A New Fog-Cloud Storage Framework with Transparency and Auditability

Yeojin Kim*, Donghyun Kim*, Junggab Son*, Wei Wang†, YoungTae Noh‡
* Department of Computer Science, Kennesaw State University, Marietta, GA 30060, USA.
E-mail: ykim89@students.kennesaw.edu, donghyun.kim@kennesaw.edu, json4@kennesaw.edu
† School of Mathematics and Statistics, Xian Jiaotong University, Xian, China. Email: wang_weiw@163.com
‡ Department of Computer Science and Information Engineering, Inha University, Incheon, 402-751, South Korea.
E-mail: ytnoh@inha.ac.kr

***Abstract*—Recently, the concept of fog-cloud storage is attracting lots of attentions to overcome the limit of the central cloud storage. A storage audit scheme aims to ensure user that his/her data on the storage is sound. So far, various audit schemes have been introduced for cloud storages. However, compared to a central cloud storage, a distributed fog-cloud storage consists of multiple local fog storages in addition to a global cloud storage and therefore it is not straightforward to directly apply an existing audit scheme for a cloud storage to a fog-cloud storage. To address this issue, this paper introduces a new fog-cloud storage architecture which can achieve much higher throughput compared to the traditional central cloud storage architecture by reducing the traffics at the routers nearby the cloud storage. The proposed architecture provides transparency such that an end user device does not know the existence of fog storages, and only needs to upload its request toward the central cloud. This means that there is no need to make a modification on the existing end user devices. Our system provides a stronger audit scheme which is naturally coupled with the initial data upload process and does not suffer from the replay attack using old proof of data soundness.**

***Index Terms*—Fog Computing, Fog Cloud Storage, Data Audit, Merkle Hash Tree, Integer Factorization**

I. Introduction

Nowadays, many researchers believe that Internet-of-Things (IoT) will play an important role in shaping our everyday lives in the near future. 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 central cloud will continuously grow, exacerbate the already high latency among them, 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-art cloud computing. 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. 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 incorporated into fog computing environment to realize various fog computing applications [1].

Before the emergence of fog computing paradigm, the centralized cloud has already been adopted to offer various services to end IoT users. 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 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 literature [2]–[4]. 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.

978-1-5386-3180-5/18/$31.00 ©2018 IEEE

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, no effort has been made so far to introduce a data storage audit scheme for fog-cloud storage service in the literature. To address these issues, this paper introduces a new fog-cloud storage framework over SDN with transparency and auditability. The summary of our main contributions is as below.

(a) New fog-cloud storage architecture with efficiency and transparency. The proposed framework consists of three main components, end-user IoT devices, SDN-compatible fog routers with regional fog storages, and global/central cloud storages. In our framework, an end-user IoT device submits a 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 throughout network routers, some of which are SDN-compatible fog routers. Each of these SDN-compatible fog router serves as a cluster head of the regional fog storages. For each request that it receives, depending on incentive and storage space available, a SDN-compatible fog router may accommodate some of the data blocks in the upload request and forward the request with the rest of the remaining data block to the next hop. The request will eventually arrive at the cloud storage and any remaining data blocks will be stored at the cloud storage. 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 (i) the amount of data blocks toward the central cloud storage greatly, (ii) the heavy traffics on the bottleneck routers nearby the cloud storage, and consequently (iii) 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.

(b) 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 node to the back-end cloud storage, the head of each fog storage cluster (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 cluster head 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.

(c) 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 fog storages or cloud storages. 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.

The rest of this paper is organized as follows. Section II discusses related work. We introduce the details of our fog-cloud storage architecture and the audit scheme as our main contributions in Section III. We analyze the security of proposed scheme in Section IV. Finally, we conclude this paper in Section V.

II. Related Work

Fog computing is an emerging network technology to overcome the limit of popular cloud computing technology and attracting 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 [5]–[9] and relevant security issues [6], [10], [11]. 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 [12], 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 [7]

and [9] 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 [6], [10], [11] have conduced a survey on the security concerns and challenges in fog computing. Alrawais [10] et. al. 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 the literature [13]–[20]. 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 [14], Yang and Jia provide 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 [13], [16], [18], [20] 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. in [13], [18] 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 [13] 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.

III. Main Contributions

A. 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; (a) end-user IoT devices $U = \{u_1, u_2, \cdots, u_{|U|}\}$, (b) SDN-compatible fog routers $F = \{f_a, f_b, \cdots, f_{|F|}\}$, and (c) the global cloud storage $cloud$. Furthermore, each SDN-compatible fog routers $f_i \in F$ serves as a cluster head of a set of regional fog storages $FS(f_i)$. In this fog-cloud storage framework, we assume the gateway

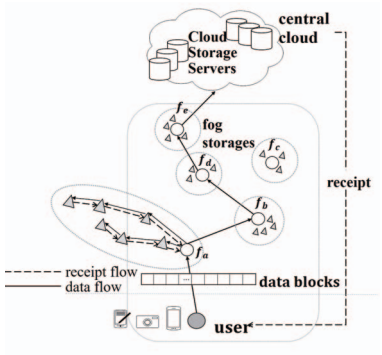

Fig. 1: The fog-storage environment is composed of five SDN-compatible fog routers with regional fog storages $F = \{f_a, f_b, f_c, f_d, f_e\}$ and a central cloud. Only four of them provide storage service to the user u_1, so the routing path from u_1 to $cloud$ becomes $\mathcal{P}_1 = u_1 \rightarrow f_a^1 \rightarrow f_b^1 \rightarrow f_d^1 \rightarrow f_e^1 \rightarrow cloud$ if f_c passes forward the data blocks to the next available fog router f_d without storing any subset of M. 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.

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. In order to simplify our discussion, we assume that M consists of k message blocks with uniform length, i.e. $M = \{m_1, m_2, \cdots, m_k\}$, 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 $M(x) \bigcap M(y) = \emptyset$. 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.

B. Proposed Fog-Cloud Storage Audit Scheme

1) Preliminaries: In fog-cloud storage processing, we introduce two auditing modes; (a) 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 (b) second audit mode, when user retrieves user's data after deleting the original data, he/she can verify data integrity.

Merkle Hash Tree. As the MHT is similar to the complete binary tree, 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

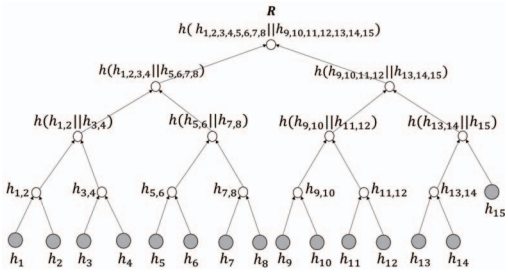

Fig. 2: An example of Merkle Hash Tree. In this figure, $h_i = h(m_i)$, $h_{i,i+1} = h(h_i||h_{i+1})$, and $h(h_{i,i+1}||h_{i+2,i+3}) = h(h(h_i||h_{i+1})||h(h_{i+2}||h_{i+3}))$, where h is a hash function that satisfies $h : \{0,1\}^* \rightarrow \{0,1\}^n$.

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 the Fig. 2 shows the example of the MHT based on the 15 data blocks.

2) Design Goal: Our fog-cloud storage auditing design goals can be summarized as the following:

Fog-cloud storage auditability. User audits if the data stored correctly through two modes, initial audit mode and second 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 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 second 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-number of coprimes and responses the results to the user. Then, user creates another linear congruences for the verification equations with the challenge k-number of 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) Our Construction: Unlike the cloud storage environment, fog-cloud storage environment composed of much more storage service providers. Thus, in the fog-cloud architecture,

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

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 following:

Initial audit mode. Fig. 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 $(\sigma_j, Sh_i, M_{rem})$ 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 the 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 the f_i, the function will return boolean value to generate Signing hash for the subtree which is taken by each SDN-compatible

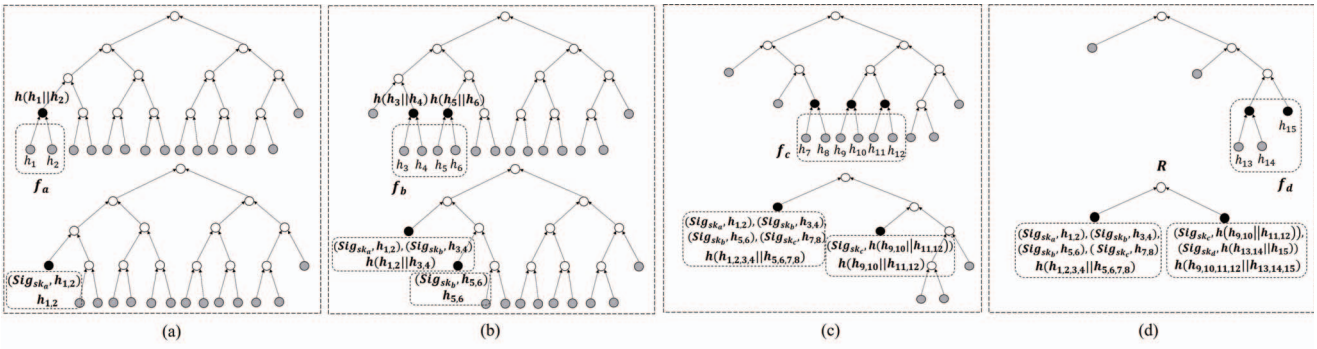

Fig. 3: (a) Re-constructed MHT by f_a, (b) Re-constructed MHT by f_b, (c) Re-constructed MHT by f_c, (d) Re-constructed MHT by f_d

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-number of 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-number of data blocks in M respectively. Fig. 3(a) 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 $M_{rem} = \{m_3, m_4 \cdots, m_{15}\}$. The MHT is reconstructed by f_a as illustrated in the bottom MHT of Fig. 3(a). 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, h_4, from the m_3, 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, h_6, from m_5, m_6, will generate Sh_b. The new updated receipt tuple, $Receipt_{f_b} = (\sigma_b, \big(Sh_{a,b}, Sh_b\big), M_{rem})$ where $\sigma_b = \{m_3, m_4, m_5, m_6\}$, $Sh_{a,b} = \big(\big((Sig_{sk_a}, h_{1,2}), (Sig_{sk_b}, h_{3,4})\big), h(h_{1,2}||h_{3,4})\big)$, $Sh_b = (Sig_{sk_b}, h_{5,6})$, and $M_{rem} = \{m_7, m_8 \cdots, m_{15}\}$. The MHT is reconstructed by f_b as illustrated in the bottom MHT of Fig. 3(b). Third SDN-compatible fog router f_c stores six data blocks $m_7, m_8, \cdots, m_{12}$, 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 is $(\sigma_c, \big(Sh_{a,b,c}, Sh_c\big), M_{rem})$ where $\sigma_c = \{m_7, m_8, \cdots, m_{12}\}$, $Sh_{a,b,c} = \big(\big((Sh_{a,b}), (Sig_{sk_b}, h_{5,6}), (Sig_{sk_c}, h_{7,8})\big), h(h_{1,2,3,4}||h_{5,6,7,8})\big)$, $Sh_c = (Sig_{sk_c}, h(h_{9,10}||h_{11,12}))$, and $M_{rem} = \{m_{13}, m_{14}, m_{15}\}$. The MHT is reconstructed by f_c as illustrated in the bottom MHT of Fig. 3(c). 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 $(\sigma_d, \big(Sh_{a,b,c}, Sh_{c,d}\big), M_{rem})$ where $\sigma_d = \{m_{13}, m_{14}, m_{15}\}$, $Sh_{a,b,c}$ is same as the previous receipt tuple, $Sh_{c,d} = \big(\big((Sh_c), (Sig_{sk_d}, h(h_{13,14,15}))\big), h(h_{9,10,11,12}||h_{13,14,15})\big)$, and $M_{rem} = \emptyset$. The MHT is reconstructed by f_d as illustrated in the bottom MHT of Fig. 3(d).

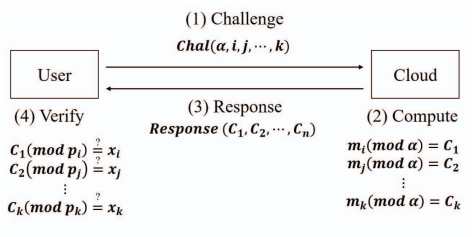

Fig. 4: Challenge and response communication in second audit mode

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 a 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.

Second 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 second audit mode. Although this auditing process will be described in a user's point of view, it can be periodically performed by a TPA [16] 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 second audit mode provides that the user can verify by computing reasonable number of linear congruence equations selectively. Before user deletes the original data, k-number of linear congruence equations are generated as follows: $m_1 \pmod{p_1} = x_1, m_2 \pmod{p_2} = x_2, \ldots, m_k \pmod{p_k} = x_k$. We assume that the p_is are RSA

moduli with safe primes [21].

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 $x_1, x_2, ..., x_k$, 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 $\alpha = (p_i \times p_j \times \cdots \times p_n) \cdot c$ when c is a randomly generated prime 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: $m_1 \pmod{\alpha} = C_1, m_3 \pmod{\alpha} = C_2, m_4 \pmod{\alpha} = C_3, m_5 \pmod{\alpha} = C_4$. 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: $C_1 \pmod{p_1} \stackrel{?}{=} x_1, C_2 \pmod{p_3} \stackrel{?}{=} x_3, C_3 \pmod{p_4} \stackrel{?}{=} x_4, C_4 \pmod{p_5} \stackrel{?}{=} x_5$. 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 second audit mode, 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 from 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 *second 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 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: $C_1 \pmod{p_2} = x_2$, $C_2 \pmod{p_5} = x_5$, $C_3 \pmod{p_6} \neq x_6$, 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.

IV. Security Analysis of the Proposed Scheme

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: (a) it computes a receipt with the actual data blocks or (b) it uses unjust way to compute the root of MHT without data blocks. The (b) can be covered by the following cases: (i) **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 second audit mode without actual data blocks. (ii) **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 second audit mode, and thus, the FSSP

will fail to pass latter auditing. We will provide security analysis for the second audit mode at the $Theorem$ 2. □

Definition 1 (Integer factorization problem [22])**.** *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 second 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 the $Definition$ 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 second 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. □

V. Concluding Remarks

This paper proposes a new audit scheme for fog-cloud storages based on MHT and does not suffer from replay attack unlike the existing MHT-based audit scheme for cloud storages. As a future work, we plan to efficient update mechanisms for the proposed approach.

Acknowledgement

This research was also supported in part by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2017M3C4A7083534).

References

[1] Z. Qin, G. Denker, C. Giannelli, P. Bellavista, and N. Venkatasubramanian, "A software defined networking architecture for the internet-of-things," *IEEE/IFIP NOMS 2014 - IEEE/IFIP Network Operations and Management Symposium: Management in a Software Defined World*, 2014.

[2] N. Daneshfar, N. Pappas, and V. Angelakis, "Resource Allocation with Service Availability and QoS Constraints in Mobile Fog Networks," no. 1, 2017, pp. 1–2. [Online]. Available: http://wivi-2020.eu/Nader_infocom.pdf

[3] 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. [Online]. Available: http://dl.acm.org/citation.cfm?id=2677046.2677052

[4] M. Aazam and E. N. Huh, "Fog computing micro datacenter based dynamic resource estimation and pricing model for IoT," in *Proceedings - International Conference on Advanced Information Networking and Applications, AINA*, vol. 2015-April, 2015, pp. 687–694.

[5] The OpenFog Consortium Architecture Working Group, "OpenFog Architecture Overview," *OpenFogConsortium*, no. February, pp. 1–35, 2016. [Online]. Available: https://www.openfogconsortium.org/wp-content/uploads/OpenFog-Architecture-Overview-WP-2-2016.pdf

[6] R. Mahmud and R. Buyya, "Future Directions," in *Cognitive Hypnotherapy*, 2016, pp. 225–228. [Online]. Available: http://arxiv.org/abs/1611.05539%0A

[7] 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. PP, no. 99, 2015.

[8] 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.

[9] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal Workload Allocation in Fog-Cloud Computing Toward Balanced Delay and Power Consumption," in *IEEE Internet of Things Journal*, vol. 3, no. 6, 2016, pp. 1171–1181.

[10] A. Alrawais, A. Alhothaily, C. Hu, and X. Cheng, "Fog Computing for the Internet of Things: Security and Privacy Issues," *IEEE Internet Computing*, vol. 21, no. 2, pp. 34–42, 2017.

[11] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, Fog et al.: A survey and analysis of security threats and challenges," 2016.

[12] Cisco Systems, "Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are," *Www.Cisco.Com*, p. 6, 2016. [Online]. Available: http://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf

[13] C. Liu, R. Ranjan, C. Yang, X. Zhang, L. Wang, and 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.

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

[15] ——, "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.

[16] Q. Wang, C. Wang, K. Ren, W. 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.

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

[18] C. Liu, C. Yang, X. Zhang, and 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.

[19] Y. Zhu, G. J. Ahn, H. 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.

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

[21] D. Naccache, "Double-speed safe prime generation," *Relation*, pp. 1–2, 2003. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.10.7139

[22] J. Katz, *Digital Signatures*. Springer Science & Business Media, 2010.