**1. INTRODUCTION**

Now-a-days, the cloud computing is considered as a promising computing paradigm, since it can provide elastic computing resources to users based on the techniques of distributed computing, virtualization, and so on . However, the prevalence of the Internet of Things (IoT) applications are now changing the main factor of computing . The centralized computing systems are starting to suffer from the unbearable transmission latency and degraded service due to the extraordinary huge volume traffic between IoT devices and cloud. Fog computing is a promising technology that takes advantage of both the paradigms of cloud computing and the IoT, which has the characteristics of location awareness, geo-distribution, low latency, mobility support, etc.

Although the great benefits brought by fog computing paradigm, security problems including data confidentiality and access control are similar to that in the area of cloud computing and IoT. Moreover, more easily compromised and low-trustworthy since fog nodes are deployed at the network edge and much lower cost than cloud servers . One promising approach to solving such problems is to encrypt data in advance before the upload. The concept of attribute-based encryption (ABE) is a one-to-many cryptographic technique that fulfills these requirements . It features a mechanism that enables an access control over encrypted data using access policies and ascribed attributes among private keys and ciphertexts. Especially, the ciphertext-policy ABE (CP-ABE) enables data owner to define the 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 . In this way, the confidentiality and fine-grained access control of data can be guaranteed.

However, existing ABE-based solutions mainly focus on how to afford secure data access for users, few works consider that there is another requirement that data owner may want to authenticate some users to update the encrypted data . For instance, Alice is a data owner and she outsources the encrypted data to cloud, she hopes that only her several friends who are regarded as valid users can renew the initial ciphertext. Thus, the key point of secure ciphertext update is that the user who renews the ciphertext should be able to prove to the cloud service provider (CSP) that he is a valid user.

The traditional approach is to sign the modified data, which means CSP must simultaneously maintain a public key list of valid users to verify the identities of users. However, it would bring a lot of extra burden to maintain the key list if existing a large number of users, and CSP can know the identities of users in this way, which discloses the user privacy. A novel cryptographic technique known as attribute-based signature (ABS) is able to help CSP to verify whether the user is valid . In an ABS system, user can sign messages with a claim policy and his attributes. Then, with the signature, the CSP can check whether the signer's attributes satisfy the claim policy while remaining completely ignorant of the identity of signer. Therefore, adopting ABE and ABS can achieve data confidentiality, fine-grained access control and user verification, but it also brings high computational cost at the same time in fog computing . The encryption, decryption and signing operations of ABE and ABS require a large number of module exponentiations, which commonly grow linearly with the number of attributes in policies. This presents a significant challenge for users who access and modify data on resource-constrained IoT devices with limited computation and storage capacity.

The author proposed a secure data access control scheme in fog computing for IoT. The main contributions are as follows:

1) Propose a fine-grained data access control scheme with ciphertext update based on CP-ABE and ABS in fog computing. First, the sensitive data from IoT devices are encrypted with multiple policies and then outsourced to cloud servers through nearby fog nodes. The authorized user whose attributes satisfy the access policy can decrypt the ciphertext stored in the cloud servers. Second, the authorized user can modify the decrypted data and re-outsource it again with his signature. If the user’s attributes in the signature satisfy the update policy, the cloud servers can renew the ciphertext.

2) Provide a secure outsourcing construction which outsources most of encryption, decryption and signing computations from end IoT devices to fog nodes, thus the computations for data owners to encrypt, end users to decrypt, re-encrypt and sign are irrelevant to the number of attributes in the policies.

The experimental results show that fog nodes perform the heavy computation operations of encryption, decryption and signing, hence the time of encryption for data owner, decryption, re-encryption and signing for users is small and constant. This paper is structured as follows. review related work in Section II, introduce the preliminaries and definitions in Section III, and provide the system model, system definition and security model in Section IV. The detailed construction of algorithms is given in Section V, and the security and performance of our scheme are analyzed in Section VI and VII respectively. Finally, conclude this paper in Section VIII.

**2. LITERATURE SURVEY**

**[1] G. Fortino, A. Guerrieri, W. Russo, and C. Savaglio, “Integration of agent-based and cloud computing for the smart objects-oriented IoT,” in *Proc. IEEE International Conference on Computer Supported Cooperative Work in Design*, Hsinchu, Taiwan, 2014, pp. 493-498.**

The Internet of Things (IoT) represents a world-wide network of heterogeneous cyberphysical objects such as sensors, actuators, smart devices, smart objects, RFID, embedded computers. These objects, which have identities, physical attributes, and communication interface for service provision, are uniquely addressable and based on standard communication protocols ,will be seamlessly embedded into the global information network to become active participants in business, logistics, information and social processes wherever and whenever needed and proper.

Different high-level approaches to model the IoT exist, such as networking-oriented, object-based, service-oriented . In this paper, focus on a smart objects-oriented IoT that is modeled as a loosely coupled, decentralized system of cooperative smart objects (CSOs). In particular, a CSO is a smart object (i.e. a physical object augmented with sensing/actuation, computing, memory and communication capabilities) able to sense, store, and interpret information created within itself and in the environment where it is situated, act on its own by also performing directed actuation, cooperate with peer CSOs, and exchange information with other kinds of IT devices/systems and human users.

The Agent-based Computing paradigm models distributed software systems in terms of multi-agent systems (MAS). In particular, agents are networked software entities that can perform specific tasks for a user and have a degree of intelligence that permits them to perform parts of their tasks autonomously and to interact with their environment in a useful manner. Agents are characterized by important features (e.g. autonomy, sociality, rationality, responsiveness, proactiveness, situatedness, mobility) , which make them very suitable to effectively model CSOs.

The Cloud Computing paradigm provides flexible, robust and powerful storage and computing resources, which supports extreme scale computation through virtualization, dynamic data integration and fusion from multiple data sources . Cloud computing layers (Infrastructure as a Service - IaaS, Platform as a Service - PaaS, Software as a Service - SaaS) and software components (e.g., databases, data mining workflow tools) can be customized to support a distributed real-time system for the management and analysis of IoT objects and data streams generated by IoT objects.

**[2] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog computing and its role in the Internet of things,” in *Proc. First Edition of the MCC Workshop on Mobile Cloud Computing*, Helsinki, Finland, 2012, pp. 13-16.**

Fog Computing is a highly virtualized platform that provides compute, storage, and networking services between end devices and traditional Cloud Computing Data Centers, typically, but not exclusively located at the edge of network. Figure 1 presents the idealized information and computing architecture supporting the future IoT applications, and illustrates the role of Fog Computing.

Compute, storage, and networking resources are the building blocks of both the Cloud and the Fog . “Edge of the Network”, however, implies a number of characteristics that make the Fog a non-trivial extension of the Cloud. Let us list them with pointers to motivating examples.

Edge location, location awareness, and low latency. The origins of the Fog can be traced to early proposals to support endpoints with rich services at the edge of the network, including applications with low latency requirements (e.g. gaming, video streaming, augmented reality).

Geographical distribution. In sharp contrast to the more centralized Cloud, the services and applications targeted by the Fog demand widely distributed deployments. The Fog, for instance, will play an active role in delivering high quality streaming to moving vehicles, through proxies and access points positioned along highways and tracks.

Large-scale sensor networks to monitor the environment, and the Smart Grid are other examples of inherently distributed systems, requiring distributed computing and storage resources.

Very large number of nodes, as a consequence of the wide geo-distribution, as evidenced in sensor networks in general, and the Smart Grid in particular.

Support for mobility. It is essential for many Fog applications to communicate directly with mobile devices, and therefore support mobility techniques, such as the LISP protocol 1 , that decouple host identity from location identity, and require a distributed directory system.

Real-time interactions. Important Fog applications involve real-time interactions rather than batch processing. Predominance of wireless access. Heterogeneity. Fog nodes come in different form factors, and will be deployed in a wide variety of environments.

**[3]**[**Mithun Mukherjee**](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=%22Authors%22:.QT.Mithun%20Mukherjee.QT.&newsearch=true)**;**[**Rakesh Matam**](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=%22Authors%22:.QT.Rakesh%20Matam.QT.&newsearch=true)**;**[**Lei Shu**](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=%22Authors%22:.QT.Lei%20Shu.QT.&newsearch=true)**;**[**Leandros Maglaras**](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=%22Authors%22:.QT.Leandros%20Maglaras.QT.&newsearch=true)**;**[**Mohamed Amine Ferrag**](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=%22Authors%22:.QT.Mohamed%20Amine%20Ferrag.QT.&newsearch=true)**;**[**Nikumani Choudhury**](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=%22Authors%22:.QT.Nikumani%20Choudhury.QT.&newsearch=true)**; "**[**Security and Privacy in Fog**](http://ieeexplore.ieee.org/document/8026115/) **Computing: Challenges"**

**SECURITY AND PRIVACY ISSUES IN FOG COMPUTING**

**A. TRUST**

IoT networks are expected to provide reliable and secure services to the EUs. This requires all devices that are part of the fog network to have a certain level of trust on one another. Authentication plays a major role in establishing initial set of relations between IoT devices and fog nodes in the network. But this is not sufficient as devices can always malfunction or are also susceptible to malicious attacks. In such a scenario, trust plays a major role in fostering relations based on previous interactions. Trust should play a twoway role in a fog network. That is, the fog nodes that offer services to IoT devices should be able to validate whether the devices requesting services are genuine. On the other hand, the IoT devices that send data and other valued processing requests should be able to verify whether the intended fog nodes are indeed secure. This requires a robust trust model in place to ensure reliability and security in fog network. Several works , have been carried out to address the issue of trust in cloud computing environment. However, the unique challenges posed by fog computing environment necessitates to revisit this problem. Contrary to cloud computing environment, the need for a fog node to quantify past interactions with IoT devices in the form of trust/reputation is to be addressed.

**B. AUTHENTICATION**

Authentication of networked devices subscribed to fog services is one of the foremost requirement in fog network. To access the services of a fog network, a device has to first become part of the network by authenticating itself to the fog network. This is essential to prevent the entry of unauthorized nodes. It becomes a formidable challenges as the devices involved in the network are constrained in various ways including power, processing and storage. Traditional authentication mechanisms using certificates and Public-Key Infrastructure (PKI) are not suitable due to the resource constraints of IoT devices. Alternatively, authentication protocols like have been proposed that is based on public-key infrastructure using multicast authentication for secure communications. In essence, like storage and processing services, authentication also needs to be offered as a service whereby a device that needs them would have to get authenticated to the fog node with the help of the intermediary that may be the Certifying Authority (CA). This model of operations would prevent unauthorized nodes from becoming part of the fog network. In addition, this would also allow the fog nodes to restrict service requests from malicious/compromised nodes. Dynamic fog nodes and EUs: Similar to mobility issue in EUs, the fog nodes also frequently join and leave the fog layer. It is required to ensure the uninterrupted service to the registered end users when a new fog node joins (or leaves) the fog layer. The EU must be able to authenticate themselves to the newly formed fog layer mutually. From EUs perspective, the complexity of registration and re-authentication phase without huge overhead.

**D. END USER’S PRIVACY**

Fog computing lies on the computational power of distributed nodes for reducing the total pressure of the data center. In fog computing, privacy preservation is more challenging since fog nodes that are in vicinity with EUs may collect sensitive data concerning the identity, usage of utilities, e.g. smart grid or location of end users compared to the remote cloud server that lies in the core network. Moreover, since fog nodes are scattered in large areas, centralized control is becoming difficult. The compromise of an poorly secured edge node can be the entry point for an intruder to the network. The intruder once inside the network can mine and steal users privacy data that is exchanged among entities. Increased communication among the three layers that constitute the fog architecture can also lead to privacy leakage. Location privacy, as discussed in , is one of the most important models for privacy, since the place of equipment can be linked to the owners. Since fog clients offload its tasks to nearest fog nodes, location, trajectory and even mobility habits can be revealed from an adversary. User habits can also be revealed from an adversary by analyzing his/her usage habits of fog services, e.g. smart grid. As shown in smart meters’ readings can disclose information about the time that the house is empty or even the TV programs that the EU prefers to watch

[**4] J. Bethencourt, A. Sahai, and B. Waters, “Ciphertext-policy attribute-based encryption,” in *Proc. 2007 IEEE Symposium on Security and Privacy*, Berkeley, California, USA, 2007, pp. 321-334.**

**Setup.**

The setup algorithm takes no input other than the implicit security parameter. It outputs the public parameters PK and a master key MK.

**Encrypt(PK, M, A).**

The encryption algorithm takes as input the public parameters PK, a message M, and an access structure A over the universe of attributes. The algorithm will encrypt M and produce a ciphertext CT such that only a user that possesses a set of attributes that satisfies the access structure will be able to decrypt the message. Assume that the ciphertext implicitly contains A.

**Key Generation(MK, S).**

The key generation algorithm takes as input the master key MK and a set of attributes S that describe the key. It outputs a private key SK.

**Decrypt(PK, CT, SK).**

The decryption algorithm takes as input the public parameters PK, a ciphertext CT, which contains an access policy A, and a private key SK, which is a private key for a set S of attributes. If the set S of attributes satisfies the access structure A then the algorithm will decrypt the ciphertext and return a message M.

**Delegate(SK, S˜)**

The delegate algorithm takes as input a secret key SK for some set of attributes S and a set S˜ ⊆ S. It output a secret key SK for the set of ˜ attributes S˜.

**[5]C. Hu, H. Li, Y. Huo, and T. Xiang, “Secure and efficient data communication protocol for wireless body area networks,” *IEEE Transactions on Multi-Scale Computing Systems*, vol. 2, no. 2, pp. 94-107, 2016.**

**1.System Model**

Consider a BAN communication system depicted in Fig. 1. There are four major entities in this system: Key Generation Center (KGC), Sensor (implanted and wearable devices), Data Sink (the BAN data controller or a mobile device such as a smart phone), and Data Consumer (doctors or nurses). In the following subsections, summarize the major functions of each entity.

**2.The Key Generation Center (KGC)**

The KGC is used to perform system initialization, generate public parameters, and assign a secret key for each of the attributes a data consumer claims to have. The public parameters should be installed into the sensors before are deployed (attached to or implanted

in a human body) in a BAN. A data consumer should be able to prove to the KGC that it is the owner of a set of attributes and the KGC will generate a secret key for each attribute. One can see that the secret keys are uniquely generated for the data consumer, which implies that random numbers need to be associated with the set of secret keys to prevent collusion attacks. Sensors have all public parameters, which means that each sensor can construct an access tree and encrypt its data according to the access tree. Once a data consumer’s attributes satisfy the access tree, it should be able to decrypt the message using the corresponding secret keys.

**3. Implanted and Wearable Sensors**

A BAN consists of wireless sensors called BAN devices either embedded on/near the surface (i.e., wearable devices) or implanted in the deep tissue (i.e., implanted devices) of a human body. These sensors are exploited to monitor vital body parameters or body movements (e.g., endoscopy capsules and motion sensors), and/or control the human body by providing life support, visual/audio feedback, etc. A BAN can be used by its human bearer for a variety of applications, including health care, military combat support, and athletic training, just to name a few. Implanted devices suffer from extremely restricted resources in terms of battery power, storage, and computation capability. Wearable devices, on the other hand, have much less stringent resource constraints are usually battery-powered and the batteries can be changed/recharged relatively easily. Wearable devices far exceed implanted ones in both quantity and heterogeneity. Example wearable devices include the sensors monitoring the cardiovascular system (electrodes on the chest to capture ECG, Peizo sensors on the wrist to measure blood pressure, optical sensors on the toe and earlobe to measure the pulse rate, microphones on the chest to measure heart sounds, etc.), the motion sensors placed on knees or in shoes, small cameras or video cameras attached to the sunglasses, and radars attached to the clothes or the stick to assist visually-disabled persons, etc. The BAN devices should have certain computation capability to encrypt the patient’s data and store the ciphertext into the data sink. When a doctor or a nurse needs the data, she/he needs to communicate with the data sink to retrieve the (encrypted) data

## 

**3. PROPOSED SYSTEM**

**A. System Model**

The system model of our proposed scheme consists of attribute authority, CSP, fog nodes, data owners and users, as shown in Fig 3.1

****

**Fig 3.1**

**1) Attribute authority.**

The attribute authority is a fully trusted party which is in charge of generating system parameters as well as secret key for each user.

**2) CSP.**

The CSP is a semi-trusted party which provides high-capacity and online data storage service. It is also responsible for verifying the signature before accepting the updated ciphertext.

**3) Fog node.**

The fog nodes are also semi-trusted parties which are deployed at the network edge and offer a variety of services. They are in charge of generating part of the ciphertext and uploading the whole ciphertext to the CSP, and also helping users to decrypt the ciphertext from the CSP. Moreover, they assist end users to sign the ciphertext update request.

**4) Data owner**.

The data owner has a great amount of data from the IoT devices to be uploaded to cloud. It is designed to define access and update policies to generate the whole ciphertext with the fog nodes.

**5) User.**

The user is attached to fog nodes and equipped with IoT devices such as smart cameras, medical sensors and smart meters . Since the IoT device has limited computation and storage ability, it wishes to gain access to the ciphertext stored in CSP with the help of fog nodes. If the user’s attribute set satisfies the access policy in the ciphertext, he is able to decrypt the underlying data. After accessing the data, the user may make a modification and wish to re-encrypt the data. If the user’s attribute set satisfies the update policy in the ciphertext, the CSP will renew the stored ciphertext.

**B. System Definition**

Define our proposed scheme by describing the following five phases and algorithms.

**Phase 1: System setup**

1) Setup(). The attribute authority takes as input security parameter , and outputs the system public key PKand master secret key MK.

**Phase 2: Key generation**

2) KeyGen(PK, MK, *S*). The attribute authority takes as input PK, MK, a set of attributes *S*, outputs the secret key SKfor the user. And the outsourcing key is sent to fog nodes. SK

**Phase 3: Data encryption**

3) Fog.Encrypt(PK). The fog node takes as input PK, an access policy , outputs a partial ciphertext . CT

4) Owner.Encrypt(PK, M). The data owner takes as input *PK*, a data *M*, an update policy , a partial ciphertext , and outputs the ciphertext CT.

**Phase 4: Data decryption**

5) Fog.Decrypt(PK, CT ). The fog node takes as input PK, a ciphertext CTand a user’s , and outputs a partial decrypted ciphertext *T* if the attributes satisfy access policy in the ciphertext CT.

6) User*.*Decrypt(*T*, SK). The user takes as input a partial decrypted ciphertext *T* and SK, then recovers the DKand outputs the plaintext *M*.

**Phase 5: Ciphertext update**

7) Fog.Sign(PK, *U* ). The fog node takes as input PK, a user’s ciphertext update request *U* and , update policy . It outputs a partial signature and the global key GK.

8) User.Sign(PK, *SK*). The user takes as input PK, a partial signature and *SK*, outputs the signature *ST*.

9) Verify(PK, ST, GK). The CSP takes as input PK, a signature STand a global key *GK*. It outputs true if *ST* is a valid signature by the signer whose attributes satisfying . u T

The work flow of our scheme is shown in Fig. 2. At the initialization phase, attribute authority uses the *Setup* algorithm to generate systems parameter. By the *KeyGen* algorithm, attribute authority generates secret keys for data owners and users. In order to achieve high encryption efficiency, the data owner first encrypts the collected data with a random *DK* by applying symmetric encryption algorithm and defines an access policy and an update policy, the fog node uses the Fog*.*Encryptalgorithm to partially encrypt the data with the access policy, and then data owner uses the Owner*.*Encryptalgorithm to finish the encryption with both the access policy and update policy and stores it to the CSP. When accessing the data, the fog node first uses the Fog*.*Decryptalgorithm to partially decrypt the ciphertext, and then the user can use the User*.*Decryptalgorithm to recover the data in fig 3.2.

After modifying the data, user also uses the algorithms in the encryption phase to encrypt the updated data. Before making the final modification, user uses the User*.*Signalgorithm to generate the signature with the partial signature returned from fog node which runs the Fog*.*Signalgorithm. Then the CSP uses the Verifyalgorithm to verify the signature and finally accepts the updated ciphertext if the signature is true. In the end, other users can obtain the updated data with the decryption algorithms. Therefore, the users with IoT devices can access and update confidential data in fog computing efficiently

****

**fig.3.2.** Work flow of our scheme

**C. Security Model**

In our scheme, assume that cloud servers and fog nodes are honest but curious, which means they execute the tasks and may collude to get the unauthorized data. Specifically, the security model covers the following aspects.

1) Data confidentiality. The unauthorized users which are not the intended receivers defined by data owner should be prevented from accessing the data.

2) Fine-grained access control. The data owner can custom expressive and flexible policies so that the data only can be accessed and updated by the users whose attributes satisfy these policies.

3) Authentication. If users could not satisfy the update policy in ciphertexts, it should also be prevented from updating the ciphertexts.

4) Collusion resistance. Two or more users cannot combine their secret and outsourcing keys and get access to the data they cannot access individually.

**4. ALGORITHM AND TECHNIQUES**

**CONSTRUCTION OF ALGORITHMS:**

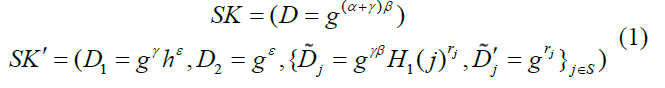
In fog computing, it is the essential requirement to make less computational complexity, since most of the IoT devices are resource-constrained. First, propose fine-grained data access control and efficient ciphertext update scheme based on CP-ABE and ABS. The authorized users whose attributes satisfy the access policy can decrypt the ciphertext, and satisfy the update policy can renew the ciphertext. Second, provide a secure outsourcing construction which outsources most of encryption, decryption and signing computations from end IoT devices to fog nodes. The construction details are as follows.

**A. System setup:**

The attribute authority runs Setupalgorithm to select a bilinear map , e:GO \* GO ->GT where GO and GT are two multiplicative groups with prime order *p*, and *g* is the generator of GO. Then the attribute authority randomly chooses h € GO and α,β € Zp , chooses cryptographic hash functions H1: {0,1}\* ->Zp H2:{0,1}\* -> GO finally outputs a system public key PK=(g,h,gα ,gβ ,hβ ,e(g,g)αβ ) master secret key MK=(α,β)

**B. Key Generation:**

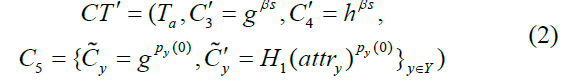
The attribute authority runs KeyGenalgorithm to select a random , which is a unique secret assigned to each user. Then the attribute authority chooses a random , and random for each attribute , where is the attribute set of user, and outputs the secret key and outsourcing key.

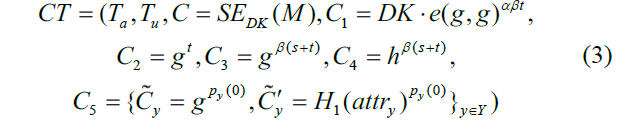


The outsourcing key of user is sent to the fog nodes, and the user only stores *SK*.

**C. Data Encryption:**

Before uploading data to the CSP, data owner first chooses a random *p DK*  , and encrypts the data *M* with *DK* using symmetric encryption algorithm, denoted as  *C* Then data owner defines an access policy *Ta* and an update policy  *Tu* , and sends  *Ta* to fog nodes. The fog nodes run Fog*.*Encryptalgorithm to perform the outsourced encryption. For each node *x* in access policy tree *a T* ,the fog nodes choose a polynomial  *px* . Beginning from the root node *R* , the  *px* is chosen in a top-down manner. For each node *x* in the tree, set the degree *x d* of the polynomial  *px* to be one less than the threshold value  *kx* of that node, that is Starting with the root node *R* , the algorithm chooses a random and sets (0) . Then, it chooses *R d* other points of the polynomial randomly to define it completely. For any other node *x* , it sets parentindex *x* and chooses  *dx* other points randomly to completely define  *p* . Let

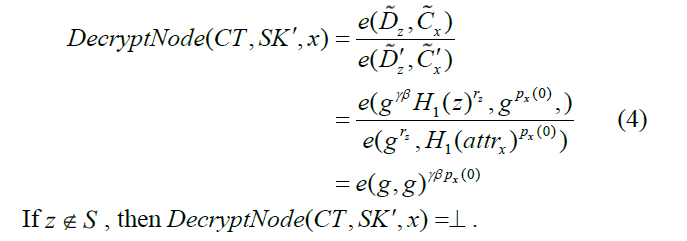




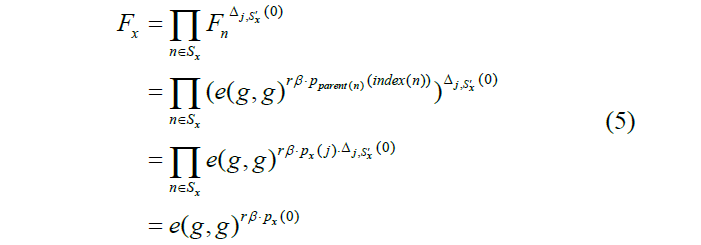
**D. DataDecryption**

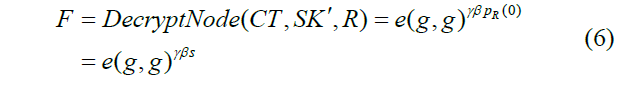
If attributes of the user satisfy the access policy , he can decrypt *CT* successfully by running the following decryption algorithm and obtain the symmetric key *DK*. Fog nodes run Fog*.*Decryptalgorithm to obtain ciphertext from the CSP. The fog nodes first run DecryptNodealgorithm which is a recursive algorithm. The algorithm takes a ciphertext *CT*, , and a node *x* from the access tree as input.

1) If the node *x* is a leaf node, then define as follows. If , then

****

2) If the node *x* is a non-leaf node, the algorithm proceeds as follows: for all nodes that are children of , it calls and stores the output as . Let be an arbitrary -sized set of child nodes such that . If no such set exists, then the node is not satisfied and the function returns⊥. Otherwise, computes and returns the result.





**5. OBSERVATIONS**

If there exists a probabilistic polynomial time (PPT) adversary can win our scheme with non-negligible advantage, then there is a PPT algorithm that can distinguish a decisional bilinear Diffie-Hellman (DBDH) tuple from a random tuple, as proofed in . Hence, our scheme is secure to the DBDH assumption. Analyze the security properties of our scheme as follows.

**A. DataConfidentiality**

The data is first encrypted using the access policy and update policy, and the confidentiality of the data can be guaranteed against users which don’t hold a set of attributes that satisfy the access policy. In encryption phase, though the fog node performs encryption computations for user, it still cannot access the data without the secret key. During the decryption phase, since the set of attributes cannot satisfy the access policy in the ciphertext, the cloud servers or fog nodes cannot recover the value to further get desired value *DK*, because it does not know the *D* of user. Therefore, only the users with valid attributes that satisfy the access policy can decrypt the ciphertext.

**B*.* Fine*-*GrainedAccessControl**

Fine-grained access control allows flexibility in specifying differential access rights of individual users. To enforce this kind of access control, Utilize CP-ABE to escort the symmetric encryption key. In the encryption phase of our scheme, the data owner is able to enforce an expressive and flexible access policy and encrypt the symmetric key which is used to encrypt the data, then outsource the ciphertext to cloud servers. Specifically, the access policy of encrypted data defined in access tree supports complex operations including both AND and OR gate, which is able to represent any desired attribute set. Thus, such construction achieves fine-grained access control.

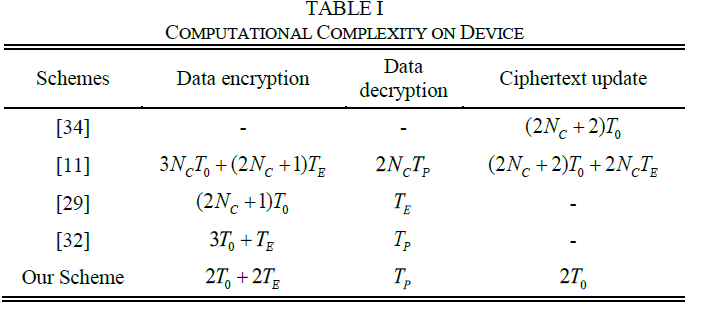
**C. Authentication**

Our scheme exploits ABS to achieve ciphertext update with authentication, even adversary may try to forge a signature with the update policy that his attributes do not satisfy. Let *F* be an adversary who makes at most and queries to random oracles *H*1, *H*2, outsourcing key generation oracle, secret keys generation oracle and signing oracle respectively, and produces a successful forgery against our scheme with a non-negligible probability . Then there exists an algorithm *B* that solves the computational Diffie-Hellman (CDH) problem with a non-negligible probability .

**D*.* PerformanceEfficiency**

Here analyze the performance efficiency of our scheme with the several IoT-based and fog-based data sharing schemes based on ABS or ABE, in terms of computational complexity on user when performing encryption, decryption and signing. The comparison result is showed in Table I. Let be the computational cost of a single pairing, be the computational cost of an exponent operation in , be the time for an exponent operation in , be the number of attributes in a ciphertext. Ignore the simple multiplication, hash, symmetric encryption and decryption operations.

In the data encryption phase, since Ruj et al. and Zuo et al. perform full ABE algorithm on local, their encryption computational cost of data owner are and respectively which both grow linearly with the number of attributes in access policy. In our scheme, the constrained IoT device only needs to cost constant time to encrypt the data with the help of fog nodes, which is similar with Zhang et al.. However, Zhang et al. cannot support ciphertext update. The similar situation appears in data decryption phase. From the end user’s point of view, computational time in our scheme and Zhang et al. is lower than that of Ruj et al. since the user only needs one pairing operation to recover the plaintext. Further, in the ciphertext update phase, compared with Ruj et al. and Su et al. which adopt standard ABS to support the update of outsourced ciphertext. The users in our scheme only need to perform two exponentiation operation in to sign the ciphertext before sending to fog nodes. Thus, the signing cost in our scheme is less than that of Ruj et al. and Su et al. which cost and respectively



**D. ExperimentalAnalysis**

Conduct simulation experiments on a laptop as fog node and an android phone as IoT device. The laptop is with Intel CPU at 2.53 GHz, 4 GB memory and Ubuntu 16.04. The android phone is Samsung G9600V with a quad core processor, 2 GB memory, and Android 6.0.1. The experimental code uses the pairing-based cryptography library to simulate the schemes. use a pairing-friendly type-A 160-bit elliptic curve group based on the supersingular curve over a 512-bit finite field. The Advanced Encryption Standard (AES) is chosen as the symmetric key encryption scheme.

The number of attributes used in the experiments is from 5 to 50, and the experimental result is the average number of 10 runs. consider this range to be representative enough for a wide range of real world IoT applications.

First, analyze the time cost of the data encryption and decryption by comparing our scheme with Zuo et al. and Zhang et al. In data encryption phase, a data owner encrypts a file with an access policy and an update policy, and posts the encrypted file to the public cloud through fog nodes. Fig.3 shows the computational overhead on data owners during this phase. can see the encryption time of our scheme and Zhang et al. is constant since most of the laborious decryption operations are delegated to the fog nodes, while it versus the number of attributes in access policy in Zuo et al.. Fig.4 shows the decryption time on users versus the number of users’ attributes. Specifically, in Zuo et al. , Zhang et al. and our scheme, the heavy computation operations of decryption are outsourced to external server, such as cloud servers and fog nodes, thus the computation operations for users to decrypt in these schemes are irrelevant to the number of attributes in the access policy.

The time complexity on users of ciphertext update which mainly refers to signing algorithms in both our scheme and Su et al. is given in Fig.5. Concerning on the local computation performed by the signer, our scheme achieves much nearly constant performance compared with the linear increasing efficiency of the scheme of Su et al. by outsourcing many computations to fog nodes. This advantage allows our scheme to be applied for the resource-constrained IoT devices to complete the signing task.

Moreover, consider that the IoT device has limited storage ability. Since the outsourcing key can be firstly generated by attribute authority and then sent to the fog nodes. Therefore, the user only needs to store a small-sized component *D* locally but still maintaining encryption, decryption and signing capability. Argue that such amount is acceptable for IoT devices such as Samsung phone used in our experiments. In summary, the experimental results show that our scheme incurs less computational cost on the encryption of data owner, the decryption and signing of user, which ensures both fine-grained data access control and efficient ciphertext update in fog computing. Hence, our scheme could be applied to smart healthcare, vehicular cloud computing, and etc.

**6. CONCLUSION**

Proposed a secure data access control scheme in fog computing for IoT based on CP-ABE . The sensitive data of users are first encrypted with both access policy and update policy, and then outsourced to cloud servers through fog nodes. Thus, the users whose attributes satisfy the access policy can decrypt the ciphertext. In order to address the issue of data modification, the CSP will check the signature, to ensure that only the users whose attributes satisfy the update policy can renew the ciphertext. Hence, our scheme achieves both fine-grained data access control and secure ciphertext update.

Moreover, our scheme presents an outsourced encryption, decryption and signing construction by delegating most of the operations to fog nodes. The extensive performance analysis and experiments are conducted, and the results indicate our scheme can well tolerate the increasing number of attributes, which is suitable for the resource-constrained IoT devices in fog computing.

**REFERENCES**

[1] C. Rong, S.T. Nguyen, and M.G. Jaatun, “Beyond lightning: A survey on security challenges in cloud computing,” *Computers & Electrical Engineering*, vol. 39, no. 1, pp. 47-54, 2013.

[2] G. Fortino, A. Guerrieri, W. Russo, and C. Savaglio, “Integration of agent-based and cloud computing for the smart objects-oriented IoT,” in *Proc. IEEE International Conference on Computer Supported Cooperative Work in Design*, Hsinchu, Taiwan, 2014, pp. 493-498.

[3] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of Things (IoT): a vision, architectural elements, and future directions,” *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645-1660, 2013.

[4] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog computing and its role in the Internet of things,” in *Proc. First Edition of the MCC Workshop on Mobile Cloud Computing*, Helsinki, Finland, 2012, pp. 13-16.

[5] P. Hu, H. Ning, T. Qiu, H. Song, Y. Wang, and X. Yao. (2017, Jan.). Security and privacy preservation scheme of face identification and resolution framework using fog computing in internet of things. *IEEE Internet of Things Journal*. [Online]. Available: https://doi.org/10.1109/JIOT. 2017.2659783

[6] Q. Huang, Z. Ma, Y. Yang, J. Fu, and X. Niu, “EABDS: attribute-based secure data sharing with efficient revocation in cloud computing,” *Chinese Journal of Electronics*, vol. 24, no. 4, pp. 862-868, 2015.

[7] Q. Huang, L. Wang, and Y. Yang. (2017, May). DECENT: secure and fine-grained data access control with policy updating for constrained IoT devices. *World Wide Web Journal*. [Online]. Available: https://doi.org/ 10.1007/s11280-017-0462-0

[8] X. Dong, J. Yu, Y. Luo, Y. Chen, G. Xue, and M. Li, “Achieving secure and efficient data collaboration in cloud computing,” in *Proc. IEEE/ACM 21st International Symposium on Quality of Service*, Montreal, QC, 2013, pp. 195-200.

[9] F. Zhao, T. Nishide, and K. Sakurai, “Realizing fine-grained and flexible access control to outsourced data with attribute-based cryptosystems,” in *Proc. Information Security Practice and Experience - 7th International Conference*, Guangzhou, China, 2011, pp. 83-97.

[10] J. Li, M.H. Au, W. Susilo, D. Xie, and K. Ren, “Attribute-based signature and its applications,” in *Proc. 5th International Symposium on Information, Computer and Communications Security*, Guangzhou, China, 2010, pp. 60-69.

[11] S. Ruj, M. Stojmenovic, and A. Nayak, “Decentralized access control with anonymous authentication of data stored in clouds,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 2, pp. 384-394, 2014.

[12] Q. Huang, Y. Yang, and M. Shen, “Secure and efficient data collaboration with hierarchical attribute-based encryption in cloud computing,” *Future Generation Computer Systems*, vol. 72, pp. 239-249, 2017.

[13] R. Lu, K. Heung, A. Lashkari, and A. Ghorbani, “A lightweight privacy-preserving data aggregation scheme for fog computing-enhanced IoT,” *IEEE Access*, vol. 5, pp. 3302-3312, 2017.

[14] K. Lee, D. Kim, D. Ha, U. Rajput, and H. Oh, “On security and privacy issues of fog computing supported Internet of Things environment,” in *Proc. 2015 International Conference on the Network of the Future*, Montreal, QC, Canada, 2015.

[15] A. Sahai and B. Waters, “Fuzzy identity-based encryption,” in *Proc. 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques Aarhus*, Denmark, 2005, pp. 457-473.

[16] A. Lewko and B. Waters, “Decentralizing attribute-based encryption,” in *Proc. 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Tallinn, Estonia, 2011, pp. 568-588.

[17] V. Goyal, O. Pandey, A. Sahai, and B. Waters, “Attribute-based encryption for fine-grained access control of encrypted data,” in *Proc. 13th ACM Conference on Computer and Communications Security*, New York, USA, 2006, pp. 89-98.

[18] J. Bethencourt, A. Sahai, and B. Waters, “Ciphertext-policy attribute-based encryption,” in *Proc. 2007 IEEE Symposium on Security and Privacy*, Berkeley, California, USA, 2007, pp. 321-334.

[19] S. Ruj, A. Nayak, and I. Stojmenovic, “Distributed fine-grained access control in wireless sensor networks,” in *Proc. 2011 IEEE International Parallel & Distributed Processing Symposium*, Anchorage, Alaska, USA, 2011, pp. 352-362.

[20] S. Yu, K. Ren, and W. Lou, “FDAC: toward fine-grained distributed data access control in wireless sensor networks,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 4, pp. 673-686, 201