Scientific Research Publishing

Journal of Information Security, 2017, 8, 203-222
http://www.scirp.org/journal/jis
ISSN Online: 2153-1242
ISSN Print: 2153-1234

Attribute-Based Secure Data Sharing with Efficient Revocation in Fog Computing

Asma Alotaibi, Ahmed Barnawi, Mohammed Buhari

Department of Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia
Email: aalotaibi0355@stu.kau.edu.sa, ambarnawi@kau.edu.sa, mesbukary@kau.edu.sa

How to cite this paper: Alotaibi, A., Barnawi, A. and Buhari, M. (2017) Attribute-Based Secure Data Sharing with Efficient Revocation in Fog Computing. *Journal of Information Security*, **8**, 203-222.
https://doi.org/10.4236/jis.2017.83014

Received: May 2, 2017
Accepted: July 11, 2017
Published: July 14, 2017

Abstract

Fog computing is a concept that extends the paradigm of cloud computing to the network edge. The goal of fog computing is to situate resources in the vicinity of end users. As with cloud computing, fog computing provides storage services. The data owners can store their confidential data in many fog nodes, which could cause more challenges for data sharing security. In this paper, we present a novel architecture for data sharing in a fog environment. We explore the benefits of fog computing in addressing one-to-many data sharing applications. This architecture sought to outperform the cloud-based architecture and to ensure further enhancements to system performance, especially from the perspective of security. We will address the security challenges of data sharing, such as fine-grained access control, data confidentiality, collusion resistance, scalability, and the issue of user revocation. Keeping these issues in mind, we will secure data sharing in fog computing by combining attribute-based encryption and proxy re-encryption techniques. Findings of this study indicate that our system has the response and processing time faster than classical cloud systems. Further, experimental results show that our system has an efficient user revocation mechanism, and that it provides high scalability and sharing of data in real time with low latency.

Keywords

Attribute-Based Encryption, Fine-Grained Access Control, Fog Computing, Proxy Re-Encryption, User Revocation

1. Introduction

Cloud computing is the most popular computing paradigm that offers its resources over the Internet. Cloud computing provides many advantages to end users, such as lower cost, high reliability, and greater flexibility. However, it has some drawbacks, which include a high latency, necessitating Internet connecti-

vity with high bandwidth and security [1].

During the last few years, a new trend of Internet deployments emerged called the Internet of Things (IoTs) that envisions having every device connected to the Internet. Its applications include ehealthcare, a smart grid, etc. Those applications require low latency, mobility support, geo-distribution, and user location awareness. Cloud computing appears to be a satisfying solution to offer services to end users, but it cannot meet the IoTs' requirements. As a result, a promising platform called fog computing is needed to provide the IoTs' requirements; fog computing was proposed by Cisco in 2012 [2].

Fog computing is a concept that extends the paradigm of cloud computing to the network edge, allowing for a new generation of services [3]. Fog computing has an intermediate layer located between end devices and the cloud computing. This leads to a model with a three-layer hierarchy: Cloud-Fog-End Users [4]. The goal of fog computing is to offer resources in a closer vicinity to the end users. As in Figure 1, each fog is located at a specific building and offers services to those inside the building [4]. Fog computing supports low latency, user mobility, real-time applications, and a wide geographic distribution. In addition, it enhances the quality of services (QoS) for end users. These features make the fog an ideal platform for the IoTs [5].

Support of location awareness is the key difference between the cloud environment and the fog environment. Cloud computing serves as a centralized global model, so it lacks location awareness. In contrast to cloud computing, fog devices are physically situated in the vicinity of end users [6].

Data sharing has great importance for many people, and it is an urgent need for organizations that aim to improve their productivity [7]. Currently, there is an urgent need to develop data sharing applications, especially for mass com-

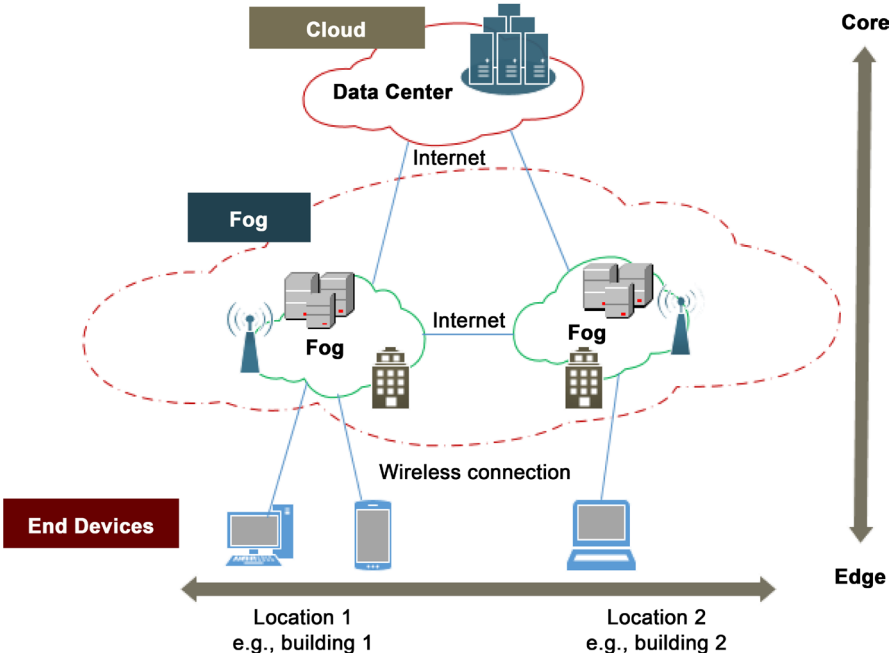

Figure 1. The fog is situated between the cloud and the edge.

munications, where the data owner is responsible for delivering shared resources to a large group of users. This one (data owner) to many (users) method needs special care, taking into consideration the challenges related to such applications. The main problems for such applications are issues related to security and privacy [8].

Like cloud computing, fog computing faces several security threats for data storage; to meet them, there are security features that were provided in the cloud environment. These security features are the enforcing of fine-grained access control, data confidentiality, user revocation and collusion resistance between entities [9].

We present a novel architecture for data sharing a fog environment. We explore the benefits brought by fog computing to address one-to-many data sharing application. Such architecture is sought to outperform the cloud-based architecture and ensure further enhancements to system performance, especially from the perspective of security. Our proposed framework provides high scalability and sharing of data in real time with low latency.

2. Related Work

We will provide a detailed overview of prior studies on secure data sharing in cloud environments.

Yu *et al.* [9] proposed a data-sharing scheme designed to provide fine-grained data access control, data confidentiality, and scalability. However, it requires updating all users' secret keys and re-encrypting all the files, thus reducing the efficiency of the user revocation operation.

Wu *et al.* [10] presented a novel technique for sharing media, especially in large distributed systems. Unfortunately, the decryption operation in low-end devices is slow, and user revocation is not addressed.

Liu *et al.* [11] designed a framework for sharing data based on the time concept. It is a better fit for an environment in which the data owner is offline and periodic user revocation occurs. However, the proposed scheme requires efficient shared time periods for all the user-related attributes.

Tu *et al.* [12] proposed a secure data-sharing framework that is secure against chosen-ciphertext attacks. Unfortunately, the proposed framework places heavy computation overhead on the process of user revocation.

Yang and Zhang [13] designed a generic scheme for sharing data. The scheme does not need to require the redistribution of keys. However, it has not addressed the scenario in which a revoked user rejoins the group with new access rights.

Hur [14] proposed a secure data-sharing scheme featuring rapid user revocation. Its major drawback is that it suffers from low scalability and high calculative complexity.

Samanthula *et al.* [15] proposed a framework with effective user revocation. Unfortunately, the proposed scheme puts a heavy burden on the cloud servers by requiring the data owner to create a token in each record for every user, which

Table 1. The main features schemes.

Design Goals	References							
	9	10	11	12	13	14	15	16
Data confidentiality	Y	Y	Y	Y	Y	Y	Y	Y
Enforcing fine grained access control.	Y	Y	Y	Y	Y	Y	Y	Y
Scalability	Y	Y	Y	Y	Y	N	N	Y
Efficient user revocation	N	N	Y	N	Y	Y	Y	N
Collusion resistance	Y	N	N	Y	N	Y	Y	Y
Real-time data sharing	N	N	N	N	N	N	N	N

increases the complexity of the system and reduces scalability.

From the previous discussion, it is evident that the previous schemes have failed to find an overall solution to achieving the previous goals, as shown in **Table 1.** Most of these desired features are realized in [9], so we will apply it in a fog environment with some enhancement to achieve all our design goals. Our proposed framework rests on a combination of previous approaches that provide secure data sharing in cloud computing, such as Attribute-Based Encryption (ABE) and Proxy Re-Encryption (PRE) techniques [9] [16] [17].

Unlike the previous system [9], the proposed revocation mechanism does not necessitate the re-encryption of all system files and updating of all secret keys. Our proposed system provides real-time data sharing to group members. Our work will focus on providing an ideal environment for secure data sharing in a fog environment to overcome the disadvantages of a cloud-based data sharing system, which includes a high latency, requiring Internet connectivity with high bandwidth and lacking location awareness.

3. Fog Based Data Sharing Architecture

3.1. Fog Based Data Sharing Model

There are four parties in the proposed system: Data Owner, Cloud Servers, many Fog Nodes and Data users.

- Data Owner (DO) has the right to access and alter the data. He encrypts the data with the attributes of a specific group and generates the decryption keys for users. Then, he uploads the encrypted data to the cloud servers.
- Cloud Server (CLD) is responsible for data storage and deploys the data to the fog nodes.
- Fog Nodes (FNs) are responsible for data storage and for addressing users' requests. They are considered as a semi-trusted party. They execute operations of user revocation phase.
- Data Users (Us) are those who request data access when they have the rights to access data. This means, only when the user's access policy satisfies the data attributes.

The fog environment scenario is shown in **Figure 2**, where a DO encrypts a data file and then outsources it to a CLD for storage. Then, the CLD deploys the

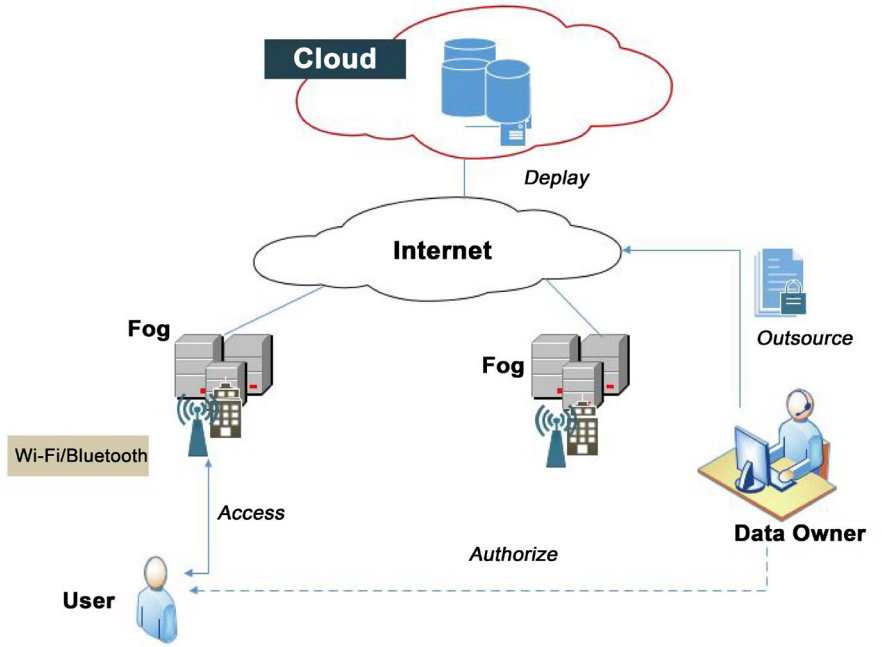

Figure 2. Fog-based data sharing model.

data file to the specific fog node via the data distribution protocol, as will be shown later. Fog nodes are geographically distributed within a specific domain, and they have fixed locations. The user can be moving, and he is requesting the data from the fog node closest to him. The fog node receives the user's request and delivers the file to the user. The DO can delegate most of the tasks to the home fog nodes, as shown in the following section. In fog-based data sharing model, fog nodes and the data owner both can be connected with the cloud via the Internet. The fog nodes are connected to each other via a wired network over Internet. The users can be connected with the fog node using a wireless connection technique such as Wi-Fi, as shown in Figure 2.

This model consists of groups of users, and each group has a set of attributes and a basic location. Each group has many users who share the same attributes. One of the group attributes refers to its location, and group members connect with a fog that has the same location. The data owner assigns many files to each group on the basis of the attributes and needs of its members.

Each fog node serves one group, and is independent in its operation, so it is not affected when a user is revoked from another group. Therefore, the proposed revocation mechanism requires the re-encryption of the affected files and the updating of the secret keys, only for a one group.

3.2. Data Distribution Protocol

Two types of fog in the data distribution architecture are defined:

- Home Fog (HF): the fog has the same location as the user's original location, where users are most likely to be found. It stores the user's data and manages the processes.
- Foreign Fog (FF): the fog is located away from the user's original location,

where the user is currently residing, as shown in Figure 3.

- The proposed system is comprised of two kinds of data centers:
- Cloud data centers (which includes the data centers for each group).
- Local fog data centers.

Each fog node is considered the "Home Fog" for the group that has the member's same location, while it is considered the "Foreign Fog" for the other groups.

A local data center is a fog storage that holds copies of secret files. It is preloaded with the data required by fog users. The fog nodes maintain communication with the cloud. The data sharing between the cloud data center and each fog node data center is performed through immediate synchronization based on the unicast method.

When the user requests a file from the fog node, if the fog is the user's HF, the fog node directly sends the file to the user. If the user is away from his/her HF, the case is processed, as shown in Figure 4.

1) Using authentication, a user logs to the fog node closest to him. He requests to join to it and identifies the period of the joined fog node through the registration process.

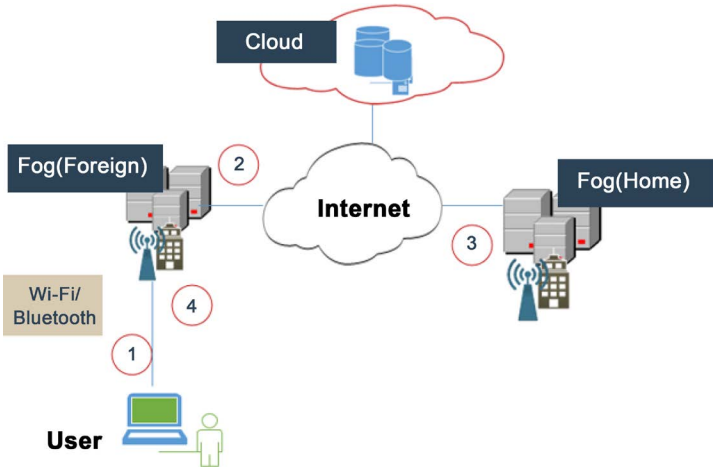

Figure 3. Data distribution architecture.

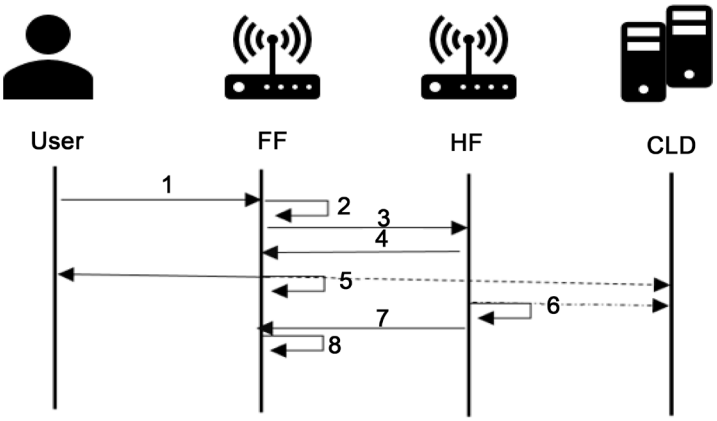

Figure 4. Data distribution protocol.

Table 2. Fog1's users locations.

User's ID	Current location	Time period
User 1	Fog2	XXXX
User 2	Fog1	-
…	…	…

2) The FF recognizes the user's home by the system user list (the cloud updates this list whenever a user is added or removed, and sends it to all fog nodes via broadcasting after each update. This list includes the user's ID and its own HF. (Note: the HF of each user is fixed).
3) The FF sends the joining message with the specified period to the user's HF.
4) The HF sends an acceptance reply to acknowledge the joining.
5) The FF accepts the user as a visitor, updates its visitor list, and then synchronizes the list with the cloud.
6) The HF updates the location of its own users in the Table 2 by changing the user's location to the FF's location and synchronizing it with the cloud. This table does not include the visitor's users; it is only for its group members.
7) The HF sends the user's secret data to the FF.
8) The FF stores the data in its data center.

If the time expires and the user is still at the FF, he must join the FF again. When the user returns to his HF, he will send a de-joined request to the HF and inform it that he is at his HF. The FF updates the current location table and synchronizes the table with the cloud.

4. The Proposed System

4.1. Technique Preliminaries

1) Key Policy Attribute-Based Encryption (KP-ABE)

In KP-ABE, data have a set of attributes linked to data by encryption with the public key. Each user has an access structure that is an access tree associated with data attributes. The user's secret key is a reflection of the user's access tree; therefore, the user can decrypt a ciphertext if the data attributes match his or her access tree [13] [18].

2) Proxy Re-Encryption (PRE)

PRE is a cryptographic primitive that allows a semi-trusted proxy to transform the cipher text of the encrypted data under the data owner's public key into a different ciphertext under the group member's public key. The semi-trusted proxy server needs a re-encryption key sent by the data owner for a successful conversion process, and it is unable to discover the underlying plaintext of the encrypted data. Only an authorized user can decrypt the ciphertext [19].

4.2. Design Goals

The design goals are as follows:

- Data confidentiality: Unauthorized users (including the fog and cloud servers) are not allowed to access the data [9].
- Fine-grained access control: The data owner can determine the access structure for each user [11].
- User revocation: Revoked users cannot access the data again.
- Scalability and efficiency: The system must maintain both efficiency and scalability, even when the number of users increases [9].
- Collusion resistance: which prohibits unauthorized parties from cooperating in order to find out the contents of sensitive data [20].
- Real-time data sharing.

4.3. Assumptions and Security Models

In the proposed framework, the data sharing system is one to many. The fog nodes have fixed locations. It may be assumed that the target user is a laptop or other mobile device. Also, that the data owner and users have already the public/private key pairs, where the public keys can be easy to get by other entities. Using the security protocols, the communication channels are secured between the data owner/cloud server and fog nodes, such as SSL. Also, the communication channel is assumed to be secured between fog nodes and users. In order to connect between the user and the fog nodes, the existing protocols such as CoAP, are used which are considered to be the promising protocol for IoTs [21], in addition to authentication the users at the fog node.

4.4. Definition and Notation

In order access control, the data owner must assign meaningful attributes to each file. The file's attributes are the same as the one group's attributes. To update the attributes, each attribute has a version number, which will be shown later. Fog servers have a copy of a group attribute history list (GAtH), as we will see later

The GAtH contains the attributes' evolution history and the PRE keys used. A PRE-key allows the data owner to assign re-encryption operations to the fog node without revealing the data contents. Additionally, one virtual attribute, denoted by AttV, must be determined for the key's management. AttV is the basic attribute in every data file's attributes and user's access structure, and is will not be updated. The user has a completely secret key, while the fog and cloud have a partially user's secret key because they lack a secret key component corresponding to a virtual attribute, where that AttV is unknown for the fog and cloud. The goal of AttV is to enable the fog to update the secret key without revealing it.

Each user has an access structure represented as an access tree [22]. The access tree has interior nodes, which are the threshold gates, and leaf nodes associated with the file's attributes. The root node must be an AND gate, with one of the child nodes associated with the virtual attribute, while the other child nodes are the threshold gates, as shown in **Figure 5**.

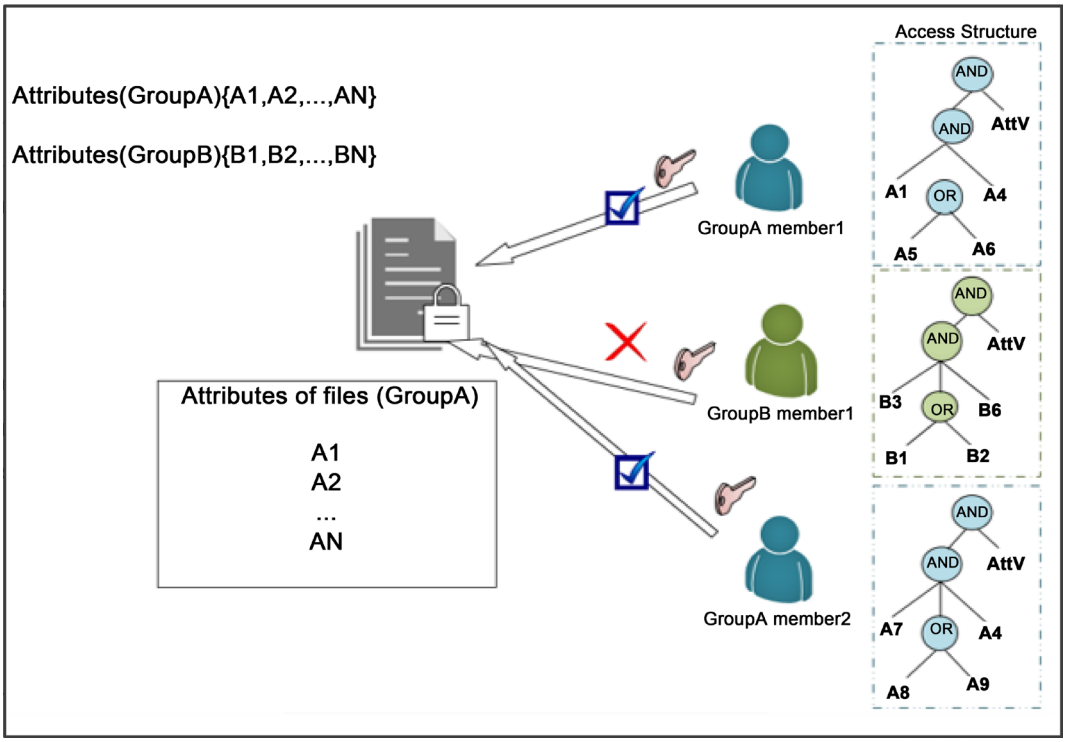

Figure 5. The simple explain of the access system.

Each group file has the same group attributes, where the members of group have the right to access the group files. Each member has a subset of the group attributes. Each member has a different secret key that reflects his or her access structure. Moreover, CLD has the system users list (UL), which includes the IDs, HF of all the authorized users. Table 3 shows the notation of the proposed system with a simple description.

4.5. System Description

The proposed framework consists of six main phases, the following subsections show details of these phases.

1) Group Creation

The DO creates groups of the system and generates its parameters.

1) The DO assigns a unique ID, name, $AttG$, and specific location for the new group.
2) A security parameter k is chosen by the DO, and then $GSetup(\kappa, AttG)$ is run, which produces GPK and GMK.
3) The DO signs GPK components.
4) DO sends the group information and the GPK to the CLD.
5) CLD stores them in data center.
6) The CLD sends GPK to group's HF.
7) The group's HF store GPK, each FN serves one group as the HF.

2) Add and Encrypt File

In this phase, the data owner processes the file before uploading as follows.

1) The DO assigns a unique ID for the new file.

Table 3. Notation of the scheme.

Notation	Description
G_{ID}	Group's ID
GPK_g, GMK_g	Group public key and master key
Q_a	Group public key component for attribute ***a***
q_a	Group master key component for attribute ***a***
pol	The access structure of user
L_{pol}	Attributes associated with leaf nodes of pol
USK	Secret key of the user
usk_a	User secret key component for attribute a
$AttG$	Group attributes set
$AttF$	File attributes set
$GAtH_a$	Group attribute history list for attribute ***a***
F_{ID}	File's ID
$AttV$	The virtual attribute
C, C_a	Ciphertext and the ciphertext component for attribute a
SEK	Symmetric data encryption key of a data file
UL_g	Group user list
$\delta_{DO,m}$	Signature of data owner on message m
$rek_{a\leftrightarrow a'}$	Proxy re-encryption key the current attribute a to the updated version a'
UL	The system users list

Figure 6. The data format.

2) The DO chooses a random data symmetric encryption key $SEK \xleftarrow{R} K$,where k refers to the key space.
3) The DO encrypts the file using SEK .
4) DO specifies the group that needs this file.
5) The DO specifies the attributes for this file. $AttF$ is same as the ($AttG$).
6) The DO encrypts SEK , by calling $GEncrypt(AttF, SEK, GPK)$, which outputs the ciphertext C .
7) The DO uploads the encrypted file with the group's ID to the CLD.
8) CLD stores them in its data center.
9) The CLD sends the encrypted file to the group's fog node.
10) The group's fog node stores the encrypted file in format, as shown in Figure 6.

3) Enroll New User

The data owner performs the following processes to grant the access right to

the system.

1) The DO defines a user unique identity w, an access structure pol and the group, where he belongs.
2) The DO runs the $GKeyGen(pol, GMK)$, which produces USK referring to the secret key of w.
3) The tuple $\left(pol, USK, GPK, \delta_{DO,(pol,USK,GPK)}\right)$ is encrypted with the user's public key by the DO (preloaded); where the ciphertext is denoted as CT.
4) The DO sends the tuple $\left(T, CT, \delta_{DO,(T.CT)}\right)$ to the CLD, where CT (from the third step) and T is $\left(w, \{j, usk_j\}_{j \in Lpol \setminus AttV}\right)$.
5) The CLD verifies the signature, then stores T in the UL.
6) The CLD sends the CT and T to the user's HF.
7) The FN stores T in its UL.
8) Then FN sends the CT to U.
9) The U decrypts the CT with his or her private key, verifies the signature, and accepts (pol, USK, GPK).

4) Delete File

This operation is performed at the DO's request

1) The DO sends the F_{ID} with his signature to the CLD.
2) The CLD verifies the signature, then removes the file.
3) The CLD sends the F_{ID} to all FNs.
4) FNs remove the file in case it was found in their data centers.

5) Revoke user

Based on the revocation method in [9], the proposed system's revocation operation works as follows.

1) To revoke user v,the DO defines the attributes' minimal set of:

$$D \leftarrow GMinimalSet(pol);$$

where pol is v's access structure.

2) The GPK and GMK components of all these involved attributes are updated accordingly.
3) The DO generates the corresponding PRE keys, for each attribute a in D,

$$\left(q'_a, Q'_a, rek_{a \leftrightarrow a'}\right) \leftarrow GUpdateAtt(a, GMK).$$

4) The DO sends the user ID, the updated attributes, the PRE keys, group ID, andthe updated GPK components to the CLD.
5) The CLD removes the revoked user from the UL and records the updated GPK in the group's table.
6) The CLD records the last version of the PRE key in $GAtH_a$ only to the updated attribute (at group's data center in cloud).
7) The CLD sends the user's ID, the updated attributes, the PRE keys, andthe updated GPK to the user's HF.
8) HF store them and remove the revoked user.
9) With $GAtH_a$, the FN will find one PRE key that can update the attribute to the latest version.
10) For each member of the group, the FN updates this user's USK compo-

nents to the latest version; for each attribute, $j \in L_{pol \setminus AttD}$;

$$usk'_j \leftarrow GUpdateSK\left(j, usk_j, GAtH_j\right).$$

11) The FN re-encrypts each file's SEK using the latest GPK version, then stores it; for each attribute $k \in AttF$;

$$C'_k \leftarrow GUpdateAtt4File\left(k, C_k, GAtH_k\right).$$

12) The FN sends the $USKs$ to users.

13) If it has an away user, HF sends the data and USK to the user's FF. Then FF stores the data.

14) The FF sends the USK to the user.

6) Download and Decrypt File

1) In this operation, fog node receives the user's access request to the data file.
2) Then FN verifies if the user is a valid user. If the user is not of its members, verifies if the user in its visitor list.
3) The FN sends the C of the requested file to the U.
4) The U checks if each attribute is the latest version of the current version he knows.
5) The U verifies if each USK components is correct. If the verification is successful, (U) replaces each usk_j of his secret key with usk'_j and updates Q_j with Q'_j.
6) The U runs $GDecrypt\left(pol, USK, C\right)$ to decrypt the SEK's.
7) The U decrypts the file using the SEK's.

5. Implementation and Evaluation

5.1. Test Environment

The proposed system was implemented on a machine running 64-bit Windows 10 with a 2.4 GHz Intel CoreTM CPU and 6 GB of memory. The system's implementation is based on a pairing-based cryptography (PBC) library, a standard symmetric key algorithm (AES), key-policy attribute-based encryption, and proxy re-encryption techniques. Our system calls the ABE- and PRE-required functions from libraries developed using the Java language, as we will show the results later. First, we built the main functions of the system: Group creation, Encryption, Key Generation, and Decryption. Then, we designed the user revocation functions: Update the master key and public key, Re-encrypt the files, and Regenerate the secret keys.

5.2. Performance Evaluation

Our performance evaluation is based time analysis in order to appreciate both the suggested protocols dealing with specific security issue "revocation" and the benefits the fog computing brings to the problem. Therefore, we have built our own simulation methods and used available packages.

Figure 7 gives an overview of the system's time factors. The data presented in the Figure above show that the length of time between $t_0 \cdot t_1$, is represents the

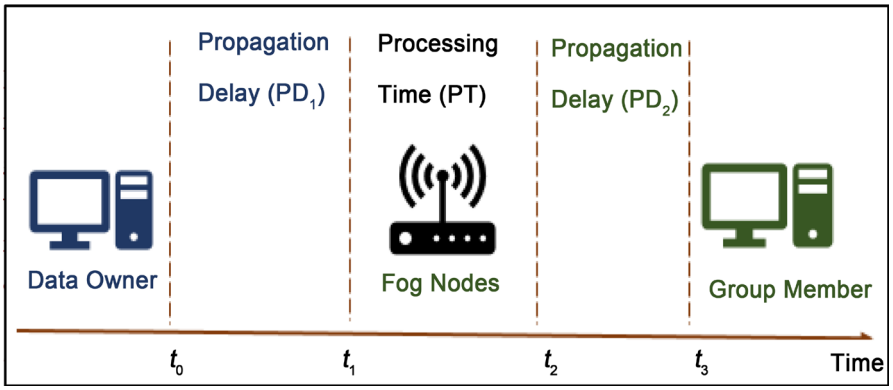

Figure 7. Time factors of the revocation phase.

first propagation delay (PD_1). This delay depends on the length of the link and the speed of propagation. The time difference ($t_2 - t_1$) corresponds to the processing time (PT), which is the time required for the fog node to execute user revocation operations. The second propagation delay (PD_2). is determined by another time difference ($t_3 - t_2$).

1) System Response Time

Fast response time is important for improving the quality of service and users' experience, it is directly related to the distance between the server and user. Fog nodes are located on the network edge, in close proximity to the users, and therefore the system has a fast response time. We can calculate the response time using the following equation:

$$\text{Response Time} = \text{Propagation Delay Time} + \text{Processing Time}.$$

The term "Propagation Time" refers to the amount of time a packet needs to arrive at its destination. It depends on the distance between the server and user and on propagation speed. Processing time can be broadly defined as the total time that a fog node requires to address a user revocation request after it is received.

In order to appreciate the impact of fog computing on response time, we build a simulation scenario that assumes the fog and cloud computing scenarios have the same resources. Naturally, in the fog computing scenario, we assume that fog resources are placed nearer to users than in the cloud. Those scenarios were implemented using the Cloud Analyst toolkit. The simulation results show that our system responds more quickly to user requests than classical cloud systems, as shown in Figure 8.

In the simulation environment, the data center responds to user requests in order according to their location. The simulation identifies response time as minimum, average, or maximum. In the figure below, we can see that the fog system's response time is 84% less than that of the cloud system; the average response time was 300 ms in cloud-based systems and 50 ms in fog-based systems.

Regarding processing time, the results in Figure 9 show that the time required to perform an encryption operation (whether in the cloud or a fog) depends directly on the number of attributes that are used, as explained in previous section.

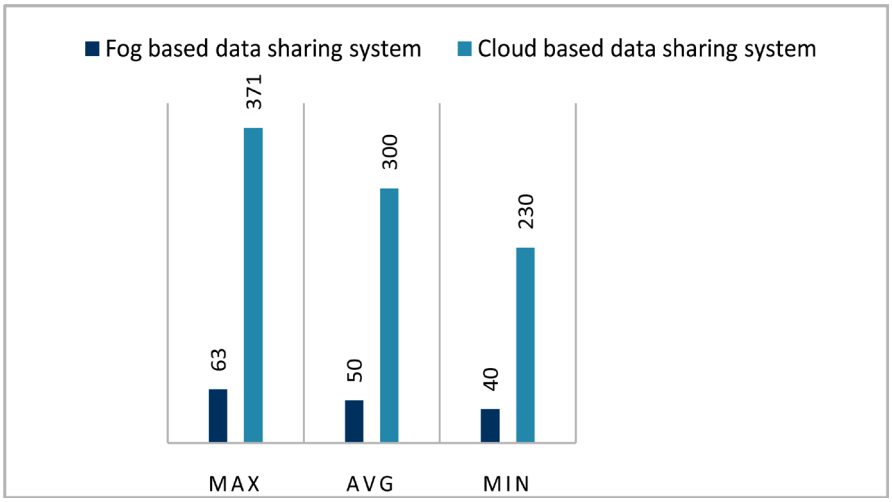

Figure 8. The simulationresults.

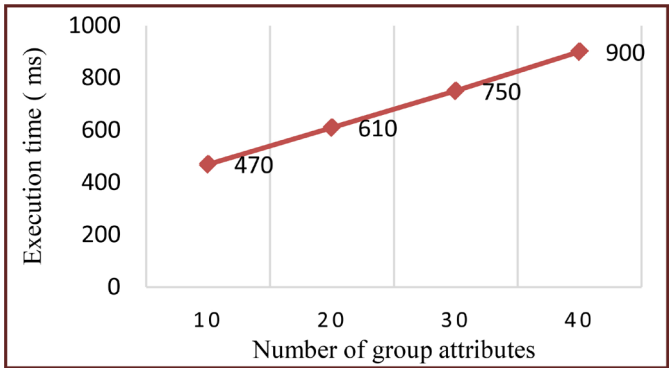

Figure 9. KP-ABE encryption.

The figure above shows that processing time for KP-ABE operations increases as the number of attributes increases. Our system groups users according to their attributes; thus, each group has a set of attributes that other groups do not have. There are not many attributes associated with each group. There is a direct relationship between a group's attributes and processing time. Thus, the number of attributes can be optimized to achieve optimal execution time. In addition, the propagation delay is significant compared to processing time, and therefore the impact of fog computing can be easily appreciated.

2) User Revocation Processing Time

The user revocation process was explained in Section IV. In this section, we will show the impact of group size per fog node on the execution time for the parameters concerned with the revocation process. We can calculate the time required for the user revocation phase as follows:

$$\text{Revocation Time} = \text{Re-generation Time}(\text{SKs}) + \text{Re-encryption Time}(\text{files})$$

The process requires that the fog node regenerate secret keys and re-encrypt files during each revocation process. Therefore, we will vary the number of users assigned to each fog node as a percentage of the overall number of users in the system.

First, we assume that there are 50,000 system users and 5000 system files. We also executed these operations on more than size of the fog node (e.g., 5%, 10%, 15%, and 20% of users) as shown in Figure 10 and Figure 11.

It has been shown that as the number of assigned users per fog node decreases, the processing time for key regeneration and file re-encryption decreases significantly in an order of about 25% as we move from one fog size to the next lesser one.

Although the reduction is significant but this could be translated in real life as adding more fog nodes at user's vicinity which could be considered as a trade off problem. In addition, our comparison of key regeneration time and file re-encryption time clearly shows that regeneration requires more processing time than re-encryption, which emphasizes that key regeneration is the dominant factor in the revocation process.

To compare our system to a cloud -based data-sharing system, we set up an experiment. The data in Table 4 shows that our system reduced the processing time required for key regeneration compared to the cloud system by about 80% (based on our setup) thanks to the fog computing architecture.

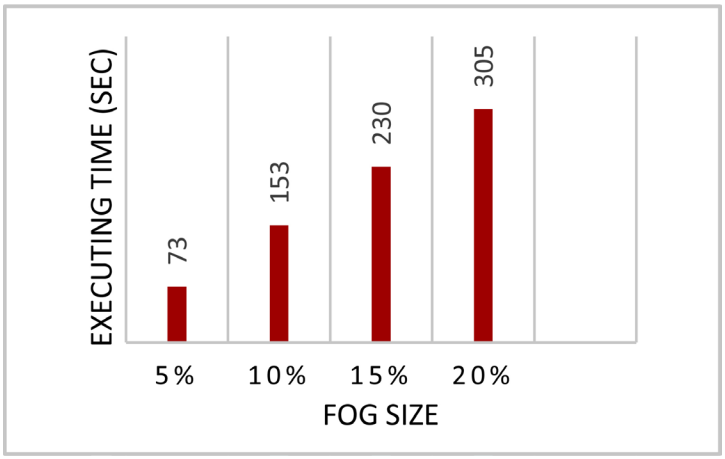

Figure 10. Secret keys updating on fog node.

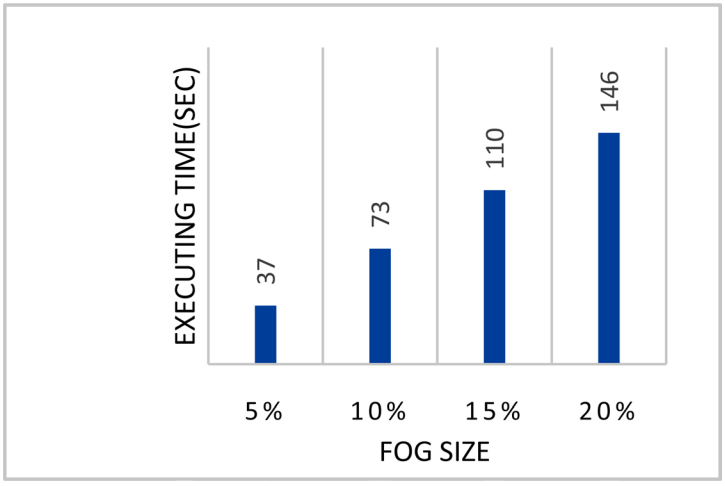

Figure 11. The re-encryption of files on fog node.

Table 4. Secret key updating.

Systems	Cloud Systems	Our System Fog size (20%)
Affected Users	All	Group users
Number of Users	50,000	10,000
Updating Time	25 minutes	5 minutes

Table 5. The files re-encryption.

Systems	Cloud Systems	Our System Fog Size (20%)
Affected Files	All	Group files
Number of Files	5000	1000
Updating Time	755 seconds	146 seconds

Table 6. Revocation time.

Systems	Cloud Systems	Our System Fog Size (20%)
Revocation Time	37.5 min	7.3 min
Improvement Rate		80%

To further demonstrate the benefit of our system, we compare the performance of cloud and fog setups with respect to the maximum number of reencryption operations per group. It is obvious that we get a better re-encryption time, as the number of the group files reduced in the fog architecture comparing to the entire cloud system files, as shown in Table 5. This is a direct result of the file distribution strategy suggested in Section 3.

3) Overall Delay Estimation Due to Revocation

Revocation time refers to the time required to regenerate secret keys and re-encrypt files during each revocation process. Based on the tables and equation above, we can calculate and compare the total time required for the revocation operation in fog and cloud systems. Table 6 shows that we achieved better revocation time compared to the cloud system; the results indicate that our system was 80% better.

5.3. Discussion

Our system responds to user requests more quickly than do classical cloud systems. As we have proven previously, the response time of a fog system is 84% less than that of a cloud system because the fog is located close to the users. The findings demonstrated the feasibility of our system, as the computation-related complexity of the system's operations does not depend on the number of users in the system, but on the number of system attributes, and is thus scalable.

The experimental results show that our revocation mechanism outperforms cloud-based systems as this phase is at the fog level in our system and thus does not affect the entire system. Also, our system requires 80% less processing time for key regeneration compared to the cloud system due to the fog computing architecture.

When comparing the performance between cloud and fog setups with respect to the maximum number of re-encryption operations, it is obvious that we get a better re-encryption time, as the number of the group files reduced in the fog architecture comparing to the entire cloud system files. Further, our system has 80% better revocation time compared to the cloud-based system.

Based on the previous discussion, our revocation mechanism outperforms cloud-based systems, and the proposed framework is a promising solution for data sharing in the emerging fog computing environment.

6. Security Analysis

We first analyze security of our framework in terms of the security features as formulated below.

6.1. Data Confidentiality

The data owner assigns the management of the data to the fog nodes and CLDs, which are considered the main adversaries against data confidentiality. They are a bigger threat than unauthorized users because of their adversarial capabilities. The fog nodes have the encrypted data and the responses to the authorized users from addressing their requests.

To achieve data confidentiality, the data owner encrypted the file using SEK by AES algorithm, and SEK is encrypted using KP-ABE, before uploading it. Thus, fog nodes cannot have access to plaintext. In addition, they have only a partial copy of the user's secret keys and lack the SEK key. To achieve data confidentiality in transmission, the communication channels are secured between the data owner/cloud server and the fog nodes using SSL/TLS protocol to overcome network attacks [23].

6.2. Fine-Grained Access Control

The data owner must have access control over his or her secret data, meaning that a valid user cannot obtain unauthorized data. The enforcement of fine-grained access control is achieved using (KP-ABE), which provides a flexible access structure for each user based on data attributes [9].

6.3. User Revocation

When an authorized user is revoked, his access right is dropped, and he is considered an outsider. The revocation operation requires the re-encryption of the files and a re-distribution of the new keys at one fog [24].

The proposed user revocation process is achieved by using the PRE technique. This reduces the data owner's burden and assigns most of the tasks to the fog node, which permits the fog node to re-encrypt the data automatically without discovering the file's contents.

6.4. Collusion Resistance

To protect against collusion, parties must not be permitted to access the data

without authorization from the data owner. To prevent collusion between servers and any party, the system must protect the users' access privilege information against the cloud and fog nodes by encrypting the user's access structure before sending it.

To prevent collusion between users, the system is a set of groups, and each group has a different public key (*GPK*); thus, users of different groups cannot combine their secret keys and decrypt files they should not be allowed to access [25].

7. Conclusions and Future Work

The aim of the present study was to design a secure data sharing framework for a fog environment. This framework achieved fine-grained access control, data confidentiality, user revocation, and collusion resistance. Our proposed framework rests on a combination of KP-ABE and PRE-techniques. The contribution of the study was the confirmation that our system outperformed the cloud-based data sharing architecture. Our framework provides high scalability and data sharing in real time and with low latency.

The findings of this study indicate that our system outperforms cloud-based data sharing systems with its faster processing time. The simulation results also show that our system responds faster to user requests than classical cloud systems. Further, the experimental results show that our system also outperforms cloud-based systems in the user revocation phase. In this paper, we proved that our proposed framework provides a promising solution to securing data sharing in the emerging fog computing environment. A future study addressing a many-to-many data sharing application in a fog environment would thus be interesting.

References

[1] Firdhous, M., Ghazali, O. and Hassan, S. (2014) Fog Computing: Will It Be the Future of Cloud Computing. *Proceedings of the* 3*rd International Conference on Informatics & Applications*, Kuala Terengganu, Malaysia, 8-15.

[2] Bonomi, F., Milito, R., Zhu, J. and Addepalli, S. (2012) Fog Computing and Its Role in the Internet of Things. *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, Helsinki, Finland, 17 August 2012, 13-16. https://doi.org/10.1145/2342509.2342513

[3] Stojmenovic, J. (2014) Fog Computing: A Cloud to the Ground Support for Smart Things and Machine-to-Machine Networks. *Australasian Telecommunication Networks and Applications Conference* (*ATNAC*), Southbank, VIC, 26-28 November 2014, 117-122. https://doi.org/10.1109/atnac.2014.7020884

[4] Luan, T., Gao, L., Li, Z., Xiang, Y., We, G. and Sun, L. (2016) A View of Fog Computing from Networking Perspective. ArXivPrepr. ArXiv160201509.

[5] Dastjerdi, A., Gupta, H., Calheiros, R., Ghosh, S. and Buyya, R. (2016) Fog Computing: Principals, Architectures, and Applications. ArXivPrepr. ArXiv160102752.

[6] Yi, S., Hao, Z., Qin, Z. and Li, Q. (2015) Fog Computing: Platform and Applications. 2015 3*rd IEEE Workshop on Hot Topics in Web Systems and Technologies* (*HotWeb*), Washington DC, 12-13 November 2015, 73-78.

https://doi.org/10.1109/hotweb.2015.22

[7] Scale, M. (2009) Cloud Computing and Collaboration. *Library Hi Tech News*, **26**, 10-13. https://doi.org/10.1108/07419050911010741

[8] Thilakanathan, D., Chen, S., Nepal, S. and Calvo, R. (2014) Secure Data Sharing in the Cloud. In: Nepal, S. and Pathan, M., Eds., *Security, Privacy and Trust in Cloud Systems*, Springer, Berlin, Heidelberg, 45-72. https://doi.org/10.1007/978-3-642-38586-5_2

[9] Yu, S., Wang, C., Ren, K. and Lou, W. (2010) Achieving Secure, Scalable, and Fine-Grained Data Access Control in Cloud Computing. 2010 *Proceedings IEEE INFOCOM*, San Diego, CA, 14-19 March 2010, 1-9. https://doi.org/10.1109/infcom.2010.5462174

[10] Wu, Y., Wei, Z. and Deng, R. (2013) Attribute-Based Access to Scalable Media in Cloud-Assisted Content Sharing Networks. *IEEE Transactions on Multimedia*, **15**, 778-788. https://doi.org/10.1109/TMM.2013.2238910

[11] Liu, Q., Wang, G. and Wu, J. (2014) Time-Based Proxy Re-Encryption Scheme for Secure Data Sharing in a Cloud Environment. *Information Sciences*, **258**, 355-370. https://doi.org/10.1016/j.ins.2012.09.034

[12] Tu, S., Niu, S., Li, H., Yun, X.-M. and Li, M. (2012) Fine-Grained Access Control and Revocation for Sharing Data on Clouds. 2012 *IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum* (*IPDPSW*), Shanghai, 21-25 May 2012, 2146-2155. https://doi.org/10.1109/ipdpsw.2012.265

[13] Yang, Y. and Zhang, Y. (2011) A Generic Scheme for Secure Data Sharing in Cloud. 2011 *40th International Conference on Parallel Processing Workshops* (*ICPPW*), Taipei City, 13-16 September 2011, 145-153. https://doi.org/10.1109/ICPPW.2011.51

[14] Hur, J. (2013) Improving Security and Efficiency in Attribute-Based Data Sharing. *IEEE Transactions on Knowledge and Data Engineering*, **25**, 2271-2282. https://doi.org/10.1109/TKDE.2011.78

[15] Samanthula, B., Howser, G., Elmehdwi, Y. and Madria, S. (2012) An Efficient and Secure Data Sharing Framework Using Homomorphic Encryption in the Cloud. *Proceedings of the 1st International Workshop on Cloud Intelligence*, Istanbul, Turkey, 31 August 2012, Article No. 8. https://doi.org/10.1145/2347673.2347681

[16] Zhang, R. and Chen, P. (2012) A Dynamic Cryptographic Access Control Scheme in Cloud Storage Services. *8th International Conference on Computing and Networking Technology* (*ICCNT*), Gueongju, 27-29 August 2012, 50-55.

[17] Do, J., Song, Y. and Park, N. (2011) Attribute Based Proxy Re-Encryption for Data Confidentiality in Cloud Computing Environments. 2011 *First ACIS/JNU International Conference on Computers, Networks, Systems and Industrial Engineering* (*CNSI*), Jeju Island, 23-25 May 2011, 248-251. https://doi.org/10.1109/cnsi.2011.34

[18] Qiao, Z., Liang, S., Davis, S. and Jiang, H. (2014) Survey of Attribute Based Encryption. 2014 *15th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing* (*SNPD*), Las Vegas, NV, 30 June-2 July 2014, 1-6.

[19] Ateniese, G., Fu, K., Green, M. and Hohenberger, S. (2006) Improved Proxy Re-Encryption Schemes with Applications to Secure Distributed Storage. *ACM Transactions on Information and System Security*, **9**, 1-30. https://doi.org/10.1145/1127345.1127346

[20] Yu, S. (2010) Data Sharing on Untrusted Storage with Attribute-Based Encryption. Ph.D. Thesis, University of Massachusetts, Lowell.

[21] Butterfield, E. (2016) Fog Computing with Go: A Comparative Study. CMC Senior Theses, Paper 1348.

[22] Li, J., Yao, W., Zhang, W., Qian, H. and Han, J. (2016) Flexible and Fine-Grained Attribute-Based Data Storage in Cloud Computing. *IEEE Transactions on Services Computing*, **PP**, 1. https://doi.org/10.1109/tsc.2016.2520932

[23] Ahmed, M. Xiang, Y. and Ali, S. (2010) Above the Trust and Security in Cloud Computing: A Notion Towards Innovation. 2010 *IEEE/IFIP 8th International Conference on Embedded and Ubiquitous Computing* (*EUC*), Hong Kong, 11-13 December 2010, 723-730. https://doi.org/10.1109/euc.2010.114

[24] Ibrahim, I., El-Din, S., Elgohary, R., Faheem, H. and Mostafa, M. (2013) A Generic, Scalable and Fine-Grained Data Access System for Sharing Digital Objects in Honest but Curious Cloud Environments. 2013 *International Conference on Cloud Computing and Big Data* (*CloudCom-Asia*), Fuzhou, 16-19 December 2013, 15-22. https://doi.org/10.1109/CLOUDCOM-ASIA.2013.21

[25] He, H., Li, R., Dong, X. and Zhang, Z. (2014) Secure, Efficient and Fine-Grained Data Access Control Mechanism for P2P Storage Cloud. *IEEE Transactions on Cloud Computing*, **2**, 471-484. https://doi.org/10.1109/TCC.2014.2378788