IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Secure IoT Data Outsourcing with Aggregate Statistics and Fine-Grained Access Control

LING LIU[1], HE WANG[1], AND YUQING ZHANG[2,1]

[1]School of Cyber Engineering, Xidian University, Xi'an 710071, China

[2]National Computer Network Intrusion Protection Center, Graduate University of Chinese Academy of Sciences, Beijing 100049, China

Corresponding author: Yuqing Zhang (e-mail: zhangyq@ucas.ac.cn).

This work was supported in part by the National Key R&D Program of China under Grant 2016YFB0800700, in part by the National Natural Science Foundation of China under Grant U1836210 and Grant 61572460, in part by the Open Project Program of the State Key Laboratory of Information Security under Grant 2017-ZD-01, and in part by the National Information Security Special Projects of the National Development and Reform Commission of China under Grant (2012)1424.

ABSTRACT With the rapid development of the Internet of Things (IoT) and the dramatic increase of IoT devices, the desire to outsource huge amounts of IoT data to the cloud becomes more urgent than ever. In order to ensure the confidentiality, IoT data are usually encrypted before they are outsourced to the cloud, which will inevitably hinder the statistical analysis of them. Homomorphic encryption is an alternative to achieve the computation of encrypted data, but its inefficiency makes it not practical in the IoT environment. Another problem comes with the encryption is how to enable IoT data to be accessed by users who possess a certain set of attributes defined by data owners. In this paper, we propose a novel and practical IoT data outsourcing scheme based on Corrigan-Gibbs et al. 's computation of aggregate statistics and the ciphertext-policy attribute-based encryption (CP-ABE). It supports both secure aggregation and fine-grained access control of outsourced IoT data. Users only have to bear a small amount of computation in the process of data upload and recovery. Security analysis demonstrates that our scheme well protects the confidentiality of IoT data. A thorough and detailed performance comparison shows that our scheme enjoys a better performance on both the client side and the fog server side.

INDEX TERMS Access control, aggregation, data confidentiality, data outsourcing, fog computing, Internet of Things.

I. INTRODUCTION

INTERNET of Things (IoT) is developing rapidly and the use of IoT devices has dramatically increased in recent years. It was forecasted that the IoT market would grow from more than 15 billion devices in 2015 to more than 75 billion in 2025 [1]. These devices have the potential to improve the living standard of their users significantly through interactions with the physical and digital worlds [2]. For example, users with smart home and wearable devices can obtain seamless and customized services from digital housekeepers, doctors, and fitness instructors [3]. Managing a constant stream of data collected from a variety of devices is a significant burden for IoT users with limited storage and computing resources. The "pay-as-you-go" Cloud Computing model is an efficient alternative to manage data for customers. Users can outsource a large amount of IoT data to the cloud and recover whenever they need it. However, since IoT embeds different kinds of sensors and other devices into a variety of things in our daily life, IoT data usually involves much private information about users [12]. It might be the heart rate of the user at a certain moment collected from the smart sphygmomanometer, user exercise data collected from the smart watch, and the like. In order to protect the security of the outsourced data, an intuitive way is to encrypt the data before outsourcing it. But there will be some new problems coming with encryption.

The first challenge is how to perform aggregate statistical analysis on encrypted data as accurate as possible. For example, we may want to learn about our health condition in a certain period of time or whether our exercise has reached the average level of people in a certain area [10] [11]. Several attempts have been made to solve this problem.

Sun et al. [12] use homomorphic encryption to encrypt the IoT data so that the service providers can process the needs of users without acquiring the plaintext data. However, the homomorphism technology is not mature currently. As we all know, its encryption and decryption are quite inefficient, and only a few homomorphism properties are supported by the privacy homomorphism. All these bottlenecks will hinder the widespread use of the Internet of Things. What's more, the cloud service provider process all the data analysis, which will inevitably cause transmission latency and degraded service when traffic between IoT devices and the cloud becomes extraordinary huge. The most important is that the trusted third party is in charge of homomorphism encryption and decryption, meaning that it can obtain all the plaintext data. Such a strong security assumption is quite problematic. In addition to homomorphic encryption, there is no better solution yet.

The second problem is how to achieve precise access control of encrypted data. The data owner may want to define an access policy and enable users that satisfy the policy to access the corresponding data. The attribute-based encryption (ABE) is a promising approach to realize this. It enables the data owner to define access policy over a universe of attributes that the user needs to possess in order to decrypt the ciphertext and enforce it on the data. ABE has two variants, key-policy attribute-based encryption (KP-ABE) [7] and ciphertext-policy attribute-based encryption (CP-ABE) [8]. The latter turns out to be well suited for access control in IoT due to its expressiveness in describing access policy of ciphertext. Huang et al. [9] achieves secure data access control using exactly the ciphertext-policy attribute-based encryption (CP-ABE) in IoT data outsourcing. But they did not consider aggregation of these encrypted data, thus data analysis is impossible.

Besides, due to the extraordinary huge volume of traffic between IoT devices and the cloud, the centralized cloud computing systems will suffer from unbearable transmission latency and degraded service. Fortunately, fog computing is a promising technology to solve this problem. Fog computing extends the cloud computing paradigm to the edge of the network, and it is characterized by low latency and widespread geographical distribution [5] [6]. Fog computing is actually a tool for cloud-based services that can be considered as an interface between the users and the cloud. In this paper, we also adopt fog as an auxiliary tool and propose a secure IoT data outsourcing scheme. As far as we know, our scheme is the first to achieve encrypted data aggregation and precise access control simultaneously in IoT data outsourcing. The main contributions of this paper are summarized as follows.

- We propose a novel and practical IoT data outsourcing scheme. Specifically, we make a combination of Corrigan-Gibbs et al.'s computation of aggregate statistics and Bethencourt et al.'s ciphertext-policy attribute-based encryption (CP-ABE) to support both secure aggregate statistics and fine-grained access control of outsourced IoT data. And the introduction of fog computing enables our scheme to provide real-time and low-latency services.
- Security analysis demonstrates that our scheme well protects the confidentiality of outsourced IoT data, and ensures that only users whose attributes sets satisfy the access policy can recover the corresponding data.
- A comprehensive performance comparison between our scheme and several present works is given, showing that our scheme behaves better in the computation overhead.

II. RELATED WORK

A. AGGREGATION OF OUTSOURCED IOT DATA

The secure aggregation of outsourced IoT data is an urgent need, but there are very few works that have solved this problem. In fact, a large number of researchers have begun to pay attention to security and privacy issues on the internet of things (IoT) [17] [18] [19]. Fan et al. [20] designed secure and privacy-preserving RFID protocols from the perspective of data collection and transmission, to make sure that the IoT data collected by RFID tags could only be transmitted to legitimate readers. Doukas et al. [11] proposed to encrypt IoT data using public key encryption to ensure the confidentiality of data. But as we all know, the computation overhead of public key encryption is extremely huge, especially in Internet of Things where quite a lot of devices are limited in computation capability. And none of the above schemes consider the aggregation and analysis of encrypted IoT data. Both Sun et al. [12] and Gong et al. [21] attempted to protect data using homomorphic encryption. Specifically, Sun et al. introduced a trusted third party to encrypt and decrypt IoT data for resource-constrained users. It means that all the private data is transparent to the trusted third party. Gong et al.'s scheme is a bit simpler, and encryption and decryption are all on the user side. But they can only decide if the result is in the region [0,1]. Another problem with these two works is that the cloud server engages in the computation of every data, meaning that the cloud needs to interact with countless IoT users frequently and can become the bottleneck of the whole system easily. Recently, Guan et al. [22] proposed an anonymous and privacy preserving data aggregation scheme for fog-enhanced IoT systems. They adopted paillier encryption, which could achieve additive homomorphism property, to outsource the aggregation of IoT data to the cloud through fog nodes. However, the smart devices still need to bear expensive computation in the process of data collection and data aggregation.

B. ACCESS CONTROL OF OUTSOURCED IOT DATA

Secure access control is another desirable function in IoT data outsourcing since IoT users may want to share their data with certain people. Attribute-based cryptography is a well-known technology to guarantee data confidentiality and fine-grained data access control. Early in 2011, Yu et al. [23] adopted KP-ABE to achieve fine-grained data access control in wireless sensor networks. Then Hu et al. [24] employed CP-ABE to realize secure data communication

between wearable sensors and data consumers. Yeh et al. [25] proposed a cloud-based fine-grained health information access control framework, which was the first scheme suitable for lightweight IoT devices. Only symmetric cryptography is required for IoT devices, such as wireless body sensors. However, the computational cost in the encryption and decryption phase is linear with the complexity of policy when using ABE in fog computing directly. Zhang et al. [26] and Huang et al. [9] introduced fog computing to alleviate the burden of IoT users. They outsourced expensive encryption and decryption operations partially to the fog servers, so that the computation that users needed to do in encryption and decryption was irrelevant to the number of attributes in the policies. Even so, users still have to bear a certain amount of computation, and such an encryption will inevitably hinder the aggregation and statical analysis of data. To our best knowledge, there is no scheme that supports both fine-grained access control and encrypted data aggregation in the context of Internet of Things (IoT).

C. FOG COMPUTING IN INTERNET OF THINGS

In this paper, we adopt fog computing as the intermediate layer between IoT users and the cloud to alleviate the burden of cloud as many previous works [4] [5] [6]. The main characteristics of fog computing include low latency, location awareness, wide-spread geographical distribution, and so on. So it can play an essential role in the Internet of Things (IoT) such as healthcare and activity tracking, connected vehicle, and smart grid. [13] proposed a distributed dataflow programming model as a basis for fog-based IoT applications. They discussed the core requirements that fog-based IoT applications needed to meet and identified several issues that had not been considered in previous works. The above works focus primarily on the principles and concepts of fog computing and its significance in the context of internet of things (IoT). Sarkar et al. [14] first proposed a mathematical formulation for fog computing and proved its significance by experiment. They showed that for a scenario where 25% of the IoT applications demand real-time and low-latency services, the mean energy expenditure in fog computing was 40.48% less than the conventional cloud computing model. Farahani et al. [15] discussed the applicability of IoT in healthcare and medicine specifically, and they presented a holistic architecture of IoT eHealth ecosystem. Oteafy et al. [16] gave a recent survey on advancements and challenges in IoT, and they also presented a number of high-yield directions that would further propagate IoT development in the fog.

III. PRELIMINARY

A. PRIO: CORRIGAN-GIBBS'S COMPUTATION OF AGGREGATE STATISTICS

Corrigan-Gibbs et al. [27] gives a simplified version and an extended version of Prio, respectively. The former does not provide robustness, meaning that a single malicious client can corrupt the protocol output completely by submitting an invalid value, while the latter provides robustness. In this paper, we just adopt the simplified version of Prio. It is because that the simplified version also provides privacy, i.e., the servers can only learn the result of aggregation but nothing else about the clients' private inputs. The other reason is that in our IoT data outsourcing scheme, clients may want to recover their data stored in the cloud at any moment, so we assume that the clients would not like to upload invalid values.

Suppose each client holds a private value x_i and that servers want to compute the sum of clients' private values $\sum_i x_i$. The simplified Prio scheme for computing sums proceeds in three steps:

- Upload. Each client i splits its private value x_i into s shares, one per server, using a secret-sharing scheme. In particular, the client picks random values $x_{i,1}, \ldots x_{i,s}$, subject to the constraint: $x_i = x_{i,1} + \cdots + x_{i,s}$. The client then sends, over an encrypted and authenticated channel, one share of its submission to each server.
- Aggregate. Each server j computes the sum of its own shares $\sum_i x_{i,j} = S_j$.
- Publish. All servers publish their values S_j, and the sum $\sum_i x_i$ will be $\sum_j S_j$.

B. BEAVER'S MULTI-PARTY COMPUTING (MPC) PROTOCOL

As in Corrigan-Gibbs et al.'s Prio [27], the implementation of multiplication needs a combination with Beaver's multi-party computing (MPC) protocol [28], which proceeds as follows.

Suppose servers want to compute the product of a private value x and a constant A, and that each server i holds a share $[x]_i$, then ith server can compute a share of Ax locally by multiplying $[x]_i$ by A, namely $[Ax]_i = A[x]_i$.

Suppose servers want to compute the product of two private values x and y, and that each server i holds the shares $[x]_i$ and $[y]_i$. Beaver showed that servers could use precomputed multiplication triples to implement the multiplication. A multiplication triple is a one-time-use triple of values (a, b, c) chosen at random, being subject to the constraint that $a \cdot b = c$. Each server i holds a share $([a]_i, [b]_i, [c]_i)$ of the triple to jointly compute the product xy. To do so, each server i uses its shares $[x]_i$ and $[y]_i$ along with the first two components of its multiplication triple to compute the following values: $[d]_i = [x]_i - [a]_i$, $[e]_i = [y]_i - [b]_i$, and then it broadcasts $[d]_i$ and $[e]_i$. Using the broadcasted shares, every server can reconstruct d and e and compute: $\sigma_i = de/s + d[b]_i + e[a]_i + [c]_i$, where s is the number of servers - a public constant. Thus σ_i is a share of the product xy since $\sum_i \sigma_i = \sum_i (de/s + d[b]_i + e[a]_i + [c]_i) = de + db + ea + c = (x-a)(y-b) + (x-a)b + (y-b)a + c = xy - ab + c = xy$.

C. ACCESS TREE

Let $\mathcal{T}$ be a tree representing an access structure. Each non-leaf node of the tree represents a threshold gate, described by its number of children num_x and a threshold value k_x, where $0 < k_x \leq num_x$. When $k_x = 1$, the threshold gate is an OR

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2961413, IEEE Access

gate and when $k_x = num_x$, it is an AND gate. Each leaf node x of the tree is described by an attribute and a threshold value $k_x = 1$. Besides, the parent of the node x in the tree is denoted as $parent(x)$, and the attribute associated with the leaf node x in the tree is denoted as $attr(x)$. The access tree $\mathcal{T}$ also defines an ordering between the children of every node by numbering the children of the node from 1 to num. The function $index(x)$ returns such a number associated with the node x, where the index values are uniquely assigned to nodes in the access structure for a given key in an arbitrary manner.

Let $\mathcal{T}_x$ be a sub tree of $\mathcal{T}$ rooted at the node x. If a set of attributes S satisfies the access tree $\mathcal{T}_x$, we denote it as $\mathcal{T}_x(S) = 1$. We compute $\mathcal{T}_x(S)$ recursively as follows. If x is a non-leaf node, evaluate $\mathcal{T}_{x'}(S)$ for all children x' of node x. $\mathcal{T}_x(S)$ returns 1 if and only if at least k_x children return 1. If x is a leaf node, $\mathcal{T}_x(S)$ returns 1 if and only if $attr(x) \in S$.

D. CIPHERTEXT-POLICY ATTRIBUTE-BASED ENCRYPTION (CP-ABE)

In the ciphertext-policy attribute-based encryption (CP-ABE) system, a user's private key is associated with an arbitrary number of attributes expressed as strings, and the encrypting party specifies an associated access structure over attributes when he encrypts a message. Only if a user's attributes set satisfies the access structure, is he able to decrypt the corresponding ciphertext. A ciphertext-policy attribute-based encryption scheme consists of the following four fundamental algorithms:

- $Setup(\lambda) \rightarrow (PK, MK)$ is the key generation algorithm that takes the security parameter λ as input and outputs the public parameters PK and a master key MK;
- $Enc(PK, m, \mathcal{T}) \rightarrow C$ is the encryption algorithm that takes the public parameters PK, a message m, and an access structure $\mathcal{T}$ over the universe of attributes as inputs and outputs a ciphertext C;
- $KeyGen(PK, MK, S) \rightarrow SK$ is the key generation algorithm that takes the public parameters PK, the master key MK, and a set of attributes S that describe the key as inputs and outputs a private key SK.
- $Dec(PK, C, SK) \rightarrow m$ is the decryption algorithm that takes the public parameters PK, a ciphertext C, and a private key SK as inputs and outputs the original message m only if the set S of attributes satisfies the access structure $\mathcal{T}$.

IV. PROBLEM FORMULATION

A. SYSTEM MODEL

There are five types of entities in our IoT data outsourcing model, including data owners, trusted authority (TA), cloud service provider (CSP), fog servers, and users, as shown in Fig. 1. The black arrows and blue arrows in the figure represent the information transmission in the process of data upload and data recovery, respectively. Note that the data recovery process begins with the download request initiated by IoT users. The concrete role that each entity play is described as follows.

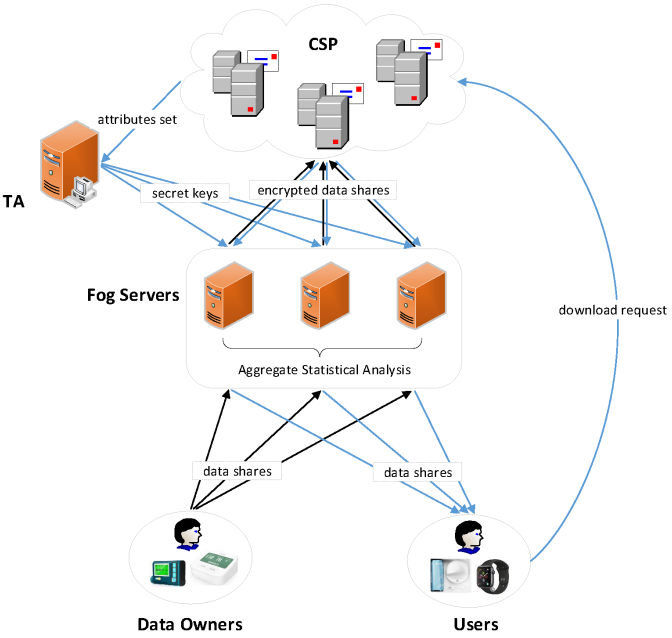


FIGURE 1: System Model

- The data owner has a considerable amount of data collected from IoT devices, and he is limited in computation and storage ability. So he wants to outsource the data to the cloud and enable legal users to access them later. Besides this, he also wants the fog servers to do some real-time aggregate statistical analysis either for his own data or for his and others' data together.
- The Trusted Authority (TA) is a party fully trusted by all the other parties. It is in charge of generating system parameters, as well as generating CP-ABE secret keys and pre-compute one-time-use multiplication triples for fog servers.
- The Cloud Service Provider (CSP) is an entity that provides cloud storage service. Explicitly, it stores t shares of ciphertext for each IoT data, where t is the number of fog servers that help encrypt the plaintext. It is also responsible for verifying if the attributes set of a user who wants to recover a data satisfies the access tree defined by the data owner.
- Fog servers are nodes deployed at the network edge. They can offer a variety of services, such as helping encrypt data shares and upload ciphertexts to the CSP for storage. What's more, some real-time data operations such as the summation, the product, and the variance are also their responsibility.
- The user equipped with IoT devices is also limited in computation and storage ability. Only when his attributes set satisfies the access policy defined by the data owner, can he gain the access of the corresponding ciphertext data stored in the CSP.

B. SECURITY MODEL

Our threat model considers two types of attackers: 1) An inside attacker refers to the CSP or fog servers. They are assumed to be "honest-but-curious" in our scheme, namely that they will execute the assigned tasks honestly, but would like to learn as much secret information as possible. For instance, they may attempt to extract useful information of data. In addition to this, we allow the collusion between fog servers. However, we require that the number of colluded fog servers is not more than $t-2$ if the number of fog servers is t, namely at least two fog servers are honest, such that original data cannot be guessed by the colluded fog servers. 2) An outside attacker refers to a malicious user that intends to obtain some knowledge about data, which is not owned by him nor does he have access to. Therefore, all the sensitive IoT data need to be fully protected against both inside attackers and outside attackers.

Moreover, pairwise authenticated and encrypted channels must be established between each IoT client and each fog server to protect data shares. Toward this end, we assume the existence of a public infrastructure and the basic cryptographic primitives (public-key encryptions, digital signatures, etc.) that make secure channels possible.

C. DESIGN GOALS

In this paper, we construct a secure IoT data outsource scheme supporting fine-grained access control and data aggregation. Specifically, the scheme aims at achieving the following security goals and functions.

- Data Confidentiality. Our scheme should guarantee the confidentiality of data, implying that the user whose attributes set does not satisfy the access tree of the data can not get any knowledge about it. Besides, fog servers can not know or infer the data that are being operated in the process of aggregate statistical analysis. The data should also be protected well against the curious CSP, even the trusted authority (TA).
- Fine-Grained Access Control. The data owner can define expressive and flexible policies so that the data can only be accessed by the users whose attributes satisfy these policies.
- Real-Time Data Aggregation. Some real-time data aggregation such as the addition, the multiplication, and the variance can be realized with the fog servers as an intermediate layer between users and the CSP. Besides, the computational complexity on the user side should be as small as possible.

V. SCHEME DESCRIPTION

Our scheme takes two aspects of both secure storage and computation of outsourced IoT data into consideration, and achieves fine-grained access control simultaneously. When a user wants to upload a data D to the CSP, he first splits it into t shares using a secret-sharing scheme, where t is the number of fog servers. For example, he can choose t random numbers such that their sum equals D. He also defines an access tree $\mathcal{T}$ for data D so that only the user whose attributes set S satisfies $\mathcal{T}$ can recover D later. Then he sends each share along with $\mathcal{T}$ to a fog server through an encrypted and authenticated channel. The fog servers will store shares of multiple data temporally so that they can jointly compute the summation, the product, and the variance of these data, without revealing the original data. They also use ciphertext-policy attribute-based encryption (CP-ABE) and symmetrical encryption to encrypt their shares, then send the encrypted data shares to the CSP for storage. The CSP finally stores a set of encrypted data shares and an access tree for data D. If a user with attributes set S wants to access a data D stored on the CSP, he sends a download request to the CSP. Once the CSP has checked that S satisfies the access tree $\mathcal{T}$ of data D, it sends S to the Trusted Authority (TA) and each encrypted data share to the fog server who uploaded it. Then TA generates a CP-ABE secret key using attributes set S and sends it to each fog server. Upon receiving the encrypted data share from the CSP and the CP-ABE secret key from the TA, each fog server can decrypt the data share and sends it to the user respectively over the encrypted and authenticated channel. Finally, the user can recover the data easily. The construction details of our scheme are described as follows.

A. SYSTEM SETUP

In this phase, two encryption schemes that will be used in our construction are initialized as follows: (1) A symmetric encryption scheme with the primitive functions $(KeyGen_{SE}, Enc_{SE}, Dec_{SE})$; (2) A ciphertext-policy attribute-based encryption (CP-ABE) scheme consisting of four algorithms $(Setup_{CP-ABE}, Enc_{CP-ABE}, KeyGen_{CP-ABE}, Dec_{CP-ABE})$. Specifically, the $Setup_{CP-ABE}$ algorithm will produce the following materials: public parameters $PK = (\mathbb{G}_0, g, h = g^\beta, e(g,g)^\alpha)$, where $\mathbb{G}_0$ is a bilinear group of prime order p, g is a generator of $\mathbb{G}_0$, $e : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$ denotes the bilinear map, $\alpha, \beta \in \mathbb{Z}_p$ are two random exponents. We also define the Lagrange coefficient $\Delta_{i,S}$ for $i \in \mathbb{Z}_p$ and a set S of elements in $\mathbb{Z}_p$: $\Delta_{i,S}(x) = \prod_{j \in S, j \neq i} \frac{x-j}{i-j}$. We will employ a hash function $H : \{0,1\}^* \rightarrow \mathbb{G}_0$ additionally that we will model as a random oracle. The master key MK is (β, g^α), which is kept by the trusted authority (TA).

B. DATA UPLOAD

1. Supposing that the user i wants to upload a private data D_i, he first splits it into t shares using a secret-sharing scheme, one per fog server. Specifically, the user chooses t random values $(D_{i,1}, ..., D_{i,t})$ subject to the constraint: $D_i = D_{i,1} + ... + D_{i,t}$. Then he sends each share of D_i along with the access policy $\mathcal{T}_i$ he defined to a fog server over an encrypted and authenticated channel.

2. Upon receiving the secret share $D_{i,j}$, each fog server j stores the share temporarily (hours, days, or weeks, depending on the needs) for data aggregation such as addition and multiplication. In addition to this, it CP-ABE-encrypts and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2961413, IEEE Access

then symmetrically encrypts the share using its secret key k_j. The specific encryption process is as follows.

(1)The fog server j invokes the Enc_{CP-ABE} algorithm to encrypt the share $D_{i,j}$ under the access $\mathcal{T}_i$. It first chooses a polynomial p_x for each node x in $\mathcal{T}_i$ in a top-down manner starting from the root node R_i. For each node x in the tree, let the degree d_x of the polynomial p_x to be one less than the threshold value k_x of that node, that is, $d_x = k_x - 1$. Starting with the root node R_i, the algorithm chooses a random $s_{i,j} \in \mathbb{Z}_p$ and sets $p_{R_i}(0) = s_{i,j}$. Then, it chooses d_{R_i} other points of the polynomial p_{R_i} randomly to define it completely. For any other node x, it sets $p_x(0) = p_{parent}(x)(index(x))$ and chooses d_x other points randomly to define p_x completely. Besides, the fog server j chooses another random $DK_{i,j} \in \mathbb{Z}_p$ as the key to symmetrically encrypt $D_{i,j}$. Let Y_i be the set of leaf nodes in $\mathcal{T}_i$, the fog server j outputs a partial ciphertext

$$\begin{aligned}&\widetilde{CT}_{i,j} = \{\mathcal{T}_i, \widetilde{C}_{i,j} = DK_{i,j} \cdot e(g,g)^{\alpha s_{i,j}},\\ &c_{i,j} = Enc_{DK_{i,j}}(D_{i,j}), C'_{i,j} = h^{s_{i,j}},\\ &\forall y \in Y_i : C_{i,j,y} = g^{p_y(0)}, C'_{i,j,y} = H(attr(y))^{p_y(0)}\}.\end{aligned} \tag{1}$$

(2)The fog server j uses its secret key k_j to symmetrically encrypt $\widetilde{C}_{i,j}$ computed in the previous step and obtains the new ciphertext

$$\begin{aligned}&CT_{i,j} = \{\mathcal{T}_i, C_{i,j} = Enc_{k_j}(DK_{i,j} \cdot e(g,g)^{\alpha s_{i,j}}),\\ &c_{i,j} = Enc_{DK_{i,j}}(D_{i,j}), C'_{i,j} = h^{s_{i,j}},\\ &\forall y \in Y_i : C_{i,j,y} = g^{p_y(0)}, C'_{i,j,y} = H(attr(y))^{p_y(0)}\}.\end{aligned} \tag{2}$$

Finally, fog server j uploads $CT_{i,j}$ to the cloud server.

3. Upon receiving ciphertexts of all the shares of D_i from t fog servers, the cloud server stores $CT_i = \{CT_{i,j}\}_{j=1,...,t}$ for each data D_i. Note that the first component of $CT_{i,j}$ uploaded by each fog server j is the same, namely $\mathcal{T}_i$, the cloud server just needs to store only one.

C. DATA AGGREGATION

1) Addition

Suppose n data $(D_1, D_2, ..., D_n)$ are to be added, and that each fog server $j (j = 1, 2, ..., t)$ stores a set of data shares $DS_j = \{D_{i,j} | i \in 1, 2, ..., n\}$. The addition is implemented as following: each fog server j computes $S_j = D_{1,j} + D_{2,j} + ... + D_{n,j}$ and publishes S_j. Then the sum of these n data is $S = S_1 + S_2 + ... + S_t = D_1 + D_2 + ... + D_n$. If the mean value of these data is needed, it can be easily computed by diving S by n.

2) Multiplication

When fog servers need to compute the product P of a constant A and a data D they stored together, each fog server j can compute a share P_j of P by multiplying D_j they stored with A, that is $P_j = A \cdot D_j$. Then $P = P_1 + P_2 + ... + P_t = A \cdot D_1 + A \cdot D_2 + ... + A \cdot D_t = A \cdot (D_1 + D_2 + ... + D_t) = A \cdot D$, where t is the number of fog servers. Note that in this case, the private data D can be easily inferred by fog servers, but in real-life, fog servers usually need to compute more complex algebraic formulas such as $A \cdot D_1 + B \cdot D_2$ or $A \cdot D_1 + D_2 \cdot D_3$, and we just give the basic multiplication of a constant and a private data here.

The multiplication of two data D_1 and D_2 can be implemented through Beaver's multi-party computing (MPC) protocol. Suppose each fog server j holds a share (a_j, b_j, c_j), where (a, b, c) is a one-time-use multiplication triple pre-computed by the trusted authority (TA) and subject to the constraint that $a \cdot b = c$. Fog server j computes $d_j = D_{1,j} - a_j$, $e_j = D_{2,j} - b_j$ and then broadcasts d_j and e_j. Thus each fog server j can construct d and e and further compute $P_j = de/t + db_j + ea_j + c_j$. Then fog server j publishes P_j and the product of D_1 and D_2 will be $P = P_1 + P_2 + ... + P_t = D_1 \cdot D_2$.

3) Variance

If we need to know the variance of n data $(D_1, D_2, ..., D_n)$, fog servers can compute the variance $V = \overline{D_i^2} - \overline{D_i}^2$, $i = 1, 2, ..., n$ as following.

Suppose fog server j stores a share $D_{i,j}$ of D_i, and that it also holds a set of shares $\{(a_{i,j}, b_{i,j}, c_{i,j})\}_{i=1,2,...n}$ of one-time-use multiplication triples $\{(a_i, b_i, c_i)\}_{i=1,2,...n}$, which are pre-computed by the trusted authority (TA) and subject to the constraint that $a_i \cdot b_i = c_i$.

For each data share $D_{i,j}$, the fog server j computes $S_j = D_{1,j} + D_{2,j} + ... + D_{n,j}$, $d_{i,j} = D_{i,j} - a_{i,j}$, $e_{i,j} = D_{i,j} - b_{i,j}$ and then broadcasts S_j, $d_{i,j}$ and $e_{i,j}$. In this way, all fog servers can compute $\overline{D}_i^2 = (\frac{S_1 + S_2 + ... + S_t}{n})^2$, $d_i = \sum_{j=1}^{t} d_{i,j} = D_i - a_i$, and $e_i = \sum_{j=1}^{t} e_{i,j} = D_i - b_i$.

Fog server j further computes $P_{i,j} = d_i e_i / t + d_i b_{i,j} + e_i a_{i,j} + c_{i,j}$ and publishes $P_{i,j}$. Thus $D_i^2 = \sum_{j=1}^{t} P_{i,j}$ and $\overline{D_i^2} = \frac{\sum_{i=1}^{n} (\sum_{j=1}^{t} P_{i,j})}{n}$.

Finally, fog servers can compute $V = \overline{D_i^2} - \overline{D}_i^2 = \frac{\sum_{i=1}^{n} (\sum_{j=1}^{t} P_{i,j})}{n} - (\frac{\sum_{j=1}^{t} S_j}{n})^2$.

D. DATA RECOVERY

1. When a user with an attributes set S wants to recover a data D_i stored in the cloud, he sends a download request for D_i to the cloud server. Since the server stores the access tree $\mathcal{T}_i$ of D_i, it first checks if $\mathcal{T}_i(S) = 1$. If not, the server will reject the download request. Otherwise, the server sends the attributes set S to the trusted authority (TA) and the ciphertext $CT_{i,j} (j = 1, ..., t)$ to each fog server j.

2. The trusted authority (TA) runs the $KeyGen_{CP-ABE}$ algorithm to generate the secret key. It first chooses a random $r \in \mathbb{Z}_p$ and then random $r_a \in \mathbb{Z}_p$ for each attribute $a \in S$. Then it computes the key as

$$SK = (K = g^{(\alpha + r)/\beta}, \forall a \in S : K_a = g^r \cdot H(a)^{r_a}, K'_a = g^{r_a}) \tag{3}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2961413, IEEE Access

and sends SK to all fog servers.

3. Upon receiving $CT_{i,j}$ from the cloud server and SK from the trusted authority (TA), fog server j first uses its secret key k_j to symmetrically decrypt $C_{i,j}$ in $CT_{i,j}$ by $\widetilde{C}_{i,j} = Dec_{k_j}(C_{i,j})$ and obtains $\widetilde{CT}_{i,j}$. Then it can invoke the Dec_{CP-ABE} algorithm to recover the data share $D_{i,j}$.

The Dec_{CP-ABE} algorithm is realized by a recursive algorithm $DecryptNode(\widetilde{CT}_{i,j}, SK, x)$. If the node x is a leaf node and if $a = attr(x) \in S$, then $DecryptNode(\widetilde{CT}_{i,j}, SK, x) = \frac{e(K_a, C_a)}{e(K'_a, C'_a)} = \frac{e(g^r \cdot H(a)^{r_a}, g^{p_x(0)})}{e(g^{r_a}, H(a)^{p_x(0)})} = e(g,g)^{rp_x(0)}$. If $a = attr(x) \notin S$, then $\widetilde{CT}_{i,j}, SK, x) = \perp$.

If the node x is a non-leaf node, the algorithm DecryptNode$(\widetilde{CT}_{i,j}, SK, x)$ proceeds as follows: For all nodes c that are children of x, it calls DecryptNode$(\widetilde{CT}_{i,j}, SK, c)$ and stores the output as DN_c. Let S_x be an arbitrary k_x-sized set of child nodes c such that $DN_c \neq 1$. If no such set exists, then the node is not satisfied and the function returns $\perp$. Otherwise, it computes

$$\begin{aligned} DN_x &= \prod_{c \in S_x} DN_c^{\Delta_{i,S'_x}(0)} \\ &= \prod_{c \in S_x} (e(g,g)^{r \cdot p_c(0)})^{\Delta_{i,S'_x}(0)} \\ &= \prod_{c \in S_x} (e(g,g)^{r \cdot p_{parent(c)}(index(c))})^{\Delta_{i,S'_x}(0)} \\ &= \prod_{c \in S_x} (e(g,g)^{r \cdot p_x(i) \cdot \Delta_{i,S'_x}(0)} \\ &= e(g,g)^{r \cdot p_x(0)} \end{aligned} \tag{4}$$

and returns the result, where $i = index(c)$ and $S'_x = \{index(c) : c \in S_x\}$. Therefore, if the tree $\mathcal{T}_i$ is satisfied by S, the fog server j can obtain $DN = DecryptNode(\widetilde{CT}_{i,j}, SK, R_i) = e(g,g)^{r \cdot p_{R_i}(0)} = e(g,g)^{rs_{i,j}}$ and then decrypt by computing

$$\begin{aligned} &\widetilde{C}_{i,j}/(e(C'_{i,j}, K)/DN) \\ &= DK_{i,j} \cdot e(g,g)^{\alpha s_{i,j}} / (e(h^{s_{i,j}}, g^{(\alpha+r)/\beta}) / e(g,g)^{rs_{i,j}}) \\ &= DK_{i,j}. \end{aligned} \tag{5}$$

Finally, each fog server j decrypts $D_{i,j}$ by $D_{i,j} = Dec_{DK_{i,j}}(c_{i,j})$ and sends $D_{i,j}$ to the user respectively, over the encrypted and authenticated channel.

4. After receiving all the t shares of D_i, the user is able to recover D_i by a simple addition$D_i = D_{i,1} + ... + D_{i,t}$.

VI. SECURITY ANALYSIS

Our scheme aims at achieving secure data aggregation and precise access control for outsourced IoT data. First of all, we should guarantee the correctness of data aggregation on the fog server side, meaning that the results of addition, multiplication, and variance computed by fog servers are all correct. Through the detailed description in Section V-C, and the hypothesis in the security model that fog servers are honest but curious, it is obvious that correctness is fulfilled. As is described in Section IV-C, the security goal of the scheme is to realize data confidentiality and fine-grained access control. We will demonstrate the security of our scheme in these two aspects in the security analysis below.

A. DATA CONFIDENTIALITY

In our scheme, an IoT data D is first split into shares $D_1, \ldots, D_t$ by its owner before being uploaded, where t is the number of fog servers and $D = D_1 + \ldots + D_t$. Then these data shares are sent to fog servers over an encrypted and authenticated channel. It can be observed that the data is secure as long as at least two fog servers do not collude. Even if fog servers $1, 2, \ldots, t-2$ collude, they can not infer D without D_{t-1} and D_t.

These t data shares will be encrypted by fog servers using the ciphertext-policy attribute-based encryption (CP-ABE) and symmetrical encryption before being uploaded to the cloud service provider (CSP) finally. Concretely, the data share D_j is symmetrically encrypted in the form $c_j = Enc_{DK_j}(D_j)$ by fog server j as in Equation 2. Here the encryption key DK_j is uniformly chosen at random in $\mathbb{Z}_p$, thus the encryption of D_j can be regarded as "one-time pad". Katz et al. have given a formal proof in [29] that one-time pad encryption scheme is perfectly-secret.

The symmetric encryption key DK_j is not kept by the fog server j in our scheme. Instead, it is first encrypted using the ciphertext-policy attribute-based encryption (CP-ABE) as in Equation 1. Bethencourt et al. have demonstrated in [8] that their CP-ABE scheme is secure against chosen plaintext attacks (CPA-Secure). They argued that no efficient adversary that acts generically on the groups underlying their CP-ABE scheme could break the security of CP-ABE scheme with any reasonable probability, and they proved their argument using the generic bilinear group model and the random oracle model. Then DK_j is further symmetrically encrypted by fog server j using its secret key k_j as in Equation 2, which will enhance the security of DK_j undoubtedly.

Another more critical consideration for further symmetrically encrypting DK_j using the fog server j's secret key is to protect DK_j against other fog servers during the data recovery process. Specifically, when a user whose attributes set S satisfies the access policy of data D wants to recover D, the trusted authority (TA) generates the secret key SK using S and sends SK to each fog server who has uploaded a share of D. If DK_j is just encrypted using CP-ABE scheme and that the ciphertext $\widetilde{CT}_j$ is, somehow, obtained by another fog server, DK_j will be decrypted by this fog server. It is because that every other fog server can recover DK_j by calling the CP-ABE decryption algorithm $Dec(PK, \widetilde{CT}_j, SK)$. But in our scheme, the final encryption of DK_j is $CT_j = \{\mathcal{T}, C_j = Enc_{k_j}(DK_j \cdot e(g,g)^{\alpha s_j}), c_j = Enc_{DK_j}(D_j), C'_j = h^{s_j}, \forall y \in Y : C_{j,y} = g^{p_y(0)}, C'_{j,y} = H(attr(y))^{p_y(0)}\}$. No other fog servers except fog server j

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2961413, IEEE Access

can decrypt C_j, nor can they call the CP-ABE decryption algorithm to obtain DK_j. Based on the above analysis, the confidentiality of data is achieved in our scheme.

B. FINE-GRAINED ACCESS CONTROL

The purpose of access control is to make sure that data can be correctly recovered by the user whose attributes set satisfies the associated access policy, while those who do not meet the policy cannot obtain anything about data. It can be easily observed in the data recovery process of our scheme that the former is achieved. For an invalid user, his attributes set can not pass through the check by the cloud on the access tree, namely $\mathcal{T}(S) \neq 1$, so he is unable to proceed the normal process of data recovery. On the other hand, even if he gets the ciphertext of a data D somehow, he can not decrypt any CT_j to obtain $\widetilde{CT}_j$, let alone recover the data.

Fine-grained is reflected in the ability to specify different access rights of individual users flexibly. By utilizing the ciphertext-policy attribute-based encryption (CP-ABE) in our scheme, the data owner is able to enforce expressive and flexible access policies. Specifically, the access policy of encrypted data supports complex operations to represent any desired attributes set. For example, we can represent a tree with "AND" and "OR" gates by using 2 of 2 and 1 of 2 threshold gates respectively. Therefore, our scheme achieves fine-grained access control by construction.

VII. PERFORMANCE ANALYSIS

In this section, we evaluate the performance of our secure IoT data outsourcing scheme, which is proposed to achieve both real-time aggregate statistics and fine-grained access control of outsourced IoT data. Since [12] [21] [22] are the very few works that consider secure IoT data aggregation, and [9] [26] are two state of the art works dealing with the access control in IoT, we will analyze our scheme in contrast to these five schemes. A overall comparison of these schemes is given in Table 1. "$\surd$" represents that the scheme supports the corresponding function, while "$\times$" the opposite. We will concentrate on the computation overhead on the client side and fog server side, including data upload and data recovery processes. In order to facilitate comparison, we choose the Advanced Encryption Standard (AES)-256 in Cipher-Block Chaining (CBC) mode as the symmetric key encryption scheme. Moreover, we utilize PBC library [30] and jPBC library [31] and choose type A pairing with 160-bit security level to conduct simulation experiments. Specifically, we adopt a desktop with Intel(R) Core(TM) i7-8700 CPU @3.20GHz and Linux version 4.19.36-1-MANJARO as a fog node, and an Android phone MI 6X with MIUI 10.3 and Android 9.0, Snapdragon 660 CPU, and 6 GB memory as the IoT device.

TABLE 1: Overall Comparison

Scheme \ Function	Data Aggregation	Fine-Grained Access Control
Sun et al. [12]	$\surd$	$\times$
Gong et al. [21]	$\surd$	$\times$
Guan et al. [22]	$\surd$	$\times$
Huang et al. [9]	$\times$	$\surd$
Zhang et al. [26]	$\times$	$\surd$
Our scheme	$\surd$	$\surd$

A. COMPUTATION OVERHEAD

We will analyze the computation overhead on the client side and fog side theoretically in this section. Let **Pair** denote one pairing operation on $e : G_1 \times G_2 \rightarrow G_T$, **Exp** denote one exponentiation in group G_1, **Mul** denote one multiplication in group G_1, and **Add** denote one addition in group $\mathbb{Z}_p$.

In Sun et al. [12], a data owner needs to perform $2n$ multiplications to encrypt a data before uploading it, where n is the number of shares that a data is divided into. Data decryption needs $2n+4$ multiplications and $2n+1$ additions. In Gong et al. [21], a data owner needs to perform 512 multiplications and 448 additions, which is derived from their adoption of eight rounds eight-order matrix multiplication. This does not include the overhead of other operations such as XOR and HASH of DES encryption. Data recovery needs the same amount multiplications and one more eight-order matrix addition. In Guan et al. [22], to enable the cloud and the fog to achieve data aggregation, the data owner needs to perform 2 exponentiations and 1 multiplication during data collection and data aggregation. Note that they outsource the data aggregation but not data storage to the cloud, thus with no need to consider data recovery. In Huang et al. [9], upon receiving the partial ciphertext computed by fog nodes, the data owner needs to perform 4 exponentiations and 3 multiplications to finally encrypt the data. In Zhang et al. [26], the computation overhead is similar, being 3 exponentiations and 3 multiplications. In the above two schemes, the computation required by the user to decrypt a data is the same, that is 2 multiplications and 1 paring operation. While in our scheme, a data owner only needs to perform $n-1$ additions to upload an IoT data, where n is the number of fog servers, also the number of shares that the data is divided into. Since data upload and data recovery are symmetric processes, the computation overhead of data recovery is also $n-1$ additions. The detailed computation overhead on the client side is listed in Table 2.

In terms of the computation overhead on the fog server side, Huang et al. [9], Zhang et al. [26] and our scheme all adopt fog computing and utilize the ciphertext-policy attribute-based encryption (CP-ABE) to accomplish fine-grained access control of IoT data, thus the computation overhead on the fog server side is related to the access policy. While Guan et al. [22] just use fog nodes to perform outsourced aggregation of IoT data, with no need to consider uploading data to the cloud or helping users recover data from the cloud. And the the computation overhead on the fog side

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2961413, IEEE Access

TABLE 2: Computation Overhead on the Client Side

Scheme \ Computation Overhead	Data Upload	Data Recovery
Sun et al. [12]	$2n$ **Mul**	$(2n+4)$ **Mul** + $(2n+1)$ **Add**
Gong et al. [21]	512 **Mul**+ 448 **Add**	512 **Mul**+ 512 **Add**
Guan et al. [22]	2 **Exp** + **Mul**	—
Huang et al. [9]	4 **Exp** + 3 **Mul**	**Pair** + 2 **Mul**
Zhang et al. [26]	3 **Exp** + 3 **Mul**	**Pair** + 2 **Mul**
Our scheme	$(n-1)$ **Add**	$(n-1)$ **Add**

TABLE 3: Computation Overhead on the Fog Server Side

Scheme \ Computation Overhead	Data Upload	Data Recovery
Huang et al. [9]	$(2\|Y\|+2)$ **Exp**	2**Pair** + 2 **Mul**
Zhang et al. [26]	$(\|Y\|+2)$ **Exp**	2**Pair** + 2 **Mul**
Our scheme	$(2\|Y\|+2)$ **Exp** + **Mul**	**Pair** + 2 **Mul**

in their scheme is related to the number of smart devices in fog. Therefore, we will compare our schemes with Huang et al. [9] and Zhang et al. [26]. Let $|Y|$ denote the number of leaf nodes of the access tree. In Huang et al. [9], the fog server needs to perform $(2|Y|+2)$ exponentiations to compute the partial ciphertext for an IoT data that needs to be uploaded, and 2 pairing operations and 2 multiplications to recover a partial ciphertext. In Zhang et al. [26], the computation overhead for data recovery is the same, while they save $|Y|$ exponentiations during data upload. In our scheme, the computation needed by each fog server for uploading a data is $(2|Y|+2)$ exponentiations and 1 multiplication, and that for recovering a data is 1 pairing operation and 2 multiplications. The detailed computation overhead on the fog server side of these three schemes is listed in Table 3.

B. EXPERIMENTAL ANALYSIS

According to the above analysis, our scheme is intuitively more efficient than the other five schemes, especially on the client side. We will demonstrate the effectiveness of our scheme through experiments in this section. Based on the experimental setting, our experiment results are as following: 1) The time required by a modular addition in $\mathbb{Z}_p$, multiplication and exponentiation in G_1, and paring on $e: G_1 \times G_2 \rightarrow G_T$ on the fog side are 2.19 us, 2.19 us, 0.94 ms, and 0.75 ms, respectively. 2) The time required by a modular addition in $\mathbb{Z}_p$, multiplication and exponentiation in G_1, and paring on $e: G_1 \times G_2 \rightarrow G_T$ on the IoT device are 0.14 ms, 0.13 ms, 31.57 ms, and 50.39 ms, respectively.

Combining the experiment results with Table 2 and Table 3, we can obtain Fig. 2 and Fig. 3, which provide a more clear and intuitive comparison between our scheme and the other works. In Fig. 2, the number of shares that an IoT data is divided into is assumed to be from 3 to 10. In Fig. 3, the number of attributes in an access policy is assumed to be from 5 to 50.

It can be seen from Fig. 2 that on the client side, our scheme has the lowest computation overhead in the process of both data upload and data recovery. Sun et al.'s computation overhead is slightly higher than ours. Gong et al. [21] has the highest computation overhead. In Fig. 2 (a), the computation overhead during data upload of Huang et al. [9], Zhang et al. [26] and Guan et al. [22] is decreased in turn. And Guan et al.'s computation overhead is much higher than ours even when the number of divided shares in our scheme reaches ten. Fig. 2 (b) shows that during data recovery, Huang et al. [9] and Zhang et al. [26] have the same computation overhead on the client side.

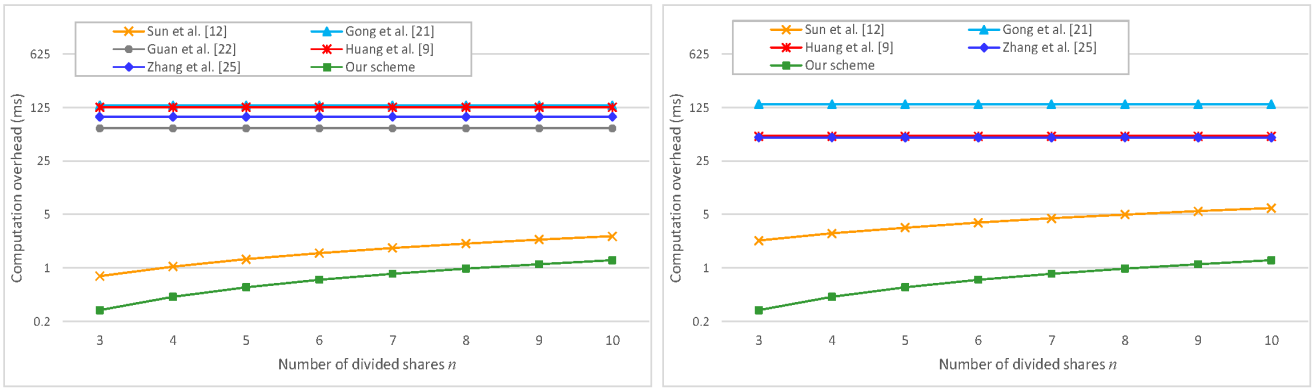
On the fog server side, Fig. 3 (a) shows that our scheme has the highest computation overhead during data upload, followed closely by Huang et al. [9] with a gap of 2.19 ms. The computation overhead of Zhang et al. [26] is the lowest, nearly half of ours. However, during data recovery, the computation overhead of our scheme is the lowest, which can be seen from Fig. 3 (b). Huang et al. [9] and Zhang et al. [26] have the same computation overhead, almost twice that of ours.

Based on the above comparison, we can draw a conclusion that our scheme enjoys a better performance as a whole, considering that we achieve both secure data aggregation and fine-grained access control of outsourced IoT data, while the other schemes just achieve one of the two functions.

VIII. CONCLUSION

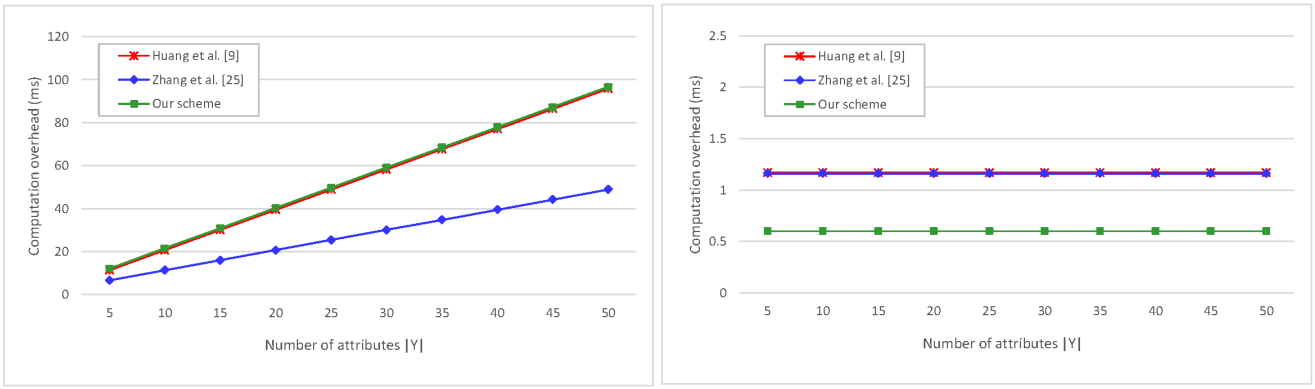
In this paper, we propose a secure IoT data outsourcing scheme, which can support both real-time aggregate statistical analysis and fine-grained access control of outsourced IoT data. By utilizing Corrigan-Gibbs et al.'s computation

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2961413, IEEE Access



(a) data upload (b) data recovery

FIGURE 2: The variation of computation overhead on the client side with the number of divided shares.



(a) data upload (b) data recovery

FIGURE 3: The variation of computation overhead on the fog server side with the number of attributes in the access policy.

of aggregate statistics - Prio and Beaver's multi-party computing (MPC) protocol, fog servers can perform aggregation such as addition, multiplication, and variance on the IoT data uploaded by the data owner, without knowing the original data. Ciphertext-policy attribute-based encryption (CP-ABE) helps us realize fine-grained access control, only allowing the user whose attributes set satisfies the access policy to recover the corresponding data. The security analysis shows that our scheme ensures correctness and data confidentiality. The extensive performance analysis and experiment demonstrate the efficiency of our scheme, meaning it is suitable for the resource-constrained IoT devices such as the MI phone used in our experiment, thus can be further used in real-time health monitoring and many other IoT environments. For our future work, we will try to seek ways to protect the confidentiality of results computed by fog servers, which is not considered in this scheme. Another problem is that the storage overhead in our scheme increases with the number of shares that a data is divided into, which is also the focus of our further research.

REFERENCES

[1] On The Edge: Solving The Challenges Of Edge Computing In The Era Of IoT (2018), [Online]. Available: https://data-economy.com/on-the-edge-solving-the-challenges-of-edge-computing-in-the-era-of-iot/.

[2] C. Li and B. Palanisamy, Privacy in Internet of Things: from Principles to Technologies[J]. IEEE Internet of Things Journal, 2019, 6(1): 488-505.

[3] B. D. Weinberg, G. R. Milne, Y. G. Andonova, and F. M. Hajjat, Internet of Things: Convenience vs. privacy and secrecy[J]. Business Horizons, 2015, 58(6): 615ÍC624.

[4] L. M. Vaquero and L. Rodero-Merino, Finding your Way in the Fog: Towards a Comprehensive Definition of Fog Computing[C]. ACM SIGCOMM Computer Communication Review, 2014, 44(5): 27-32.

[5] M. Chiang and T. Zhang, Fog and IoT: An Overview of Research Opportunities[J]. IEEE Internet of Things Journal, 2016, 3(6): 854-864.

[6] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, Fog Computing and Its Role in the Internet of Things[C]. MCC'12 Proceedings of the first edition of the MCC workshop on Mobile cloud computing, Helsinki, Finland, August 17, 2012: 13-16.

[7] V. Goyal, O. Pandey, A. Sahai, and B. Waters, Attribute-based encryption for fine-grained access control of encrypted data[C]. Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, 2006: 89-98.

[8] J. Bethencourt, A. Sahai, and B. Waters, Ciphertext-Policy Attribute-Based Encryption[C]. 2007 IEEE Symposium on Security and Privacy (SP'07), Berkeley, CA, USA, 2007: 321-334.

[9] Q. Huang, Y. Yang, and L. Wang, Secure Data Access Control With Ciphertext Update and Computation Outsourcing in Fog Computing for Internet of Things[J]. IEEE Access, 2017, 5: 12941 - 12950.

[10] H. Abie and I. Balasingham, Risk-Based Adaptive Security for Smart IoT in eHealth[C].ACM 7th International Conference on Body Area Networks - Oslo, Norway, 2012: 269-275.

[11] C. Doukas, I. Maglogiannis, V. Koufi, F. Malamateniou, and G. Vassilacopoulos, Enabling Data Protection through PKI encryption in IoT m-

Health Devices[C]. Proceedings of the 2012 IEEE 12th International Conference on Bioinformatics & Bioengineering (BIBE), Larnaca, Cyprus, 11-13 November 2012.

[12] G. Sun, S. Huang, W. Bao, Y. Yang, and Z. Wang, A privacy protection policy combined with privacy homomorphism in the Internet of Things[C]. International Conference on Computer Communication & Networks. IEEE, 2014.

[13] N. K. Giang, M. Blackstock, R. Lea, and V. C. M. Leung, Developing IoT applications in the Fog: A Distributed Dataflow approach[C]. IEEE 2015 5th International Conference on the Internet of Things (IOT), Seoul, South Korea, 2015: 155-162.

[14] S. Sarkar and S. Misra, Theoretical modelling of fog computing: a green computing paradigm to support IoT applications[J]. Iet Networks, 2016, 5(2): 23-29.

[15] B. Farahani, F. Firouzi, V. Chang, M. Badaroglu, N. Constant, and K. Mankodiya, Towards fog-driven IoT eHealth: Promises and challenges of IoT in medicine and healthcare[J]. Future Generation Computer Systems, 2017, 78(2): 659-676.

[16] S. M. A. Oteafy and H. S. Hassanein, IoT in the Fog: A Roadmap for Data-Centric IoT Development[J]. IEEE Communications Magazine, 2018, 56(3): 157-163.

[17] S. Chen, H. Xu, D. Liu, B. Hu, and H. Wang, A Vision of IoT: Applications, Challenges, and Opportunities With China Perspective[J]. IEEE Internet of Things Journal, 2014, 1(4): 349-359.

[18] P. Porambage, M. Ylianttila, C. Schmitt, P. Kumar, A. Gurtov, and A. V. Vasilakos, The Quest for Privacy in the Internet of Things[J]. IEEE Cloud Computing, 2016, 3(2): 36-45.

[19] L. Chen, S. Thombre, K. Järvinen, E. S. Lohan, A. Alén-Savikko, H. Leppäkoski, M. Z. H. Bhuiyan, S. Bu-Pasha, G. N. Ferrara, S. Honkala, J. Lindqvist, L. Ruotsalainen, P. Korpisaari, and H. Kuusniemi, Robustness, Security and Privacy in Location-Based Services for Future IoT: A Survey[J]. IEEE Access, 2017, 5: 8956-8977.

[20] K. Fan, W. Jiang, H. Li, and Y. Yang, Lightweight RFID Protocol for Medical Privacy Protection in IoT[J]. IEEE Transactions on Industrial Informatics, 2018, 14(4): 1656-1665.

[21] T. Gong, H. Huang, P. Li, K. Zhang, and H. Jiang, A Medical Healthcare System for Privacy Protection Based on IoT[C]. 2015 Seventh International Symposium on Parallel Architectures, Algorithms and Programming (PAAP), Nanjing, China, 2015.

[22] Z. Guan, Y. Zhang, L. Wu, J. Wu, J. Li, Y. Ma, and J. Hu, APPA: An anonymous and privacy preserving data aggregation scheme for fog-enhanced IoT[J]. Journal of Network and Computer Applications, 2019, 125: 82-92.

[23] S. Yu, K. Ren, and W. Lou, FDAC: Toward Fine-Grained Distributed Data Access Control in Wireless Sensor Networks[J]. IEEE Transactions on Parallel & Distributed Systems, 2011, 22(4): 673-686.

[24] C. Hu, H. Li, H. Yan, X. Tao, and X. Liao, Secure and Efficient Data Communication Protocol for Wireless Body Area Networks[J]. IEEE Transactions on Multi-Scale Computing Systems, 2017, 2(2):94-107.

[25] L.-Y. Yeh, P.-Y. Chiang, Y.-L. Tsai, and J.-L. Huang, Cloud-Based Fine-Grained Health Information Access Control Framework for LightweightIoT Devices with Dynamic Auditing andAttribute Revocation[J]. IEEE Transactions on Cloud Computing, 2018, 6(2): 532-544.

[26] P. Zhang, Z. Chen, J. K. Liu, K. Liang, and H. Liu, An efficient access control scheme with outsourcing capability and attribute update for fog computing[J]. Future Generation Computer Systems, 2018, 78(2): 753-762.

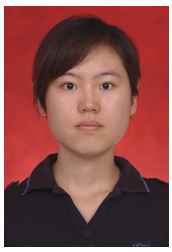
[27] H. Corrigan-Gibbs and D. Boneh, Prio: Private, Robust, and Scalable Computation of Aggregate Statistics[C]. Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI'17), Boston, MA, USA, 2017.

[28] D. Beaver, Efficient Multiparty Protocols Using Circuit Randomization[C]. CRYPTO, 1991: 420-432.

[29] J. Katz and Y. Lindell, Introduction to Modern Cryptography[M]. Chapman & Hall/CRC, Cryptography and Network Security Series, 2007.

[30] PBC Library-The Pairing-Based Cryptography Library, [Online]. Available: https://crypto.stanford.edu/pbc/.

[31] The Java Pairing-Based Cryptography Library (JPBC), [Online]. Available: http://gas.dia.unisa.it/projects/jpbc/#.XN0OclUzaUk.



LING LIU received the B.S. degree from the School of Telecommunications Engineering, Xidian University, in 2013. She is currently pursuing her Ph.D. degree in the School of Cyber Engineering at Xidian University. Her research interests include cloud computing security, network and system security, and applied cryptography.

HE WANG received her Ph.D. degree in Information Security from Xidian University, China. Dr. Wang is a lecturer in the School of Cyber Engineering at Xidian University. Her research interests include information security and quantum cryptography protocol.

YUQING ZHANG received his Ph.D. degree in Cryptography from Xidian University, China. Dr. Zhang is a Professor in the School of Cyber Engineering at Xidian University, and the Director of the National Computer Network Intrusion Protection Center at University of Chinese Academy of Sciences. His research interests include network and system security, and applied cryptography. He has published more than 100 research papers in international journals and conferences, such as ACM CCS, USENIX, IEEE TPDS, and IEEE TDSC. He has served as program chair for more than 5 international workshops (e.g., SMCN-2017), and PC member for more than 10 international conferences in networking and security, such as IEEE Globecom 16/17, IEEE CNS 17, and IFIP DBSec 17.