair was generated based on sufficiently large number $N_B, N_B = q_3 \cdot q_4$. Then, a malicious adversary captures the two numbers and sends a query to $\mathcal{O}$ including $N_A \cdot N_B$ for the input. The $\mathcal{O}$ will return four prime numbers, and finally, the adversary will be able to compute private keys of both the user A and the user B. Therefore, we can say that the proposed scheme is as secure as RSA cryptosystem by reduction. ■

4 A Lightweight and Transparent Compensation Mechanism for Fog-Cloud Storage

In this chapter, we introduce a new compensation mechanism for the proposed fog-cloud storage framework that we have proposed in the previous chapter.

4.1 Proposed compensation mechanism

Once data is uploaded to the fog-cloud storage in our fog-cloud storage framework, a proper compensation has to be paid within every interval for the continuity of the service. The proposed compensation scheme consists of the following three distinct phases: (1) compensation preparation, (2) compensation distribution, and (3) compensation acknowledgement. In the followings, we will explain each phase in detail.

4.1.1 Compensation preparation

Once the data is uploaded, the user prepares the following for the payment. In detail, for each message block m_i, the user computes a key $key_i = H(m_i)$, where H is a cryptographic hash function. Consider $K = \{key_1, key_2, \ldots, key_n\}$ is the key sequence computed in this way. Then, the user prepares n encrypted payment coupons $P = \{p_1, p_2, \ldots, p_n\}$ for the message blocks, where p_i is a combination of (1) an integer i (corresponding message block id), and (2) a compensation ticket CT_i and a random number q_i encrypted by a symmetric encryption scheme such as AES with the key key_i. The philosophy of this design is that once p_i is given to a router which has witnessed m_i, it will be able to extract CT_i and q_i with the key $key_i = H(m_i)$. Then, the router will be able to use CT_i to deposit the payment, while it can use q_i to report back to the user that the payment is correctly received.

4.1.2 Compensation distribution

After the compensation preparation phase, the user sends P to the central cloud. Starting from the cloud, the payment coupon is distributed in the reverse order of the routers appeared in R, which is the order of routers over which the message M is delivered toward the central cloud. For each router r_i (starting from $i = c$), it will take the payment tuples corresponding to the blocks stored in the regional storage $C(r_i)$. For instance, if r_i has m_{j-1} and m_j, p_{j-1} and p_j will be taken by r_i (its operator). Then, the rest of payment

tuples in P will be forwarded to the next router r_{i-1}. Figure 5 illustrates the processes of uploading the data blocks M and payment coupons P.

4.1.3 Compensation acknowledgement

For each router r_i in the reverse order of the appearance in R, it decrypts corresponding payment tuples and obtains random variables inside the tuples. For example, suppose r_c decrypted (CT_{n-2}, y_{n-2}), (CT_{n-1}, y_{n-1}), and (CT_n, y_n). Then, it first computes the first block of the hash chain $HC_n = H(y_n)$. Then, the second block of the hash chain $HC_{n-1} = H(y_{n-1}, HC_n)$ is computed. Finally, it computes the third block of the hash chain $HC_{n-2} = H(y_{n-2}, HC_{n-1})$. Then, HC_{n-2} is passed to the next hop router in the reverse sequence, r_{c-1}. After this process is recursively executed, r_1 will have HC_1, and then pass it to r_c. Finally, r_c passes to the user. As the user has the knowledge of P as well as all random variables in the payment coupons $y_1, y_2, \ldots, y_n$, it will be able to compute HC_1 by itself and compare it with what is received from r_c. If both match, this indicates that all payment coupons are successfully received.

4.1.4 Merit of proposed system

In this subsection, we discuss three major merits of the proposed scheme: (1) low computation overhead, (2) low communication overhead, and (3) transparency to end-user devices.

Low computation overhead at user side. For the compensation distribution, the user needs to compute a hash value for each of n messages and encrypt a payment coupon for the message with the hased value. For the verification of acknowledgement, it needs to compute a hash chain of the n random values. As a result, the whole process requires $2n$ hash function computations and n symmetric encryptions. As both computations are not heavy, the proposed system incurs low computation overhead at the user terminal device.

Low communication overhead. In the proposed scheme, in addition to upload the n message blocks, the user needs to upload the set P of the n payment coupons and download a single hash chain value to verify the correct receipt of the payment coupons. As a result, the whole process requires only $n + 1$ message exchanges other than the upload of the n message blocks. Therefore, the proposed system does cause low communication overhead.

Transparency. The user uploads the message blocks toward the central cloud, and during the course of routing of the blocks toward the central cloud, each regional fog cluster absorbs some part of the blocks autonomously, and the user receives the confirmation of the receipt of stored data from the cloud. On the other hand, the payment coupons go directly to the

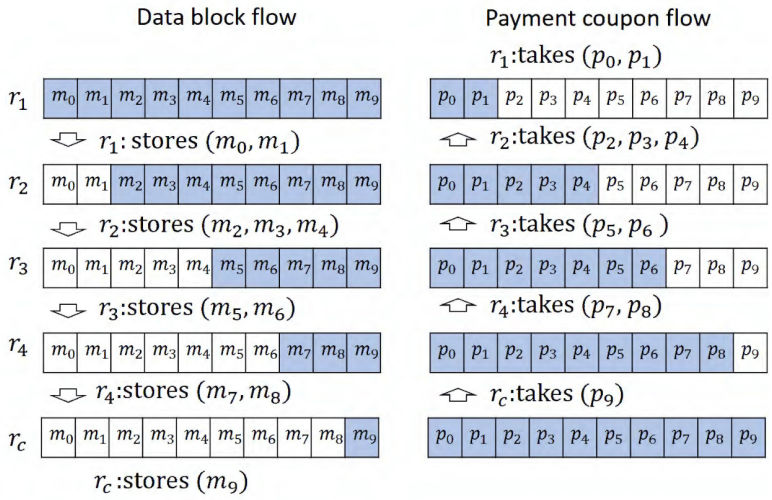


Fig. 5 Process of uploading the data block M and payment coupons P. In this example, u uploads a message M with 10 blocks $\{m_0, m_1, \ldots, m_9\}$. M first arrives at r_1 which stores $M(r_1) = \{m_0, m_1\}$. Then, the residual message blocks $M - M(r_1)$ are forwarded to r_2. r_2 stores $M(r_2) = \{m_2, m_3, m_4\}$ and forwards $M - (M(r_1) \cup M(r_2))$ to r_3. r_3 stores $M(r_3) = \{m_5, m_6\}$ and forwards $M - (M(r_1) \cup M(r_2) \cup M(r_3))$ to r_4. r_4 stores $M(r_4) = \{m_7, m_8\}$ and forwards $M - (M(r_1) \cup M(r_2) \cup M(r_3) \cup M(r_4))$ to r_c. r_c stores all of the residual message blocks which were not stored so far. Once the message blocks in M are stored across the networked storage clusters, the set of payment coupons P is sent to r_c. r_c takes $\{p_9\}$ which is corresponding to the data block it stores, which is $P(r_c) = \{m_9\}$. Then, the rest of the payment coupons $P - \{p_9\}$ are sent to r_4. In the following, each of r_4, r_3, r_2, and r_1 takes $P(r_4) = \{p_7, p_8\}$, $P(r_3) = \{p_5, p_6\}$, $P(r_2) = \{p_2, p_3, p_4\}$, and $P(r_1) = \{p_0, p_1\}$, respectively, and each of the subset of payment coupons is corresponding to the data blocks each networked storage cluster store.

central cloud and distribute among the responsible local fog storages. During the whole process, the user only can see the cloud and interact only with the cloud. Therefore, the proposed system does not require any change on existing end-user terminal devices as long as they are capable of interacting cloud infrastructure.

4.2 Analysis of proposed system

We divide the proposed system into two phases: (1) compensation distribution phase, and (2) compensation acknowledgement phase, and provide the correctness analysis for each of them.

4.2.1 Correctness of the proposed compensation scheme

In this subsection, we prove the correctness of our compensation scheme. Let us introduce some notations and definitions first. Then, we will prove the correctness of the proposed scheme.

We first introduce some additional notations to make our discussion easier. Given a router r_i, $M(r_i)$ is the set of message blocks stored at $C(r_i)$, the storage cluster whose head is r_i. $W(r_i)$ is the subset of message blocks of M which have been witnessed (stored or forwarded to the next hop router) by r_i. $P(r_i)$ is the subset of payment coupons of P which have been witnessed (taken or forwarded to the next hop router) by r_i. $N(r_i)$ is the subset of payment coupons of P which have been taken by r_i. For each $m_j \in M$, we say p_j is the corresponding payment coupon to m_j, i.e., using m_j, p_j can be decrypted and CT_i can be revealed. Then, we define $\hat{m_j} = p_j$ and $\hat{p_j} = m_j$, respectively. Now, we present some important definitions.

Definition 2 (Effect payment tuple) A tuple (m_i, p_j), where m_i is a message block and p_j is a payment coupon, is an effective payment tuple if $i = j$.

Definition 3 (Overpayment) A compensation scheme $\mathcal{C}$ for the proposed storage system allows an overpayment if an r_i is able to access the compensation ticket CT_i inside a payment coupon p_i which is corresponding to a message block m_i which is not stored in $M(r_i)$.

Definition 4 (Underpayment) A compensation scheme $\mathcal{C}$ for the proposed storage system allows an underpayment if an r_i is not able to access the compensation ticket CT_i inside a payment coupon p_i which is corresponding to a message block m_i which is stored in $M(r_i)$.

Definition 5 (Correct compensation scheme) A compensation scheme $\mathcal{C}$ for the proposed storage system is correct if it does not allow both overpayment and underpayment.

Now, we show that the proposed scheme is correct. In particular, we show the proposed compensation scheme will not allow any underpayment (Theorem 3) as well as overpayment (Theorem 4).

Lemma 1 (Partitioning property) For every pair of i, j such that $1 \leqslant i, j \leqslant c$, and $i \neq j$, we have $M(r_i) \bigcap M(r_j) = \varnothing$. Furthermore, for each m_i, there exists a $j \in [1, \ldots, c]$ such that $m_i \in M(r_j)$.

Proof By the construction of the system, each message block m_j arrived at r_i has two options. The first option is stored inside $M(r_i)$. The second option is being forwarded to the next storage cluster, whose head is a fog-aware SDN router, r_{i+1}. Therefore, if a router r_i decided to store a message m_j into $M(r_i)$, this means that all routers r_k such that $k < i$ denied to store m_j. Furthermore, as r_i will not forward to any m_j to the next hop router, no storage cluster $M(r_q)$ with $i < q$ will have a chance to store m_j. Consequently, each block in M will be stored at a unique storage cluster. Furthermore, if m_j will be always stored by some storage cluster, $M(r_c)$ will always store a message block denied by all other storage clusters. Therefore, this theorem is true. ■

Lemma 2 r_i can extract a payment ticket CT_j if $m_j \in W(r_i)$ and $p_j \in P(r_i)$.

Proof As the key to decrypt p_j to take CT_j out is $k = H(m_j)$, this lemma is true. ■

Lemma 3 $W(r_i) = M - \bigcup_{p_l \in P(r_{i-1})} \hat{p_l}$.

Proof By the construction of the message block uploading process, we have $W(r_j) \subseteq W(r_i)$ for any $1 \leqslant i < j \leqslant c$. Now, consider r_i for some $1 \leqslant i \leqslant c$. Then, $W(r_i)$ can be interpreted as a subset of message blocks that are witnessed by any of r_i. By the construction of our payment coupon distribution scheme, as a part of P is originally given to r_c, then the remainders are forwarded to r_{c-1}, and so on, when the residual payment coupons are arrived at r_i, all payment coupons associated with $W(r_i)$ are used and the rest will be given to r_{i-1}. Due to the reason, M can be partitioned into two message block subsets $W(r_i)$ and M', which is the set of message blocks whose associated payment coupons are given to r_{i-1}. In other word, we have

$$M = W(r_i) + \bigcup_{p_l \in P(r_{i-1})} \hat{p_l}.$$

Therefore this lemma is true. ■

Theorem 3 The proposed compensation scheme

does not cause an underpayment.

Proof In order to prevent an underpayment, for each r_i, if $m_j \in M(r_i)$, then $p_j \in N(r_i)$ should be true. In other word, r_i should have an effective payment tuple (m_j, p_j), otherwise an underpayment will occur by Lemma 2. First, by the construction of the message block uploading process, we have

$$M(r_i) = W(r_i) - W(r_{i+1}) \tag{1}$$

Next, by the construction of the payment coupon distribution process, the subset of payment coupon taken by r_i, $N(r_i)$, is equivalent to

$$P(r_i) - P(r_{i-1}) \tag{2}$$

Furthermore, from Lemma 3, we have

$$W(r_i) = M - \bigcup_{p_l \in P(r_{i-1})} \hat{p}_l \tag{3}$$

and

$$W(r_{i+1}) = M - \bigcup_{p_l \in P(r_i)} \hat{p}_l \tag{4}$$

By combining Eqs. (1), (3), and (4), we have

$$\begin{aligned} M(r_i) =& W(r_i) - W(r_{i+1}) = \\ & \Big(M - \bigcup_{p_l \in P(r_{i-1})} \hat{p}_l\Big) - \Big(M - \bigcup_{p_l \in P(r_i)} \hat{p}_l\Big) = \\ & \bigcup_{p_l \in P(r_i)} \hat{p}_l - \bigcup_{p_l \in P(r_{i-1})} \hat{p}_l = \\ & \bigcup_{p_l \in P(r_i) - P(r_{i-1})} \hat{p}_l \end{aligned} \tag{5}$$

Therefore, $m_j \in M(r_i)$ means

$$m_j \in \bigcup_{p_l \in P(r_i) - P(r_{i-1})} \hat{p}_l,$$

and this implies

$$p_j \in P(r_i) - P(r_{i-1}),$$

where $P(r_i) - P(r_{i-1})$ is in fact $N(r_i)$, the subset of payment coupons taken by r_i (see Eq. (2)). As a result, this theorem is true. ■

Theorem 4 The proposed compensation scheme does not allow an overpayment.

Proof By Lemma 1, each block m_j is stored by a unique r_i, but witnessed by all r_k such that $1 \leqslant k < i$. Therefore, some of such r_k may exploit its knowledge on m_j to achieve an overpayment if they also have a knowledge of p_j. In the proposed payment scheme, in order to prevent an overpayment, for any $m_j \notin M(r_i)$, r_i should not know p_j, i.e., $p_j \notin P(r_i)$, otherwise an overpayment will occur by Lemma 2. By the construction of the message block uploading process, for any m_j, if $m_j \in W(r_i)$ and $m_j \notin M(r_i)$, then $m_j \in W(r_{i+1})$, i.e., r_i forwards m_j to the next hop router. Meanwhile, $m_j \in W(r_{i+1})$ implies that $\exists q \geqslant i+1$ such that $m_j \in M(r_q)$ by the construction of the message block uploading process. By Theorem 3, r_q should know p_j. At the same time, by the construction of the payment coupon distribution process, for all l such that $1 \leqslant l < q$, $p_j \notin P(r_l)$. As $i < i+1 \leqslant q$, $p_j \notin P(r_i)$ should be also true. As a result, $p_j \notin P(r_i)$ is true and this theorem is proved. ■

Theorem 5 The proposed compensation scheme is correct and each router r_i will be paid exactly for the messages stored in $M(r_i)$.

Proof By Definition 5, the proposed compensation scheme is correct if no overpayment or underpayment is possible. By Theorem 4, no overpayment is possible in the proposed compensation scheme. By Theorem 3, no underpayment is possible in the proposed compensation scheme. Therefore, this theorem is correct and the proposed compensation scheme is correct. ■

4.2.2 Correctness of compensation acknowledgment scheme

The compensation acknowledgment scheme is based on the construction of a hash chain using the random value embedded in each payment coupon. The random value can be extracted only if the compensation ticket inside the payment coupon is correct extracted.

Suppose $r_c \rightarrow r_{c-1} \rightarrow \cdots \rightarrow r_2 \rightarrow r_1$ is the order of routers (fog storage cluster heads) receiving the payment coupons. For each r_i in this sequence starting from $i = c$, it takes a set of payment coupon $P(r_i)$ and decrypts the coupon. If the decryption was successful, then r_i can obtain both a compensation ticket CT_j and a random number y_j for each $m_j \in M(r_i)$.

By the construction of the hash chain, r_i which owns m_j can compute HC_j based on y_j from p_j and the previous computed hash chain value HC_{j+1} from r_{i+1} or r_i itself. It is important to notice that by the construction of the hash chain, regardless the ownership of m_{j+1}, HC_j is uniquely computed as long as every r_k for $k > i$ decrypted the payment coupons associated with the message blocks stored at them and extracted the random value to compute valid hash chain. Due to the same reason, r_1 will be able to compute a unique HC_1 corresponding all p_is.

Once HC_1 is given to the user through r_c, the user can compute HC_1 on its own, which does not require the knowledge of which r_i and which m_j and p_j. Due to the collision freeness of hash function,

the output of a hash chain operation with n random variables $\{y_1^1, y_2^1, \ldots, y_n^1\}$ will be different from m random variables $\{y_1^2, y_2^2, \ldots, y_m^2\}$ if $n \neq m$, there exists a pair of random values $y_i^1 \neq y_i^2$, or the order of random variables applied to the hash chain computation is different. Otherwise, the output of two hash chain operations will be exactly same.

By comparing HC_1 from r_c with the HC_1 computed by the user in the same order (i.e., starting from y_n, decrease the subscript by 1 up to y_1), the user can verify if all payment coupons are correctly received.

5 Discussion and Concluding Remarks

In our discussion of the storage framework and audit scheme, we focus on how to storage a given set of data and how to produce receipt for audit. As a result, one may wonder how to deal with constantly incoming data stream as well as how to deal with dynamic stored data update. In the following, we outline our idea to deal with the situations. Now, consider that given an application data $\mathcal{M}$, we can always split $\mathcal{M}$ into messages $\{M_1, M_2, \ldots, M_a\}$ with maximum size limit, e.g., 64 KB. Then, each $M_i \in \mathcal{M}$ can be further divided into a number of message blocks $\{m_{i,1}, m_{i,2}, \ldots, m_{i,b}\}$ such that $b \leqslant b_{max}$, where b_{max} is the maximum number of message blocks which can be included in a single message. This means that for any root of a hash tree, a message M_i consists of at most b_{max} leaf values.

From the beginning of a new data uploading flow, if the total number of data blocks uploaded so far is less than b_{max}, we can also add them to the current hash tree. Once the number of blocks reaches at b_{max}, then a new hash tree has to be constructed. In order to add a new block to a hash tree, it is better to use a delayed update. That is, if we want to add a new block each time it arrives, we need to recompute at most h hash values, where h is the height of the current hash tree. This means that to add k new blocks, we may need to perform $h \times k$ hash computations. The overhead of repeatedly computing hash values (and signing on the values) could be significantly reduced if we add k new blocks all together. The approach described above, i.e., pending update, is also useful to improve the efficiency of the proposed system when the update is needed.

In this paper, we propose a new fog-cloud storage framework with transparency such that users only see the existence of the cloud, and therefore there is no need to modify the existing end-user IoT terminal devices. Along with the new storage framework, we introduce two essential mechanisms, stored data integrity audit mechanism as well as storage service provider compensation mechanism. We believe that our seminary work will open a new research direction in the area of fog-cloud storage. However, it will also bring a number of security and privacy issues by nature as it promotes further distribution of user data over distributed storages each of which could be operated by different entities. Therefore, we plan to further study these issues and develop security measure of them.

Acknowledgment

This work was supported by Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. 2016-6-00599, a study on functional signature and its applications). This work was also supported in part by the Soonchunhyang University Research Fund.

References

[1] D. Kim, E. Ko, J. Son, Y. Kim, and J. Seo, A lightweight and transparent compensation mechanism for fog-cloud storage framework, in *Proc. 2008 IEEE 4^{th} Int. Conf. Big Data Computing Service and Applications*, Bamberg, Germany, 2018, pp. 254–259.

[2] Y. Kim, D. Kim, J. Son, W. Wang, and Y. Noh, A new fog-cloud storage framework with transparency and auditability, in *Proc. 2018 IEEE Int. Conf. Communications*, Kansas City, MO, USA, 2018, pp. 1–7.

[3] Z. J. Qin, G. Denker, C. Giannelli, P. Bellavista, and N. Venkatasubramanian, A software defined networking architecture for the internet-of-things, in *Proc. 2014 IEEE Network Operations and Management Symposium*, Krakow, Poland, 2014.

[4] N. Daneshfar, N. Pappas, and V. Angelakis, Resource allocation with service availability & QoS constraints in mobile fog networks, in *Proc. 2017 IEEE Conf. Computer Communications Workshops*, Atlanta, GA, USA, 2017, pp. 1018–1019.

[5] L. M. Vaquero and L. Rodero-Merino, Finding your way in the fog: Towards a comprehensive definition of fog computing, *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 27–32, 2014.

[6] M. Aazam and E. N. Huh, Fog computing micro datacenter based dynamic resource estimation and pricing model for IoT, in *Proc. 2015 IEEE 29^{th} Int. Conf. Advanced Information Networking and Applications*, Gwangiu, South Korea, 2015, pp. 687–694.

[7] T. X. Zhu, T. Shi, J. Z. Li, Z. P. Cai, and X. Zhou, Task scheduling in deadline-aware mobile edge computing systems, *IEEE Internet of Things Journal*, doi: 10.1109/JIOT.2018.2874954.

[8] L. Yu, L. H. Chen, Z. P. Cai, H. Y. Shen, Y. Liang, and Y. Pan, Stochastic load balancing for virtual resource management in datacenters, *IEEE Transactions on Cloud Computing*, doi: 10.1109/TCC.2016.2525984.

[9] L. Yu, H. Y. Shen, Z. P. Cai, L. Liu, and C. Pu, Towards bandwidth guarantee for virtual clusters under demand uncertainty in multi-tenant clouds, *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 2, pp. 450–465, 2018.

[10] L. Yu, H. Y. Shen, K. Sapra, L. Ye, and Z. P. Cai, CoRE: Cooperative end-to-end traffic redundancy elimination for reducing cloud bandwidth cost, *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 2, pp. 446–461, 2017.

[11] L. Yu and Z. P. Cai, Dynamic scaling of virtual clusters with bandwidth guarantee in cloud datacenters, in *Proc. 35th Annu. IEEE Int. Conf. Computer Communications*, San Francisco, CA, USA, 2016, pp. 1–9.

[12] R. Mahmud, R. Kotagiri, and R. Buyya, Fog computing: A taxonomy, survey and future directions, arXiv:1611.05539, 2017.

[13] S. Sarkar, S. Chatterjee, and S. Misra, Assessment of the suitability of fog computing in the context of Internet of things, *IEEE Transactions on Cloud Computing*, vol. 6, no. 1, pp. 46–59, 2018.

[14] A. Munir, P. Kansakar, and S. U. Khan, IFCIoT: Integrated fog cloud IoT: A novel architectural paradigm for the future Internet of Things, *IEEE Consumer Electronics Magazine*, vol. 6, no. 3, pp. 74–82, 2017.

[15] R. L. Deng, R. X. Lu, C. Z. Lai, T. H. Luan, and H. Liang, Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption, *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1171–1181, 2016.

[16] A. Alrawais, A. Alhothaily, C. Q. Hu, and X. Z. Cheng, Fog computing for the Internet of Things: Security and privacy issues, *IEEE Internet Computing*, vol. 21, no. 2, pp. 34–42, 2017.

[17] R. Roman, J. Lopez, and M. Mambo, Mobile Edge Computing, Fog et al.: A survey and analysis of security threats and challenges, *Future Generation Computer Systems*, vol. 78, pp. 680–698, 2018.

[18] Cisco Systems, Fog computing and the Internet of things: Extend the cloud to where the things are, https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf, 2015.

[19] C. Liu, R. Ranjan, C. Yang, X. Y. Zhang, L. Z. Wang, and J. J. Chen, MuR-DPA: Top-down levelled multi-replica Merkle hash tree based secure public auditing for dynamic big data storage on cloud, *IEEE Transactions on Computers*, vol. 64, no. 9, pp. 2609–2622, 2015.

[20] K. Yang and X. H. Jia, Data storage auditing service in cloud computing: Challenges, methods and opportunities, *World Wide Web*, vol. 15, no. 4, pp. 409–428, 2012.

[21] K. Yang and X. H. Jia, An efficient and secure dynamic auditing protocol for data storage in cloud computing, *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 9, pp. 1717–1726, 2013.

[22] Q. Wang, C. Wang, K. Ren, W. J. Lou, and J. Li, Enabling public auditability and data dynamics for storage security in cloud computing, *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 5, pp. 847–859, 2011.

[23] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. J. Lou, Privacy-preserving public auditing for secure cloud storage, *IEEE Transactions on Computers*, vol. 62, no. 2, pp. 362–375, 2013.

[24] C. Liu, C. Yang, X. Y. Zhang, and J. J. Chen, External integrity verification for outsourced big data in cloud and IoT: A big picture, *Future Generation Computer Systems*, vol. 49, pp. 58–67, 2015.

[25] Y. Zhu, G. J. Ahn, H. X. Hu, S. S. Yau, H. G. An, and C. J. Hu, Dynamic audit services for outsourced storages in clouds, *IEEE Transactions on Services Computing*, vol. 6, no. 2, pp. 227–238, 2013.

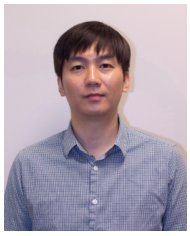
[26] L. F. Wei, H. J. Zhu, Z. F. Cao, X. L. Dong, W. W. Jia, Y. L. Chen, and A. V. Vasilakos, Security and privacy for storage and computation in cloud computing, *Information Sciences*, vol. 258, pp. 371–386, 2014.

[27] D. Naccache, Double-speed safe prime generation, https://eprint.iacr.org/2003/175.pdf, 2003.

[28] J. Katz, *Digital Signatures*. Boston, MA, USA: Springer Science & Business Media, 2010.

Donghyun Kim received the BS degree from Hanyang University, South Korea in 2003, and the MS degree from Hanyang University, South Korea in 2005. He received the PhD degree from the University of Texas at Dallas, TX, USA in 2010. From 2010 to 2016, he was an assistant professor in the Department of Mathematics and Physics at North Carolina Central University, Durham, NC, USA. Currently, he is an associate professor in the Department of Computer Science at Kennesaw State University, Marietta, GA, USA. His research interests include cyber physical, next generation computing and networking, and algorithm design and analysis. He is a member of ACM and a senior member of IEEE.

Daehee Seo received the BS degree from the Dongshin University, South Korea in 2001, and the PhD degree from Soonchunhyang University, South Korea in 2006. Currently, he is an assistant professor in the Department of Computer Science at Kennesaw State University, Marietta, GA, USA. His research interests include cyber security, Blockchain and networking, Attacker trace and Traffic analysis, and SCADA.



Junggab Son is currently an assistant professor in the Department of Computer Science, College of Computing and Software Engineering, Kennesaw State University (KSU), Marietta, GA, USA. He was a limited-term assistant professor from January 2018 to May 2018 and was a research fellow/part-time assistant professor from October 2016 to December 2017 in Department of Computer Science, KSU. Before joining KSU, he was a post doctoral research associate in the Department of Mathematics and Physics, North Carolina Central University, Durham, NC, USA from September 2014 to September 2016. He received the PhD degree in 2014 and the MS degree in 2011 from Hanyang University, South Korea. He received the BS degree in 2009 from Hanyang University, South Korea. His research interests include applied cryptography and data science to address security/privacy issues in critical applications such as Cloud/Fog/Edge Computing, Internet of Things (IoT), Vehicular Ad-hoc Network (VANET), Social Network Services (SNS), and bioinformatics.

Yeojin Kim received the BS degree from Kennesaw State University in 2016, and currently she is pursuing the MS degree at Kennesaw State University. Her research interest includes computer networking, algorithm, and cybersecurity.

Hyobin Kim received BS and MS degrees from Soonchunhyang University, South Korea, in 2017 and 2019, respectively. His research interest includes IoT, platform, blockchain, analyzing security threats, Cyber Security, SCADA systems, and control engineering computing.

Jung Taek Seo received the PhD degree from Korea University in 2006. From November 2000 to February 2016, he has worked for National Security Research Institute as a senior researcher as well as the head of Infrastructure Protection Research Department. Currently, he is an assistant professor in Department of Information Security Engineering, Soonchunhyang University. He has been a principal investigator of several government sponsored research projects in SCADA, Smart Grid, and nuclear power plants. Recently, he has been actively working in the area of smart grid, in particular with respect to standard and policies. His research interest includes SCADA, smart grid, nuclear power plants, smart city, and Cyber Physical System (CPS).