

Efficient Auditing Scheme for Secure Data Storage in Fog-to-Cloud Computing

XINGJUN ZHANG AND WEI SI

School of Computer Science and Technology, Department of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China

Corresponding author: Xingjun Zhang (xjzhang@xjtu.edu.cn)


ABSTRACT Fog-to-cloud computing has now become a new cutting-edge technique along with the rapid popularity of Internet of Things (IoT). Unlike traditional cloud computing, fog-to-cloud computing needs more entities to participate in, including mobile sinks and fog nodes except for cloud service provider (CSP). Hence, the integrity auditing in fog-to-cloud storage will also be different from that of traditional cloud storage. In the recent work of Tian et al., they took the first step to design public auditing system for fog-to-cloud computing. However, their scheme becomes very inefficient since they uses intricate public key cryptographic techniques, including bilinear mapping, proof of knowledge etc. In this paper, we design a general and more efficient auditing system based on MAC and HMAC, both of which are popular private key cryptographic techniques. By implementing MAC and HMAC, we give a concrete instantiation of our auditing system. Finally, the theoretical analysis and experiment results show that our proposed system has more efficiency in terms of communication and computational costs.

INDEX TERMS MAC, homomorphic MAC, cloud storage, fog-to-cloud computing.

I. INTRODUCTION

Fog computing, which is first proposed by Bonomi *et al.* in 2012 [6], has now been a popular technique for kinds of industrial fields based on Internet-of-Things (IoT) devices [15], [16], [19], [32]. As a middleware between IoT devices and clouds, fog computing nodes have their own basic computing, storage as well as resources to achieve the requirements for data preprocessing and transmission. Therefore, the model of fog-to-cloud computing emerges as an attractive solution for data storage in some resource-constraint large-scale industrial applications.

However, fog-to-cloud computing has also to face some classical problems appeared in traditional cloud computing. One of the most famous concerns is how to ensure the integrity of stored in cloud service provider (CSP). The reason is as follows. Some CSP may try to conceal the fact that some important data of IoT devices or fog nodes has been lost or corrupted due to kinds of internal or external attacks [25]. Hence, developing efficient auditing techniques for secure data storage in fog-to-cloud computing are also very necessary and significant just like in traditional cloud computing.

The associate editor coordinating the review of this manuscript and approving it for publication was Massimo Cafaro .

Although, in past years, many auditing schemes are presented for traditional cloud storage [12], [22], [25], [26], [31], [33], including many private and public auditing schemes, all of them are not directly applicable to fog-to-cloud computing for two main reasons [23], [24]. The first one is that the data from IoT is generated by various devices and hence it is inadvisable for those users (or data owners) to first retrieve these data and generate corresponding authenticators before outsourcing. The second one, which is also more important, is that the existing auditing systems do not involve fog computing nodes, which are rather crucial entities for fog-to-cloud computing because those nodes can help to efficiently process and rapidly transmit for large-scale of IoT data. Hence, it is urgent to develop new auditing techniques to ensure data's integrity for fog-to-cloud computing. In recent work of [23], Tian et al. took the first step to this direction and try to fill this gap. In fact, they designed a privacy-preserving public auditing system based on bilinear mapping and the so-called tag-transforming strategy. In addition, they also evaluated the performances of their scheme by theoretical analysis and comprehensive experiments.

It is well-known that, in public auditing scheme, the task to verify the integrity of users' data is suitable to be outsourced to another authorized third-party auditor (TPA), which may have more professional knowledge on auditing and more computational resources. However, it should also

be noted that, generally speaking, public auditing systems have lower efficiencies than private ones. Just as Zhang et al. illustrated in [33], for a same data file, the time consumptions for proving, verifying and outsourcing in public auditing scheme are hundreds (or even thousands) of times of the corresponding process in their private scheme. Hence, in some pursuing-efficiency scenarios, especially for the resource-constrained mobile sinks in fog-to-cloud computing, we believe the private auditing system may be more popular. Therefore, it is also necessary and significant to design efficient private auditing schemes for the fog-to-cloud computing.

In this paper, we try to take the step to this direction. More specifically, we propose a new auditing system based on private authentication techniques: message authentication code (MAC) [14] and homomorphic MAC (HMAC) [2], [8]–[10] schemes, both of which are important primitives in cryptography. The MAC technique is used in the transmission process between mobile sinks and fog nodes while the HMAC scheme is used to verify the integrity of data blocks stored in CSP. Since a common private key is needed for the parties in MAC or HMAC when generating or verifying the tags, this model is not suitable to introduce TPA into it.

Moreover, we give a concrete instantiation of the system by instantiating the hash-based MAC scheme in [14] and the efficient HMAC scheme designed by Agrawal and Boneh in [2].

Finally, we also analyze the performances of our proposed system and compare them with that of Tian et al. as well as two related traditional cloud auditing schemes in [20] and [18]. The experiment results show that our system outperformed Tian et al.'s system in terms of communication costs and computational efficiency. Moreover, our protocol is suitable for fog-to-cloud computing and hence prior to the two schemes in [18], [20].

A. RELATED WORKS

One of the earliest work to consider the integrity of data stored in remote clouds is proof of retrievability (PoR) suggested by Jues and Kaliski [13]. In PoR, one can combine error-correcting code with spot-checking of data blocks to ensure the data's integrity. But this technique only supports a limited number of verification operations. At the same time, Ateniese et al. proposed provable data possession (PDP) based RSA-homomorphic authenticators, which can support both unlimited number of challenges and public auditing [3]. Subsequently, many works focused on the improvement of communication efficiency [4], [7], [11], [20]. Some other researches considered the dynamic update of PDP schemes [12], [22], [26], [28]. To support data dynamics, kinds of authenticated data structures are widely introduced into the public auditing schemes. For example, in 2011, Wang et al. presented the Merkle-hash-tree-based public auditing for dynamic data [26]. Later, Zhu et al. proposed a new data structure, called index hash table, to achieve data dynamics [34]. In 2017, Tian et al. further suggested a two-dimensional

data structure, named dynamic hash table, to achieve both public auditing and dynamic data updating [22]. At the same year, Shen et al. proposed another novel structure, which includes a doubly linked info table and a location array, to achieve dynamic data [21].

However, few of them can be directly extended to achieve efficient and secure verification for data storage in the fog-to-cloud based IoT scenarios, although there are fruitful schemes suggested in the traditional cloud storage. The two main reasons are as follows. First, in fog-to-cloud case, the data are usually generated by various IoT devices, instead of the data owners themselves. Second, some new entities, like fog nodes, are introduced and also play important roles for processing and transmission in fog-to-cloud scenario. But in the traditional cloud storage, they are never considered. Therefore, in the recent works, Tian *et al.*, [23] and Kashif and Mohammed [18] respectively filled this gap in the public auditing setting based on different techniques. Nevertheless, the more efficient private key auditing schemes are not considered in both papers.

As for the fog-computing, we note that, in the recent work [27], Wu et al. proposed a fog-computing-enabled cognitive network functions virtualization approach for an information-centric future Internet and also designed a communication scheme between the fog nodes and the future Internet Nodes for the forwarding process.

B. ORGANIZATIONS

The organizations of this paper are as follows. In Section 2, we introduce the background of fog-to-cloud computing as well as some preliminaries, including basic notations and primitives. In Section 3, we presented our proposed auditing system. The security analysis can be found in Section 4. In Section 5, we give a concrete instantiation of the general system presented in Section 3 and discuss the problem of key-distribution. Then we evaluate our system's performances from the aspects of communication and computational costs. Finally, conclusions can be found in Section 7.

II. BACKGROUND AND PRELIMINARIES

A. SYSTEM MODEL

As shown in Fig. 1, a highly modernized company consists of many workshops, each of which is equipped with a large number of sensors, gathering the related environmental data, such as humidity, temperature as well as brightness etc.. Moreover, in each workshop, many mobile sinks are also pre-arranged at proper sites, whose actions are to collect the environment data from the sensors and transmit them to fog nodes.

Generally speaking, every fog node is a local server cluster and is able to conduct some data processing and analysis. The the original data from mobile sinks and the "new" generated ones are transmitted to CSP for cost-efficient storage.

In order to ensure the integrity of data, each mobile sink should compute and generate the authenticatable metadata (i.e. tag), which is bound to the original data. Then the pair of

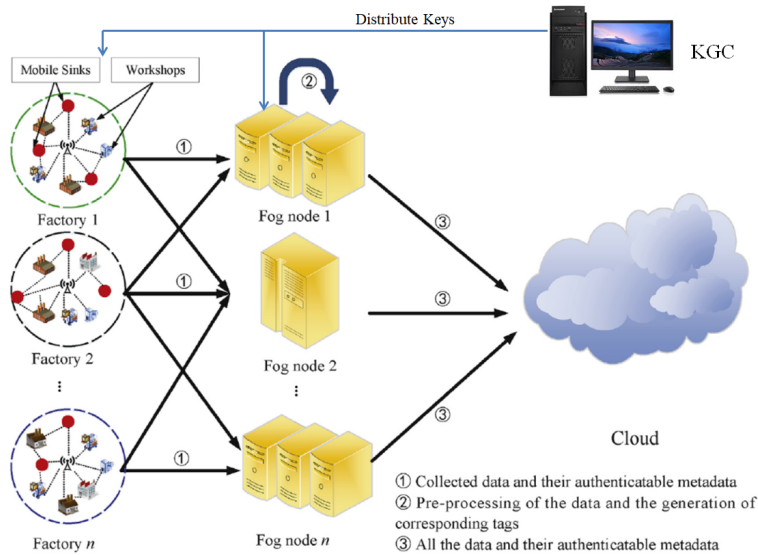


FIGURE 1. System model.

data-tag is sent to the corresponding fog node. After receiving the pair from the mobile sink, the fog node first verifies it. If it can pass the verification, then accept the original data as one block that will be stored to CSP. Otherwise, drop it and require the mobile sink to resubmit it.

Similarly, the fog node also needs to compute authenticated tags for the data blocks from mobile sinks and the pre-processed data from itself so that it is able to audit the integrity of those data after transmitting all the data and their tags to CSP.

B. BASIC NOTATIONS

In the whole paper, we denote by λ the security parameter of algorithms. For a set X , the symbol $x \xleftarrow{\$} X$ means that randomly choose element x from X . For a positive number q , $|q|$ denotes its binary length and $[q]$ denotes the set $\{1, 2, \dots, q\}$. The symbol $\langle \cdot, \cdot \rangle$ means that the inner-product operation of two vectors. PPT is an abbreviation of probabilistic polynomial time. A function $f(\lambda)$ is called negligible if, for any $c > 0$, there exists a $\lambda_0 \in \mathbb{Z}$ such that for any $\lambda > \lambda_0$, it holds that $f(\lambda) < \lambda^{-c}$.

Now, we list other symbols appeared in this paper as well as their definitions in Table 1.

C. PSEUDORANDOM FUNCTIONS

Now, we recall the notion of pseudorandom function (PRF). In particular, consider a set of functions from $\mathcal{K} \times D$ to R . Denote by $func(D, R)$ all the functions from D to R . Then $f \xleftarrow{\$} func(D, R)$ means that, for $x \in D$, the value $f(x)$ is random in R . Randomly choose a K from \mathcal{K} and let $\mathcal{O}_0 = F(K, \cdot)$ be an oracle, which will compute and output the value $F(K, x)$ when given the input x . Finally, for a PPT adversary A , who can make queries to the oracle \mathcal{O}_0 or $f(\cdot)$, it is computational indistinguishable to decide which one it is interacting with. Since the performance of F looks like a truly random function f , we call the set consisting of F is a pseudorandom function set.

TABLE 1. Symbols and the corresponding definitions.

Symbols	Definitions
IoT	Internet-of-Things
CSP	Cloud Service Provider
TPA	Third-Party Auditor
KGC	Key Generation Center
MAC	essage Authentication Code
PoR	Proof of Retrievability
PDP	Provable Data Possession
PRF	Pseudorandom Function
s_1	the number of mobile sinks
f	the number of fog nodes
$MS_{i,j}$	the j -th mobile sink of i -th fog node
$m_{i,j}$	the data block generated by $MS_{i,j}$
$T_{i,j}$	the augmented vector of $m_{i,j}$
$t_{i,j}$	the MAC-tag of $m_{i,j}$
F_i	the i -th fog node
$T_{i,j}$	the HMAC-tag of $v_{i,j}$
$k_{i,j}$	the common key between $MS_{i,j}$ and F_i
k_i	the set $\{k_{i,1}, \dots, k_{i,s_1}\}$
K_i	the common key between the cloud and F_i
(j_τ, r_τ)	the challenge message for verifier
ℓ	the numbers of challenge messages
$\mathcal{K}^{(i)}$	the key space of PRF F_i
MAC-KeyGen	the key-generation algorithm of MAC
MAC	the tag-generation algorithm of MAC
MAC-Verify	the verification algorithm of MAC
HMAC-KeyGen	the key-generation algorithm of HMAC
HMAC	the tag-generation algorithm of HMAC
HCombine	the combination algorithm of HMAC
HVerify	the verification algorithm of HMAC
TagGenMS	the tag-generation for mobile sinks
TagGenFN	the tag-generation for fog node
ChalGen	the generation of challenge message
ProofGen	the generation of proof
Verify	the verification on the returned proof

D. MESSAGE AUTHENTICATION CODE

A message authentication code (MAC) MAC consists of three PPT algorithms MAC-KeyGen, MAC, and MAC-Verify, which have the following forms.

- MAC-KeyGen : For the input λ , this algorithm randomly chooses a key $k \xleftarrow{\$} \{0, 1\}^\lambda$ and output it.

- **MAC** : Take some key k and a message $m \in \{0, 1\}^*$ as inputs. Then compute a tag t for m and output it.
- **MAC-Verify** : Take some key k , a message m and a tag t as inputs. This algorithm will output a bit $b \in \{0, 1\}$. If $b = 1$, then accept t as a valid tag for m . Otherwise, t is not a valid tag for m .

The correctness requires that, for any λ , every $k \in \{0, 1\}^\lambda$ and all $m \in \{0, 1\}^*$, it holds that

$$1 \leftarrow \text{MAC-Verify}(k, m, \text{MAC}(k, m)).$$

The security requires that, any PPT adversary can generate a valid tag on a “new” message (i.e., a message not sent by the communicating parties) with at most negligible probability.

E. HOMOMORPHIC MAC

Informally, a homomorphic MAC scheme is an authentication technology, which allows “legal” users to verify the correctness of the generated tag t for a message v , which in fact is an $(n + s)$ -dimensional vector in some finite field \mathbb{F}_q , and recompute a new tag on a combined message.

Formally, a (q, n, s) homomorphic MAC scheme consists of four PPT algorithms **HMAC-KeyGen**, **HMAC**, **HCombine** and **HVerify**, which have the following forms.

- **HMAC-KeyGen** : For the input λ , this algorithm generates and outputs a secret key K .
- **HMAC** : Take as inputs of a secret key K , an identifier id , an augmented vector $v \in \mathbb{F}_q^{n+s}$, and $j \in [s]$ indicating that v is the j -th basis vector of the vector space identified by id . This algorithm will output a tag T for v .
- **HCombine** : For the inputs of ℓ ($\ell < s$) constants $r_1, \dots, r_\ell \in \mathbb{F}_q$, vectors $v_1, \dots, v_\ell \in \mathbb{F}_q^{n+s}$ and the corresponding tags T_1, \dots, T_ℓ , this algorithm outputs a tag T for the combined vector $y := \sum_{i=1}^{\ell} r_i v_i \in \mathbb{F}_q^{n+s}$.
- **HVerify** : For the inputs of a secret key K , an identifier id , a vector $y \in \mathbb{F}_q^{n+m}$ and a tag T , this algorithm outputs 0 (reject) or 1 (accept).

The correctness requires that, for any secret key K , $r_1, \dots, r_\ell \in \mathbb{F}_q$,

$$T_j = \text{HMAC}(K, id, v_j, j),$$

it holds that

$$1 \leftarrow \text{HVerify} \left(K, id, \sum_{j=1}^{\ell} r_j v_j, \text{HCombine} \left(\left\{ (r_j, v_j, T_j)_{j=1}^{\ell} \right\} \right) \right).$$

The security requires that any PPT attacker, obtaining the signatures on arbitrary vector spaces of its choice, is not able to give a valid triple (id, y, T) , where id is new, or equals to some identifier id_i that it had obtained the corresponding signature but y does not belong to the space identified by id_i .

III. THE PROPOSED AUDITING SYSTEM

In this section, we present our general auditing system for fog-to-cloud computing based on a MAC scheme

$$\text{MAC} = (\text{MAC-KeyGen}, \text{MAC}, \text{MAC-Verify})$$

and a (q, n, s) homomorphic MAC scheme

$$\text{HMAC} = (\text{HMAC-KeyGen}, \text{HMAC}, \text{HCombine}, \text{HVerify}).$$

In fact, the whole system mainly includes six PPT algorithms: a key-generation algorithm **KeyGen**, a tag-generation algorithm for mobile sinks **TagGenMS**, a tag-generation algorithm for fog nodes **TagGenFN**, a challenge-generation algorithm **ChalGen**, a proof-generation algorithm **ProofGen** and a verification algorithm **Verify**. All the entities are equipped with those algorithms as follows.

- A key generation center (KGC)¹ runs the key-generation algorithm to obtain some keys and distributes them to the entities in the system, including s_1 mobile sinks, f fog nodes and CSP.
- After obtaining a distributed key $k_{i,j}$ ($1 \leq i \leq f$, $1 \leq j \leq s_1$) from KGC, each mobile sink $MS_{i,j}$ runs the algorithm **TagGenMS**, which in fact consists of the authentication algorithm of a MAC scheme, to generate an authenticated tag $t_{i,j}$ for its data block $m_{i,j} \in \mathbb{F}_q^n$ and transmits the pair $(m_{i,j}, t_{i,j})$ to its fog node F_i .
- Then the fog node F_i will first check the correctness of the transmitted pair $(m_{i,j}, t_{i,j})$ by using the verification algorithm **MAC-Verify** of the same MAC scheme based on the common key $k_{i,j}$, which is also distributed by KGC. If it can not pass the verification, then drop it and require the mobile sink to resubmit a new pair. Else, discard the tag $t_{i,j}$ and only save the data block $m_{i,j}$. Meanwhile, the fog node is also allowed to make analysis on the data blocks collected from all the mobile sinks and hence generates new data blocks denoted by $m_{i,s_1+1}, m_{i,s_1+2}, \dots, m_{i,s}$. For all the data blocks $m_{i,1}, m_{i,2}, \dots, m_{i,s}$, the fog node F_i binds them into an intact file with filename ID_i , and respectively computes tags for each block by running **TagGenFN**, which in fact is a homomorphic MAC algorithm, based on another key K_i , and submits all the data-tag pairs to CSP for secure storage.
- If some fog node wants to check the integrity of its data blocks, it can run **ChalGen** to obtain a challenge message $chal$ and gives them to CSP.
- Now, CSP can run **ProofGen** to get a related proof Γ based on the storage data blocks and returns it to the fog node.
- Finally, the fog node can check the response Γ by running **Verify** based on the same key K_i , which is also distributed by KGC.

Specifically, the six algorithms included in the above system are described as follows.

- **KeyGen** : The key generation algorithm takes the security parameter λ as its input and will generate keys for all the entities except for CSP in the system.

¹Here, we remark that, the KGC must be honest and cannot be compromised, just as the basic assumptions in [29], [30].

Run $f \cdot s_1$ times $\text{MAC-KeyGen}(\lambda)$ to get the MAC-keys

$$\{\{k_{1,1}, \dots, k_{1,s_1}\}, \dots, \{k_{f,1}, \dots, k_{f,s_1}\}\},$$

which is denoted by $\{k_1, \dots, k_f\}$. Then run f times $\text{HMAC-KeyGen}(\lambda)$ to get HMAC-keys

$$K_1, \dots, K_f.$$

The j -th mobile sink $MS_{i,j}$ for the i -th fog node F_i , who gets the keys k_i and K_i , will obtain the key $k_{i,j}$. That is, all the keys of the mobile sinks corresponding to some fog node are also shared by this fog node.²

- **TagGenMS** : The tag generation algorithm for mobile sinks takes data block and a MAC-key as inputs, and will output a tag under the MAC-key for this data block. Concretely, for the data block $m_{i,j} \in \mathbb{F}_q^n$ collected by the mobile sink $MS_{i,j}$, this algorithm computes

$$t_{i,j} \leftarrow \text{MAC}(k_{i,j}, m_{i,j}),$$

and outputs it.

- **TagGenFN** : This algorithm is run by fog nodes, which takes a HMAC-key K_i , an identifier ID_i , data block $m_{i,j} \in \mathbb{F}_q^n$ and an index j as inputs, and will output a tag under the HMAC-key of some fog node for this data block. Specifically, for the data block $m_{i,j}$, this algorithm first augments it as

$$v_{i,j} = \left(m_{i,j}, \underbrace{0, \dots, 0, 1, 0, \dots, 0}_s \right) \in \mathbb{F}_q^{n+s}.$$

Then compute and output the tag

$$T_{i,j} \leftarrow \text{HMAC}(K_i, ID_i, v_{i,j}, j).$$

- **ChalGen** : This algorithm generates the challenge messages for each fog nodes. In particular, for the input of parameter ℓ , it randomly chooses ℓ different indices $1 \leq j_1 < j_2 < \dots < j_\ell \leq s$ as well as coefficients $r_1, \dots, r_\ell \in \mathbb{F}_q$. Denote by $chal$ the set of challenge messages

$$\{(j_1, r_1), \dots, (j_\ell, r_\ell)\}$$

and output it.

- **ProofGen** : Once receiving the challenge $chal$ from the i -th fog node F_i , the CSP first parses $chal$ as

$$\{(j_1, r_1), \dots, (j_\ell, r_\ell)\}.$$

Then run

$$(v, T) \leftarrow \text{HCombine} \left(\{(r_\tau, v_{i,j_\tau}, T_{i,j_\tau})\}_{\tau=1}^\ell \right).$$

Denote by Γ the proof of (v, T) and return it to the corresponding fog node F_i .

- **Verify** : When receiving the returned proof Γ , the fog node F_i first parses Γ as (v, T) and runs

$$b \leftarrow \text{HVerify}(K_i, ID_i, v, T).$$

If $b = 1$, then it thinks that its original data blocks are still securely stored in CSP. Else if $b = 0$, then the integrity of those blocks is broken.

IV. SECURITY ANALYSIS

In this section, we analyze the security of our proposed system in Section III, which can be divided into two cases: data security from mobile sinks to fog node and data integrity from fog node to CSP.

A. DATA SECURITY FROM MOBILE SINKS TO FOG NODE

Theorem 1: If MAC is a secure MAC scheme, then the transmission of data blocks from mobile sinks to fog node is secure.

Proof: For the i -th fog node F_i , who has the MAC-key

$$k_i = \{k_{i,1}, \dots, k_{i,s_1}\},$$

it shares the key $k_{i,j}$ ($1 \leq j \leq s_1$) with the j -th mobile sink $MS_{i,j}$. Consider a PPT adversary \mathcal{A} , who does not know the key $k_{i,j}$ but intends to denote $MS_{i,j}$ and submit a forged message-tag pair (m, t) to the fog node F_i . If it can pass the verification of fog node, that is,

$$1 \leftarrow \text{MAC-Verify}(k_{i,j}, m, t),$$

then this (m, t) is just a successful forgery for the MAC scheme MAC . Therefore, according to the security of MAC , we know that any ‘‘illegal’’ PPT adversary can not generate a valid message-tag pair with non-negligible probability. That is, the transmission process of data blocks from mobile sinks to the corresponding fog node is secure.

B. DATA INTEGRITY FROM FOG NODE TO CSP

Theorem 2: If $HMAC$ is a secure HMAC scheme, then the data integrity (stored in CSP) for the system presented in Section III can be guaranteed.

Proof: To respond to the challenge message

$$chal = \{(j_1, r_1), \dots, (j_\ell, r_\ell)\}$$

from the i -th fog node, the CSP needs to compute the proof Γ as

$$(v, T) \leftarrow \text{HCombine} \left(\{(r_\tau, v_{i,j_\tau}, T_{i,j_\tau})\}_{\tau=1}^\ell \right). \quad (1)$$

Therefore, we only need to consider the unforgeability of Γ . If some block m_{i,j_0} (or its augmented vector v_{i,j_0}) is not securely stored in CSP, then the randomly chosen ℓ indices j_1, j_2, \dots, j_ℓ include j_0 with probability ℓ/s . Meanwhile, CSP can also not generate a correct proof Γ from (1). Hence, the CSP has to return a forged proof $\Gamma' = (v', T')$ without using m_{i,j_0} (or v_{i,j_0}). According to the security of $HMAC$, we know that the forged proof Γ' can pass the verification of HVerify with at most negligible probability.

²In this case, if some of the mobile sinks are compromised or revoked, then it does not affect the key's security of other mobile sinks. However, if the i -th fog node is compromised or revoked from the system, then the secret keys of $MS_{i,1}, \dots, MS_{i,s_1}$ are all exposed hence need to be updated.

As a result, the fog node F_i accepts this proof (i.e. running $\text{Verify}(K_i, v', T')$ and obtaining the output 1) with also negligible probability. Hence, the data integrity from fog node to CSP for our proposed system is hold if the underlying HMAC scheme HMAC is secure.

V. A CONCRETE INSTANTIATION AND SOME DISCUSSIONS

A. A CONCRETE INSTANTIATION

In this subsection, we give a concrete instantiation for our proposed system presented in Section III by instantiating the MAC and HMAC schemes. In particular, we choose the hash-based MAC scheme in [14] and the HMAC scheme proposed by Agrawal and Boneh in [2]. Then the concrete instantiations of the six algorithms are as follows.

- KeyGen' : The key generation algorithm takes the security parameter λ as its input. Randomly choose

$$(k_{1,1}, \dots, k_{1,s_1}, \dots, k_{f,1}, \dots, k_{f,s_1}) \xleftarrow{\$} \{0, 1\}^{\lambda f s_1},$$

and set k_i as $(k_{i,1}, \dots, k_{i,s_1})$ for $1 \leq i \leq f$. Then randomly choose $u_i \xleftarrow{\$} \mathbb{F}_q^{n+s}$ and $b_{i,j} \xleftarrow{\$} \mathbb{F}_q$ for $1 \leq i \leq f, 1 \leq j \leq s$. Set

$$K_i = (u_i, b_{i,1}, \dots, b_{i,s}).$$

The j -th mobile sink $MS_{i,j}$ for the i -th fog node F_i , who gets the keys k_i and K_i , will obtain the key $k_{i,j}$.

- $\text{TagGenMS}'$: For the data block $m_{i,j} \in \mathbb{F}_q^n$ collected by the mobile sink $MS_{i,j}$, this algorithm computes

$$t_{i,j} = H(IV || (k_{i,j} \oplus \text{opad}) || H(IV || (k_{i,j} \oplus \text{ipad}) || m_{i,j})),$$

in which H is a standard hash function, IV is an arbitrary fixed initial vector, opad is formed by repeating the byte 36 in hexadecimal as many times as needed and the string ipad is formed in the same way using the the byte 5C. Output $t_{i,j}$ as the tag of the data block $m_{i,j}$.

- $\text{TagGenFN}'$: This algorithm is run by fog nodes, which takes a HMAC-key

$$K_i = (u_i, b_{i,1}, \dots, b_{i,s}),$$

an identifier ID_i , data block $m_{i,j} \in \mathbb{F}_q^n$ and an index j as inputs. For the data block $m_{i,j}$, this algorithm first augments it as

$$v_{i,j} = \left(m_{i,j}, \underbrace{0, \dots, 0}_j, \underbrace{1, 0, \dots, 0}_s \right) \in \mathbb{F}_q^{n+s}.$$

Then compute and output the tag

$$T_{i,j} = \langle u_i, v_{i,j} \rangle + b_{i,j} \in \mathbb{F}_q.$$

- $\text{ChalGen}'$: Same as ChalGen .
- $\text{ProofGen}'$: Once receiving the challenge chal from the i -th fog node F_i , the CSP first parses chal as

$$\{(j_1, r_1), \dots, (j_\ell, r_\ell)\}.$$

Then compute

$$v = \sum_{\tau=1}^{\ell} r_\tau v_{i,j_\tau}, \quad T = \sum_{\tau=1}^{\ell} r_\tau T_{i,j_\tau}.$$

Finally, output the proof $\Gamma = (v, T)$.

- Verify' : When receiving the returned proof Γ , the fog node F_i first parses Γ as (v, T) and computes

$$T'_{i,j} := \langle u_i, v \rangle + \sum_{j=1}^s (v_{n+j} b_{i,j}), \quad (2)$$

where v_{n+j} is the $(n+j)$ -th component of v . If $T'_{i,j} = T_{i,j}$, then output 1. Otherwise, output 0.

In addition, we remark that the fog node F_i will check the correctness of the pair $(m_{i,j}, t_{i,j})$ transmitted from the mobile sink $MS_{i,j}$ as follows. Compute

$$t'_{i,j} = H(IV || (k_{i,j} \oplus \text{opad}) || H(IV || (k_{i,j} \oplus \text{ipad}) || m_{i,j})),$$

and check if $t_{i,j} = t'_{i,j}$. If it is not, drop it and require the mobile sink to resubmit a new pair. Otherwise, discard the tag $t_{i,j}$ and continue to do the next steps.

B. KEY-DISTRIBUTION

Note that in the concrete instantiation presented in Section V-A, each mobile sink owns a key $k_{i,j} \in \{0, 1\}^\lambda$. But each fog node F_i will be distributed s_1 MAC-keys $k_{i,1}, \dots, k_{i,s_1}$ and a HMAC-key

$$K_i = (u_i, b_{i,1}, \dots, b_{i,s}) \in \mathbb{F}_q^{n+2s}.$$

Therefore, the distribution of keys for all the entities (especially for fog nodes) becomes a heavy work in our proposed system. In this subsection, we adopt cryptographic PRF to change the situations and try to alleviate the heavy task of key-distribution.

Concretely, let

$$\begin{aligned} \mathcal{F}_1 &: \mathcal{K}^{(1)} \times \{0, 1\}^* \rightarrow \mathbb{F}_q, \\ \mathcal{F}_2 &: \mathcal{K}^{(2)} \times \{0, 1\}^* \rightarrow \{0, 1\}^\lambda \end{aligned}$$

be two families of PRFs. The i -th fog node F_i randomly chooses $k_i^{(1)} \in \mathcal{K}^{(1)}$ and $k_i^{(2)} \in \mathcal{K}^{(2)}$. Then for $1 \leq j \leq s$, compute

$$b_{i,j} \leftarrow \mathcal{F}_1(k_i^{(1)}, j), \quad (3)$$

and

$$k_{i,1} \leftarrow \mathcal{F}_2(k_i^{(2)}, 1), \dots, k_{i,s_1} \leftarrow \mathcal{F}_2(k_i^{(2)}, s_1).$$

Distribute $k_{i,j}$ to its j -th mobile sink (corresponding to F_i) while keeping

$$K_i = (u_i, b_{i,1}, \dots, b_{i,s})$$

as its own HMAC-key.

C. DYNAMIC UPDATE AND PREVENTING REPLAY ATTACKS

In this subsection, we discuss the problem of dynamic update and security against replay attacks for our proposed protocol, which is based on the key-distribution technique discussed in Section V-B. Here, we adopt the map-version table \mathcal{T} , which is a small data structure on the verifier side (i.e. fog nodes F_i 's) to record the update information and proposed by Barsoum and Hasan in [5]. Concretely, the map-version table consists of three columns: Serial number $SN^{(i)}$, block number j , and block version $BV^{(i)}$. The serial number is an indexing to the file blocks and indicating the physical position of a block in a data file. The block number is a counter used to make a logical indexing to the file blocks and the block version indicates the current versions of the file block. If a specific block is updated, then its $BV^{(i)}$ is incremented by 1.

1) DYNAMIC UPDATE

To support dynamic update, the original key-generation in (3) will be further changed as

$$b_{i,j} \leftarrow \mathcal{F}_1(k_i^{(1)}, j, BV^{(i)}). \quad (4)$$

When the i -th fog node F_i wanting to dynamic update the outsourced blocks, it sends a request in the form:

$$\langle ID_i, BlockOp, j, m_{i,j}^*, T_{i,j}^* \rangle,$$

where ID_i is F_i 's identity, $BlockOp$ corresponds to block modification (denoted by BM), block insertion (BI), or block deletion (BD), and $(m_{i,j}^*, T_{i,j}^*)$ is the new block-tag pair computed as follows.

- **Block Modification.** First update $BV^{(i)} = BV^{(i)} + 1$ in the map-version table \mathcal{T} and compute

$$\begin{aligned} T_{i,j}^* &= \langle u_i, m_{i,j}^* \rangle + b_{i,j} \\ &= \langle u_i, m_{i,j}^* \rangle + \mathcal{F}_1(k_i^{(1)}, j, BV^{(i)}). \end{aligned} \quad (5)$$

Upon receiving the request, the CSP replaces $m_{i,j}$ with $m_{i,j}^*$ and $T_{i,j}$ with $T_{i,j}^*$.

- **Block Insertion.** If F_i wants to insert a new block m_i^* into the outsourced data file, it first defines $SN = SN_{max} + 1$, $BN = BN_{max} + 1$, $BV^{(i)} = 1$, where SN_{max} and BN_{max} are respectively current maximum of serial and block numbers, and generates the tag T_i^* as in (5). Upon receiving this request, the CSP adds (m_i^*, T_i^*) into the data file outsourced by F_i .
- **Block Deletion.** The deletion request is the form $\langle ID_i, BD, j, null, null \rangle$. The map-version table \mathcal{T} is updated by deleting the j -th item and the CSP deletes the j -th data-tag pair.

When verifying the returned proof from CSP, which is based on some challenge message $chal$, the fog node F_i looks up \mathcal{T} and finds out the block information (j_1, \dots, j_ℓ) as well as the block version $(BV_{j_1}^{(i)}, \dots, BV_{j_\ell}^{(i)})$. Then re-compute $b_{i,j}$ according to (4) and continue the next steps in the algorithm $Verify'$.

Obviously, the additional operations to support dynamic update only consist of an appended map-version table and an additional input for \mathcal{F}_1 . Therefore, in terms of communication and computational costs, it does not affect efficiency.

2) PREVENTING REPLAY ATTACKS

Here, we discuss the security of preventing replaying attacks for our concrete instantiation protocol in the above Subsection. Of course, in order to obtain security against replay attack, we need further modify (4) as

$$b_{i,j} \leftarrow \mathcal{F}_1(k_i^{(1)}, j, BV^{(i)}, Time_{i,j}), \quad (6)$$

where $Time_{i,j}$ denotes the time stamp generating $b_{i,j}$. Then we have

Theorem 3: If H is a secure hash function, then the modified protocol of Section V-A (according to (6)) is secure against replay attacks.

Proof. Recall that the replay attack means that a part of the returned proof from CSP, which is based on some challenge message

$$chal = \{(j_1, r_1), \dots, (j_\ell, r_\ell)\},$$

may be replaced by some previous data information. Without loss of generality, we assume that the first block's information is replaced by its former block v_{i,j_1}^* as well as the corresponding tag T_{68,j_1} . In other words, the CSP computes its proof $\Gamma^* = (v^*, T^*)$ as

$$v^* = r_1 v_{i,j_1}^* + \sum_{\tau=2}^{\ell} r_\tau v_{i,j_\tau}, \quad T^* = r_1 T_{i,j_1}^* + \sum_{\tau=2}^{\ell} r_\tau T_{i,j_\tau}.$$

According to the equation (2) in the algorithm of $Verify'$, we know that this Γ^* can pass the verification with at most negligible probability if the time stamp is considered.

VI. PERFORMANCE EVALUATIONS

In this section, we make the performance analysis on our concrete instantiation for the system in Section V-A in terms of communication and computational costs. In fact, we choose three classical and related protocols: [20], [23], and [18], and compare the performances of them with our proposed one. We remark that, in [20], Shacham and Waters proposed one private key protocol and another public key verification protocol for traditional cloud storage, respectively. Meanwhile, the proposed protocol by Munir and Mohammed in [18] is almost same as the public key verification one in [20]. However, as we said in the part of Introduction, the existing authentication protocols for traditional cloud storage can not be directly used to fog-to-cloud computing and do not consider how the mobile sinks securely transmit data to fog nodes. For convenience to compare, the fog node is seen as the data owner in traditional cloud auditing.

A. COMMUNICATION COSTS

1) MOBILE SINK TO FOG NODE (MStoFG)

In the process of transmission from mobile sink to fog node, each mobile sink $MS_{i,j}$ needs to compute a MAC-tag $t_{i,j}$ for

TABLE 2. The total comparisons of communication costs for our system and other ones.

Systems	MStoFG	FGtoCSP	ChFGtoCSP	ReCSPtoFN
Ours	$ H $	$s \cdot \mathbb{F}_q $	$\ell \cdot (s + \mathbb{F}_q)$	$(n + s + 1) \cdot \mathbb{F}_q $
[23]	$ \mathbb{G} $	$(s + s_1) \cdot \mathbb{G} $	$\ell \cdot (s + \mathbb{F}_q) + (2 + s_1) \cdot \mathbb{G} + \Lambda $	$ \mathbb{G} $
[20]	" \times "	$s \cdot \mathbb{F}_q $	$\ell \cdot (s + \mathbb{F}_q)$	$2 \cdot \mathbb{F}_q $
[18]	" \times "	$s \cdot \mathbb{G} $	$\ell \cdot (s + \mathbb{F}_q)$	$ \mathbb{F}_q + \mathbb{G} $

its data block $m_{i,j}$ and transmit the data-tag pair $(m_{i,j}, t_{i,j})$ to fog node. Hence, the communication overhead for this process is the length of the tag $t_{i,j}$ (i.e. $|t_{i,j}|$), which in fact is the length of the chosen hash function H (denoted by $|H|$). Accordingly, the tag $t_{i,j}$ in [23] consists of an element in the basis group \mathbb{G} and hence, its communication overhead for this process is $|\mathbb{G}|$. In addition, since in the systems of [20] and [18], the process of transmitting from mobile sink to fog node is not considered, the communication costs from mobile sink to fog node for both protocols are null.

2) FOG NODE TO CSP (FGtoCSP)

For the process of transmitting from fog node to CSP, each fog node F_i needs to recompute HMAC-tag

$$T_{i,1}, T_{i,2}, \dots, T_{i,s_1}$$

for all the corresponding data blocks $m_{i,1}, m_{i,2}, \dots, m_{i,s_1}$ collected from its mobile sinks. In addition, it also needs to compute the HMAC-tags

$$T_{i,s_1+1}, T_{i,s_1+2}, \dots, T_{i,s}$$

for the data blocks $m_{i,s_1+1}, m_{i,s_1+2}, \dots, m_{i,s}$ generated by itself. Therefore, the communication overhead in our system for this process is

$$|T_{i,1}| + |T_{i,2}| + \dots + |T_{i,s}| = s \cdot |\mathbb{F}_q|.$$

In turn, the communication overhead in [23] equals to

$$\begin{aligned} &|T_{i,1}| + \dots + |T_{i,s_1}| + |T_{i,s_1+1}| + \dots + |T_{i,s}| \\ &= (2|\mathbb{G}| + \dots + 2|\mathbb{G}| + |\mathbb{G}| + \dots + |\mathbb{G}|) \\ &= (2s_1 + (s - s_1)) \cdot |\mathbb{G}| \\ &= (s + s_1) \cdot |\mathbb{G}|. \end{aligned}$$

If the fog nodes are seen as the data owners in traditional cloud storage, then for total s data blocks m_1, \dots, m_s , the communication overheads for [20] and [18] are also the generated tags for all the data blocks, which respectively equal to $s \cdot |\mathbb{F}_q|$ and $s \cdot |\mathbb{G}|$.

3) CHALLENGE MESSAGES FROM FOG NODE TO CSP (ChFGtoCSP)

In our system, if a fog node wants to check the integrity of its stored data blocks, then it only needs to submit a challenge message

$$chal = \{(j_1, r_1), \dots, (j_\ell, r_\ell)\}$$

to CSP. Hence, the communication cost for fog node equals to

$$\begin{aligned} |chal| &= |j_1| + \dots + |j_\ell| + |r_1| + \dots + |r_\ell| \\ &= |s| + \dots + |s| + |\mathbb{F}_q| + \dots + |\mathbb{F}_q| \\ &= \ell \cdot (|s| + |\mathbb{F}_q|). \end{aligned}$$

However, the corresponding challenge message in the system [23] consists of

$$(j_1, \dots, j_\ell, r_1, \dots, r_\ell)$$

and another group elements $\phi_i \in \mathbb{G}_1, \theta_1, \dots, \theta_{s_1}, \zeta \in \mathbb{G}$ as well as a knowledge proof Λ , where \mathbb{G}_1 is the codomain of pairing from \mathbb{G} . Hence, the communication cost will be

$$\begin{aligned} |chal| &= |j_1| + \dots + |j_\ell| + |r_1| + \dots + |r_\ell| \\ &\quad + |\phi_i| + |\theta_1| + \dots + |\theta_{s_1}| + |\zeta| + |\Lambda| \\ &= \ell \cdot (|s| + |\mathbb{F}_q|) + |\mathbb{G}_1| + (1 + s_1) \cdot |\mathbb{G}| + |\Lambda| \\ &= \ell \cdot (|s| + |\mathbb{F}_q|) + (2 + s_1) \cdot |\mathbb{G}| + |\Lambda|. \end{aligned}$$

Similarly, both the communication costs from data owners in [20] and [18] equal to $\ell \cdot (|s| + |\mathbb{F}_q|)$.

4) RESPONSES FROM CSP TO FOG NODE (ReCSPtoFN)

After receiving a challenge message $chal$ from fog node (or TPA in [23]), the CSP will compute and return a proof Γ based on this $chal$ as well as the stored data-tag pairs. Hence, the communication cost in our system equals to the length of $\Gamma = (v, T) \in \mathbb{F}_q^{n+s} \times \mathbb{F}_q$ (i.e. $|\Gamma| = (n + s + 1) \cdot |\mathbb{F}_q|$). Accordingly, the proof Γ in [23] consists of an element $\Theta \in \mathbb{G}_1$ and hence its communication cost is $|\mathbb{G}_1| = |\mathbb{G}|$. In addition, we can compute the length of the responses from CSP to the verifier (i.e. fog nodes or TPA) in [20] and [18] respectively equal to $2 \cdot |\mathbb{F}_q|$ and $|\mathbb{F}_q| + |\mathbb{G}|$.

All the comparisons of communication costs for the four protocols are summarized in Table 2, where " \times " denotes the corresponding item is not considered.

B. COMPUTATIONAL COSTS

Now, we consider the computational costs of our system. In particular, we set the parameter q as 2^8 so that each element in \mathbb{F}_q can be represented in 1 Byte. Let $n = 1024$ and hence each data block $m_{i,j}$ has size of 1024 Bytes (i.e. 1 M). In addition, we choose $s_1 = 100, s = 150$, which means that each fog node will obtain 100 data blocks from 100 mobile sinks and it will generate 50 additional blocks.

Since Tian et al.'s system is based on different techniques from ours, we can not instantiate them in a same frame. But all the experiments are done based on the same hardware device. That is, a laptop with the configuration of Intel(R) Core(TM) i7-7500 CPU 2.7GHz 2.9 GHz and 8 GB RAM. The experiment on our system is run in Matlab with version R2014a since all the computations consist of linear operations of vectors. We also implement Tian et al.'s system within the framework of "Charm" [1] and choose the 512-bit SS elliptic curve from pairing-based cryptography library [17] as the basis of whole scheme. Then the running time for all the

TABLE 3. The comparisons of running time for our and other systems.

Systems	TagGenMS' (s)	TagGenFN' (s)	FG-Verify1 (s)	ChalGen' (s)	ProofGen' (s)	FG-Verify2 (s)
Ours	0.89	127.7	77	0.001	1.22	1.76
[23]	216.3	19364.8	19487.9	21.2	400.16	491.04
[20]	"×"	129.4	"×"	0.0012	1.31	1.87
[18]	"×"	15877.5	"×"	0.0011	282.6	293.4

processes are listed in the Table 3. We remark that, in Table 3, TagGenMS' is the tag-generation algorithm for single data block, which is run by mobile sink. TagGenFN' is the tag-generation algorithm for 150 data blocks. FG-Verify1 denotes fog node's verification of data-tag pairs transmitted from 100 mobile sinks while FG-Verify2 means the verification on the CSP's response. In [23], the check of CSP's response is done by TPA. However, in our system, the fog node verifies it. Finally, ChalGen' and ProofGen' are respectively the generations of challenge messages for FG (or TPA in Tian et al.'s scheme) and proof for CSP.

In addition, we also implement the two traditional auditing schemes in [20] and [18] based on the same hardware devices and software platform, respectively. Still note that, there are not mobile sinks in traditional cloud storage and hence the running time of TagGenMS' and FG-Verify1 is null, which is denoted by "×" in Table 3.

From Table 3, we know that our proposed system obviously outperforms that of Tian et al. in terms of computational efficiency. Moreover, our protocol is suitable for fog-to-cloud computing and hence prior to the two schemes in [18], [20], which are designed only for traditional cloud storage.

VII. CONCLUSION

In this paper, we propose an efficient auditing system for fog-to-cloud computing. Although our system is not public auditing, it obviously outperforms the one proposed by Tian et al. in terms of communication and computational efficiencies. The simulation results illustrate the computational efficiency. We believe that our proposed system must be an interesting choice for securely storage of data in fog-to-cloud computing.

REFERENCES

- [1] J. A. Akinyele, C. Garman, I. Miers, M. W. Pagano, M. Rushanan, M. Green, and A. D. Rubin, "Charm: A framework for rapidly prototyping cryptosystems," *J. Cryptogr. Eng.*, vol. 3, no. 2, pp. 111–128, Jun. 2013.
- [2] S. Agrawal and D. Boneh, "Homomorphic MACs: MAC-based integrity for network coding," in *Applied Cryptography and Network Security*, (Lecture Notes in Computer Science), vol. 5536. Berlin, Germany: Springer, 2009, pp. 292–305.
- [3] G. Ateniese, R. Burns, R. Curtmola, Joseph Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, Alexandria, VA, USA, 2007, pp. 598–609.
- [4] G. Ateniese, S. Kamara, and J. Katz, "Proofs of storage from homomorphic identification protocols," in *Advances in Cryptology*. Berlin, Germany: Springer, 2009, pp. 319–333.
- [5] A. F. Barsoum and M. A. Hasan, "Provable multicopy dynamic data possession in cloud computing systems," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 3, pp. 485–497, Mar. 2015.
- [6] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st Ed. MCC Workshop Mobile Cloud Comput.*, New York, NY, USA, 2012, pp. 13–16.
- [7] K. Bowers, A. Juels, and A. Oprea, "Proofs of retrievability: Theory and implementation," in *Proc. ACM Workshop Cloud Comput. Secur.*, 2009, pp. 43–54.
- [8] J. Chang, Y. Ji, M. Xu, and R. Xue, "General transformations from single-generation to multi-generation for homomorphic message authentication schemes in network coding," *Future Gener. Comput. Syst.*, vol. 91, pp. 416–425, Feb. 2019.
- [9] J. Chang et al., "Secure network coding from secure proof of retrievability," *Sci. China Inf. Sci.*, early access, Oct. 2020.
- [10] J. Chang, H. Wang, F. Wang, A. Zhang, and Y. Ji, "RKA security for identity-based signature scheme," *IEEE Access*, vol. 8, pp. 17833–17841, 2020.
- [11] Y. Dodis, S. Vadhan, and D. Wichs, "Proofs of retrievability via hardness amplification," in *Proc. Theory Cryptogr. Conf.*, 2009, pp. 109–127.
- [12] C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *Proc. CCS*, 2009, pp. 213–222.
- [13] A. Juels and B. J. Kaliski, "PORS: Proofs of retrievability for large files," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, Alexandria, VA, USA, 2007, pp. 584–597.
- [14] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*. London, U.K.: CRC Press, 2007.
- [15] A. Kaswan, V. Singh, and P. K. Jana, "A multi-objective and PSO based energy efficient path design for mobile sink in wireless sensor networks," *Pervasive Mobile Comput.*, vol. 46, pp. 122–136, Jun. 2018.
- [16] X. Lin, J. Li, J. Wu, H. Liang, and W. Yang, "Making knowledge tradable in edge-AI enabled IoT: A consortium blockchain-based efficient and incentive approach," *IEEE Trans. Ind. Informat.*, vol. 15, no. 12, pp. 6367–6378, Dec. 2019.
- [17] B. Lynn. *The Standard Pairing Based Crypto Library*. Accessed: Jul. 27, 2016. [Online]. Available: <http://crypto.stanford.edu/pbc>
- [18] M. Kashif and L. Mohammed, "Secure third party auditor (TPA) for ensuring data integrity in fog computing," *Int. J. Netw. Secur. Appl.*, vol. 10, no. 6, pp. 13–24, Nov. 2018.
- [19] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, Fog et al.: A survey and analysis of security threats and challenges," *Future Gener. Comput. Syst.*, vol. 78, pp. 680–698, Jan. 2018.
- [20] H. Shacham and B. Waters, "Compact proofs of retrievability," *J. Cryptol.*, vol. 26, no. 3, pp. 442–483, 2013.
- [21] J. Shen, J. Shen, X. Chen, X. Huang, and W. Susilo, "An efficient public auditing protocol with novel dynamic structure for cloud data," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 10, pp. 2402–2415, Oct. 2017.
- [22] H. Tian, Y. Chen, C.-C. Chang, H. Jiang, Y. Huang, Y. Chen, and J. Liu, "Dynamic-hash-table based public auditing for secure cloud storage," *IEEE Trans. Serv. Comput.*, vol. 10, no. 5, pp. 701–714, Sep. 2017.
- [23] H. Tian, F. Nan, C.-C. Chang, Y. Huang, J. Lu, and Y. Du, "Privacy-preserving public auditing for secure data storage in fog-to-cloud computing," *J. Netw. Comput. Appl.*, vol. 127, pp. 59–69, Feb. 2019.
- [24] T. Wang, Y. Li, G. Wang, J. Cao, M. Z. A. Bhuiyan, and W. Jia, "Sustainable and efficient data collection from WSNs to cloud," *IEEE Trans. Sustain. Comput.*, vol. 4, no. 2, pp. 252–262, Apr. 2019.
- [25] C. Wang, K. Ren, W. Lou, and J. Li, "Toward publicly auditable secure cloud data storage services," in *IEEE Netw.*, vol. 24, no. 4, pp. 19–24, Jul./Aug. 2010.
- [26] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 5, pp. 847–859, May 2011.
- [27] J. Wu, M. Dong, K. Ota, J. Li, W. Yang, and M. Wang, "Fog-computing-enabled cognitive network function virtualization for an information-centric future Internet," *IEEE Commun. Mag.*, vol. 57, no. 7, pp. 48–54, Jul. 2019.
- [28] Y. Wu, Z. L. Jiang, X. Wang, S. Yiu, and P. Zhang, "Dynamic data operations with deduplication in privacy-preserving public auditing for secure cloud storage," in *Proc. IEEE Int. Conf. Comput. Sci. Eng. (CSE) IEEE Int. Conf. Embedded Ubiquitous Comput. (EUC)*, Jul. 2017, pp. 21–24.

- [29] Y. Yu, M. H. Au, G. Ateniese, X. Huang, W. Susilo, Y. Dai, and G. Min, "Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 4, pp. 767–778, Apr. 2017.
- [30] Y. Yu, L. Xue, M. H. Au, W. Susilo, J. Ni, Y. Zhang, A. V. Vasilakos, and J. Shen, "Cloud data integrity checking with an identity-based auditing mechanism from RSA," *Future Gener. Comput. Syst.*, vol. 62, pp. 85–91, Sep. 2016.
- [31] J. Yuan and S. Yu, "Public integrity auditing for dynamic data sharing with multiuser modification," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 8, pp. 1717–1726, Aug. 2015.
- [32] T. Zhang. *Fog Boosts Capabilities to Add More Things Securely to the Internet*. Accessed: Mar. 3, 2016. [Online]. Available: <http://blogs.cisco.com/innovation/fog-boosts-capabilities-to-add-more-things-securely-to-the-internet>
- [33] R. Zhang, H. Ma, Y. Lu, and Y. Li, "Provably secure cloud storage for mobile networks with less computation and smaller overhead," *Sci. China Inf. Sci.*, vol. 60, no. 12, 2017, Art. no. 122104.
- [34] 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 Trans. Services Comput.*, vol. 6, no. 2, pp. 227–238, Apr. 2013.



WEI SI received the master's degree in management from Changan University, China, in 2013. He is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Xi'an Jiaotong University, China. His research interests include high-performance computing and big data storage systems.

• • •



XINGJUN ZHANG received the Ph.D. degree in computer architecture from Xi'an Jiaotong University, China, in 2003. From January 2004 to December 2005, he was a Postdoctoral Fellow with the Computer School, Beihang University, China. From February 2006 to January 2009, he was a Research Fellow with the Department of Electronic Engineering, Aston University, U.K. He is currently a Full Professor and the Dean of the School of Computer Science and Technology, Xi'an Jiaotong University. His research interests include high-performance computing, big data storage systems, and machine learning acceleration.