5-2021

Securing Fog Federation from Behavior of Rogue Nodes

Mohammed Saleh H. Alshehri
University of Arkansas, Fayetteville

Securing Fog Federation from Behavior of Rogue Nodes

A doctoral dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in Engineering with a Concentration in Computer Science

by

Mohammed Saleh H Alshehri
King Khalid University
Bachelor of Science in Computer Science, 2010
University of Colorado at Denver
Master of Science in Computer Science, 2014

May 2021
University of Arkansas

This dissertation is approved for recommendation to the Graduate Council

---

Brajendra Panda, Ph.D.
Dissertation Director:

Susan Gauch, Ph.D.
Committee Member

Qinghua Li, Ph.D.
Committee Member

Paul Cronan, Ph.D.
Committee Member

ABSTRACT

As the technological revolution advanced information security evolved with an increased need for confidential data protection on the internet. Individuals and organizations typically prefer outsourcing their confidential data to the cloud for processing and storage. As promising as the cloud computing paradigm is, it creates challenges; everything from data security to time latency issues with data computation and delivery to end-users. In response to these challenges CISCO introduced the fog computing paradigm in 2012. The intent was to overcome issues such as time latency and communication overhead and to bring computing and storage resources close to the ground and the end-users. Fog computing was, however, considered an extension of cloud computing and as such, inherited the same security and privacy challenges encountered by traditional cloud computing. These challenges accelerated the research community's efforts to find practical solutions. In this dissertation, we present three approaches for individual and organizational data security and protection while that data is in storage in fog nodes or in the cloud. We also consider the protection of these data while in transit between fog nodes and the cloud, and against rogue fog nodes, man-in-the-middle attacks, and curious cloud service providers. The techniques described successfully satisfy each of the main security objectives of confidentiality, integrity, and availability. Further we study the impact of rogue fog nodes on end-user devices. These approaches include a new concept, the Fog-Federation (FF): its purpose to minimize communication overhead and time latency between the Fog Nodes (FNs) and the Cloud Service Provider (CSP) during the time the system is unavailable as a rogue Fog Node (FN) is being ousted. Further, we considered the minimization of data in danger of breach by rogue fog nodes. We demonstrate the efficiency and feasibility of each approach by implementing simulations and analyzing security and performance.

ACKNOWLEDGMENTS

I have been privileged to have benefitted from the expertise, guidance, and support of many throughout the research and writing of this dissertation. I would like to express my deepest appreciation for the support of my advisor, Professor Brajendra Panda. This dissertation is based on three published papers; two of them awarded best paper. Without the meaningful support and direction by Professor Panda, that would not have been possible. Thank you, Professor Panda. It was an honor working under your supervision. I would like to thank those who served on my doctoral committee: Dr. Susan Gauch, Dr. Qinghua Li, and Dr. Paul Cronan. You honor me with your approval of my work. Thank you for your valuable comments and directions throughout this dissertation. My deepest gratitude to my parents, who have given me unwavering love, prayers, and support all my life. It is time to pay you back. Particular thanks to my wife, and kids. Your unconditional love, support, and patience have sustained me as I pursued my Ph.D. Thank you. To Ministry of Education and Najran University, Saudi Arabia. This dissertation would not be possible without the generous scholarship you offered me to pursue my graduate education.

DEDICATION

To my parents, brothers, sisters, my wife and my children.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 Introduction

1.1 Introduction to the Research Problem

Since the beginning of the 21^{st} century, popularly known as the internet revolution age, the world has seen swift changes in terms of processing and storage facilities. This has been attributed to consumer demands as a result of the dynamism surrounding computer technologies. This makes the new computing resources very affordable and widely accessible than during any other human age [1][2]. One element of the computer technologies revolution is the cloud, whereby the cloud computing model has gained popularity as a result of the on-demand services it provides, such as storage over the internet to the end-users [3][4][5]. The popularity among the consumers and end-users can be attributed to its accessibility, scalability, and reasonably priced packages. As mentioned above, cloud computing has been largely utilized for storage and therefore providing solutions to individuals to store their data safely and securely in the cloud [6][7][8]. Beyond storage, cloud computing is also used for other computing services such as networking, databases, providing intelligence, and analytics. Therefore, organizations and individuals have resorted to the use of cloud storage in place of local storage. One of the advantages of this computing resource is that data can be processed regardless of the size of data or the software or hardware needs. It also is secure as data is backed up safely in the cloud. There are, however, drawbacks associated with this form of computing service.

1.1.1 Drawbacks of Cloud Computing

Cloud computing, though important in the aspects of storage, also has its limitations that can reduce efficiency in its operability. One of these drawbacks is the issue of high end to

end delay, also referred to as high latency. High latency is actually unavoidable, especially in the public cloud due to the high number of users and also due to differences in geographical locations for end-users. This form of latency is the delay that occurs when a user requests a cloud service provider [9][10]. These delays cause a slowdown in communication and a high number of users in public clouds worsens the situation accelerating data delays causing communication overhead [11].

Another drawback concerns security for end-users' data and privacy, which are prone to malicious violations. This violation can occur as the data is being transported or during the process of cloud storage. Cloud computing is known to possess a weakness of having semi-trusted third parties who can control the cloud [12][13]. Access control can be advantageous and also disadvantageous in cloud computing. The drawback of access control in cloud computing is that organizations or individuals have limited to no control over the data or the security [14].

Secure processing and data storage for individuals and organizations pose a huge challenge for users as they are concerned about their data security. Security issues concerning data are important to both software and hardware. This has led to the implementation of cryptographic systems and algorithms to handle security and privacy issues in cloud computing. The problem of latency with relation to cloud computing is not well resolved; therefore, it is a gap for technological innovation [9][15]. The issue of latency is a problem for individuals and organizations because of delays and timeliness. This can be a problem, especially during retrieval of data such as medical records data for patients, hence influencing the provision of quality services for clients [9].

1.1.2 Fog Computing and Related Issues

The various issues concerning cloud computing such as high latency led to the introduction of fog computing in 2012 by CISCO [10]. Although cloud computing has assisted individuals and organizations to a wide array of computing resources, the time needed to access cloud applications can be too long and, therefore, may not perform well when needed. This also applies to communication overhead problems, which are resolved through fog computing [9][15]. Due to the dynamism of technologies and the changing needs of consumers, and the amount of data being generated and the rising number of devices connected need cloud resources to be on the edge, which is close to the user for easy access [15]. Fog computing, more importantly, provides linkages between the cloud and end-users such as through Internet of Things (IoT) nodes [16]. Therefore fog computing helps with issues of latency by offering low latency, high bandwidth, security, location awareness, network load efficiency, real-time, capacity to process a high number of nodes distributed appropriately, and finally, mobility [17][18].

Fog computing aides computing, storage, networking, and management of data near IoT devices to enhance access for users [17]. Therefore, this enhances the security and privacy of the users through measuring, processing, and analyzing devices [13][19]. Fog computing is an extension of cloud computing [19]. Fog computing achieves this through the numerous edge nodes that send data directly to IoT devices. This can also cause issues while data travels through the different fog nodes that help to transit the data to the end-users.

Because fog computing is an extension of cloud computing, there is the issue of 'inheriting' problems related to security and privacy from cloud computing. Security and privacy challenges can occur while data moves from the Fog Nodes (FN) and Cloud service provider. Privacy can also be manipulated while data is being stored. Security of data must target to avert the loss of integrity, loss of availability, and loss of confidentiality.

Manipulation of data can be in many forms: 1) A fog node can go rogue and expose sensitive data in transit. 2) End-users' confidential data can be compromised when fog nodes are compromised through malicious attacks. 3) Interception of private data by malicious party in transit between fog nodes and end-user, or between fog nodes and the cloud. End-users' confidential data could be accessed by malicious unauthorized actors through rogue fog nodes, hence breaking the data security goal of confidentiality. This brings up the element of man-in-the-middle (MITM), who is a malicious attacker who intercepts communication between end-users and fog nodes. Encryption and integrity check algorithms can be applied to enhance the integrity and thereby, privacy. Access keys should be robust to prevent MITM attacks such as brute-force-attack that uses the trial and error method to gain access. Additionally, cryptographic keys should never be handed to the cloud service provider as curiosity can lead them to access data, hence breaking user confidentiality. Therefore, this research considered the protection of end-users' confidential data through the provision of data security from rogue fog nodes, man-in-the-middle attack, and curious cloud storage provider/administrator in the context of fog computing.

Fog computing has a vital feature of reducing time latency that exists between cloud providers and end-users' devices (such as IoT devices). Fog nodes are location-sensitive, and therefore, each must serve the intended purpose for users in that location; however, fog nodes can fail, and this can consequently disrupt computing services. This means that user access will be limited by the failed fog node. This calls for the development of approaches or techniques or even systems to ensure the continuity of system processes such as storage even with a failed fog node.

Communication overhead includes the time used to complete assigned work. Communication overhead issues should be reduced in order to lower latency. This could be between the Cloud Service Provider (CSP) and also Fog Nodes (FNs). Therefore, reducing the com-

munication time between FNs and CSP through different approaches such as Fine-grained data access control in CSP and FNs can aide in subsiding the latency in the fog computing system. A fog node is used to transfer data from the end-users to the cloud. In the process, it can be difficult to protect other fog nodes not being in use at that particular moment in the process of transit. A malicious attacker can use this weakness as his/her point of attack and illegal access. Therefore, an attack on the fog node would negatively influence the security of users' data. The same case applies if the fog node goes rogue. This can be, however, resolved through the use of secure communication methods in order to transfer data to and from fog nodes securely and without the worry of compromise by fake end devices.

For that reason, it is essential to design methods for the purpose of shielding users' data by keeping it confidential and available by ousting rogue fog nodes. From the discussion above, we can deduce that there are two forms of threats. 1) Data modification which occurs when an malicious third party is able to access end-users' private data. In this process, the attacker violates their confidentiality and integrity. 2) Unauthorized access; which occurs when an attacker disrupts a fog node. This can result in exposure of a user and their confidential private data leading to a breach in data security. Additionally, data availability will be hindered in case a malicious attacker accesses data that was in transit. This makes the availability of data an important aspect of data security and privacy.

1.2 Research Overview

This section will provide a summary of the contributions of this dissertation. We have offered three research contributions to the issues surrounding data security through the recommendation of approaches.

1.2.1 An Encryption-Based Approach

The application of fog computing has been majorly on streaming, whether it is video streaming or playing games online. Such applications require the use of fog computing because the fog nodes are located near the user hence significant for quick decisions such as in health care services [10]. A Fog computing system presents diverse security matters besides the numerous benefits that it provides for users. A fog node, for example, needs to be active for it to perform its intended task. If it is not, then, end-users cannot get value of the features provided by fog computing. If a similar occurrence of having an inactive fog node happens in a healthcare facility, a malicious attacker may intercept data and risk the confidential patient data. It also could cause high time latency and communication overhead to retrieve end-users' data from the cloud.

These issues have not been addressed by previous literature in the context of fog computing; hence it creates a research gap that is addressed in this approach.

This approach, therefore, proposes an Encryption-Based-Approach, a method developed to hinder unauthorized access of encrypted data by rogue fog nodes through the development of fog-federations. This ensures that data security goals of availability, integrity, and confidentiality are upheld. End-users need to be protected from attacks on their confidential data as well as be protected from the irregularities that occur when rogue fog nodes attack other fog nodes. For example, rogue FNs may deceive other FNs to breach end-users' private data. Fog Nodes (FNs) also need to communicate or store data in the cloud without the problem of communication overhead and also latency. The idea of Fog-Federation (FF) allows the organization of Fog-Nodes (FNs) into arrangements referred to as Fog-Federations (FFs). Grouping is in relation to the attributes they possess, such as location attributes and classification by the form of services they provide, therefore, restricting the rogue Fog-Node (FN) from accessing encrypted data stored whether in the cloud or other FNs in the same

FF. The stratification of files before encryption is for tracking purposes while assessing their individual needs. Every group has its specific set of attributes (e.g., locations and services). Also, every fog node serves only one group and cannot be applied for any other purpose. It will, therefore, remain intact even if an FN from another different group is retracted. Re-encryption in this case applies only for that one group accesses files. This process enhances time management, hence low latency is achieved. This is achieved by dividing files to be encrypted into blocks of the same size and equipping the Cloud-Service-Provider CSP) with a tracking table to track each FN current possess blocks and needs. Rogue/malicious FN will not be able to access the data stored in the CSP or other FNs in the same FF upon the re-encryption process.

This will be broadly discussed in Chapter 3 through the introduction of its objectives, design, algorithms, simulation, and experimental evaluation.

1.2.2 Multi-Level Approach

Fog computing has revolutionized the idea of computing faster and efficiently through the numerous resources it makes accessible to users. The various security issues limit these benefits that fog computing is meant to provide, especially if the fog node goes rogue or is not working. For this problem, we came up with a proposal of the Fog-Federation (FF) in 1.2.1. This approach would work by preventing fog nodes from violating the user's data security. Rogue and malicious fog nodes are ejected to ensure that integrity and confidentiality are conserved by inhibiting them from gaining access to the encrypted data kept in the cloud or in other FNs in the same FF group. This cuts down on access time and hence time latency and communication overhead between FNs and the cloud.

FF enhances capacities in terms of data aggregation, processing, and, finally, the storage aspect. These aspects can become problematic because there is a possibility of an FN

accessing all the data encrypted with the set attributes for that specific FF. In this situation, if one of the FNs in the FF goes rogue, there is a possibility of breaching confidential and encrypted data.

Other potential security concerns related to fog computing is data hijacking that results in breaching of data or loss of data. This can happen in different ways, such as malicious actor taking charge of the intercepted fog node or obtaining data maliciously from the fog node. Also, a fog node might go rogue. In this case, end-users' data security would be in danger of being breached. A fake fog node, once it has access to private data, can breach confidentiality and violate data integrity as private data will be at risk.

To address these issues, we propose this approach as a method that categorizes the FNs found in FF into Fog-Federation-Subgroups (FFSGs) to reduce the amount of data has a possibility of being breached by the rogue fog nodes. In this case, files are divided into three blocks whereby each block is available if FNs meet the established FFSG attributes—therefore meeting the data security goals (e.g., confidentiality, integrity, and availability) for protection against security issues related to fog nodes such as the rogue fog nodes, Man-in the Middle Attack and the curious cloud service provider.

This approach, therefore, has filled the gap of having no literature in the exact topics, as discussed in the context of fog computing. Thereby, Chapter 4 provides a depiction of the FFSG concept, multilevel objectives, the multilevel approach design, the designed model, and the algorithms intended to accomplish the objectives related to the proposed approach. After that, a performance evaluation will be available. It contains the correctness of security analysis, the multilevel approach design, and our performed experiments to demonstrate the feasibility and applicability of our approach. Later, discussion and conclusion parts will be provided to provide a summary of the approach.

1.2.3 A Blockchain-Encryption-Based Approach

Blockchain related approaches provide security and a fine-grained data access scheme. The discussed approaches above intend to get rid of rogue fog nodes that hinder the security of data. However, these methodologies are dependent on third parties for the execution and also are not able to differentiate between a legitimate and a rogue fog node. A blockchain, therefore, plays a role in preventing FNs from violating the security of data for the user, especially when an FN has been ousted. Also, they lack tracking the identity of legitimate FNs in the same FF and what encrypted data they are allowed to access. Fog nodes are classified into fog federations depending on their attributes (e.g., locations and services).

In Blockchain-Encryption-Based Approach, authorization occurs in a distributed manner, and they also have the capability of providing every fog node with an off-line database that keeps the recently accessed data to accurately track the encoded data and inhibit other FNs from altering the tracking data and, also an on-chain database that now protects data from being interfered with by rogue fog nodes. Therefore, the blockchain is unique in that it is an 'interfere free' system. This approach also reduces time latency and communication overhead, which involves the Cloud-Service-Provider (CSP) and FNs through off-line database tracking. This approach further exploited and sought to incorporate the blockchain technology with the Ciphertext-Attribute-Based-Encryption algorithm. In this case, blockchain is used in an on-chain file tracking-table for verification before any access is accorded to the data. Also, each fog node would be identified using the smart contract before any access on the CSP. Blockchain, therefore, provides services not provided by other approaches and is therefore feasible.

A more detailed discussion is provided in chapter 5, broadly discussing the idea behind the use of blockchain and the approach's objectives. Chapter 5 also provides an illustration of the proposed approach in full details together with assumptions made to achieve the

objectives. It also discusses the threat model and designed algorithms. Then, chapter 5 provides performance evaluation and security analysis. The final parts are the discussion and the conclusion of the chapter.

1.3 Conclusion and Dissertation Outline

This chapter provides a short analysis of the aspects surrounding security and privacy issues in fog computing while considering the approaches that might be applicable for mitigating the security concerns. This chapter presented the research problem and a summary of the dissertation's contributions. Chapter 2 discusses the background of the topic as well as related literature. Further, chapter 3 provides a discussion of the Encryption-Based approach, chapter 4 introduces the Multilevel Approach, and finally, chapter 5 discusses the Blockchain-Encryption-Based Approach. Chapter 6 concludes this dissertation.

2 Background and Literature Review

This dissertation focuses on multiple purposes regarding the security of data in the environment of fog computing. The first purpose addresses security issues and proposes methods to secure fog-computing through approaches intended to satisfy the information security goals (i.e., confidentiality, integrity, and availability) for data while being in transit or during computation/storage in fog nodes or cloud. The second purpose is to provide a discussion of security issues rogue fog nodes could cause to end-users' confidential data and prevent them from breaching these confidential data. Thereafter, this dissertation focuses on providing approaches to detect and oust rogue fog nodes with maintaining data security. Finally, we discuss the secure communication among fog nodes in a trust-less environment.

In this chapter, advanced cryptographic primitives will be reviewed and exploited to achieve the proposed approaches, as well as the blockchain technology. This is followed by a broad study of the literature's approaches that have been developed for managing rogue fog nodes and protecting and securing data in fog computing. Subsequently, we provide a critical analysis of the current approaches for protecting data security in fog computing. Finally, a conclusion summarizes the dissertation.

2.1 Attribute-Based Encryption (ABE) Algorithms

In this section, we reviewed the literature of cryptographic and technological primitives we exploited in this dissertation. We start with the Attribute-Based Encryption Scheme (ABE) as we adopted the Ciphertext-Policy Attribute-Based Encryption (CP-ABE) algorithm in our approaches.

ABE was introduced by Sahai and Wates in 2005, as an amendment of the identity-

based encryption [20]scheme. In this scheme, every user who accesses the system has a set of attributes assigned to him/her. This scheme aims to support users and owners in sharing or distributing confidential data with others, depending on the access policies they provide. Encryption is usually undertaken depending on the receivers' attributes, which makes ABE algorithms broadly applied to perform multiple encryption processes.

According to Hohenberger and Waters [21], ABE guided by available user attributes, allows users to encrypt and decrypt messages. ABE is one of the cryptographic schemes that ensure fine-grain data access control in cloud computing. One of those ABE schemes is Ciphertext-Policy Attribute-Based Encryption (CP-ABE). The attribute-Based Encryption (ABE) is an advanced cryptographic primitive that can be utilized to run multiple encryption systems. In ABE systems, the ciphertext can be encrypted to multiple recipients depending on the predefined attributes of the recipients [22]. In this cryptographic system, for example, ABE, the private keys assigned to users, and ciphertext are related to a predefined formation of attributes or an arrangement of policies over attributes [23]. Users that fulfill the attributes associated with the ciphertext have the right to access the ciphertext.

There are two variants of the ABE system: ciphertext-policy attribute-based encryption (CP-ABE) [24] and key-policy attribute-based encryption (KP-ABE) [25]. The CP-ABE variant has been further explored and described below.

2.1.1 Ciphertext-Attribute-Based Encryption (CP-ABE)

Ciphertext-Policy Attribute-Based Encryption is a system that provides complex access control on data to limit compromising of the data in the cloud storage. In CP-ABE, users' private keys are linked with a set of attributes, while ciphertext is linked with an access structure constructed over the predefined attributes. Encryption occurs depending on the credentials described by users; therefore, users are identified using their specified

attributes, making it hard for malicious users whose attributes are not described to access data. Therefore, users are unable to gain access to encrypted data unless their attributes mathematically align with the access structure. CP-ABE is fundamentally comprised of four algorithms:

1. **Setup:** It takes security parameter as input to produce a public key and a master key (PK, MK).

2. **Encrypt:** The inputs are described as attributes, public parameters, and the message, while its outputs are ciphertext CT. No user can decrypt CT unless their inputted attributes fulfill the requirements of the access structure A in the policy. The ciphertext implicitly contains A.

3. **KeyGen:** In this algorithm, the inputs comprise the master key and a set 'S', a block of attributes descriptive of the key. A private key SK is generated.

4. **Decrypt:** It takes the public parameters PK , the Ciphertext CT, and the private key as inputs. The output is the message (e.g., plaintext) generated if the inputted user attributes meet the requirements of the access structure A attached to CT.

2.1.2 Bilinear Mapping

As described in [20], let G_1 and G_2 be a cyclic mathematical groups, multiplicative, of large prime order p, with g as a generator. Also, Let e be a bilinear map: $e : G_1 X G_1 \rightarrow G_2$. Such that, $G2$ is the output space of the bilinear mapping. Therefore, bilinear map e must comply the following conditions:

1. **Bi-Linearity:** $\forall\ u, v \in G_1, e(u^a, v^b) = e(u, b)^{ab}$ where $a, b \in Z_p^*$

2. **Non-degeneracy:** No pair in $e : G_1 X G_1$ is mapped to the element of identity .

3. **Computability** This means that the bilinear map $e(u, v)$ must be efficiently computable.

2.1.3 Access Structure

With reference to [26], let the set $\{P_1, P_2, ..., P_n\}$ be a set of parties. We can say that collection $\{\mathrm{A}\} \subseteq 2^{(P_1, P_2, ..., P_n)}$ is monotone if $\forall\ B \in A$ and $B \subseteq C$ then $C \in A$. In this scenario, the collection A of non-empty subsets of $\{P_1, P_2, ..., P_n\}$ is considered a monotone access structure. $\forall$ sets $\in A$, they can be referred to as authorized sets while the parties $\notin A$ are the unauthorized parties. As same as provided in [22], our approaches also represent the parties with attributes; all fog nodes with the same attributes will produce a fog federation (FF). Therefore, as described further in chapters 3,4, and 5, the fog nodes with similar attributes were categorized into fog federation subgroups.

2.2 Blockchain Technology

Blockchain technology has shown a promising and flourishing future for applications, especially for recording data such as transactions. The history of blockchain shows that many non-financial applications are adopting it besides the cryptocurrency [27]. For example, blockchain has been used in healthcare, agriculture, the food industry, and many other applications. A blockchain fundamentally allocates record database for all transactions as implemented and shared among contributing parties [28]. Every transaction in the public register is substantiated by consensus favored by the majority of the parties in the system. When information is entered and proofed, it cannot be erased or changed. Blockchain has developed to be a widely available infrastructure for the expansion of decentralized applications. Those applications constructed on the blockchain can attain components such as immutability, integrity, transparency, and non-repudiation [29].

2.2.1 Smart Contracts

Blockchain assists users in applying parts of software, usually known as smart contracts. The blockchain technology is comprised of three generations that enhance their operations and include: money transaction, assets, and smart contract. Blockchain was first published by Satoshi Nakamoto [30], but at that time, it was focused on money transactions only. Thereafter it began focusing on exchanging assets among different parties [31]. Blockchain has been enhanced by introducing the smart contract concept [32]. These pieces of software are usually seen by everyone in the network as described in [32], and must be controlled by the network parties.

Smart contracts work through self-execution of contracts whereby users use codes on their contracts found on a blockchain [33]. Through smart contracts, transactions are made safe for users, such as in banks, through their applications. Smart contracts, therefore, store valuable data safely and can be retrieved. However, the issue of security is present due to the fact that most blockchain systems are in the public domain; therefore it is susceptible to malicious attacks [33]. In this dissertation, we adopted the smart contract concept in our third approach, see chapter 5. For further reading about blockchain and smart contract, read through [34][35].

2.3 Fog Computing

Fog computing is also known as fogging and is like an extended form of cloud computing. The difference between fog and cloud computing is that in fog, data is processed locally while in the cloud, it is sent to the cloud [36]. The concept was first introduced by Cisco in 2012. When describing fog computing, we should visualize the mid-layer that falls between cloud computing providers and devices employed by end-users, including PCs or other devices [10]. Fog computing is therefore, closer to the end user than the cloud comput-

ing. Fog computing enables computation, storage, and data management to be shifted close to the ground, reducing time latency and communication overhead between the far cloud computing providers and end-users' devices. Fog computing sorts out the issue of having data so close to the data source. For example, compacting the GPS data can be flowed at the edge of the system before being transferred to the cloud in Intelligent Transportation Systems [37].

The OpenFog Consortium in 2016 defined fog computing as: "a horizontal system-level architecture that dispenses computing services, storage, control and networking tasks nearer to the users along a cloud-to-thing continuum." In the following section, we define and discuss fog computing services with relation to computation and storage, and the associated features. The security issues in fog computing will also be reviewed as well as a critical analysis of fog nodes.

2.3.1 Fog Computing Services and Characteristics

This section summarizes fog computing services and the associated characteristics for organizations and individual users.

2.3.1.1 Services

Due to the significant distance between the cloud data center and the end-users' devices, cloud computing suffers from end-to-end delay, traffic congestion, and communication overheads [38]. Cloud computing cannot be used for activities such as real-time activities e.g., gaming; therefore, fog computing comes at play. It gets its name from fog, which is a cloud that is usually located near the ground. It is applied in smart home applications which are becoming common especially when a wide variety of services are needed in one application such as home security systems which involve numerous aspects such as alarm,

cameras and recorders, motion sensors and aspects such as key in locking system or even face recognition sensors. Fog computing also provides services in data management, especially in the healthcare industry[39]. This is because of the sensitivity that surrounds medical records and related data. Each provider, therefore, will possess a fog node that has medical data for that specific patient, and this can be stored in a smartphone.

2.3.1.2 Characteristics

Fog computing was formerly initiated by CISCO to provide the following characteristics [10][40]:

1. Support End-users' devices mobility.
2. Real-time applications rather than group processing to ensure timely service.
3. Low time latency between the far cloud providers and end-users. Latency in the form of execution time, offloading time for a task, and the speed of decision making.
4. It provides location awareness and distribution.

Due to immense support from big information technology companies such as CISCO, Dell, Microsoft Corp., Cloudlet, and Intelligent Edge, fog computing has a promising future.

2.3.2 Security and Privacy Issues in Fog Computing

Fog computing is faced with security and privacy challenges that change every time due to the dynamism of the IT systems. Previous measures utilized in cloud computing are not applied to fog computing because of the characteristics associated with fog computing [17]. As described in the previous section, fog computing has features that distinguish it from cloud computing such as: Low latency, support of the geographic distribution, end

device mobility, the capability of processing a large number of devices, wireless access, real-time use, and heterogeneity [17]. The security vulnerability of fog computing is certainly high as it exists in a principal network for end-devices and cloud. The Research community acknowledges that security and privacy issues for fog computing should be dealt with. However, current literature has discussed data security issues related to fog computing in terms of data security objectives (Confidentiality, trust, Integrity, and Availability), data access control, and communication security among fog nodes, and DoS attack.

2.3.2.1 Data Security Violation in Fog Computing

End-users utilize devices the Internet of Things (IoT), laptops, servers ...etc. in processing, transfer, and storage of confidential data. Privacy and security for end-users is a challenge as devices transmit confidential data to fog nodes for processing and further to the cloud where extended processing and storage occur. In the process, end-user security and privacy compromised as sensitive data may be collected. This confidential data is susceptible to malicious compromises. These privacy and security issues may occur through the interception by malicious third parties or by fog nodes or cloud services providers. The vast distribution of fog nodes also makes centralized control difficult. The fog node can also turn out to be malicious or rogue, which would result in the violation of end-users' confidential data while traveling through the malicious fog node to the cloud and from the cloud. Data can also be compromised if a fog node was poorly installed, inviting intrusion by a malicious actor. This is known as man-in-the-middle (MITM) attack- which violates data confidentiality, trust, and integrity in an unapproved manner. A provider of cloud services may exhibit curiosity about the aspects surrounding outsourced encrypted data resulting in a violation of data security and privacy. Additionally, new challenges in security and privacy have emerged. Ni et al. in [41] recommended the concept of Fog-based Vehicular Crowd

Sensing (FVCS) whereby these can store data momentarily in order to offer local services and also to play the role of cloud servers. The exchange of data about local situations can expose location data for the user [17].

Security requirements cannot be offloaded. Securing communication among the fog nodes themselves is a significant security issue. The fog nodes require an effective way to exchange data in a secure mode without violating data security. Data availability and low time latency found in the exchanges between the cloud and end-users' devices would be enhanced in this way. Fog nodes interact with one another when there is a need for network management. Additionally, when two fog nodes aim to exchange confidential data, they must have a safe way of protecting sensitive data from being exposed to malicious entities that might carry out unauthorized changes to the data being transferred. The fog node that seeks to retrieve data from another fog node must be certain that the fog node it is communicating with is legitimate; otherwise, the end-users' data security will be maliciously violated. Communication amongst fog nodes needs to be secured as the nodes utilized in the multi-hop route may be untrustworthy [17]. Research communities have studied these security concerns and have come up with different schemes to defend these confidential data, whether being in transit or in storage.

A Rogue fog node by definition, is a fog node that seems to be legitimate, but is, in reality malicious. A rouge fog node in fog computing is a device that is attempting to imperil a legitimate fog node [42]. Additionally, when a devices is damaged by malicious actors or invaders and acts as a genuine fog node in the fog network, provoking a connection by the user, it too becomes rogue [42]. It introduces, to the fog network, incorrect data either purposely, with harmful intent, or as a result of faulty devices in the system. Any fog node that has become rogue is, therefore, considered to be an entity of malice in fog networking.

The existence of a rogue, or fake, fog node causes a massive risk to end-user' data

privacy and the general security in the fog computing network. An example of this is when a malicious user influences a legitimate fog node in an insider attack, whereby a legitimate fog administrator knowingly or mistakenly introduces a rouge fog node instead of a legitimate fog node. Malicious node or rogue node will try to produce dissimilar attacks to get a trust value by familiarizing the system with its attributes. In this case, rogue fog nodes have the capacity of accessing end-users' cloud stored, encrypted data or encrypted data stored in other fog nodes.

2.3.3 Security Oriented Approaches to Protect Data in Fog Computing

This section investigated three areas: data security and access control in fog computing, rogue fog nodes and secure communication, and data exchange among fog nodes.

2.3.3.1 Security and Access Control In Fog Computing

The research community has presented different schemes such as Wang et al. [43], Wang et al. [44] and Goyal et al. [25] through the adoption of Attribute-Based Encryption schemes in the cloud computing storage and processing environment. Cloud storage means that security can become a challenge through external and internal threats. Yu et al. [45] proposed attribute-based data sharing with attribute revocation. Attribute revocation is about the dropping of some attributes by the user, an action that limits access to the system [46]. It was demonstrated to be resilient to selected plaintext attacks founded on the Decisional Bilinear Diffie-Hellman (DBDH) assumption [47]. A drawback of this revolved around the element of key generation, encryption, and decryption computation processes included in all likely attributes in the universe of attributes. As a result, this scheme had a computation cost liability on users.

Li et al. [48] proposed a new data sharing scheme, attribute-based, in a bid to sort

out the challenge and possibility of ineffective data security. They would accomplish it by changing part of the system into offline status. Therefore, before decryption begins, the development and inputting of a public ciphertext would be carried out in a bid to reduce high computation overhead. Offline ciphertexts shield the ciphertext produced when the system goes online.

Tysowski et al. [49] recommended a scheme that would work by combining cryptographic primitives, namely CP-ABE and proxy re-encryption, to achieve a user revocation process. They separated users into groups; each group of users shared a similar secret key. However, the scheme was made vulnerable to a collusion attack by making all users in the same group share and use the same private key.

ABE cryptographic schemes have been applied by researchers in cloud computing to enhance security and enable fine-grained data access control of outsourced data. Like every system, the ABE scheme also has its downsides. High computation overheads is one of the issues , which burdens the inadequate resources accessible to end-user devices as encryption and decryption process occur. A scheme proposed by Yu et al. [50] would ensure security as well as fine-grained data access control in cloud computing. This would work through the delegation of tasks to an untrusted proxy server. The process includes the combination of three elements; proxy re-encryption, lazy re-encryption, and KP-ABE in order to design a scheme that is characteristic of fine-grained data access control. All but one (dummy attribute) user private keys were stored in the cloud.

Another study conducted by Green et al. [21] provided a new prototype for ABE that would eliminate the involved overheads for ABE, such as time-related drawbacks or those related to size. This can be achieved through outsourcing of decryption into the cloud server. A single transformation key can enable any form of action about time and size. A blind secret key was issued to a user using a random choosen number that was hidden from

the user. The covered key with the server to execute a decryption that had been outsourced. Other researchers have come up with some schemes, such as Ning et al. [51], who introduced schemes with white-box traceability with the capability to trace malicious attackers. Ning et al. [52] introduced the use of large universe ciphertext-policy attribute-based encryption and the potential to trace users who maliciously sharing their private keys.

Additionally, Li et al. [53] adopted a system centered around CP-ABE to accomplish efficient revocation of user keys with the likelihood of subcontracting, to cloud servers, the encryption and decryption processes. Huang et al. [54] proposed a scheme allowing the sharing and accessing of data to accredited users, collectively. This scheme is based on the CP-ABE, and ABS in fog computing, whereby sensitive data received from the Internet of Things devices are encrypted with a handful of policies then released to servers in the cloud. The user who has attributes that content with the policies released can alter the decrypted data and re-outsource.

ABE systems have had the weakness of not having a hierarchical filing system based on ABE systems in cloud computing; therefore, having the limitation of disorganized files in the cloud. This gap led Wang et al. [55] to propose a scheme for files hierarchy access control based on attribute-based encryption. The layers access structures are incorporated into a one single access construct.

Many schemes, therefore, have been proposed to enhance the operability of ABE systems. Kaaniche et al. [56], saw the need to have privacy-preserving mechanisms and, therefore, proposed an encryption-based multilevel access policies scheme using ABE systems to enhance selective access to encrypted data based on users' granted access rights. An owner of encrypted data has the power to determine who will access that data through the use of privacy-preserving mechanisms. This model is highly applicable to cloud computing. However, it does not put into consideration the aspect of sharing data on fog nodes that

service the end-users.

2.3.3.2 Security and Access Control In Fog Computing

Security and privacy issues are expected in Fog Computing, some of which have been existed in cloud computing platforms while others are new [42]. When exploiting fog nodes in developed systems, it is crucial to consider the threats to end users' data security and privacy and secure it. When talking about security concerns in this research, we mean access control, rogue fog nodes, and trust issues among fog nodes and between fog nodes and cloud service provider/end-users.

Researchers have applied ABE to devices from IoT to resolve issues of data access control. The FDAC scheme has been proposed by Yu et al. [16] to attain fine-grained data access control and the issues that arise in wireless server networks. In the FDAC scheme proposed, each sensor node is assigned a specific set of attributes, while each user was allocated an access structure that contained access rights for that particular user. Similarly, another method was proposed by Hu et al. [57] whereby users with wearable devices were studied to assess the aspect of communication security that existed between the users and the device, in a wireless body area network. A safe and secure scheme was implemented through the influence of CP-ABE in WBAN. Besides, Jiang et al. [58] pointed out the issue of key-delegation abuse in the ABE system and specifically a CP-ABE scheme. He, therefore, came up with a CP-ABE scheme that would trace traitors and misuse of delegating access keys in fog computing.

Another access control scheme was proposed in 2018 by Zuo et al. [59]. This scheme named CCA-secure ABE scheme to ensure the security of outsourced decryption to ensure that the data is secure. Regrettably, this scheme came with high computational costs, and only performed with the integration of AND-gate encryption attributes. Xiao et al.

[60] also recommended another hybrid fine-grained search and admission authorization for fog computing, which would be built on a CP-ABE cryptographic primitive. This scheme reinforced both index encryption and data encryption making it hybrid. It also reinforced AND-gates bases. Consequently, Mao et al. [61] proposed a generic anonymous identity-based encryption with ciphertext security to ensure confidentiality and anonymity associated with ciphertext attacks.

Recently, researchers have started adopting ABE algorithms into fog computing to resolve issues involving data confidentiality and fine-grained access control. Li at el. [62] proposed a time-domain multi-authority outsourcing attribute-based encryption scheme for acquiring data in edge computing. This will boost data security and users' privacy in edge computing. Edge computing moves computing and storage devices near the user to save on time and cost. In this context, encryption attributes have been divided into two parts: time attribute and universal attributes. Time-domain attributes were combined with the encryption algorithm. This scheme, therefore, works by outsourcing computation to edge nodes for increased security and operability.

Xu at el. [63] proposed PMADC-ABSC scheme that controls data access founded on Ciphertext policy ABSC to offer a fine-grained control mechanism as well as a multi-authority data access control scheme for fog computing. The recommended structure based on Multi-authority Ciphertext-Policy Attribute-Based Signcryption (ABSC) supports the outsourced processing for data users and owners. In this structure, parameters are managed by a central and trusted authority. The drawbacks in this scheme are the high levels of computation and communication overhead.

Dsouza et al. [64] proposed policy management of resources needed for fog computing. This scheme would ensure the security of collaboration activities and enable interoperability among different users in fog computing. This framework comprises of a set of rules, attributes

repository, and an administrator that work at different levels to enforce capability policies.

A Fog Computing-based Security (FOCUS) is a system proposed by Alharbi et al. [65] to prevent Malware attacks on the Internet of Things (IoT) devices. They would use a virtual private network (VPN) to safeguard communications channels to IoT devices. They also protect against DDoS attacks through the use of a challenge-response authentication technique. Traffic analysis and firewall were also placed into consideration as IoT devices validate each other using a challenge and response method. Traffic analysis through FOCUS can identify malicious or suspicious activities and sources as they are labeled. The source has to pass the set test to attain access to the VPN.

Imine et al. [66], proposed the implementation of new authentication protocols because traditional authentication protocols had issues related to latency and lack of mutual identification. These new authentication schemes would mutually identify users through information inputted concerning that user and the identifiable attributes. A fog node usually upholds a blockchain and enables users to validate any fog node. Fog nodes are also able to authenticate each other.

Research community has developed distinct systems to deal with challenges to allow different user's access sub-parts of the encrypted data according to their access policies in cloud computing.

Similarly to cloud computing, fog computing stores data for users in fog nodes. In Alotaibi et al. [67], an attribute-based data sharing scheme was proposed in relation to fog computing. Data sharing has its challenges, and these would be addressed using ABE systems and proxy re-encryption. Through this scheme, the owners of data can encrypt and provide their access policies of the users who have the right to access the encrypted data. Specifically, a Key-Policy Attribute-Based encryption and proxy re-encryption mechanisms were considered by Alotaibi et al. [67], so that fog nodes will carry out re-encryption processes

without revealing the data. Data users whose attributes fulfill the access policies can access the encoded data and provide security in communication and data sharing.

Zhang et al. [68] proposed an access control Ciphertext-Policy Attribute-Based encryption scheme in fog computing. This scheme would promote outsourcing capacities as well as attribute update for fog computing. This would involve a redefinition of the access structure to alter the attributes as well as generate ciphertext partially. Besides, it enabled data owners to delegate part of the ciphertext generation into the fog nodes. The reason is to handle the issue of time that is usually needed to encrypt, decrypt as well as updating the attribute is very long. The adoption of this scheme would reduce time and make it to be constant.

Fog computing is likely to be faced by authentication issues and authorization challenges that might arise once in a while. Stojmenovic et al. [69] provided an understanding of authorization and authentication issues in fog devices and between fog and cloud through the ABE algorithm. They provided an argument that end-users would still be authenticated and authorized into fog devices even when there was proof of vulnerability of the connection. The study also assesses different forms of attacks such as typical attacks and man in the middle attack. Similarly, Li et al. [70] proposed a framework for the collection of smart devices' attributes and incorporating them with the ABE algorithm to validate access authority in real-time. Existing schemes are assessed and compared with the new schemes to assess vulnerabilities that could result. Mollah et al. [71] proposed a lightweight cryptographic model to offer a mechanism for control of access sharing of data. It has all security operations unloaded to nearby fog servers. A search scheme proposed would enable a user to search data either in their storage or on shared storage systems securely without an interception.

Mohammed et al. in [72] proposed a lightweight revocable hierarchical ABE scheme. This is a scheme that embraces Attribute-Based Encryption for IoT that can benefit from

the cloud systems as the cloud will carry out on its behalf the most expensive operations (e.g., encryption and decryption operations) without prior knowledge of the features of the underlying data. This scheme also employs the hierarchical model to enhance flexibility such that attributes can be handled using different authorities.

Blockchain is a widely used method for enhancing the privacy of systems and reducing tampering. This scheme was proposed by Guo et al. [73] in an attempt to authenticate the outsourced decryption of attribute-based encryption (ABE) through the inclusion of blockchain into fog computing. Blockchain help to guarantee the validity of outsourced data to limit invasion. An additional authentication layer in decryption operations would verify the precision of the ciphertext by the data users. Blockchain technology keeps data that are used for verification. That would assist in inhibiting inside or outside attackers from meddling with the available data.

When enabling data sharing among multiple actors, it is imperative to consider that one of those actors become rogue/malicious. As end-users outsource confidential data to fog nodes then to the cloud, if one of those fog nodes goes rogue or were maliciously compromised, the end-users' confidential data would be at risk from being maliciously breached. Therefore, this paper focuses on reducing the amount of data rogue fog nodes can breach in fog computing, in the case of a federation of fog nodes.

Rogue fog nodes can pretend being legitimate, to be trusted, where they strike once the system has trusted them. They can be of different forms, such as an access point (AP), fog node, or Internet of Things (IoT), according to Alrawais et al. [74]. Our work focuses, therefore, on the minimization of data that is in the danger of breach by rogue fog node. After scanning the literature, we found that rogue fog node has not been studied in the context of fog computing.

An interesting perspective that would assess authentication protocols was proposed

by Ibrahim [75], whereby fog users and fog nodes would need to authenticate each other mutually at the edge of the network. This scheme would have a random user roam the server. Therefore, fog nodes will be required to keep credential information on attributes for all users so that they can identify each other with the attributes stored. This scheme did not put into consideration the aspect of a selfish fog node that becomes rogue and even attacks other good fog nodes, an element that our study has considered.

Technological advancements require us to be innovative to ensure that problems that arise have solutions developed. The application of ABE systems in the real world has seen innovations in the transport systems. The use of VANETs would enhance communication between vehicles and also communication between the vehicle and infrastructure. This information was provided by Al-Otaibi et al. [76]. They targeted developing a scheme for the detection of vehicular rogue fog nodes with the fog computing paradigm. This scheme would limit traffic data exchange among vehicles, and at the same time allowing roadside communication through roadside units (RSU). Through this important scheme, roadside units also capable of identifying vehicles that provide false data to enhance the privacy of a vehicle.

Another author in [77] also came up with a similar proposal like that of Al-Otaibi described above. He raised the concern of security of VANETs whereby some malicious parties may violate the privacy of a user. To counter this, he proposed a threshold anonymous announcement model to detect rogue users by enabling the receiver to check the information provided by the sender (other vehicles) and choose to either accept or reject the projected information. The role of this scheme is to meet the goals of dependability, privacy, and audibility.

A rogue fog node imitates the genuine to demand the sharing information from other fog nodes, end-users devices, or the cloud. The primary, and hidden, intent is to violate owners' data security and privacy and retrieve important information that could jeopardize

the security of the user. For example, the use of debit cards can be maliciously attacked, and information such as Personal Identification numbers intercepted. Having control of compromised a fog node, an attacker can infringe the privacy of end-users' data as it transits the fog nodes to the cloud. This issue has been widely researched with almost 2,500 journals and scholarly articles on the same.

Man in the Middle attacks are one of the most frequent attacks in the security of computers. A middle malicious party intercepts communication between users or user platforms [78]. Since fog is known for its close proximity to the end-user, they are bound to have frequent and are more susceptible to attacks than in cloud computing. Stojmenovic et al. [19] suggested a scheme to show the probability of a man-in-the-middle attack in which the gateway is either being compromised or replaced by a fake one. The threat of Man in the Middle attacks has been widespread because of the spread of WLAN networks in public places. This scheme led to the proposal by Han et al., [79] on the utilization of a measurement-based scheme to aid users to stop connections to fake access points. This system identifies the rogue AP by computing the time taken between end-users and the DNS server.

To create an efficient and effective fog computing environment, it is essential to ensure the security of the devices used in transmission. The idea of a cyber-security framework was proposed by Sohal et al. [80], a framework that works by detecting and recognizing malicious control devices at the edge of the network, popularly known as outside device attacks. This identification is critical in the securing of fog computing platforms. Therefore, they proposed the inclusion and use of three frameworks that would work effectively if applied concurrently and in the real environment. These include the Hidden Markov model (HMM), Intrusion Detection Systems (IDS), and Virtual Honeypot technologies for their framework [80]. The HMM would be used to recognize and classify edge components. At the same time, the

Virtual Honeypot technologies would work by keeping and sustaining the log comprising all earlier detected malicious components. This scheme is important as it helps the system develop a self-defense mechanism from known and unknown malicious attacks. Finally, the IDS is the genesis of identifying suspicious events and ensuring that they are malicious. However, they can sometimes produce false positives, which is why other frameworks, as mentioned above, have been applied.

One of the most significant problems that bother network administrators is the presence or possibility of rogue access points, as pointed out by Shivaraj et al. in [81]. They thereby proposed a methodology known as the Hidden Markov Model (HMM) to detect the presence of rogue AP in the WLAN. This approach is applied passively in two stages. The first stage involves training HMM and another, which is detection based on the trained HMM. Data is usually attained through routers, whereby assessment is done on packet arrival times to determine any dissimilarities. This method proposed in [81] is very efficient in that rogue access points can be realized within seconds.

Insider data thefts can result in leakage of confidential data, and therefore defense mechanisms need to be adopted as control is of such attacks is almost impossible [82]. Stolfoet et al. in [83] developed an approach that is able to detect and handle insider data theft attacks in the cloud. They concentrated on user performance profiling and decoy technology to monitor user data access patterns into the system. Therefore, if a user with malicious intent were sensed by the technology, which operates by profiling user behavior, it would then send the snare document to that user.

2.4 Recommendations

The critical question that no researcher has explored is how to mitigate the effects of rogue fog nodes on encrypted data stored in the cloud storage or fog nodes. This research

tries to address the study gap about security issues related to rogue fog nodes, as previous literature has not discussed in the environment of fog computing. Therefore, the absence and inconclusiveness of research about security issues facing fog nodes in previous studies renders the current literature incomplete and limited. This omission must be managed the context of fog computing.

However, this leads us to the primary aim of this research. It investigates all issues facing fog computing paradigm and, consequently, recommends approaches that can minimize the impact of rogue fog on the data security of end-user transfers of information between fog nodes and the cloud Service Providers (CSPs). The main focus of this research is to assess fog nodes and cloud storage platforms and the approaches to ensure the data security of end-users.

2.5 Conclusion

In this chapter, we reviewed the background of the cryptographic primitives and technologies exploited in this research. This chapter included the application of ABE systems through the adoption of Ciphertext ABE systems. Besides, we reviewed the blockchain technology as we exploited the idea of smart contracts in our work. The background of fog computing was reviewed while considering the services it offers and associated security and privacy issues. Then, a discussion of security issues of fog computing was carried out. Finally, a survey of the literature of the Attribute-Based Encryption (ABE) schemes developed by researchers to provide data security issues in the cloud and fog computing paradigms was undertaken. Several considerations need to be made before selecting the mitigating methods for rogue fog nodes, which are majorly related to the specific privacy or security issue.

3 An Encryption-Based Approach

In this chapter, we developed an Encryption-Based approach to protect fog federations from rogue fog nodes. Sections that are shown in this chapter has been published in [84] verbatim.

3.1 Introduction

The cloud computing paradigm, over the past few years, has gained a well-respected reputation due to the tremendous on-demand services it provides to end users over the internet. Cloud computing has innovative features— availability, scalability, and economy— that help to meet the huge demand for storage and computation resources [6]. End users can remotely store and process their data in the core network on the cloud, but it is stored far from users; therefore, the time between a user request and the cloud response is high. Thus, there is a demand for new technology to resolve the cloud latency issue [9][15]. With huge growth in the use of the cloud and outsourcing data to the cloud, cloud computing can expose users to various challenges such as data access control, data security, and high latency. However, the fog computing paradigm emerged in 2012 [10] to reduce latency between the cloud and end users' devices [9][15]. The fog computing paradigm locates a cloud close to the end-users' devices, which enables customers to perform computing at the edge of the network [10]. In the context of fog computing, fog nodes provide resource services to end-users [16]. The main distinguishing features of fog computing include mobility, location-awareness, and low latency [18][42]. The research community has examined the topic of access control in cloud computing, but that is not the focus of this chapter. For a comprehensive overview of access control, read [85][50][86]. The fog computing paradigm is susceptible to different

threats ranging from malicious attacks to technical issues. Therefore, our focus of this chapter is to demonstrate ways to oust rogue (malicious) fog nodes that present threats to fog computing. One reason for these vulnerabilities is that end-users outsource sensitive data to a nearby fog node to be processed and then sent to the cloud. Since these fog nodes are the connector between end-users' devices and the cloud, it is challenging to secure the cloud and other fog nodes from malicious attacks. This reveals a need to protect the owner's sensitive data from being compromised. In general, fog computing threats take two forms: 1) Data modification: if the adversary gets ahold of the fog node, he/she might violate the user's data confidentiality/integrity. Thus, it is important to introduce a security technique to check data confidentiality/integrity among fog nodes as well as between fog nodes and the cloud. 2) Unauthorized access: when a fog node is compromised, an adversary can get his/her hands on users' sensitive data. Therefore, it is essential to define a security approach to revoke(oust) the fog node as soon as it goes rouge for whatever reason. To defend users' data security while continuing the low latency that fog computing provides, there is a fundamental demand for a security approach that will meet the users' security needs. Attribute-based encryption (ABE) is a cryptographic primitive that enables one-to-many encryption. It consists of two types: ciphertext policy attribute-based encryption (CP-ABE) [24] and key policy attribute-based encryption [25]. CP-ABE, introduced by Bethencourt at al [24], is one of the most cryptographic approaches, providing fine-grained data access control. In this chapter, we present a CP-ABE based approach to oust (rogue) malicious fog nodes and minimize the communication overhead between the cloud service provider (CSP) and fog nodes (FNs).

Contribution This chapter presents a secure and fine-grained data access control approach, based on CP-ABE algorithm[24], which achieves the following features:

- Classifying the fog nodes (FNs) into different fog federations(FF) depending on FNs' attributes —namely, location and provided services.

- Ousting a rogue (malicious) fog node (FN) that acts maliciously, so it cannot access the encrypted data in cloud.

- It directly prevents the departure FN from accessing encrypted data in the cloud or in other FNs in same FF, besides it helps remaining FNs to access re-encrypted data with no latency.

- It helps in reducing communication time to transfer files from the CSP to FNs. The FN divides the file into multiple blocks. The CSP tracks what each fog node currently has/needs. After the re-encryption process, the CSP sends the needed block to the legitimate FNs. This feature makes our approach time efficient.

3.2 Classifying Fog Nodes into Fog Federation

The Encryption-Based Approach was based on classifying the Fog-Nodes (FNs) into Fog-Federations (FF) based on their locations, and services they provide to end-users, Fig. 3.1. A FN can belong to multiple FFs based on its attributes, therefore it can access the data encrypted to the FFs' attributes where it belongs. For example, Fig. 3.1 shows that FN_1 belongs to FF_1 and FF_2. In this case, FN_1 is capable to access data encrypted to both FFs. The benefit of FF here is to manage the FNs accessibility to encrypted data stored in the Cloud-Service-Provide (CSP). Fog computing was proposed to deal with the time latency issue between end-devices and the far cloud. Therefore, researchers have seen bringing the computing and processing close to end-devices would be helpful. However, what if the fog node serving a certain region is down or goes rogue. This means end-users have to get back to the far cloud to outsource or retrieve their data. In another word, it

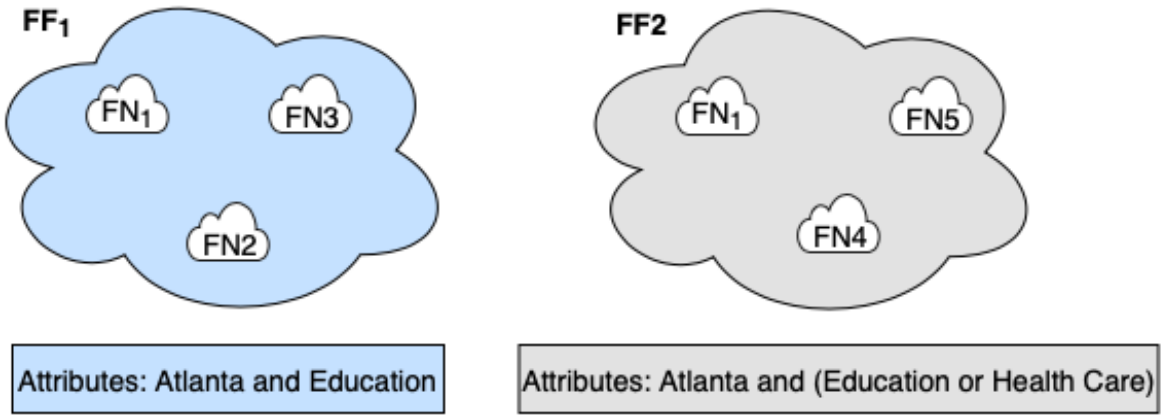


Figure 3.1: Fog Federation Concept.

means the computing and processing would move back to the far cloud. In this approach, we strongly believe that the FF concept would help in keeping the computing and processing services close to the ground with defeating the rogue fog nodes from accessing the encrypted data stored on the cloud or in other FN in the same FF. However, FF concept is a great contribution to FNs management.

3.3 An Encryption-Based Approach Objectives

In this section, we present a description of the objectives of Encryption-Based Approach in aspects to confidentiality, integrity, availability, and performance.

3.3.1 Confidentiality

Encryption-Based Approach provides and guarantees data confidentiality as follow. The encrypted files/blocks in the Cloud-Service-Provider (CSP) or Fog-Nodes (FNs) are protected against ousted fog nodes by the following process: when updating the Fog-Federation-List (FFL), the TAS updates the System Public Key (SPK) and propagates it through a

secure channel to fog nodes that are not yet members of the FF. In the meantime, it sends notifications to the CSP to re-encrypt the files encrypted with a specific FFs attributes, and it notifies the fog nodes remaining in the FF to update their secret keys. In this case, the ousted fog node does not have the SPK and, therefore, cannot update its private key for a future decryption process. This feature makes this approach robust against unauthorized fog nodes. In addition, when a FN receives a request from another FN in the same FF, it verifies the request using the digital signature [87]. Once the remaining FNs and the CSP receive the updated SPK, the ousted FN (rogue/malicious FN) becomes unable to forge the signature to retrieve encrypted files/blocks from either the CSP or FNs. In this senario, the end-users' data confidentiality is preserved against rogue FNs.

3.3.2 Integrity

Encryption-Based Approach provides data integrity by hashing [88] each block of the files to be encrypted. Then, it outsources these blocks along with their hashes into the cloud service provider (CSP) for storage. Encryption-Based Approach is eventually capable of verifying whether a potential damaged block is corrupted or not. If a downloaded block is corrupted, Encryption-Based Approach can report the corrupted block to the CSP to mark it as damaged block. This approach allows the FNs in same FF of checking the block/blocks integrity by calculating the decrypted blocks hashes, if matched, it is verified; otherwise it is not.

3.3.3 Availability

Encryption-Based Approach provides data availability for end-users' by classifying Fog Nodes (FNs) into Fog Federations (FF) based on their attributes. Fog computing emerges to solve the time latency issues between end-users and the cloud service provider.

However, if the FN providing services to end-devices in its region goes rogue, or is being down or maliciously compromised, the fog computing roles will be dysfunction in this case. In particular, end-users encrypted data retrieving must be from the far cloud. One great advantages of the Encryption-Based Approach is to let fog nodes (FNs) in same Fog Federation (FF) to serve users in same region. In this scenario, the FN stores the retrieved files for a period of time in order to make it closely available to end-users and other FNs ion same FF. On the other hand, FNs must flush their storage in a specific time in sake of maintaining data security as FNs might unexpectedly go rogues.

3.3.4 Performance

One of the Encryption-Based Approach objectives is to reduce the communication overhead between the CSP and the FNs. The CSP and FNs exchange encrypted messages. The Encryption-Based Approach divides the message into blocks before encrypting and uploading the blocks to the CSP. At the same time, the CSP maintains a progress list which helps to track the needs of all FNs in each FF. When a FN in a specific FF needs to decrypt a file, the CSP will check the progress list in case some of the intended file's blocks have already been shared with another FN in the same FF. If that is the case, the CSP directs the requester to contact the FN in the same FF to get the needed blocks. This feature gives the Encryption-Based Approach more momentum as it plays a vital role in reducing time latency between the CSP and the FNs.

3.4 Overview of the Encryption-Based Approach

As Fig. 3.2 illustrates, the Encryption-Based Approach comprises of the following entities: a Cloud-Service-Provider (CSP), Fog-Nodes (FNs), and a Trusted-Authority-Server (TAS). Fog-Nodes (FNs) with same attributes form a Fog-Federation(FF). Time latency to

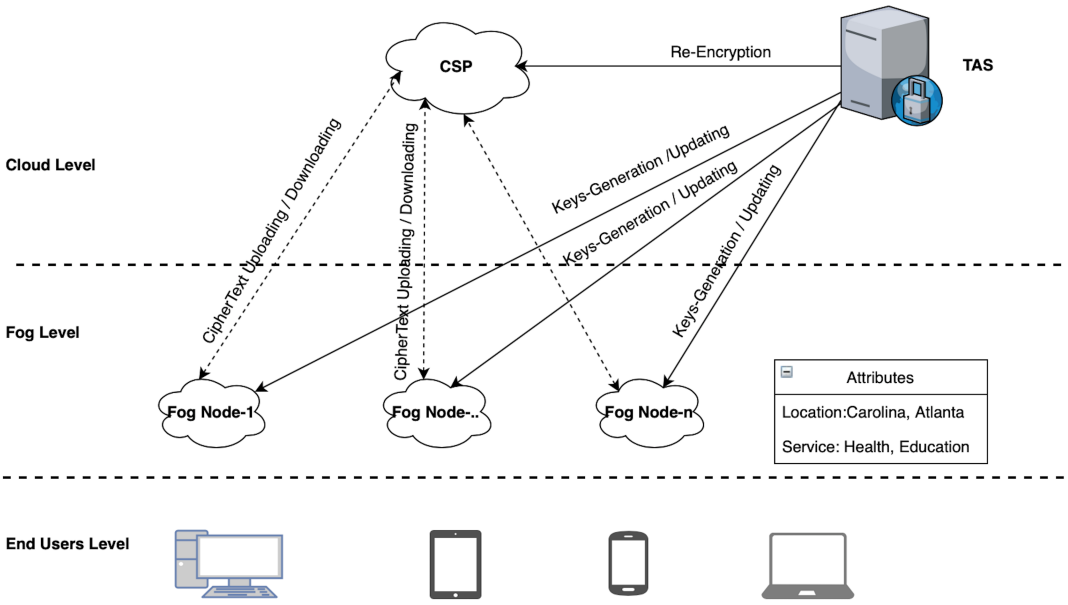


Figure 3.2: The Encryption-Based Approach System Model.

deliver data from cloud storage to end users' devices is high; therefore, fog computing emerged to address this issue. However, FNs are susceptible to malicious attack, and accordingly the end user's confidentiality could be violated.In order to maintain the confidentiality, integrity and availability for end-users' outsourced data traveling through FNs to CSP, we utilized the following advanced cryptographic primitives: CP-ABE algorithm, which enforces fine-grained data access control to guarantee a secure communication between FNs and the CSP, and hash function and digital signature concepts to guarantee data integrity. This approach equips each FN with a set of attributes and associates each ciphertext with an access policy that is defined by these attributes. Only a FN that satisfies the access policy structure

Table 3.1: Progress List

FN_{ID}	$File_{ID}$	Available-Blocks	Needed-Blocks
FN_1	$File_1$	B_1	B_2,B_3,B_4,B_5
FN_2	$File_1$	B_1,B_2,B_3	B_4,B_5

associated with the ciphertext can decrypt the ciphertext to obtain the plaintext.

This approach classifies the FNs into different fog federations (FFs) based on a set of attributes which characterizes the FF. Consequently, we let the FN divide the file to be encrypted into blocks, then perform the hashing, and encryption processes, and finally outsource it into the cloud storage. More specifically, CSP maintains a progress list, which is a list of all FNs in a specific FF besides blocks of files they currently need/have, Table. 3.1. Such a construction reduces the file retrieving and decryption time. For the sake of clarity, we will consider the following example:

Suppose File_1 is encrypted and uploaded to the CSP. FN_1 and FN_2 already have accessed some blocks of this file. Suppose FN_2 becomes unauthorized to decrypt this file for any reason whatsoever. The TAS simultaneously notifies the remaining FNs to update their private keys and notifies the CSP to perform a re-encryption operation. The CSP then defers re-encrypting blocks already shared with any FN in the FF. It instead starts by re-encrypting the blocks that have not been accessed yet and then re-encrypts the ones that were already shared. This feature boosted this approach's success as it is designed to reduce the time latency when retrieving files from the cloud. When an authorized FN accessed an encrypted file, the CSP checked the progress list and accordingly returned the file or the blocks that had not been accessed by others in the same FF. Then, the CSP returned a list of FNs and which blocks they already had. Then, the FN, which requires the encrypted file, communicates the FN in the same federation to get the intended blocks. Thus, the time consumption to retrieve the encrypted data from FNs in same FF was much less than retrieving the same file from CSP , so this approach provides low latency and less traffic overhead between the FNs and the CSP. This is attributed to the idea of fog federations (FF).

A FN could leave the system for whatever reason, or it could go rogue by being

compromised by a malicious attack. Thus, to guarantee the owners' data confidentiality there was a need to take precautions against the rogue FN. This approach efficiently acts as a barrier against an ousted fog node to keep it from accessing the encrypted data stored in the cloud. This approach was designed to instantaneously and simultaneously prevent the ousted FN from accessing the encrypted data and delivering the re-encrypted data to the remaining FNs. Fog Federation-updating, and re-encryption algorithms were executed by TAS and CSP in a parallel manner: TAS excluded the ousted FN from the Fog-Federation-List (FFL). Then, it updated the current *Fog-Federation-Public-Key* ($FFPK_{ver}$) to $FFPK_{ver+1}$, then propagated $FFPK_{ver+1}$ to the FNs remaining in the FFL. CSP started re-encrypting the files which could not be decrypted by the ousted FN. The re-encryption key was $FFPK_{ver+1}$.

In addition, this approach was designed to maintain data integrity which protects encrypted block/blocks from being damaged while in transit or in storage. This was achieved by calculating the hash of the decrypted blocks when retrieving encrypted data either from the CSP or other FN.

3.5 Threat Model

The Encryption-Based Approach presents three security models that reflect the following cases:

3.5.1 Rogue Fog Nodes

The fog node was called rogue when it seems acting appropriately, but it was acting maliciously. When it comes to real application, rogue fog nodes would badly affect the computing ecosystems in different ways. For example, suppose we have a Fog-Federation (FF) of Fog-Nodes (FNs) that share same attributes. Accordingly, FNs in the same FF can access the same data encrypted to the FF's attributes. The question is: what if one

of those FNs in same FF goes rogue, this would lead to breaching the confidential data that must be kept secret, is not it? The idea of FF is centring about minimizing the time latency between the Cloud-Service-Provider (CSP) and the end-users, but with considering the rogue fog nodes issues. In case a rogue fog node manages to continue retrieving the encrypted files/blocks from the CSP and from other FNs in same FF, this would lead to a major violation to the end-users' data confidentiality and integrity.

3.5.2 Man-in-the-Middle (MITM) Attack

A man-in-the-middle (MITM) attack occurs when an attacker intentionally intercept the traffic between two parties to modify the communication between them. In the Encryption-Based Approach, Fog-Nodes (FNs) in same Fog-Federation (FF) exchange confidential data among them and with the Cloud-Service-Provider (CSP). Therefore, this approach aims to protect end-users' confidential data from being damaged or exposed by the MITM attack.

3.5.3 Curious Cloud Service Provider

The Encryption-Based Approach intends to secure end-users confidential data against the rogue FNs and the CSP as well. In this approach, we consider the Cloud-Service-Provider (CSP) partially trusted as it could violate end-users' data security by handing it to unwanted third party. This is because the CSP was run by companies which not fully trusted to end-users. The aim of this approach is to prevent the CSP from inferring the stored encrypted data.

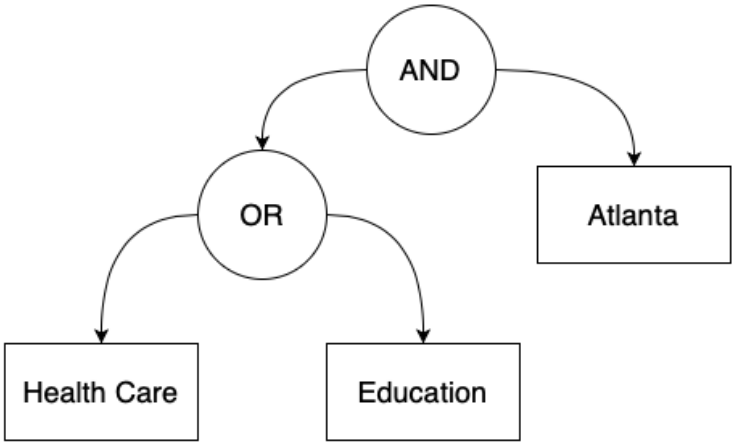


Figure 3.3: Encryption-Based Approach Access Structure.

3.6 Design of the Encryption-Based Approach Algorithms

In this section, we present the description of the Encryption-Based Approach based on the CP-ABE algorithm, hash function and digital signature concepts. We begin by reviewing the access tree model presented in [24][25] as we exploit it as an access structure as shown in Fig. 3.3. Then, we describe our algorithms. To satisfy the access tree conditions, we follow the same methods in [24][25]. Read through [24][25] for more information. Next we provide the algorithms needed for the Encryption-Based Approach.

Algorithm 3.1 which based on [24] is executed by TAS. It inputs a security parameter and outputs System Public Key (SPK) along with System Master Key (SMK). It constructs a universal set of attributes to be used in our approach as USAT = FN Location, Service Provided.

TAS also executes Algorithm 3.2. Its input is SPK. It contains two parts. First: it creates a new Fog-Federation-List (FFL) of FNs depending on the provided location and services. It contains hash of FN_{ID} and FF_{ID} to prevent against id-impersonation. Second: when TAS receives a request from FN to join the system, TAS will check a predefined FFL. If FN-Attributes (FN_{Att}) are suitable with one or multiple fog federations attributes (FF_{Att}),

Algorithm 3.1 Setup (λ)

1: Choose bilinear cyclic group G_1 of prime order p with generator g_1 ;
2: Randomly generates exponents $\alpha, \beta \in G_p^*$;
3: Random Oracle H: $USAT \rightarrow \{0,1\}^*$ // to map each att in USAT into random element $\in G_1$;
4: SMK and SPK as:
5: $SMK = \{\beta, g_1^{\alpha}\}$
6: $SPK = \{G_1, g_1, g_1^{\beta}, e(g_1, g_1)^{\alpha}\}$;
7: USAT = FN location, Services
8: The SPK is public to all fog nodes, but SMK is private for TAS

it will be added to the corresponding Fog Federations. Otherwise, it will create new FF and add the new FN to it. Additionally, this algorithm outputs fog federation master key (FFMK), which is only available for TAS, and fog federation public key (FFPK) that is public to the federation's members.

Algorithm 3.3 is also executed by TAS which is based on [24]. It takes as inputs FFMK, SPK, FN_{ID}, SMK and a set of attributes S. In this algorithm, the set S along with the F_{ID} identify the FN's private key ($FN_{PR-S-ID}$). FN_{ID} is used here to assign each fog node with a distinct private key and to track each FN's activities. It outputs $FN_{PR-S-ID}$. The $FN_{PR-S-ID}$ is associated with FN attributes and FN_{ID} simultaneously, and this helps in preventing an unauthorized fog node from accessing data stored in the CSP.

Authorized FNs executes Algorithm 3.4 which is based on [24]. This algorithm takes the file to be encrypted (M), the access tree T, SPK, and $FFPK$ as input. It outputs the ciphertext to be uploaded to the CSP. In our approach,the FN divides the file to be encrypted into multiple blocks ($M_1, M_2, \ldots, M_n$). Then, it encrypts all blocks and uploads the ciphertext to the CSP.

CSP executes Algorithm 3.5. Our approach is designed to reduce the communication overhead between the CSP and the FNs. CSP maintains a progress list as shown in Table-1. CSP will check the progress list upon receiving a request from an authorized fog node. In

Algorithm 3.2 FF Creation and Adding new FN (SPK)

Part 1: FFL Creation

Create FFL to contain list of FNs in the system,as
$FFL[FF_{ID}|FFPK_{ver}|FF_{att}|H(FN_{ID}FF_{ID})]_{ver}$

Part 2: Addign New FN to FF

if $FN_{att} = FF_{att}$ **then**
  assign new-FN $FN_{ID} = FN_{ID} + FF_{ID}$
  add FN to the corresponding FF
  send $FFPK_{ver}$ to new FN
else
  create new FF and assign it unique FF_{ID}
  randomly choose $FFMK_{ver} = \gamma \in Z_p^*$
  Compute $FFPK_{ver} = \{FF_{ID}, g_1^{\gamma}, e(g_1, g_1)^{\gamma}\}$
  assign $FN_{ID} = FN_{ID} + FF_{ID}$
  add new FF to FFL
  add new FN to new FF
  send $FFPK_{ver}$ to the new FN
end if

Algorithm 3.3 FN_{PR} Generation $(FFMK_{ver}, SPK_{ver}, SMK, S, FN_{ID})$

TAS gets request from new FN to join the system
if FN_{ID} is verified **then**
  continue
else
  decline
end if
randomly generate $r \in Z_P^*$
for every $att_i \in S$ **do**
  randomly generate $r_i \in Z_P^*$
end for
for *every* $att \in S$ **do**
  compute FN_{PR-S} as follows
$FN_{PR-S-ID} = (D = g^{(\alpha+r)/\beta}, \forall i \in S : D_i = g^r.H(FN_{ID})^{\gamma_{ver}}.H(i)^{r_i},$
$D_i{}' = g^{r_i}.H(FN_{ID})^{\gamma_{ver}})$
end for
send $FN_{PR-S-ID}$ to the corresponding FF and update FFL

case the requested file or some blocks of it has/have already been accessed by other FNs in same FF, the CSP will notify the requester to communicate which FNs have those blocks. But in case the file has not been accessed by any FN in the same FF, the CSP will send the file to the FN.

Each authorized FN executes Algorithm 3.6 which is based on [24] to obtain plaintext. Its inputs are the current ciphertext, the current version of the $\text{FN}_{PR-S-ID}$, the defined attributes in the access structure. The decryption algorithm outputs either plain-text or $\perp$. In (1), we associate the fog node private key with FN_{ID} and FN_{att} set to prevent a fog node's impersonation. The FN performs decryption for each block and then combines all the decrypted blocks to form the plaintext file.

TAS and CSP simultaneously execute Algorithms 3.7 and 3.8. TAS would oust a FN from the system for multiple reasons (e.g. being attacked or becomes a rogue FN). Thus, to guarantee the owner's data security, there is a need to take precautions against those ousted FNs. Our approach efficiently prevents an ousted FN from accessing the encrypted files stored in the CSP. To prevent an unauthorized FN from accessing the encrypted files, Ousting rogue FN and Ciphertext Re-encryption algorithms are executed simultaneously by TAS and CSP, respectively. More specifically, such feature ensures that our approach is secure. TAS will exclude the ousted FN from the FFL. Then, it will update the current FFPK_{ver} to FFPK_{ver+1} and propagate FFPK_{ver+1} to the FNs remaining in the FFL. CSP will start re-encrypting the files, which cannot be decrypted by the ousted FN. The re-encryption key is FFPK_{ver+1}. The CSP starts re-encrypting the files/blocks that have not been accessed by any FN in the given FF. This feature makes our approach beneficial for data owners in terms of data confidentiality.

Algorithm 3.9 is executed by the legitimate FNs remaining in the FF after executing Algorithms 3.7 and 3.8. The TAS sends the new FNPK_{ver+1} and a random large number m

Algorithm 3.4 Data Encryption (M, T, SPK, $FFPK$)

A is set of atts represented by monotone access structure tree T
for each node x in T **do**
  set a polynomial
  if node x is root node in T **then**
    set $q_{(0)R}$ = s // s is random value $\in Z_P^*$
  else
    $q(0)_x = q_{parent(x)}(index(x))$
    then choose $d_x = k_x$ - 1 as polynomial degree to interpolate with q_x
  end if
end for
specify n as number of blocks in each file
specify Block's size in certain file = M size/n
for each y $\in$ Y **do** //Y is the set of leaf nodes in T
  $C_y = g^{q_y(0)}$
  $C'_y = H(att(y))^{q_{y(0)}}$
end for
Compute: $C = g^{\beta s}$
Call Algorithm 3.11-Part 1
for every $M_i \in M$ **do**
  Compute $C_{i_{ver}} = M_i.e(g,g)^{\alpha s}$
  $C_{ver} += C_{i_{ver}}$
end for
Upload CipherText as $\{T, C_{ver}, C, C_y, C'_y, MD\}$ to the cloud

Algorithm 3.5 Progress List creation/ Requesting File from CSP

FN submits a request to CSP
CSP verifies FN, by checking the legitimate FNs list
if FN is verified and cipher-text access structure policy is satisfied **then**
  Check the progress list
  if available blocks $\in$ requested file blocks **then**
    Respond "list of FN_{IDs} that have copy of this file" to let the requester get the file from
  else
    send the requested file/blocks to FN
  end if
else
  decline and notify other FNs in same federation that this FN is not legitimate
end if

Algorithm 3.6 Data Decryption $(CT_{ver}, FN_{PK-ver}, x)$

1: **for** each leaf node attribute "x" in access structure T **do**
2: Set i= att(x)
3: **if** i ∈ S **then**
4: Recursively Compute

$$DecryptNode(CT_{ver}, FN_{PK-S-ver}, x) = \frac{e(D_i, C_x)}{e(D'_i, C'_x)} = \frac{e(g^r.H(FN_ID)^{\gamma_{ver}}.H(i)^{r_i}, g^{q(0)_y})}{e(g^{r_i}.H(FN_ID)^{\gamma_{ver}}, H(i)^{q_{(0)y}})} = \frac{e(g,g)^{rq_{(0)y}}, e(g, H(FN_{ID}).H(i))^{\gamma_{ver} r_i q_{(0)y}}}{e(g, H(FN_{ID}).H(i))^{\gamma_{ver} q_{(0)y} r_i}} = e(g,g)^{rq_{(0)y}} \quad (3.1)$$

5: **if** i ∈ S and x not leaf node **then**
6: **for** every child node z of x **do**
7: $F_z = DecryptNode(CT_{ver}, FN_{PK-S-ver}, x)$
8: **end for**
9: **if** $F_z! = \perp$ **then**
10: //by lagrange and polynomial interpolation as in [24]
$F_z = \prod_{z \in S_x} F_z^{\triangle_{i,S'_x}(0)} where i = index(z) and S'_z = index(z) : z \in S_x$
$= e(g,g)^{r.q_{(0)x}} = e(g,g)^{rs}$
11: **if** x =root **then**
12: $A = DecryptNode(CT_ver, FN_{PK-S-ver}, r) = e(g,g)^{rq_{(0)R}} = e(g,g)^{rs}$
13: **else** $DecryptNode(CT_{ver}, FN_{PK-S-ver}, x) = \perp$
14: **end if**
15: **end if**
16: **end if**
17: **end if**
18: Decrypt Message by Computing:
$M_i = \frac{C_{ver}}{e(C,D)/A} = \frac{M_i.e(g,g)^{\alpha s}}{e(g^{\beta s}, g^{(\alpha+r)/\beta})/e(g,g)^{rs}} = \frac{M.e(g,g)^{\alpha s}}{e(g,g)^{(\beta s\alpha+\beta sr)/\beta}/e(g,g)^{rs}}$
$= \frac{M_i.e(g,g)^{\alpha s}}{e(g,g)^{(s\alpha+sr)}/e(g,g)^{rs}} = \frac{M_i.e(g,g)^{\alpha s}}{e(g,g)^{s\alpha}}$
19: **for** M_i=1 to M_i=n **do** //n is the number of blocks in given file
20: Plain-Text+ = M_i
21: **end for**
22: Call Algorithm 3.11 Part 2

Algorithm 3.7 Ousting rogue FN ($FFMK_{ver}, FFPK_{ver}, FFL$)

1: update $FFMK_{ver} \rightarrow \gamma + 1 \in Z_p^*$
2: update $FFPK_{ver} \rightarrow FFPK_{ver+1} = FF_{ID}, g^{\gamma+1}, e(g,g)^{\gamma+1}$
3: remove ousted FN
4: **for** $FN_i \in$ updated FFL **do**
5: propagate $FFPK_{ver+1}$
6: **end for**=0

Algorithm 3.8 Re-Encryption ($CT_{ver}, FFPK_{ver+1}$)

1: Cloud service checks the progress list
2: Start re-encrypting blocks which have not been requested by any FN in same FF.
3: updated-file = $C_{ver}.e(FFPK_{ver+1}, C)$
$= M_i.e(g,g)^{\alpha s}.e(g^{\gamma_{ver+1}-\gamma_v er}, g^{\beta s})$
$= M_i.e(g,g)^{\alpha s+\gamma_{ver+1} s}$
4: CT_{ver+1} = updated-file

to the updated list of fog nodes remaining in the FF via a secure channel. Upon receiving this message, each FN in the FF updates its FNPK.

Algorithm 3.10 is executed by FNs. When a FN intends to download and decrypt a file, it first needs to contact the CSP. When the CSP checks the progress list and finds out the required file is already shared with another FN in the same FF, the CSP will reply with a notification to contact the other FN which already has the file in the same FF. The requester (FN) will contact the FN posses the file to provide the desired file.

Algorithm 3.11 is executed by authorized FNs prior to performing encryption process and after performing decryption process to make sure that files/blocks not been maliciously modified.

Algorithm 3.9 PrivateKeyUpdate

1: Remaining Fog Nodes in the federation receives $FNPK_{ver+1}$
2: $FN_{PR-S-ver+1} =$
$(D = g^{(\alpha+r)/\beta},$
$\forall i \in S : D_i = g^r.H(FN_{ID})^{m(\gamma_{ver+1})},$
$D_i^{'} = g^{r_i}.H(FN_{ID})^{m(\gamma_{ver+1})}$

Algorithm 3.10 Fog Nodes Secure-Communication

FN receives a request,from other FN in same FF, to send file X or Blocks of file X
if FN requester is verified **then** // in the same FF
    sender sends file/blocks to the requester
else
    decline the connection
end if
requester performs decryption process by calling algorithm-6

Algorithm 3.11 Integrity Check

Part 1: Blocks Hash Calculation
for every $M_i \in$ M **do**
    Hash(M_i) = MD
end for
Part 2: Blocks Hash Checking
When a FN retrieves Encrypted Data from CSP or from another FN;
if Hash(M_i == MD **then**
    Return ("Integrity is Accurate");
else
    Return False;
end if

3.7 Performance Evaluation

This section presents the correctness and security analysis of avoiding the threats to this approach as per discussed in section 1.5. Next, we discuss the Encryption-Based Approach design and setup. Then, we conclude by analyzing the performance of the proposed approach by performing different experiments with various factors.

3.7.1 Security Analysis

In this section, we first show the correctness of our approach. Then, we present the security analysis in aspects of security against a rogue fog node, MITM attack, and a curious cloud service provider.

3.7.1.1 Correctness

The correctness of this approach is that the designed algorithms can effectively achieve the desired objectives and security expectation. In this approach, the correctness is based on authorized fog nodes ability to encrypt and decrypt the files through the algorithms we design. We demonstrate that our proposed protocol is correct as follow. The fog node must first divide the file to be encrypted into multiple blocks of predefined size. Then, FN encrypts every block as $C_{i_{ver}} = M_i.e(g,g)^{\alpha s}$. After that, it aggregates all the encrypted blocks to form the cipher-text as $C_{ver} += C_{i_{ver}}$ in the data encryption algorithm.

The authorized FN can contact the CSP to download and decrypt a certain cipher-text. Specifically, the FN executes the recursive function $DecryptNode(CTver, FN_{PK-S-ver}, r)$ on the root R of the subtree T to find $A = e(g,g)^{rs}$. The FN can decrypt the cipher-text by decrypting each block $M_i \in$ cipher-text. Then, FN aggregates decrypted blocks to obtain the cipher-text, as follow:

$M_i = \frac{C_{ver}}{e(C,D)/A} = \frac{M_i.e(g,g)^{\alpha s}}{e(g^{\beta s}, g^{(\alpha+r)/\beta})/e(g,g)^{rs}}$

$= \frac{M.e(g,g)^{\alpha s}}{e(g,g)^{(\beta s\alpha+\beta sr)/\beta}/e(g,g)^{rs}}$

$= \frac{M_i.e(g,g)^{\alpha s}}{e(g,g)^{(s\alpha+sr)}/e(g,g)^{rs}} = \frac{M_i.e(g,g)^{\alpha s}}{e(g,g)^{s\alpha}}$

3.7.1.2 Security against Rogue Fog Nodes

A Fog-Node (FN) might go rogue for whatever reasons, such as being maliciously compromised. In this case, the encrypted data stored either in the Cloud-Service-Provider (CSP) or in other FNs are in dnager from the rogue nodes.In our approach, the encrypted files in the cloud are protected against the ousted FNs by the following process: when updating the FFL, the TAS updates the SPK and propagates it through a secure channel to the FNs that are not yet members of the FF.In the meantime, it sends notifications to the CSP to re-encrypt the files encrypted with a specific FF's attributes, and it notifies the FNs remaining

in the FF to update their secret keys. In this case, the ousted FN does not have the updated SPK and, therefore, cannot update its private key for a future decryption process. This feature makes our approach efficient in protecting end users' data confidentiality.

3.7.1.3 Security against MITM Attack

In this approach, we assume that man-in-the-middle attack occurs when a third party (attacker) tries to intercept encrypted data transmission among FNs and between the CSP and FNs. The attacker can intercept block/blocks of the file to tamper with end-users' encrypted data. This approach protects end-users' data integrity when being in transit among FNs and between the FNs and the CSP in two methods as follow.

First, the encryption process is preceding by a hash calculation for each block in the files to be encrypted. Next, the encrypted file is uploaded to the CSP along with its blocks hashes. In this context, whenever another FN in the same FF retrieves the encrypted files, it calculates the hash of each block after the decryption process and then matches the hashes against the received hashes. If they match, the integrity is preserved; otherwise the corresponding blocks are damaged.

Second, the adversary ,that wants to tamper with the transmitted data, needs to have a private key related to the current version of FFPK in order to decrypt the transmitted blocks. Therefore, the adversary to generate a private key needs the following: 1) a correct ID gained from the TAS upon joining the system. 2) a knowledge about the FF's attributes "S" and what is r_i assigned to each attribute in S, see algorithm[]. 3) a current version of the System-Public-Key (SPK) and the Fog-Federation-Public-Key (FFPK) which are propagated by TAS to all authorized FNs in the FF.

Because the adversary cannot get any of the previous elements to calculate a legitimate private key without an insider collaboration which we assume it is not occur in this

approach for now, and because of the hash function concept exploited in our approach, we can conclude that our approach is resilient against MITM attack.

3.7.1.4 Security against a curious Cloud Service Provider

The CSP receives an encrypted data along with its hashes for storage. This approach aims to prevent the CSP from having clue what the encrypted files' contents. Therefore, the only information the CSP has are: 1) the Fog-Federation-Public-Keys (FFPKs), 2) the encrypted data with their hashes, 3) list of the Fog-Nodes in each FF. Someone might argue that the CSP can generate a private keys related to the FFPK. However, since the CSP has no clues about the System-Public-Key (SPK) and the System-Master-Key (SMK), and the random $r_i \in Z_{P^*}$,that TAS used to generate the FFPK, it cannot generate a private keys from those elements. In this case, this approach successfully prevents the CSP from decrypting the encrypted data.

3.7.2 The Encryption-Based Approach Design and Setup

We run the experiment of our approach on macOS Mojave 10.14 with an intel core i7 CPU 2.2 GHz and 16.00 GB RAM. We utilize the java pairing-based cryptography (JPBC) library used in [89]. JPBC library provides a port of the pairing-based cryptography library (PBC) [90] to perform the fundamental mathematical operations of pairing-based cryptosystems in java.

We simulated the CSP and TAS as servers and the FNs as local clients. We have simulated the following Algorithms:1,2,3,4,5, and 6. Then, depending on that we collected the experiment results. Our experiment tends to show that our approach is time efficient in case of retrieving data from the same FF other than from the CSP. Any FN can access data file encrypted by another FN in the same FF. For example, when specific FNx requests to

access encrypted data in the CSP, the CSP checks the progress list if any FN has obtained the requested file in the same FF, if yes the CSP will notify the requester to communicate the FN that has the encrypted file to send it. Otherwise, the CSP sends the encrypted file that has not been accessed by any FN in the same FF to the authenticated requester. Our results show that the time it takes the FN to retrieve and decrypt a particular file from other members in the same FF is less than obtaining it from the CSP. We let the FN divide the file into blocks of 50MB size, and then perform other processes. However, fog computing has been introduced to reduce the overhead burden on the cloud and to minimize the time latency between the end users and the cloud. We have studied that and collect results supporting such a claim.

3.7.3 Experimental Evaluation

In this section, we provided the experiments we carried on to show the feasibility and efficiency of the Encryption-Based Approach.

3.7.3.1 Experiment 1: Time to Encrypt, to Hash and to Upload Files to the CSP

Section 3.7.1 shows that this approach is robust in terms of algorithms correctness and security against rogue Fog-Nodes (FNs). In this experiment, we measure the time it takes a Fog-Node (FN) in a given Fog-Federation (FF) to hash, encrypt, and upload files to the Cloud-Service-Provider (CSP). In this approach, the number of attributes is fixed to two attributes that are location and service FNs provide. Prior to encryption process, we let FNs to split files to be encrypted into blocks of 50MB each. We repeated the experiment on our generated input files 30 times, then we took the average to analyze our approach.

As shown in Figure. 3.4, the splitting and hashing processes grow linearly for files with

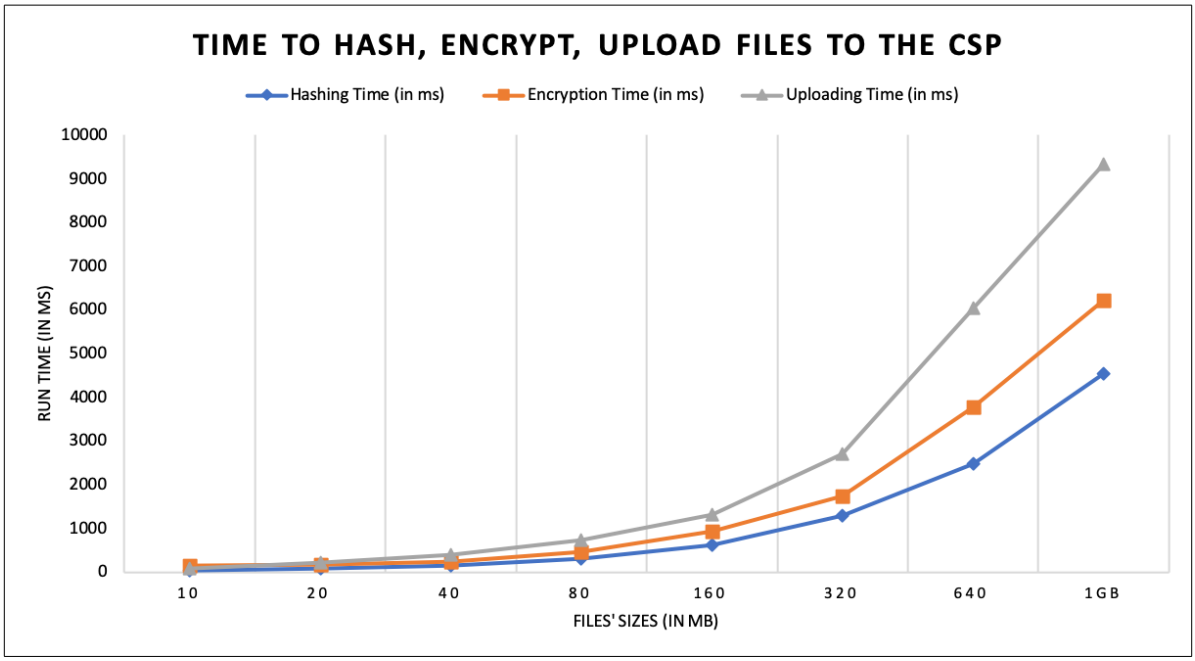


Figure 3.4: Hashing, Encryption, Uploading of Files with Various Sizes

one block. On the other hand, they take a slight exponential behaviour for files with more than one block and sizes less than 160 MB. For files with sizes greater than 160 MB, they grow in noticed exponential time. In addition, the files encryption process grow linearly in case of files with one block, while they grow slightly exponentially for files of two blocks with sizes less than or equal 160MB. It is obvious that the time increases sharply exponentially for files of greater that 160 MB. The uploading process takes linear time for for files of one block while it exponentially grows for files of more that one block. Figure. 3.4 also shows that the hashing, encryption, and uploading operations behave exponentially for files with more than a block. This behavior is attributed to our exploiting of the Attribute-Based-Encryption (ABE) algorithm as it is based on interpolation method. Also each block is encrypted independently from other blocks.

3.7.3.2 Experiment 2: Time to Retrieve and to Decrypt, and to Check the Integrity of the Ciphertext from the CSP

In addition to that this approach is able to remove a specific FN of the FF, it is efficient in reducing the communication overhead between the CSP and FF's members as the CSP will not send any file/blocks, to the FN, which already been shared with another node in the same federation. CSP rather notifies the requestor to communicate the FN, in same FF, which already has a copy of the file or blocks of the file. As justification, we performed Experiments 2 and 3 to support this hypothesis.

In Experiment 2, we performed a study to measure the time it takes the encrypted files/blocks to be downloaded from the CSP to the FN (requestor). Then, we studied the decryption time takes a certain FN to decrypt the encrypted files/blocks. And finally we studied the integrity check time of the decrypted files/blocks.

Fig. 3.5 shows that the running time it takes a FN to retrieve a ciphertext from the CSP linearly grows for encrypted files with one block. For encrypted files with more that one block, the retrieving time surges exponentially. The same behavior was inferred about the decryption operation in which it grows linearly for files with one block while it grows exponentially for files with more than one block.

This approach guarantees the integrity of the encrypted data to be protected against any unauthorized modification. When a FN in a given FF retrieves an encrypted files/blocks either from the CSP or from another FN in same FF, it is able to check the retrieved files integrity. The integrity checking operation grows linearly for files with size of 80 MB or less, while it increases exponentially for files greater that 80 MB, as shown in Fig. 3.5.

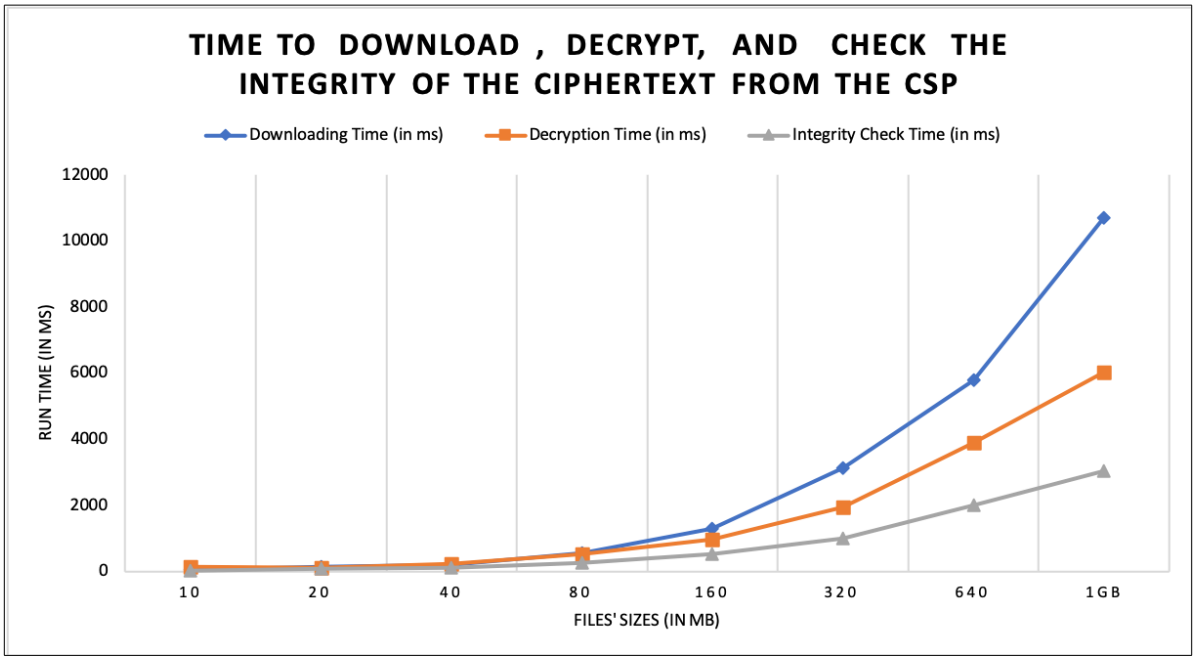


Figure 3.5: Retrieving Ciphertexts from the CSP, Decryption and Integrity Check

3.7.3.3 Experiment 3: Time to Retrieve and to Decrypt Files from same FF

One of the substantial feature this approach provides is FN capability to retrieve encrypted files/blocks from another FNs in same FF in which they share same FF. For example, suppose FN_1 wants to retrieve Encrypted-File (EF_1). Suppose another FN in same FF as FN_1 has an updated copy of EF_1, in this case FN_1 will retrieve the required EF from same FF rather than from the CSP. This means less communication overhead between the FFs and CSP.

We performed Experiment-3 to show the efficiency of our proposed approach in reducing communication overhead between FFs and CSP. This experiment studied the time to retrieve and decrypt encrypted files from another FN in same FF. Fig. 3.8 illustrates that the time consumed by the FN to retrieve the ciphertext from another FN in the same FFgrows exponentially. However, while retrieving the same ciphertext from the CSP, it performs better as shown in Fig. 3.7

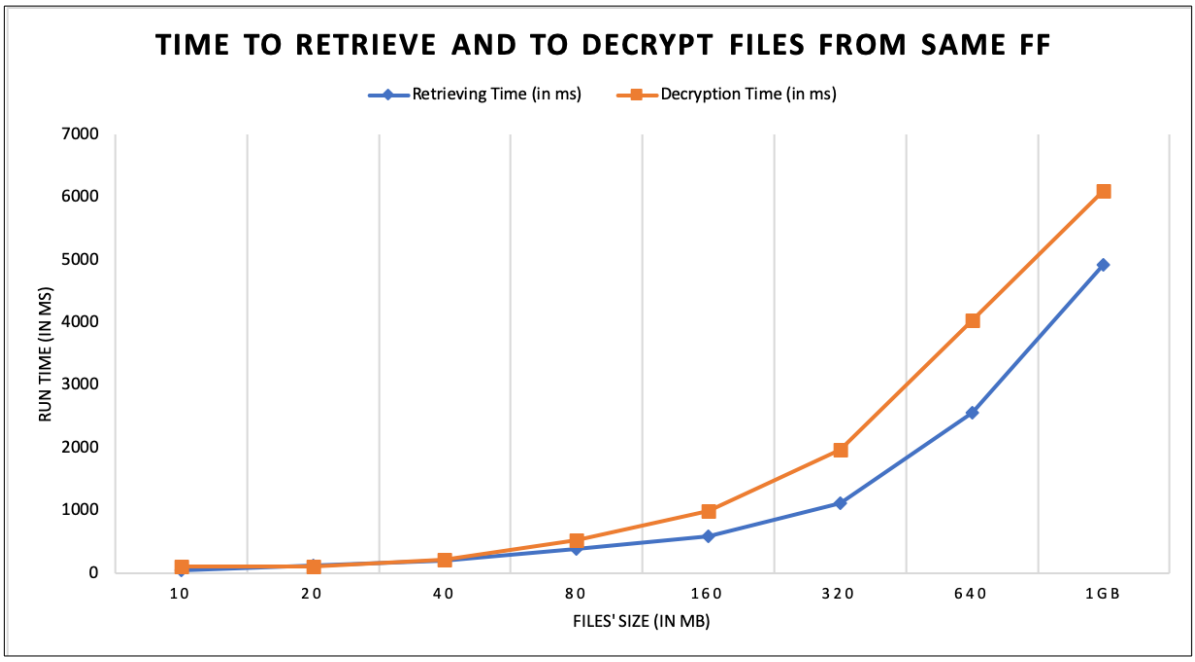


Figure 3.6: Retrieving and Decryption of Ciphertexts from the same Fog Federation

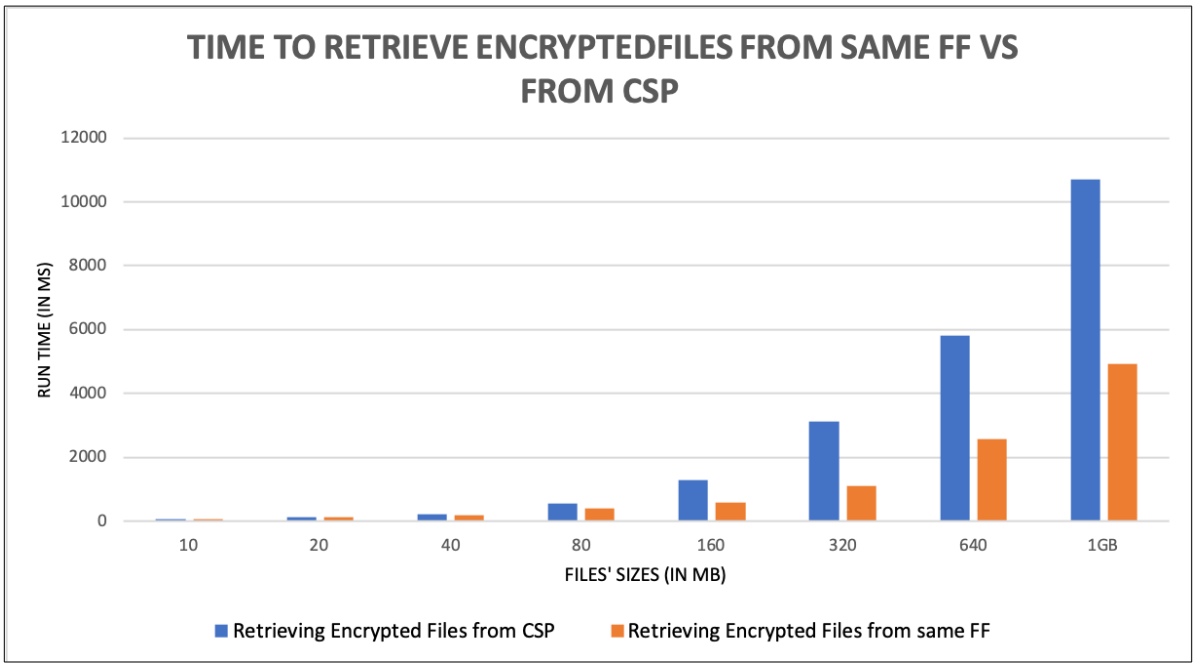


Figure 3.7: Retrieving and Decryption of Ciphertexts from the same Fog Federation VS from CSP

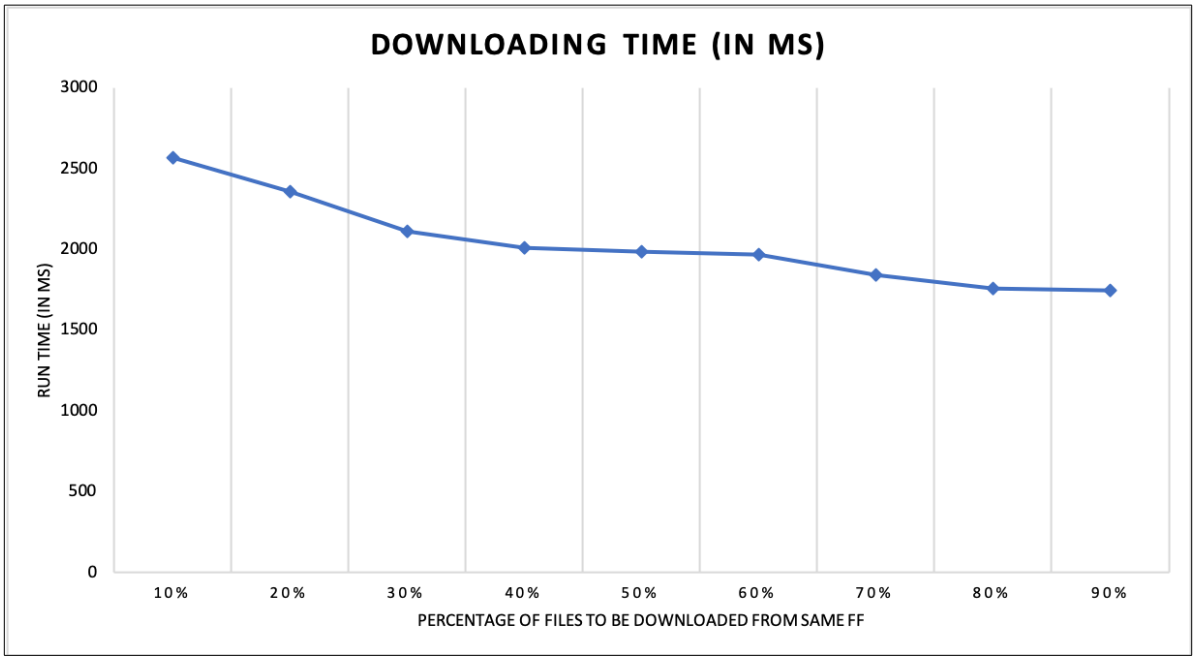


Figure 3.8: Retrieving Various Percentages of Ciphertexts from the Fog Federation and the Cloud Service Provider

3.7.3.4 Experiment 4: Time to Download Various Size of the Ciphertext from the CSP and same FF

We performed Expriment-4 to show that the Fog-Federation concept is efficient in terms of encrypted data retrieving. We have studied the running time of downloading different sizes of the ciphertext from the CSP and the same FF. Our work, for now, is focusing on the side between fog nodes and the cloud.

Fig. 3.8 shows that the running time decreases linearly when the ciphertext size to be retrieved from the same FF is increased, and this result supports our argument. However, the best case for the FN is to retrieve the whole file from other FNs in same FF, while the worst case is to retrieve the whole file from the CSP.

3.8 Discussion

Based on the performed security analysis, we observed that this approach is correct by proving our algorithms correctness followed by showing our approach robustness against the rogue fog nodes, MITM attack, and curious cloud service providers. This means that this approach supports the primary goals each system must have to be secure: Confidentiality, Integrity, and Availability. This approach also provides fine-grained access control. For more details read through section 3.7.1.

Because this work focuses on securing the channels between the FF and the CSP, we assumed that communication channels between end devices and the FNs were fully secure. Because FNs can divide the messages to be encrypted into multiple blocks and the CSP maintains a progress list of each message, the CSP only sends the messages or blocks that have not been accessed by any FN in the FF. Therefore, our approach minimized the communication overhead between the CSP and FFs. We assumed that the FNs in a specific FF were not busy, so the communication among the FNs in same FFs was time efficient. In our proposed approach, we assumed that the channel between FNs in the same FF was secure. We also have not considered a malicious fog node detection as it is out of our scope for now.

3.9 Conclusion

End users' data confidentiality could be violated while travelling through the FNs to the CSP. Besides, the FNs would go rogue for what so ever reason. When the FN providing services to end users is down, the data would not be available. In this chapter, we proposed an efficient, secure, and a fine-grained data access control approach to create FFs of FNs with same attributes. We utilized the ciphertext-policy attributed-based encryption (CP-ABE) algorithm. This approach is efficient in protecting the FF against rogue FNs by

ousting them off the FF. So they cannot access the encrypted files in the CSP.Our approach classified the FNs into different FFs depending on the location and services that each FN provides; one FN can belong to multiple FFs. Moreover, our approach enabled the FN to divide the messages to be encrypted into multiple blocks prior to performing encryption and uploading operations. This helps to reduce communication overhead between the CSP and the FFs. Based on our security analysis, the owners' data confidentiality and availability is maintained.

4 Multi-Level Approach

In this chapter, we developed a multi-level approach to protect fog federations from rogue fog nodes' behaviours. Most sections that are shown in this chapter has been published in [91] verbatim.

4.1 Introduction

The past few years have witnessed expeditious and successful developments in processing and storage technologies. Cheaper, more powerful computing resources are now ubiquitously available [2]. The most exceptional technology created in this technological evolutionary era is, unarguably, cloud computing. Cloud Computing has gained a huge popularity providing on-demand services to individuals and organizations via the internet[3][4][5]. Cloud computing paradigm features unprecedented availability, dynamic scalability, and economy that fulfill huge demands for computation and storage resources[7][8]. For these reasons, individuals and organizations are substituting cloud storage for local storage. Customers have generated significant amounts of data to outsource to cloud data centers for processing and storage without concern for hardware or software requirements or data size. However, data owners faced issues when using cloud services even considering the outstanding benefits provided by those same services. First, the cloud computing paradigm suffers from high end-to-end delay, communication overhead, and traffic congestion because data is stored in geographically distinct locations, typically far from data owners [9][10]. The slowing in communication and end-to-end delays are exacerbated by the increase in the number of end-user devices now connecting to the internet [11]. Second, individuals' and organizations' data are at risk in terms of security and privacy while in transit to and stored in, the

cloud [12][13]. Last but not least, the issue of data access control has hindered individuals' and organizations' adoption of cloud services [14]. Maintaining data security and privacy in cloud computing was a bothersome issue for data owners and a research hot topic for many years. Accordingly researchers have adopted variant and advanced cryptographic techniques and made great contributions toward resolving security and privacy issues. However, a new technology resolving the time latency and communication overhead issues remains open to research and implementation [9][15].

CISCO introduced a fog computing paradigm in 2012 [10] to reduce communication overhead and time latency between cloud service provider and end-users and to provide new services and applications to end-users [92][93]. Fog computing was deployed as multiple clouds close to the ground so that end-users could utilize computing and storage services at the edge of the network [15]. Fog computing subsumes multiples fog nodes which provide end-users services [16]. Fog computing has introduced multiple features including real time applications, low latency, location awareness, mobility, heterogeneity, and the capability of processing high number of nodes [17][18].

Fog computing initiates data storage, computing capacity, and networking close to data owners [17] and in doing so, fog computing inherits the security and privacy issues cloud computing has faced [13][19]. Fog computing is a connector between the cloud and end-users' devices [19]. End-users data travels through fog nodes to the cloud and vice versa. As data travels, it becomes challenging to protect it from attack or breach. Data can be attacked in different ways: 1) The fog node to which the data is outsourced could go rogue and breach this sensitive data. 2) The fog node providing services to the end-user could be maliciously compromised. 3) Confidential data could be intercepted and en-damaged malicious parties in the media between fog nodes and the.

From here, there is a significant and necessary need for a approach to protect end-

user data security (e.g. confidentiality, availability, and integrity), while it is stored in the cloud or fog nodes or transiting between. There is also a significant need for a approach to minimize the amount of data breached by rogue fog nodes. This chapter aims to design a new approach conceived as a means of securing end-user data by ousting rogue fog nodes, and minimizing the amount of data endangered of by those fog nodes being ousted. It is intuitive that end-user data have multiple levels of confidentiality. Therefore, this approach aspires to minimize the danger rogue fog nodes can cause to end-user data and to other fog nodes.

In this approach, the advanced cryptographic primitive, Attribute-Based Encryption (ABE), is utilized to accomplish our proposed goals. ABE enables one-to-many encryption. It has two variants: ciphertext-policy attribute-based encryption (CP-ABE) [24], and key-policy attribute-based encryption (KP-ABE) [25]. In this chapter, we propose a CP-ABE based approach to defend end-user data security and fog nodes against rogue fog nodes while maintaining low time latency and low communication overhead between Fog Nodes (FNs) and Cloud Service Providers (CSPs) as provided by the fog computing paradigm. The work here focuses on the FNs and CSPs. This chapter major contributions are as follow:

1. This approach classifies fog nodes (FNs), depending on their attributes, into fog federations (FFs) to create trust among those FNs and for the exchange of encrypted data.

2. This approach aims to minimize the impact of a breach by rogue/malicious FNs to encrypted data . Therefore, this approach divides files to be encrypted into multiple blocks depending on the level of data sensitivity. Then, FNs are distributed into subgroups within each FF depending on the portion of the encrypted files those FNs can access. There are shared attributes among all subgroups enhancing data sharing among those FNs. This means if one FN goes rogue, the whole encrypted file will not

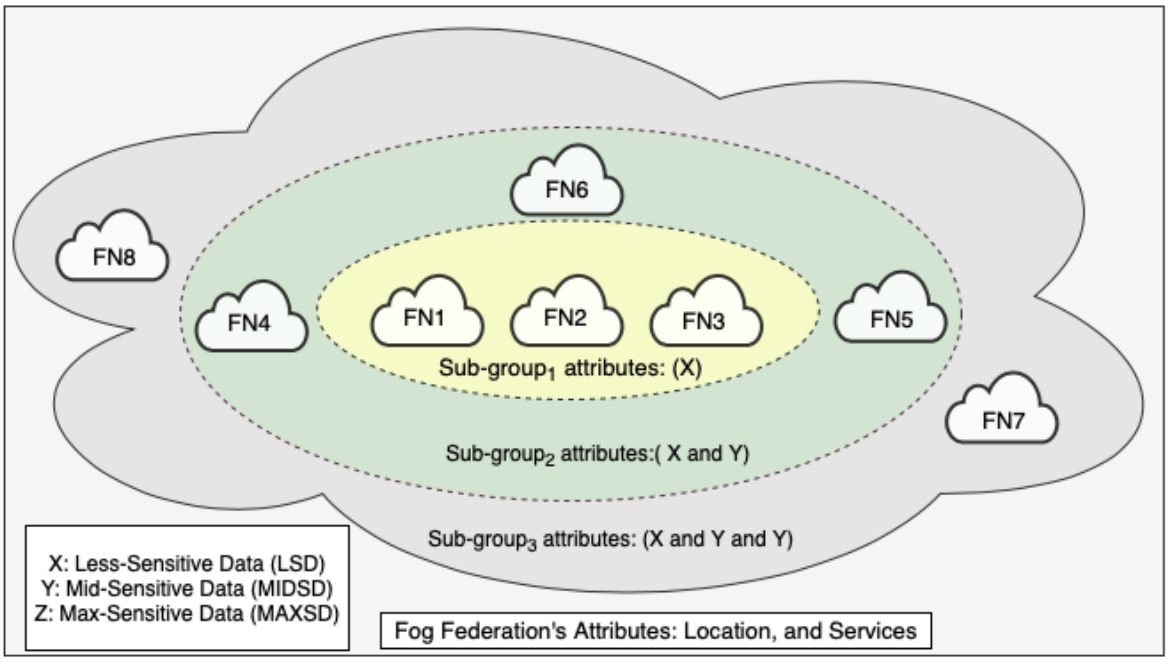


Figure 4.1: Sub-groups Classification

be in danger of breach.

3. This approach prevents rogue FNs from accessing encrypted data in a CSP or in another FN in the same Fog Federation Subgroup (FFSG). Legitimate FNs remaining in the FFSG are allowed to access encrypted data with no latency.

4. This approach maintains the low communication overhead feature the fog computing paradigm has proposed.

4.2 Classifying Fog Nodes into Subgroups

The concept of Multi-Level Approach is predicated on the classification of Fog-Nodes (FNs) in each Fog-Federation (FF) into subgroups, e.g. Fog-Federation-Subgroups (FFSGs), subject to the access level granted to each (FN) in a given (FF), Figure 4.1. The main goal of this Approach is to minimize the danger of data breach or compromise by rogue/malicious

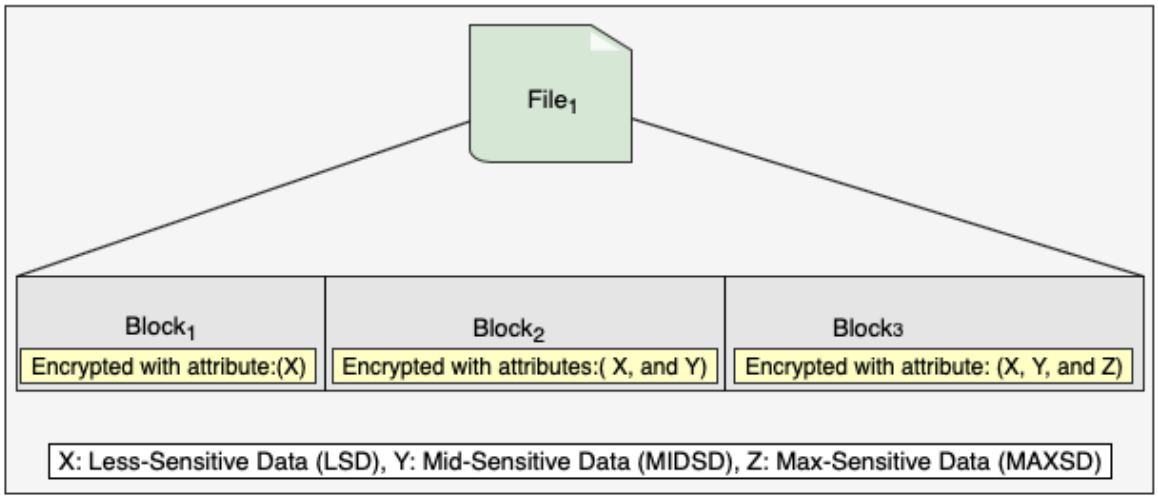


Figure 4.2: Splitting Files to be Encrypted into Blocks.

FNs. This goal is achieved by distributing FNs into FFSGs based on the parts of the encrypted files they can access. There are shared attributes among all subgroups. This enhances data sharing among among FNs. Consider FN_1, FN_2 and FN_3; three FNs with authorization to access only Less-Sensitive-Data (LSD), Figure 4.1. Should one of these FNs go rogue, the Mid-Sensitive Data (MIDSD) and Most-Sensitive Data (MAXSD) are not in danger of breach by these rogue nodes.

4.3 Splitting Files into Blocks

In the Multi-Level Approach, FNs only serve as intermediaries between the Cloud-Service-Provider (CSP) and end-user devices. At the beginning, the Trusted-Fog-Node (TFN) oversees division of the files identified for encryption into three blocks, each encrypted to the targeted FFSGs attributes. Each block is therefore accessible only to FNs satisfying the FFSGs attributes to which it has been assigned, see Figure 4.2.

4.4 Multi-Level Objectives

4.4.1 Confidentiality

The Multi-Level Approach provides data confidentiality on the FFSG level by utilizing the Cipher-Text Attribute-Based Encryption approach []. The Multi-Level Approach splits a file into multiple blocks and performs the encryption process for each block using the FFSG's attributes. More specifically, each block is accessible only to FNs that satisfy the FFSG's attributes. No block contains complete information regarding an encrypted file. This Approach assures that FNs in a FF are divided into FFSGs whose members can access only blocks of files encrypted to their FFSG's attributes. FNs can not access encrypted data stored in the Cloud-Service-Provider (CSP) unless their identities and the FFSG in which they dwell, are proven legitimate to the CSP. This is accomplished by a digital signature concept []. FNs must sign the request sent to the CSP with their Fog-Node-Fog-Federation-private-key (FNFFPK). Then, the CSP verifies the provided signatures using the Fog-Federation-Public-Key (FFPK). The CSP verifies the FN's (requestor) FFSG before returning an encrypted block(s), returning to the FN only blocks encrypted to the FN's FFSG's attributes. The means use of the FFSG concept boosts data confidentiality in this Approach as it reduces the amount of data in danger of breach by rogue or malicious FNs by dividing the data.

4.4.2 Integrity

The Multi-Level Approach provides data integrity to stored files, preventing any intentionally malicious modification to the encrypted data. A Trusted-Fog-Node (TFN) first divides the file deposed for encryption into blocks, calculates the hash of each block, and encrypts these blocks. The blocks and their hashes are then uploaded into the CSP for storage. When an authorized FN in a FFSG retrieves a block(s) from the CSP, the CSP

returns the requested block(s) and their hash(es) to the FN. The FN decrypts the block(s), calculates the hash(es), and upon verifying that the hash(es) match, accesses the block. Mismatched hash(es) will verify corruption of the encrypted block(s).

4.4.3 Availability

Fog computing plays a crucial role in the current computing paradigm. It works as a middle connector between the cloud and IoT devices as it filters and pre-processes data before it is sent to the cloud [**?**]. Fog computing provides availability and faster service to end-user devices. Considered a great feature of fog computing, nodes are deployed close to the ground eliminating the need for end-devices to wait for a centralized cloud to provide services []. The Multi-Level Approach provides availability service to end-users devices via the concept of the FF. End-users cannot benefit from the services a FN provides in their region if that FN goes rogue or is down. The FF concept provides end-users with services via any FN in the FF, bypassing a rogue or compromised FN.

4.4.4 Performance

The Multi-Level Approach guarantees data availability close to end-users devices by enabling the FNs in each FFSG to retain the encrypted blocks retrieved from the CSP for a period. The idea of FF boosts this approach performance as the FN (requestor) can retrieve the needed block(s) from another FN in the same FF. To accomplish this, the CSP maintains an Encrypted-Files-Tracking-Table. This allows the CSP to track and find files and the FNs in current possession of them. When the CSP receives a retrieval request from a FN, it checks the Encrypted-Files-Tracking-Table. If another FN in the same FF has an updated copy of the requested block(s), the CSP directs the FN (requestor) to the FN in possession of the block(s) for retrieval. This feature plays a huge role in reducing the communication overhead

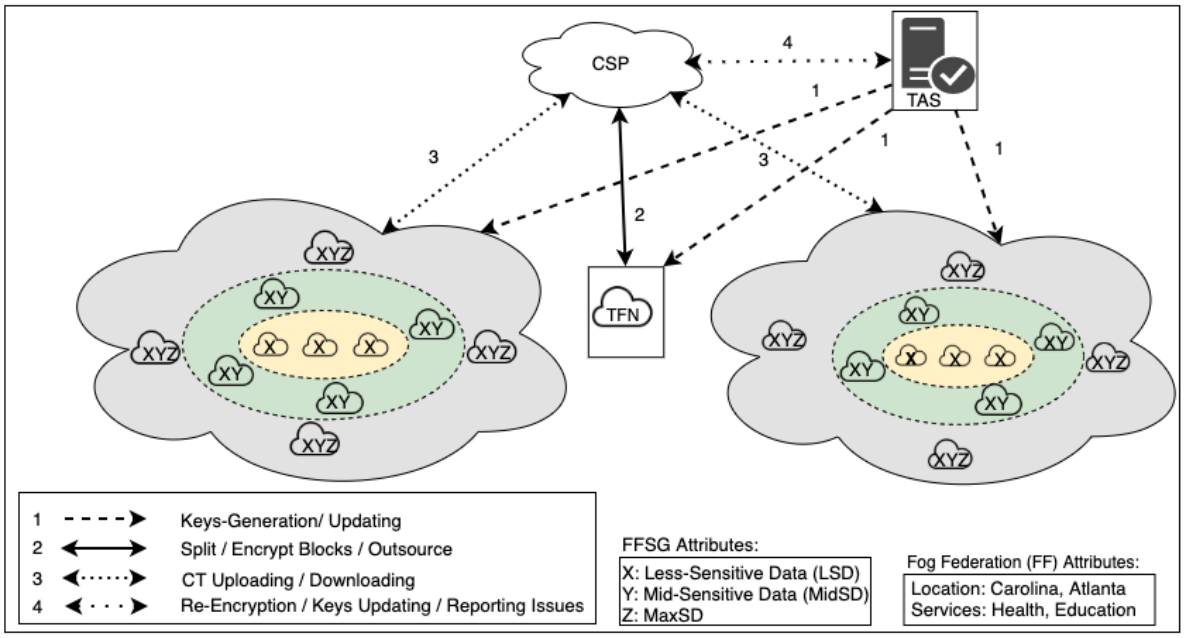


Figure 4.3: The Multi-Level Approach System Model.

between the CSP and FFs. After a period, the CSP considers the encrypted block(s) held by FNs in a FF invalid as a precaution, further protecting files if an FN goes rogue or is maliciously compromised.

4.5 Multi-Level Approach Overview

As Figure 4.3 demonstrates, Multi-Level Approach is comprised of the following entities: a Cloud Service Provider (CSP), Fog Nodes (FNs), a Trusted Authority Server (TAS), and a Trusted Fog Node (TFN). FNs with the same attributes form a Fog Federation (FF).

Fog computing emerged as an extension allowing CSPs to address data delivery time

Table 4.1: Encrypted Files Tracking Table

FN_{ID}	FN_{att}	Available EF
FN_1	X	EF_1, EF_6
FN_2	X and Y	EF_5
FN_3	X and Y and Z	EF_7
...	...	...

latency. However, fog computing inherited cloud security issues. Fog computing is prone to malicious attacks, and accordingly, end-user data security is at risk.

To maintain individuals' data security (e.g. confidentiality, integrity, and availability) as it is in transit from the FNs to the CSP and vice versa, the advanced cryptographic primitive, CP-ABE algorithm was exploited. The CP-ABE enforces fine grained data access control securing communication between FNs and the CSP. This approach equips each FN with a set of attributes and associates each encrypted file with an access policy which is defined by these attributes. This means that no FN can access encrypted data unless it satisfies the access policy associated with the data. In addition, to guarantee an authentic communication among FNs in same FF, the digital signature concept was utilized.

This approach manages to minimize the amount of data compromised by a rogue FN or attack. This is achieved by classifying the FNs into different FFs of FNs based on a set of attributes that characterizes each FF. Consequently, FNs in each FF are classified into subgroups described by a set of attributes from inner to outer levels. The FNs located in the outermost subgroup can access data encrypted to FNs located in inner subgroups. Contrarily, FNs located in the inner subgroups cannot access data encrypted to FNs in the outer subgroups. Such a construction reduces the amount of data impacted if a FN in an inner subgroup goes rogue or is attacked. The CSP maintains a progress list of the encrypted files to demonstrate the possession of each FN in the FF, as shown in Table. 4.1. For the sake of clarity, consider the following scenarios as an examples:

Suppose five fog nodes described as (FN_1, ..., and FN_8) are grouped by the TAS into a FF according to their location and the services they provide. Then, the TAS classifies those FNs into subgroups and characterizes them by one or multiple of these attributes: X = Less-Sensitive Data (LSD), Y= Mid-Sensitive Data (MIDSD), and Z= Max-Sensitive Data (MAXSD), see Figure4.1. In particular, the TAS characterizes each FN with single

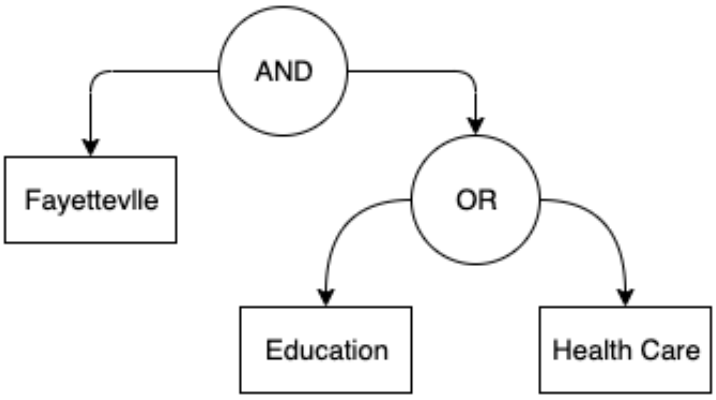


Figure 4.4: Access Structure for FF Attributes

or multiple attributes according to their subgroup in addition to the FF's attributes they belong to. FN_1, FN_2, and FN_3 are characterized by attribute X which means they can only access data encrypted with attribute X. FN_4, FN_5 and FN_6 are characterized by attributes X and Y, then FN_7 and FN_8 are characterized by attributes X, Y, and Z. This means FNs may be granted access to encrypted files marked with one or all of these attributes.

In the beginning, the Trusted Fog Node (TFN) splits $File_1$ into blocks based on data sensitivity, as Figure 4.2 shows, then performs hashing and encryption processes for each block based on the subgroup's attributes. After that, the TFN outsources the Encrypted File (EF_1) along with its hashes to the CSP for storage.

Suppose FN_2 seeks retrieval of $EF_{1,x}$, which is encrypted and can be accessed to FNs in the Fog-Federation Subgroups (FFSG) with attributes X. FN_2, the requestor, sends a retrieval request to the CSP for $EF_{1,x}$. The CSP checks the encrypted file tracking table. If another FN in the same FF has recently accessed the requested block, then the CSP will direct the requestor (FN_2) to the FN in possession of that block, which is FN_1 or FN_3. This feature reduces the communication overhead and time latency between the CSP and FNs. In this example, FN_2 is in the most inner FFSG which can access blocks of $EF_{1,x}$ characterized by attribute X. As seen in the encrypted file tracking list, FN_1 is in possession of the required

block, so FN_2 can retrieve this block along with its hash from FN_1. If no FN in the same FF has an updated copy of the requested file/block and FN_2 satisfies the FF's and block's attributes, then the CSP returns the requested block and its hash to the requestor (FN_2).

A FN could go rogue or be compromised by an attack. Therefore, to guarantee end-user data security and reduce the amount of data endangered by a rogue FN, precautions are necessary. This approach acts as a barrier against a rogue, ousted FN and reduces the volume of breach endangered data. To clarify that, suppose FN_1 goes rogue or is compromised by an attack. Our approach is designed to oust the rogue fog node (FN_1) and render it incapable of accessing any other encrypted file blocks characterized by attribute X in the CSP or other FNs in same FF. FN_1 does not have access to blocks characterized by Y and Z attributes. This design feature means the data volume breached through a rogue or compromised FN is minimized.

This approach is designed to instantaneously and simultaneously prevent ousted FNs from accessing encrypted files, while maintaining data delivery to legitimate FNs remaining in the FF. The CSP and TAS Ousting Rogue FN and Re-encryption algorithms are accomplished at the same time.

In addition, this approach helps in maintaining data integrity which protects encrypted data from interference when stored or in transit between a CSP and FNs or FNs in the same FF . This feature is accomplished by verifying the hash of encrypted files every time communication occurs among FNs or between the CSP and FF members.

4.6 Threat Model

In this proposed approach, three security models are demonstrated, reflecting the following cases:

4.6.1 Rogue fog nodes

Where an attacker compromises a fog node, or a fog node goes rogue. A compromised or rogue FN may attempt access to file blocks which are encrypted to FNs placed in other FFSGs. In addition, this kind of node may enact retrieval of encrypted blocks, without proper access rights, from the CSP. Gaining access to the encrypted data of upper level FFSG members can lead to violation of end-user data confidentiality and integrity. For example, when a rogue or malicious FN becomes capable of accessing blocks encrypted to other FNs in other FFSGs, end-user data confidentiality and data integrity is compromised. End-user privacy would be exposed to unauthorized parties.

4.6.2 Man-in-the-Middle (MITM) Attack

A man-in-the-middle (MITM) attack means that an interceptor seizes the information sent by the FNs, and tampers with it, and resends it to the CSP. This tampering creates the potential for serious issues including the spread of false information and damage to data in transition

4.6.3 Carious Cloud Service Provider

A curious CSP may have access to confidential data about individuals, including names, addresses, salaries, and private images. A CSPs discovery of and access to any encrypted blocks has the potential to create negative impact on end-user privacy and jeopardize data confidentiality and integrity.

4.7 Algorithms Design and Workflow of the Multi-Level Model Approach

This section describes the Multi-Level Approach based on the CP-ABE algorithm, hashing and digital signature concepts. The access tree model presented in [26, 24] is reviewed

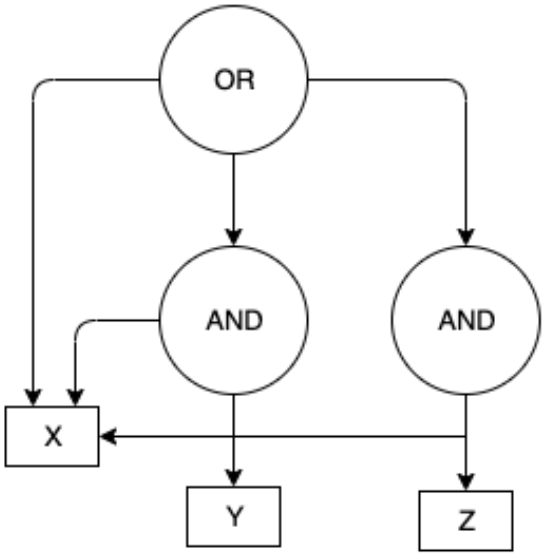


Figure 4.5: Access Structure for FF with SG Concept

first as it is exploited as an access structure. This approach has two access structures: one is for Fog Federations' (FF) attributes Figure.4.4 and one for the Fog Federation Subgroup (FFSG) attributes Figure.4.5. A description of the algorithms follows.

To satisfy the access tree conditions, we follow the same methods in [24, 25]. Let T be a tree describing an access policy. In T, each leaf node characterizes an attribute, and each non-leaf node expresses a threshold gate. For example, consider a node x with; num_x as number of children x has; and k_x as a threshold value which $0 < k_x <= num_x$. In case $k_x = num_x$, the threshold gate is AND gate, and it is OR gate when $k_x = 1$. The threshold value k_x=1 for each leaf node. For simplicity, another functions are described: $att_(x)$ is for node x's attribute only if x is leaf node, $parent(x)$ is for node x's parent in the tree, and $index(x)$ is for returning the number uniquely assigned to x. In order to satisfy the access tree, let T is an access tree rooted at node R, and let T_x be a subtree of T rooted at node x. In case set of attributes S satisfies the access tree T_x, $T_x(S)$=1. $T_x(S)$ is recursively computed as follows: 1) If x is not a leaf node, evaluate $T_x(S)$ for x's children; iff at least one of x's children k_x returns 1, $T_x(S)$ =1. 2) If x is a leaf node, $T_x(S)$ =1 iff $att(x) \in S$.

Fog Federations are created based on the access tree conditions demonstrated in Figure.4.4. Then, to satisfy the features of this approach, a Trusted Fog Node (TFN) splits each file into three blocks as described in the previous section. The Fog Nodes (FNs) in a given Fog Federation Subgroup (FFSG) are capable of retrieving and decrypting designated blocks if the access tree conditions in Figure.4.5 are satisfied. Next the algorithms required for this approach are provided.

The Trusted Authority Server (TAS) executes Algorithm 1, which is based on [24]. The input to Algorithm 4.1 is a security parameter, and the outputs are a System Public Key (SPK) and a System Master Key (SMK). The SPK is public to all FNs in the same FF, the SMK is private to the TAS. The TAS constructs two universal sets of attributes. First: Fog-Federation-Universal-Set-of-Attributes (FF_{USAT}) is used to classify FNs into FFs. FNs are classified into FFs based on location and the services they provide. Second: Fog-Federation Subgroups Universal-Set of Attributes(FFSG_{USAT}) is a set of attributes used to specify which blocks of encrypted files the FNs can access. Access is dependent upon the FFSG's attributes in which the FN is placed.

Algorithm 4.2 is also executed by the TAS. Its inputs are SPK, FF_{USAT}, and FFSG_{USAT}. It is comprised of three parts. First: it creates a new Fog Federation List (FFL) of FNs sharing the same FF_{att}. This list contains the FN_{ID} and FF_{ID} hashed to prevent id impersonation. It also contains the FF_{att}, and current version of Fog-Federation-Public-Key $FFPK_{att-ver}$. Second: it divides the FNs residing in the same FF into subgroups (FFSG). These subgroups are created according to the data access privilege(s) each FN is granted upon joining the system. It creates a Fog-Federation-Subgroup-List (FFSGL) which contains: a Fog-Federation-Subgroup-ID (FFSG_{ID}), a current version of Fog-Federation-Subgroup-Public-Key ($FFSGPK_{att-ver}$), Fog-Federation-Subgroup-Attributes (FFSG_{att}), and the FN_{ID} and FFSG_{ID} which are hashed to prevent FNs in each FFSG from impersonating other FNs.

Third: when the TAS receives a FN join request , the TAS will confirm the legitimacy of the FN by checking the predefined FFL. If the requestor (FN) attributes match one or multiple FF_{att} and FFSG_{att}, then it will be added to the corresponding FFs and the correct FFSG. Otherwise, the TAS will create a new FF with a new ID and attributes based on location and provided services. Then, the TAS divides the FF into FFSGs, assigns the requestor (FN) to the appropriate FFSG, and grants the requisite data access level to the FN requestor. Additionally, the FN is assigned an ID and added to the Fog-Federation-Subgroup-List (FFSGL). Algorithms 4.3 and 4.4 will be triggered and the $\text{FFSGPK}_{att-ver}$ will be securely sent to the FN requestor.

The TAS also executes Algorithm 4.3. Its inputs are SPK, and SMK. The aim of this algorithm is to generate a Fog-Federation-Public-Key ($FFPK_{att}$) and a Fog Federation Master Key ($FFMK_{att}$) based on the FF_{att}. The ($FFPK_{att}$) is public to all FNs in the same FF while the ($FFMK_{att}$) is known only to the TAS. Then, it generates a Fog-Node-Secret-Key ($FFSK_{FF-att}$) based on the attributes of the FF to which it belongs. The aim of the ($FNSK_{FF-att}$) is to ease authentic communication among FNs in the same FF and between the CSP and FNs. When a FN requests encrypted data from the CSP or another FN in the same FF, it signs the request using its own ($FNSK_{FF-att}$). The receiver verifies the request using the ($FFPK_{att}$).

In addition, the TAS executes Algorithm 4.4. The purpose of this algorithm is to

generate a Fog-Federation-Subgroup-Master-Key (FFSGMK) and Fog Federation-Subgroup-Public-Key (FFSGPK) for each attribute assigned to a FFSG. The FFSGMKs are private to the TAS while the FFSGPKs are public to all FNs granted access privileges to data encrypted with the FFSG attributes.

The TAS executes Algorithm 4.5. The function of this algorithm is to generate a Fog Node Secret Key ($\text{FNSK}_{FFSG_{att}}$) based on the FFSG attributes each FN is accorded.

Algorithm 4.1 Setup(λ)

1: Choose bilinear cyclic group G_1 of prime order p with generator g_1;
2: Randomly generates exponents $\alpha, \beta \in {G_p}^*$;
3: Random Oracle H: $USAT \rightarrow \{0,1\}^*$ // to map each att in USAT into random element $\in G_1$;
4: SMK and SPK as:
5: $SMK = \{\beta, g_1^{\alpha}\}$;
6: $SPK = \{G_1, g_1, g_1^{\beta}, e(g_1, g_1)^{\alpha}\}$;
7: FF_{USAT} = FN location, Services
8: FFSG_{USAT} = X,Y,Z
9: SPK is public to all fog nodes, but SMK is private to TAS

This algorithm boosts the generation of a secret key for each FN premised on the encrypted files/blocks it is privileged to access. It takes as inputs (FFSGMK, SPK, SMK, FN_{ID}) and outputs a distinct secret key ($\text{FNSK}_{FFSG_{att-ID}}$) for each FN in the FF.

This model is designed to reduce the communication overhead and time latency between CSPs and FNs. In a further effort to accomplish this, the CSP will maintain a list called the Encrypted Files Tracking List. This allows CSPs to track FNs in current possession of encrypted files, as shown in Table.4.1. Algorithm 4.6 initiates creation of the Encrypted Files Tracking List. A FN in a FF sends an encrypted file retrieval request signed with its $\text{FNSK}_{FF_{att-ID}}$ to the CSP. The CSP verifies the request using ($\text{FFPK}_{att-ver}$). If verified, the CSP will check the Encrypted Files Tracking Table. If a FN in the FF has a copy of the requested data/blocks, the CSP will return the FN_{ID} currently in possession of the file/block to the requestor. Otherwise, the CSP will return the file/block directly to the privileged requestor FN. If the requestor FN does not pass the verification step, the CSP will decline the request and report the FN as illegitimate to the TAS and other FNs in same FF.

Algorithm 4.7 is executed by a Trusted Fog Node (TFN). The benefit of adding a TFN to this design is in enabling classification of files to be encrypted into blocks based on the sensitivity level of the data. In this approach, there are three levels of data sensitivity:

Algorithm 4.2 FF and FFSG Creation and Adding new FN (SPK, FF_{USAT}, FFSG_{USAT})

Part 1: FFL Creation

Create list of FF (FFL) to contain list of FNs in the system,as
$FFL[FF_{ID}|FFPK_{ver}|FF_{att}|H(FN_{ID}FF_{ID})]_{ver}$

Part 2: Creating FFSGL of FNs Based on Data Sensitivity Level they could Access

Classify FNs in FFs into FFSGs:
$FFSGL[FFSG_{ID}|FFSGPK_{ver}|FFSG_{att}|H(FN_{ID}FFSG_{ID})]_{ver}$

Part 3: Adding new FN to FFSG

if $(FN_{att} == FF_{att} \& (FN_{att} \subseteq FFSG_{att})$ **then**
  Assign the new FN an ID as, $FN_{ID} == FN_{ID} + FFSG_{ID}$
  Add the new FN to the corresponding FFSG
  Securely, send the $FFSGPK_{ver}$ to the new FN
else
  Create new FF and assign it a unique FF_{ID}
  Divide the FF into one or multiple FFSG
  Assign the new FN an ID, $FN_{ID} == FN_{ID} + FFSG_{ID}$
  Add new FN_{ID} to the FFSGL
  Call Algorithms 3 and 4
  Securely, send the $FFSGPK_{att-ver}$ to the new FN_{ID}
end if

Algorithm 4.3 FF Keys Generation (SPK, SMK)

Randomly choose $FFMK_{ver} = \gamma_1 \in Z_p^*$
Compute $FFPK_{ver} = FF_{ID}, g_1^{\gamma_1}, (g_1, g_1)^{\gamma_1}$
$FNSK_{FF_{att}-FN_{ID}} = (D = g^{(\alpha+r)/\beta}, \forall_i \in S : D_i = g^r.H(FN_{ID})^{\gamma_1 ver}.H(i)^{r_i}.D_i^{'} = g^{r_i}.H(FN_{ID})^{\gamma_1 ver})$

X = Less-Sensitive-Data (LSD), Y= Mid-Sensitive-Data (MIDSD), and Z= Max-Sensitive-Data (MAXSD). The FNs in each FF are classified into FFSG as explained in the previous section. This algorithm has three parts.

The TFN triggers the classification of the files to be encrypted into blocks and then uploading of them to the CSP. This design boosts the approach's ability to minimize the data volume endangered by rogue/compromised FNs. Not all FNs in the same FF can access all blocks of an encrypted file. Authorized FNs in each FFSG are charged, through this algorithm, with updating retrieved data from the CSP or another FN in the same FF. In general, the TFN starts encrypting blocks of data using FFSGPK_{att}. This means no FN

Algorithm 4.4 FFSG Keys Generation

for every $i \in FFSG_{att}$ **do**
  Randomly choose $FFSGMK_{ver} = \gamma_2 \in Z_p^*$
end for
for every $i \in FFSG_{att}$ **do**
  Compute $FFSGPK_{att-ver} = FFSG_{ID}, g_1^{\gamma_2}, e(g_1, g_1)^{\gamma_2}$
end for

Algorithm 4.5 $FNSK$ Generation $(FFSGMK_{ver}, (SPK, SMK)_{ver}, S, FN_{ID})$

TAS revieves a request from a FN to join the system
if FN_{ID} is not verified **then**
  Decline;
else
  Randomly generate $r \in Z_p^*$
  for every $att_i \in S$ **do**
    randomly generate $att_i \in Z_P^*$
  end for
  for every $att \in S$ **do** //S is blocks' attributes FN can access
    $FNSK_{FFSG_{att-ID}} =$
    $(D = g^{(\alpha+r)/\beta}, \forall_i \in S : D_i = g^r.H(FN_{ID})^{\gamma_{2ver}}.H(i)^{r_i}.H(FN_{ID})^{\gamma_{2ver}})$
  end for
end if
send $FNSK_{FFSG_{att-ID}}$ to the corresponding FN and update FFSGL

in the FF can access encrypted file blocks unless it satisfies the FFSG attributes.

Algorithm 4.8 is responsible for decrypting encrypted files/blocks in order to obtain plain text, based on [24]. It is executed by authorized FNs in each FF. Its input is the block(s) of the encrypted file the requestor FN is privileged to access, the current version of $FNSK_{FFSG_{att-ID-ver}}$, and the defined attributes in the access structure on the encrypted blocks. The decryption algorithm outputs either the blocks to be retrieved or $\perp$. Each Fog Node Private Key is associated with the FN_{ID}, and FFSG_{att}. This feature makes it impossible for a FN to retrieve encrypted data unless it satisfies the associated FFSG_{att}.

Algorithms 4.9 and 4.10 are simultaneously executed by the CSP and the TAS. The TAS will oust a FN from the system for multiple reasons including report/detection as rogue or malicious FN. Rogue FNs Detection is out of our scope for now. To protect end-user

Algorithm 4.6 Encrypted Files/Blocks Tracking Table Creation/Requesting EF from the CSP($FFPK_{att-ver}$)

1: FN submits a request (intended EF/blocks) to the CSP signed with $FNSK_{FF_{att-ID}}$ of the intended block
2: CSP verifies FN, using $FFPK_{att-ver}$
3: **if** FN (requestor) not verified and cipher-text access structure policy is not satisfied **then**
4: Decline;
5: **else**
6: check the encrypted files tracking table
7: **if** available EF $\in$ requested EF **then**
8: return list of FN_{ID} have the required EF to let the requestor get it from
9: **else**
10: Send the Blocks of the EF match the requestor (FN_{att})
11: **end if**
12: **end if**

data security precautions against those rogue nodes are necessary. This approach efficiently prevents rogue FNs from accessing blocks of the file previously accessible to it. Only the FFSGPK of the FNs located in the FFSG where the rogue node originated would be updated. This means that there is no need to update the FFSGPK for all FNs in the FF. This mechanism of the approach signifies that FNs can access only data/blocks characterized by its FFSG attributes.

To prevent ousted FNs from accessing encrypted files/blocks in the CSP or in other FNs in same FF, ousting rogue FNs and re-encryption algorithms are simultaneously executed by the TAS and the CSP. This feature ensures that the approach is secure and robust in its treatment of ousted FNs. The TAS starts by removing a rogue FN from the prestored FN list. Then it updates the $FFSGMK_{att-ver}$ and $FFSGPK_{att-ver}$ to $FFSGMK_{att-ver+1}$ and $FFSGPK_{att-ver+1}$. The TAS then sends $FFSGPK_{att-ver+1}$ to the CSP for re-encryption, and to the FNs with access privileges to the blocks characterized by the FFSG attributes to update the private key(s).

Algorithm 4.11 is executed by the authorized FNs still in the FF after rogue FNs have been ousted from the FF. After the TAS updates the FFPK and FFSGPK, it will

Algorithm 4.7 Files Splitting and Encryption (SPK,$Block_i$, T, $FFSGPK_{att}$)

Part 1: by FNs

A is set of attributes represented by monotone access structure tree T
for each node x in T **do**
  set a polynomial
  if node x is a root node in T **then**
    set $q_{(0)R}$ = s // s is random value $\in Z_P^*$
  else
    $q(0)_x = q_{parent(x)}(index(x))$
    then choose $d_x = k_x$ - 1 as polynomial degree to interpolate with q_x
  end if
end for

Part 2:by TFN

divide the file to be encrypted into 3 blocks based on data sensitivity
for each $block_i \in$ the file **do**
  assign att as: $block_{i-att}$ = X=(LSD)—Y=(MIDSD)—Z=(MAXSD)
end for

Part 3:by TFN and FNs

for every $y \in Y$ **do**
Y is the set of leaf nodes in T
  C_y= $\mathrm{q}^{q_y(0)}$
  $C_y^{'}$=$\mathrm{H(att(y))}^{q_y(0)}$
end for
Compute $C = g^{\beta s}$
for every $block_{i-att} \in file$ to be encrypted **do**
  $C_{i-ver} = e(g,g)^{\alpha\gamma_2 s}$
  $C_{ver} += C_{i-ver}$
end for
Call Algorithm 12 part 1
Upload CipherText(CT) to the CSP for storage as $\{T, C_{ver}, C, C_y, C_y^{'}, MD\}$ to the CSP

Algorithm 4.8 Files/Blocks Decryption $(CT_{ver}, FNSK_{FFSG-att-ver}, x)$

1: **for** each leaf node attribute "x" in access structure T **do**
2: Set i= att(x)
3: **end for**
4: **if** i ∈ S **then**
5: Recursively Compute

$$DecryptNode(CT_{ver}, FNSK_{FFSG-att-ID-ver}, x) = \frac{e(D_i, C_x)}{e(D_i', C_x')} = \frac{e(g^r.H(FN_{ID})^{\gamma_{ver}}.H(i)^{r_i}, g^{q_{(0)y}}}{e(g^{r_i}.H(FN_{ID})^{\gamma_{ver}}, H(i)^{q_{(0)y}})} = \frac{e(g,g)^{rq_{(0)y}}, e(g, H(FN_{ID}).H(i))^{\gamma_{ver} r_i q_{(0)y}}}{e(g, H(FN_{ID}).H(i))^{\gamma_{ver} q_{(0)y} r_i}} = e(g,g)^{rq_{(0)y}} \quad (4.1)$$

6: **if** i ∈ S and x not leaf node **then**
7: **for** every child node z of x **do**
8: $F_z = DecryptNode$ $(CT_{ver}, FNSK_{att-ID-ver}, x)$
9: **end for**
10: **if** $F_z! = \perp$ **then**
11: //by lagrange and polynomial interpolation as in [24]
$F_z = \prod_{z \in S_x} F_z^{\triangle_{i,S_x'}(0)} where i = index(z) and S_z' = index(z) : z \in S_x$
$= e(g,g)^{r.q_{(0)x}} = e(g,g)^{rs}$
12: **if** x =root **then**
13: $A = DecryptNode\ (CT_{ver}, FNSK_{att-S-ver}, r) = e(g,g)^{rq_{(0)R}} = e(g,g)^{rs}$
14: **else**
15: $DecryptNode\ (CT_{ver}, FNSK_{att-S-ver}, x) = \perp$
16: **end if**
17: **end if**
18: **end if**
19: **end if**
20: Decrypt Message by Computing:

$$block_{i-att} = \frac{block_i.e(g,g)^{\alpha\gamma s}}{e(g^{\beta s}, g^{(\alpha+r)/\beta})/e(g,g)^{rs}} = \frac{block_i.e(g,g)^{\alpha\gamma s}}{e(g,g)^{(\gamma\alpha s + sr)}/e(g,g)^{rs}} = \frac{block_i.e(g,g)^{\alpha\gamma s}}{e(g,g)^{s\alpha\gamma}/e(g,g)^{s\gamma\alpha}} = block_i \quad (4.2)$$

21: Call Algorithm 12 part 2

Algorithm 4.9 Re-Encryption

1: CSP checks the encrypted files tracking table
2: Start re-encrypting blocks accessible to the ousted FN
3: updated-Ciphertext (CT)
$= C_{ver}.e(FFSGPK_{att-ver+1}, C)$
$= Block_{i-att}.e(g,g)^{\alpha\gamma_2 s}.e(g^{\gamma_2+1-\gamma_{2ver}}, g^{\beta s})$
$= Block_{i-att}.e(g,g)^{\alpha s+\gamma_{2ver+1}s}$
4: CT_{ver+1} = Updated-Ciphertext(CT)

Algorithm 4.10 Ousting rogue FN

1: Update $FFSGMK_{ver} \rightarrow FFSGMK_{att-ver+1} = \gamma_2 + 1 \in Z_p^*$
2: Update $FFSGPK_{ver} \rightarrow FFSGPK_{att-ver+1} = FFSG_{ID}, g_1^{\gamma_2+1}, e(g_1,g_1)^{\gamma_2+1}$
3: Update $FFMK_{ver} \rightarrow FFMK_{ver+1} = \gamma_1 + 1 \in Z_p^*$
4: Update $FFPK_{ver} \rightarrow FFPK_{ver+1} = FF_{ID}, g_1^{\gamma_1+1}, e(g_1,g_1)^{\gamma_1+1}$
5: Remove ousted FN
6: **for** $FN_i \in$ updated FFL **do**
7: Securely broadcast $FFSGPK_{att-ver+1}, FFPK_{ver+1}$
8: **end for**

securely send them to the authorized FNs still in the FFL and to the CSP. FNs update their own secret keys as they can access the encrypted data under their attributes. Our approach is secure against the breach or compromise of end-user data confidentiality as the ousted FN becomes incapable of accessing the encrypted data as soon as the CSP receives the new FFPK and FFSGPK, and the FNs update their secret keys. This feature proves the approach's engagement in protecting end-user data security.

Algorithm 4.11 Fog-Node-Secret-Keys Update

1: Remaining Fog Nodes in the fog federation subgroup receive $FFSGPK_{att-ver+1}$ and $FFPK_{ver+1}$
2: $FNSK_{FF_{(att-ID)+1}} =$
$(D = g^{(\alpha+r)/\beta},$
$\forall i \in S : D_i = g^r.H(FN_{ID})^{m(\gamma_{1(ver+1)})},$
$D_i^{'} = g^{r_i}.H(FN_{ID})^{(\gamma_{1ver+1})})$
3: $FNSK_{FFSG_{(att-ID)+1}} =$
$(D = g^{(\alpha+r)/\beta},$
$\forall i \in S : D_i = g^r.H(FN_{ID})^{m(\gamma_{2(ver+1)})},$
$D_i^{'} = g^{r_i}.H(FN_{ID})^{(\gamma_{2ver+1})})$

Algorithm 4.12 CT Integrity // between FNs and CSP, and among FNs in same FF

After performing algorithm 7
TNS performs:
Part 1:
for every $Block_i \in Plaintext$ **do**
    Hash($Block_i$)=MD
end for
Part 2:
When a FN retrieves encrypted file/blocks from the CSP or from another FN
if Hash($Block_i$==MD) **then**
    Return ("Integrity is Accurate");
else
    Return False
end if

Algorithm 4.12 is designed to ensure the integrity of encrypted files/blocks as they transit among and between FNs and the CSP and while stored in the CSP or FNs. In short, this algorithm ensures data consistency over its lifecycle. At beginning the TFN hashes each data to be encrypted with the $FFSGPK_{att-ver}$ of the FNs that have the access rights to those blocks. When a privileged FN retrieves a block, it verifies block integrity using its own $FNSK_{FFSG_{att-ID-ver}}$.

4.8 Performance Evaluation

In this section, the security aspects of the Multi-Level Approach and the ability of this model to evade threats are analyzed, see Section.4.6. Next, the simulation setup will be explained, followed by the performance analysis and discussion of the experiments. This section concludes the performance analysis.

4.8.1 Correctness and Security Analysis

Here the correctness of the Multi-Level Approach is discussed, and a security analysis is presented focusing on security issues involving rogue FNs, an MITM attack, and a curious

CSP.

4.8.1.1 Correctness

The correctness of this approach is based on the following: the algorithms and their ability to achieve the intended objectives and security expectations; the authorized FNs ability to encrypt and decrypt files through the algorithms.

The proposed protocol is demonstrated as correct through the following: the Trusted-FogNode (TFN) must first divide the file to be encrypted into three blocks based on data sensitivity level, see Figure 4.2; Then the TFN encrypts each block using the Fog-Federation-Subgroup-Public-Key (FFSGPK$_{att-ver}$) of the FFSG's fog nodes it is encrypted to. This means no FN can decrypt the encrypted blocks unless satisfying its FFSG's attributes. The TFN performs file splitting as follow:

for each $block_i \in$ the file **do**

assign att as: $block_{i-att}$ = X=(LSD)—Y=(MIDSD)—Z=(MAXSD)

end for

Then the TFN encrypts each block with the corresponding ($FFSGPK_{att-ver}$), as follow:

for every $block_{i-att} \in file$ to be encrypted **do**

$C_{i-ver} = e(g,g)^{\alpha\gamma_2 s}$

$C_{ver} += C_{i-ver}$

end for

The TFN also calculates the hash of every block as follow:

for every $Block_i \in Plaintext$ **do**

Hash($Block_i$)=MD

end for

The TFN then uploads the encrypted blocks and their hashes to the CSP for storage

as CipherText (CT_{ver}), see Algorithms. 4.7 and 4.12.

An authorized FN can contact the CSP to download and decrypt a block(s) of a certain cipher-text, Algorithm. 4.8. Specifically, the FN executes the recursive function $Decrypt - Node(CT_{ver}, FNSK_{FFSG-att-ID-ver}, x)$ on the root R of the sub-tree T to find $A = e(g,g)^{rq_{(0)y}}$. This takes place if, and only if, the FN satisfies the FFSG's attributes for the encrypted block(s). The FN can decrypt the intended block(s) of ciphertext by decrypting each block in cipher-text. Then, the FN aggregates decrypted blocks to obtain the cipher-text, if the FN (requestor) can access more than one block, as follows:

Decrypt Message by Computing:

$block_{i-att} = \frac{block_i.e(g,g)^{\alpha\gamma s}}{e(g^{\beta s}, g^{(\alpha+r)/\beta})/e(g,g)^{rs}} =$

$\frac{block_i.e(g,g)^{\alpha\gamma s}}{e(g,g)^{(\gamma\alpha s+sr)}/e(g,g)^{rs}} = \frac{block_i.e(g,g)^{\alpha\gamma s}}{e(g,g)^{s\alpha\gamma}/e(g,g)^{s\gamma\alpha}} = block_i$

The authorized FN can guarantee the block(s) decryption accuracy by verifying the integrity of the decrypted block(s) as shown in Algorithm. 4.12.

4.8.1.2 Security against Rogue Fog Nodes

Suppose a rogue FN is detected by the system. The Multi-Level Approach is designed to oust rogue FNs and prevent them from accessing encrypted blocks to the FFSG where they reside. A rogue FN with access to blocks of the file encrypted with the $FFSGPK_{att-ver}$, once ousted, renders the encryption invalid. Therefore, the TAS will update $FFSGMK_{att-ver}$, and $FFSGPK_{att-ver}$ to $FFSGMK_{att-ver+1}$, and $FFSGPK_{att-ver+1}$. The TAS will update the $(FFMK_{ver}, FFPK_{ver})$ to $(FFMK_{ver+1}, FFPK_{ver+1})$. Next, the TAS will remove the rogue FN from the FNL, then send the updated $FFSGPK_{att-ver+1}$ and $FFPK_{ver+1}$ to the FNs remaining in the FFL and to the CSP. The authorized FNs remaining in the FFL update their $(FNSK_{FF_{att-ID}})$ and $(FNSK_{FFSG_{att-ID}})$ to new versions as: $(FNSK_{FF_{(att-ID)+1}})$ and $(FNSK_{FFSG_{(att-ID)+1}})$, as shown in Algorithms. 4.10 and 4.11 respectively. The CSP

performs the re-encryption process as shown in Algorithm. 4.9. It also updates the hash of each block in order to maintain data integrity. In this case, the ousted FNs cannot access re-encrypted blocks

4.8.1.3 Security against MITM Attack

A Man-in-the-Middle Attack is considered a consequential security issue in Fog Computing. An MITM Attack is conducted by a malicious adversary situated between FNs and the CSP. In our system design the adversary can intercept the traffic between FNs and the CSP. In the Multi-Level Approach, the hash function algorithm is leveraged. The TFN divides the files deposed for encryption into blocks, then hashes each block, and uploads the block and its hashes to the CSP. When an authorized FN retrieves a file from the CSP it can detect any changed block by checking its hash. If a FN discovers a corrupted block, it reports it to the TFN to regenerate the damaged block. In this scenario the Multi-Level Approach is subtle in regard to detection of corrupted blocks by intermediary's malicious attack.

4.8.1.4 Security against a Curious Cloud Service Provider

One of the crucial goals of the Multi-Level Approach is the preservation of end-user data security. The CSP in this approach is considered trusted. However, as a precaution from further compromise against the CSP and to preserve end-user data security at storage, the Multi-Level Approach merely stores encrypted data in the CSP without exposing the decryption keys to the CSP. The TAS is responsible for key management in this Approach; The TAS sends the decryption keys to the corresponding FNs in the FFSG they locate.

4.8.2 Performance Analysis

This section presents the Multi-Level Approach experiment design followed by the Multi-Level Approach setup.

4.8.2.1 Multi-Level Model Approach Experiment Design

This experiment aims to evaluate the performance and the applicability of the Multi-Level Approach. In the Multi-Level Model Approach experiment design, Fog Nodes (FNs) are classified into different Fog-Federations (FFs) by attributes based on their locations and the services they provide to end-users. Then, FNs in each FF are further divided into Fog-Federation-Subgroups (FFSGs) according to the degree of data sensitivity an FN can access. As a result of this division the amount of data that can be breached by a rogue or compromised FN is reduced. This approach splits each file to be encrypted into three blocks. Each block is encrypted with the corresponding FFSG's attributes so that no FN from one FFSG has access to blocks encrypted to another FFSG's attributes. In this approach, every FN stores the block/blocks of data retrieved from the Cloud-Service-Provider (CSP) for a specified amount of time, allowing other FNs in the same FF and satisfying the FFSG attributes, to retrieve the block/blocks from the FN currently in possession of them. This feature boosts the approach's ability to reduce communication and time latency between the CSP and FF.

Studying and evaluating the system's performance in experiments using various factors to confirm its feasibility and applicability. Java programming language and the following file sizes were used to test this approach: 10MB, 20MB, 40MB, 80MB, 160MB, 320MB, 640MB, and 1GB. The system's experiments were conducted on macOS Mojave 10.14 with an Intel Core i7 CPU 2.2 GHz and 16.00 GB RAM, utilizing the java pairing-based cryptography (JPBC) library used in [89]. The JPBC library provides a port of the Pairing-Based

Cryptography Library (PBC) [90] to perform the fundamental mathematical operations of pairing-based cryptosystems in Java.

4.8.2.2 The Multi-Level Approach Setup

The CSP, TAS, and TFN are simulated as servers and the FNs as local clients. The algorithms in this approach are intentionally designed to show that dividing a FF's members into FFSGs and splitting the files to be encrypted into blocks will reduce the amount of data in danger of breach by rogue or malicious FNs. Algorithms 1, 2, 3, 4, 5, 6, 7, 8, 9, and 12 are simulated.

Data is retrieved from FNs residing within the same FF rather than from the CSP server. Any FN in the same FFSG can access a block(s) encrypted to that FFSG's attributes. An otherwise encrypted block(s) is not accessible. For instance, suppose $FN_x \in FF_{Arkansas,Health}$ requires retrieval of $File_1$, which has three blocks, each encrypted to the FFSG corresponding to the block's level of data sensitivity. FN_x sends a request, signed with its FFPK, to the CSP for retrieval of the encrypted file ($File_1$). The CSP checks the signature to verify FN_x legitimately belongs to the $FF_{Arkansas,Health}$. Then, the CSP verifies the data sensitivity level accessible to FN_x (the requestor) through its FFSG. The CSP further searches for another FN in the same FF with a current, updated, and accessible copy of the block(s) requested. If found, the CSP directs FN_x to the FN in current possession for retrieval of the block(s) and their hashes. Otherwise, the CSP returns the requested data to FN_x, again, if and only if FN_x satisfies the FFSG attributes for data accessibility.

It is also the goal here to compare the time latency between retrieval of encrypted data from a CSP and retrieval from other FNs in same FF. Fog computing undoubtedly reduces the time latency from the far cloud to end-user devices. However, if the FN serving end-user devices in a specific region goes rogue or is down for any reason, the end-user will

not benefit from fog computing services. From this insufficiency the Fog-Federation (FF) and Fog-Federation Sub-Group (FFSG) concepts originated; the latter specifically designed to reduce the impact of a rogue or compromised FN on end-user data security. Experimentation shows that obtaining and decrypting a file from other FNs in the same FFSG requires less time than obtaining it from the CSP. The Multi-Level Approach guarantees data integrity by utilizing the hash concept. Fog computing was introduced to reduce the overhead burden on the cloud and to minimize the time latency between end-users and the cloud. That has been studied and results collected to support the claim.

4.8.3 Experimental Evaluation

The experiments are designed to use multi-threading to create multiple servers and multiple FNs in each FF. Three servers were created as follows: Cloud Service Provider (CSP), Trusted-Authority-Server (TAS), and a Trusted-Fog-Node (TFN). We also created multiple Fog-Nodes (FNs) as clients that can communicate with the TAS and the CSP. Those FNs are divided into Fog-Federation-Subgroups (FFSGs). During the encryption process, the TFN splits the file into three blocks of equal size and then hashes and encrypts each block with the FFSG's attributes.

This approach aims to show the benefit of splitting files into blocks by data sensitivity level and allowing FNs to retrieve only blocks where the required data is housed, thus resulting in cost effective data retrieval. In addition, authorized FNs in each FFSG store the retrieved data for a period of time and provide access to that data to other authorized FNs in the same FFSG. The FN (requestor), sends a request to the CSP. The CSP scans the Encrypted-Files-Tracking-Table (Table. 4.1) to check for an authorized FN in same FFSG with an updated copy of the required block(s), if found, the CSP returns the address of the FN currently in possession of the required block(s). This is achieved with the FFSG

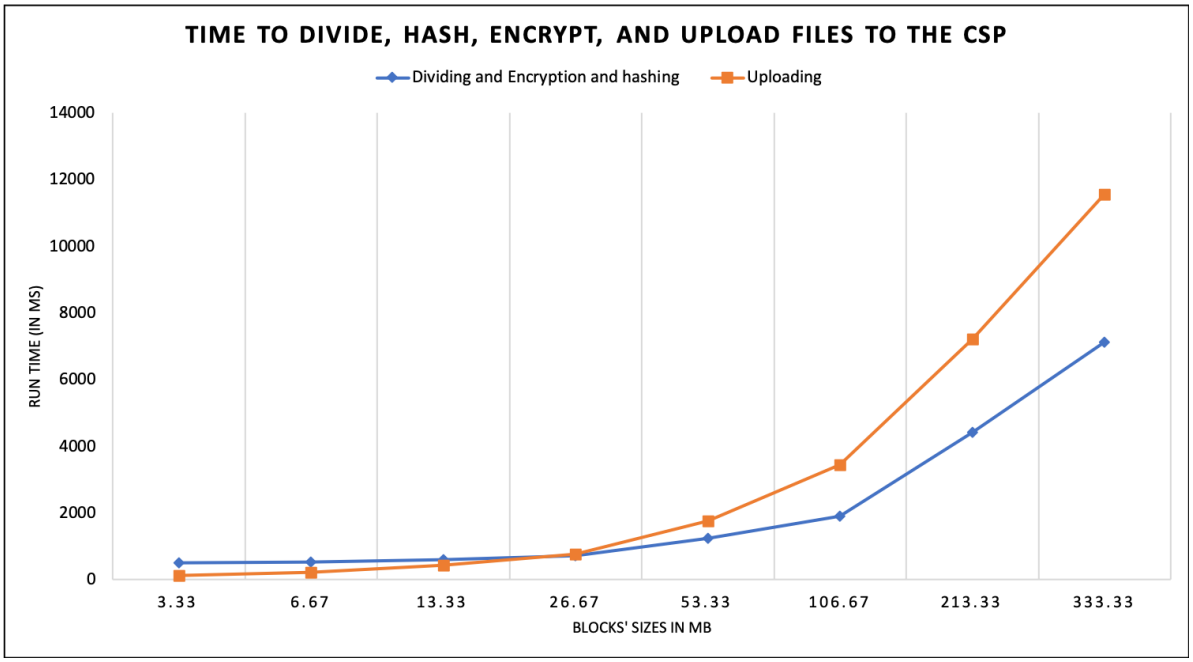


Figure 4.6: Dividing, Encrypting, Hashing of Files with Various Sizes

proposal. This also minimizes the communication overhead and time latency between the CSP and FFs. Next, the performance of our proposed approach is presented through the results of experimentation.

4.8.3.1 Experiment 1: Time Analysis for Files: Dividing, Hashing, Encrypting, and Uploading to a Cloud-Service-Provider by a Trusted-Fog-Node

In this experiment the TFN computation time (execution cost) to divide, encrypt, and upload block(s) to the CSP, is measured. The TFN divides the files disposed for encryption into blocks of the same size; meaning that block-size = file-size/3. The TFN begins calculating the hash of each block, then it encrypts and uploads them to the CSP for storage. The experiment is repeated on input data sets 30 times (See Section 1.8.2.1..... description of data sets). The average is taken to analyze the approach.

The TFN rebounds linear time as blocks less than or equal to 27MB are divided, hashed, and encrypted, show Fig. 4.6. The time grows exponentially when the block size is

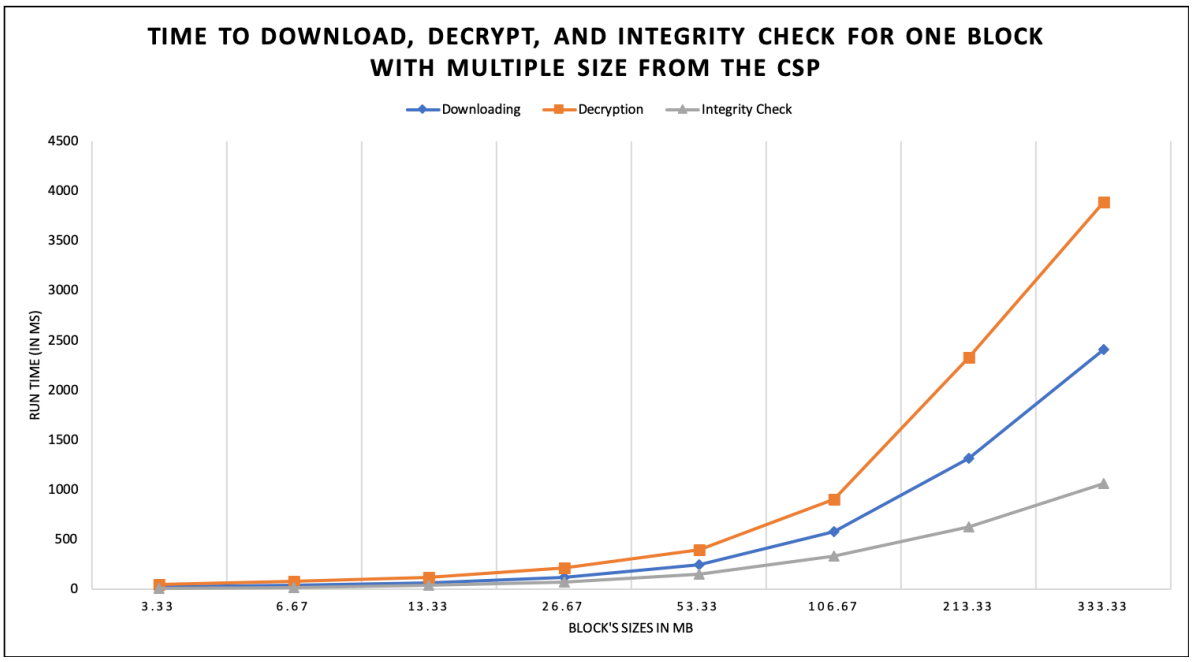


Figure 4.7: Retrieving One Block with Various Sizes from the CSP

greater than 50MB. On the other hand, the TFN uploading of blocks less than or equal to 27MB and their hashes to the CSP grows linearly, while it grows exponentially for blocks greater than 50MB. Fig. 4.6 also indicates that uploading requires less time than dividing, hashing and encrypting blocks less than or equal to 14MB, while they are almost equal for blocks of 26.67MB. The uploading process time is greater than other processes for blocks of 50MB or more.

4.8.3.2 Experiment 2: Time Analysis for File Download, Decryption, and Integrity Check for One Block from the CSP by Authorized FN in FFSG

This experiment evaluates and analyzes the time required for an authorized FN in a FFSG to download, decrypt, and check the integrity of one block of various sizes from the CSP. This experiment assumes that no other FN in the same FFSG has an updated copy of the required block. Suppose an authorized FN in a specific FFSG requests a block from the CSP. The CSP authenticates the FN (requestor) and returns the requested block. The

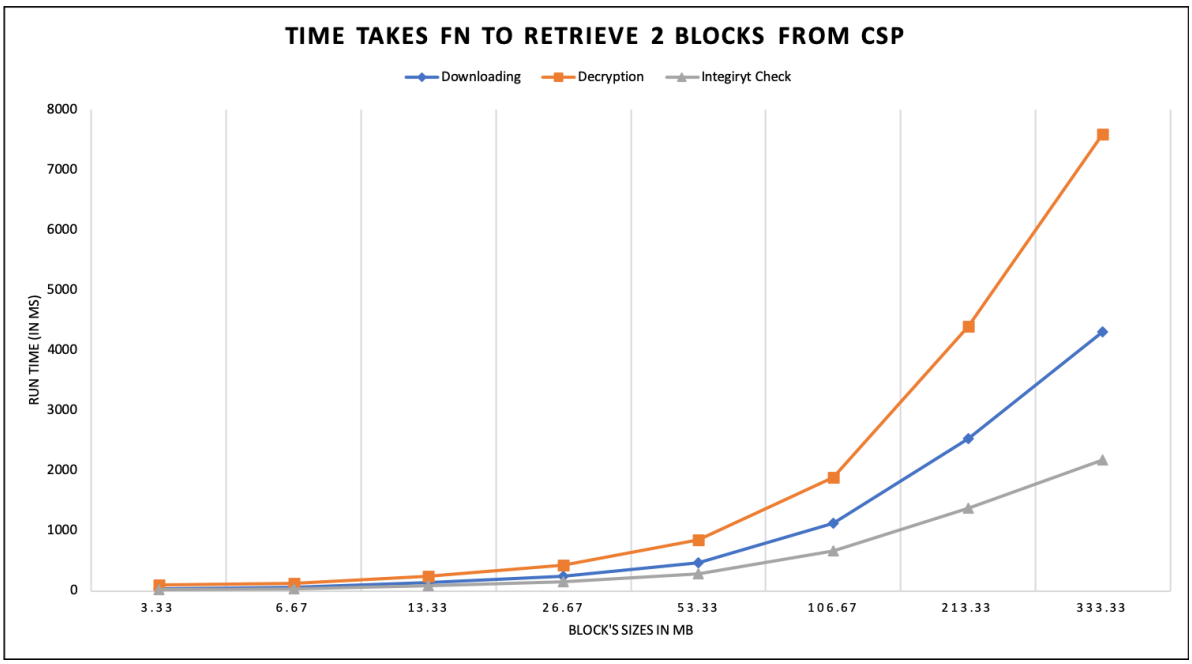


Figure 4.8: Retrieving Two Blocks with Various Sizes from the CSP

experiment is repeated 30 times on the input data sets. An average time to complete the request is determined for use in the evaluation of this model. This experiment demonstrates that the time it takes a FN in a FFSG to download one block from the CSP grows linearly as block size increases from 3MB to 53.33MB, see Fig. 4.7. On the other hand, the time grows exponentially for blocks exceeding 53.33MB. Fig. 4.7 also shows that the decryption time for one block moves almost steadily for blocks between 3.33MB and 13.33M. It grows linearly as block size increases from 13.33MB to 53.33MB, and increases exponentially for blocks surpassing 53.33MB. The integrity check time for the decrypted block increases linearly for blocks from 3.33MB to 26.67MB, while it exponentially increases for blocks larger than 27MB.

4.8.4 Experiment 3: Time Analysis for File Downloads, Decryption, and Integrity Checks for Two-Three Blocks from the CSP by Authorized FN in FFSG

This experiment measures and evaluates the time it takes an authorized FN in a certain FFSG to retrieve, decrypt, and conduct an integrity check on multiple blocks from the CSP. This experiment also assumes that no FN in the FFSG has an updated copy of the required blocks. This means the FN (requestor) has no other option than retrieving the required blocks from the CSP. This retrieval is applicable when a FN needs piece of data that are stored in two or three blocks of an encrypted file. The experiment is repeated 30 times and time is averaged for use in time analysis.

It is observable through this experimentation that the download time for an authorized FN in a specific FFSG to retrieve two and three blocks of an encrypted file from the CSP grows linearly for files less than or equal to 53.33MB per block, while it exponentially surges for blocks greater than 100MB. Furthermore, the time required for a FN to decrypt the retrieved blocks (two or three blocks) from the CSP linearly increases for blocks less than or equal to 53.33MB, while exponentially increasing for blocks greater than 54MB. Verification of block integrity increases linearly for blocks of 53.33MB or less, while it exponentially grows for blocks greater than 54MB, see Fig. 4.8 and Fig. 4.9. Experiments 2 and 3 support our Multi-Level Approach. The experiments validate the efficiency of this approach wherein FNs can only access blocks encrypted to the FFSG attributes they satisfy. This means that a rogue or compromised FN in an FFSG will not breach the entirety of the encrypted file as it would not have access. The Multi-Level Approach is also time efficient as it enables FNs to retrieve block/s of the encrypted file rather than the whole.

4.8.4.1 Experiment 4: Retrieving Block(s) from other FNs in the Same FFSG

In addition to the ability of the Multi-Level Approach to oust rogue FNs by rendering them incapable of accessing encrypted data in the CSP or in other FNs in same FFSG, it efficiently minimizes the communication overhead between the CSP and the FFs. This capability is attributed to a feature that enables the CSP to check the Encrypted-Files-Tracking-Table, (Table. **??**, and search for another FN in same FFSG with an updated copy of the requested block(s). If found; the CSP returns the address(es) of those FNs to the FN (requestor) for local retrieval. This experiment demonstrates the time necessary for a FN in a FFSG to retrieve one, two, or three blocks from the same FFSG. The time is compared against that of retrieving the same blocks from the CSP. Fig. 7.1 illustrates linear growth in FN download time of a variable number of blocks from the same FFSG for blocks equal to or less than 13.33MB, while it exponentially increases for blocks equal to or greater than 26.67MB. It becomes obvious that retrieval time increases as the number of blocks requested increases. Another experiment compares the time required for downloading a variable number of blocks of various sizes from the CSP and from the same FFSG. This experiment demonstrates that retrieval time from the same FFSG outperforms retrieval of the same blocks from the CSP, Fig. 4.11. As indicated, this study has been implemented utilizing variable numbers of blocks of various sizes and the argument appears to hold true.

4.9 Discussion

The performance evaluation in Section 1.8, page .., illustrates this approach's ability to accomplish the security objectives: mainly fine-grained data access control, confidentiality, integrity, and availability. Also, this approach focuses on minimizing the communication overhead and time latency between the Cloud-Service-Provider (CSP) and Fog-Federation-Subgroup (FFSG) members. This is accomplished by enabling the CSP to maintain the

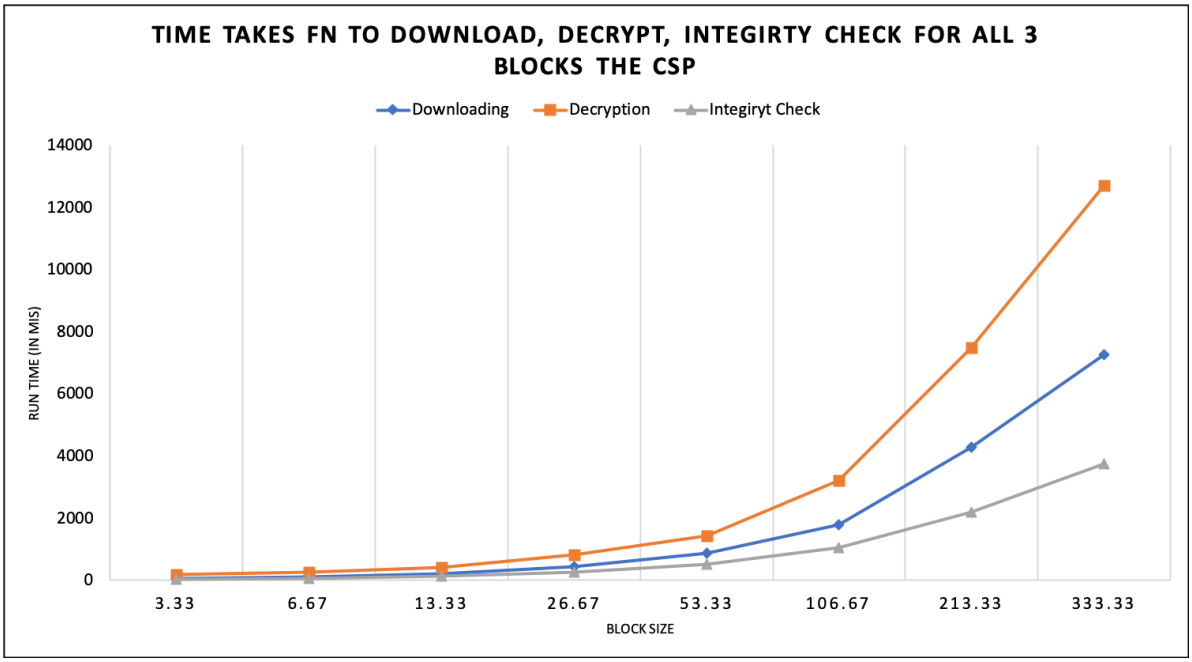


Figure 4.9: Retrieving Three Blocks with Various Sizes from the CSP

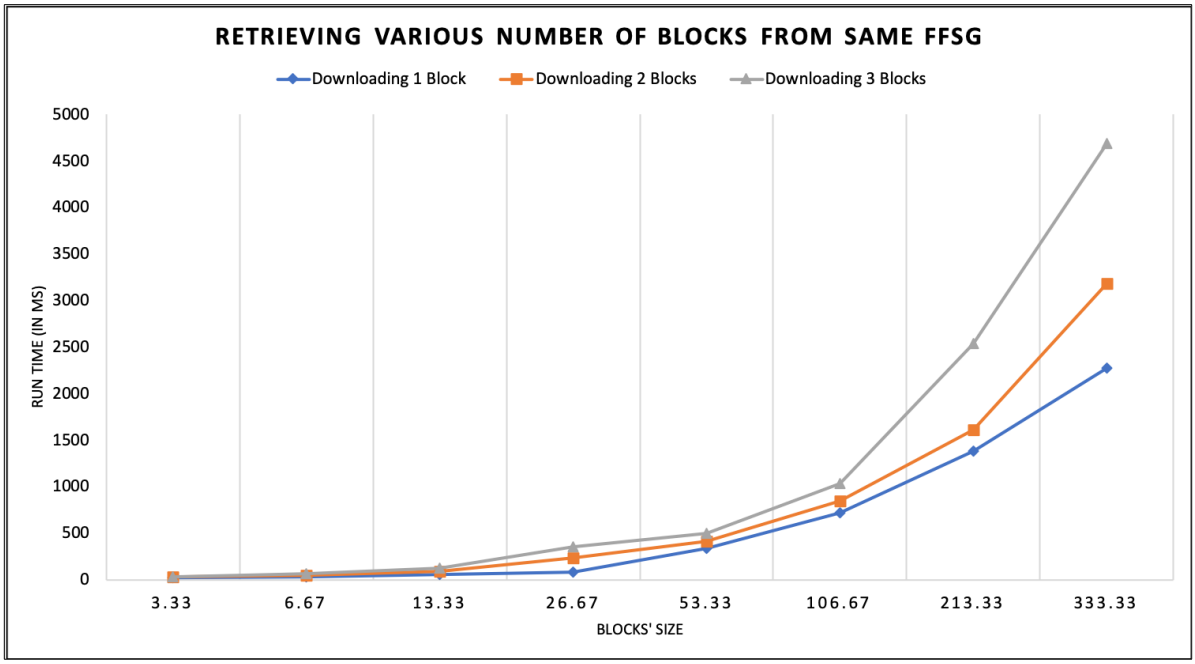


Figure 4.10: Retrieving Different Number of Blocks with Various Sizes from same FFSG

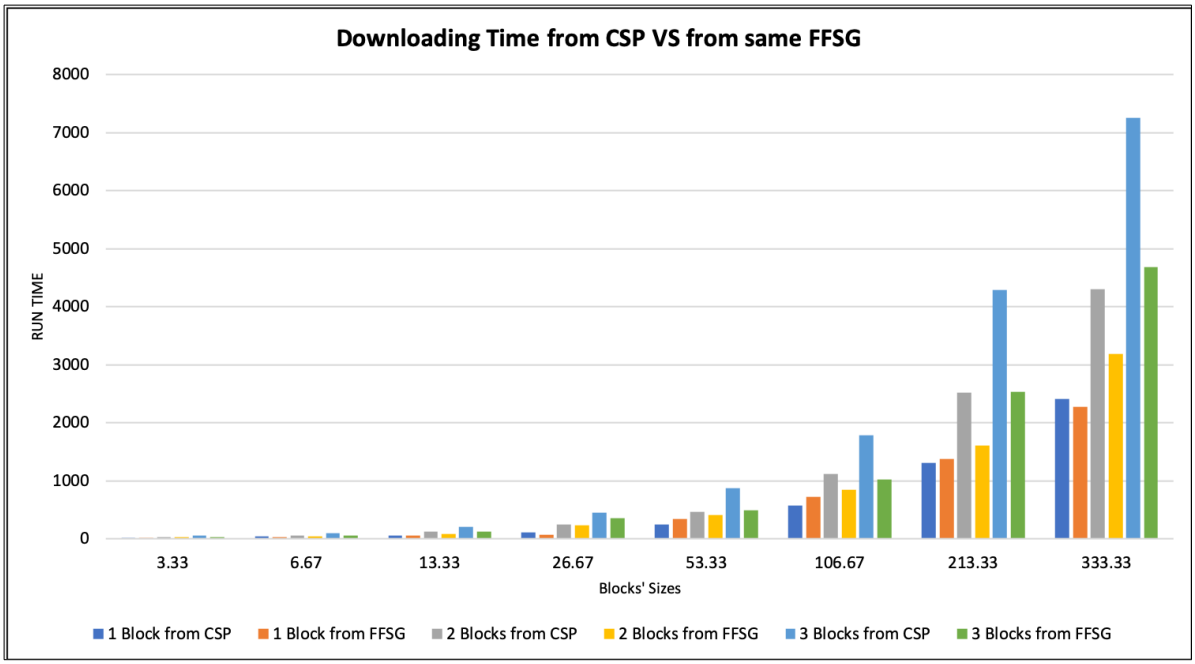


Figure 4.11: Downloading Time from CSP VS from same FFSG

Encrypted-Files-Tracking-Table, Table. **??**. This table helps the CSP to return only blocks in not currently in the possession of another FN in the same FFSG. The proposed approach secures the communication link between the CSP and FF, and among FNs in the FF, by exploiting the hash function mechanism. Thus, the data receivers (FNs) can check the received blocks for damage, meaning that it is resistant to MITM attack. Besides, the proposed approach is secure against rogue FNs; this is accomplished by executing algorithms 9, 10, and 11. Consequently, the ousted FN is incapable of accessing encrypted blocks in either the CSP or other FNs in the same FFSG. The approach assumes that FNs in FFs are not busy; the requests are instantly processed. Rogue FN detection is not considered here as it is outside the scope of this research for now.

4.10 Conclusion

End-user data security (e.g., confidentiality, integrity, availability) can be maliciously violated either by rogue FNs or while in transit between the FNs and the CSP. The Multi-

Level Approach is proposed here as an efficient, secure, and fine-grained data access control, to minimize the amount of data in danger of breach by rogue FNs, and from damage while in transit. The Ciphertext-Policy-Attribute-Based Encryption (CP-ABE) algorithm and the hash-function mechanism are utilized in the design of this approach. The algorithm design and performance evaluation prove that the approach is efficient in minimizing the amount of data in danger of malicious breach by rogue FNs. This is achieved by dividing files to be encrypted into blocks and then encrypting each block to the corresponding FFSG's attributes based on the block level of data sensitivity. The approach classifies the FNs into FFs based on the FF's attributes; mainly by location and services provided by FNs. FNs are further classified to sub-groups called FFSGs based on the access right granted to the FN upon joining the FF. FFSGs are characterized by one or multiple of these attributes: X = Less-Sensitive Data (LSD), Y = Mid-Sensitive Data (MIDSD), and Z = Most-Sensitive Data (MAXSD). This means each FN can only access the blocks encrypted to the FFSG attributes in which the FN resides. Each FN can belong to multiple FFSGs depending on the access-rights granted to it by the Trusted-Authority-Server (TAS). The performance evaluation expresses the efficiency and applicability of this approach.

5 A Blockchain-Encryption-Based Approach

In this chapter, we developed a Blockchain-Encryption-Based approach to protect fog federations from rogue fog nodes. Most sections that are shown in this chapter has been published in [94] verbatim.

5.1 Introduction

Cloud computing is a thriving paradigm due to the enormous on-demand services it provides to end users over the internet. Cloud computing has provided customers with innovative features such as availability, scalability, and economy that help to satisfy the substantial demand for storage and computation resources [6]. End users outsource their data, to the core network on the cloud, for processing and storage. However, there are many obstacles facing data owners. First, the response time between users and the cloud is high because the data is stored in far from the data owners. Second, end-users' data security and privacy are susceptible to violation because of the semi-trusted third party controls the cloud. The research community has studied the issues of data security and privacy in cloud computing by adopting and applying advanced cryptographic techniques. However, it is still demanding to invent a new technology to resolve the cloud latency issue [9][15].

From here, fog computing was introduced in 2012 [10] to reduce the time latency between the cloud and end users' devices [9][15] and to provide new services and applications to end users. Fog computing paradigm is as small clouds close to end-user's devices, which allows customers to utilize the computing and storage services at the edge of the network [10]. More specifically, fog computing comprises of multiple fog nodes which provide services to end users [16]. The fog computing paradigm has supported end devices with distinguish-

ing features such as mobility, location awareness, and low time latency [18]. Because fog computing is deemed as an extension of the cloud computing paradigm; therefore, it inherits some of the security and privacy obstacles in cloud computing[13]. In particular, a fog computing paradigm is susceptible to different threats, such as malicious attacks and technical issues. In this context, the end users' data traveling through fog computing nodes to the cloud is vulnerable to different violations. Therefore, it is necessary to design a approach to protect end-users' data confidentiality, availability, and integrity by ousting the rogue (malicious) fog nodes. One reason for these vulnerabilities is that end users outsource sensitive data to the nearby fog node for processing and then to the cloud for further processing and storage. Because this fog node is seen as a connector between end users and the cloud, it is challenging to protect other fog nodes and the cloud from malicious attacks. End users' data security would be difficult to defend if the fog node providing services in terms of storage and computation is compromised or goes rogue. We have been motivated by those issues to protect customers' private data from being compromised. Also, a method to secure communication among fog nodes is required to exchange encrypted data for reducing the time latency to retrieve it from the far cloud.

In general, threats in the context of fog computing takes two forms. 1) data modification: if an adversary gets hands-on end users' private data, he/she might violate its confidentiality and integrity. Therefore, introducing a security approach is necessary to prevent data confidentiality and integrity violation among fog nodes; and between fog nodes and the cloud. 2) unauthorized access: when an adversary compromises a fog node, he/she can get unauthorized access to the end users' private data. Besides, data availability is in danger of malicious violation in this way. Therefore, it is essential to introduce a security approach to protect against rogue (malicious) fog nodes and to oust them off the network while maintaining low time latency feature fog computing provides. Additionally, it is vital

to enabling the fog nodes to carry on a distributed authorization process and to communicate in a trust-less environment.

Attribute-Based Encryption (ABE) is a cryptographic primitive that enables one-to-many encryption. It comprises two types: ciphertext policy attribute-based encryption (CP-ABE) [22] and key policy attribute-based encryption (KP-ABE) [25]. CP-ABE was introduced by Bethencourt at el. [22] which is deemed as one of the most significant algorithms to provide fine-grained data access control.

Blockchain has attracted the attention of researchers since it was introduced in late 2008 [95]. Blockchain is a shared distributed ledger, which is secured, immutable, and transparent, that helps in processing and tracking resources without depending on a centralized trusted third party [29]. With this promising technology, peer-to-peer nodes can communicate, and exchange resources where the decision is carried on in a distributed manner by the majority of the network's nodes rather than a single centralized trusted third party.

This chapter proposes a approach to oust rogue (malicious) fog nodes and to minimize the communication overhead between the cloud service provider (CSP) and fog nodes (FNs) by integrating the CP-ABE algorithm and blockchain technology. Blockchain is adopted as a medium to store on-chain tracking table to verify the identity of each fog node using the smart contract before they could access the encrypted data on the CSP. The blockchain immutability feature prevents fog nodes from maliciously changing the on-chain tracking table, if such a change detected, the FN which issues the request is reported as a rogue fog node.

The rest of this chapter is organized as: Section (II) presents the related work. Section (III) focuses on the motivation of this chapter. Section (IV) explains the proposed approach, and the approach description is broadly explained in section (V). Finally, the security analysis and the conclusion consecutively are in sections (VI) and (VII).

5.2 Idea Behind using Blockchain as Fog Federation

In this chapter, we exploited the Ethereum blockchain smart contract as intermediary among Fog-Nodes (FNs) that belong to the same Fog-Federation (FF). We constructed each FF as a private blockchain in which the FF's members can perform an authorization process in distributed manner using the Smart Contract (SC) concept. In addition, this approach deploys the On-chain Tracking Table (see Table 5.1) which has to be protected from being tampered with by malicious/rogue FNs. This is attributed to the tamper proof nature of blockchain technology. Therefore, that was the main reason for exploiting the concept of blockchain technology. Also, this table is used as an access control table smart contract uses to verify the FN (requestor) identity in order to return the stored encrypted file credentials to the FN to be able to retrieve the file from the IPFS.

Our approach allows the Fog-Node that originates the encrypted data to encrypt it only to the other FNs in same FF. This approach precisely detects and blocks malicious/rogue FNs from retrieving encrypted data either form the Shared-Storage (SS) or from other FF's member (FN).

This approach must efficiently achieve the approach security objectives in the sense that the consumption gas takes smart contract to perform approach's functions should be affordable to FNs. One of the most important feature of this approach is to detect and oust the rogue fog nodes which would positively improve the security of encrypted data stored in the Shared-Storage.

5.3 A Blockchain-Encryption-Based Approach Objectives

5.3.1 Confidentiality

The Blockchain-Encryption-Based Approach provides data confidentiality to the data stored in the Shared-Storage (SS). This approach integrates the Ciphertext-Attribute-Based-Encryption (CP-ABE) algorithm with the Ethereum blockchain smart contract to reach the maximum level of protecting the confidential data. This approach assures data confidentiality by enabling the Fog-Node (FN) to encrypt data to the Fog-Federation (FF) it belongs to, and in the meantime, no FN can access this encrypted data unless being proofed by the smart contract. The FN (data encryptor) generates a random secret key to encrypt the file. Then, it uploads the Encrypted-File (EF) to the SS for storage. The SS returns the index-hash of the file to the FN (file owner). Then, the FN encrypts the SK using the Fog-Federation-Public-Key (FFPK), and after that it uploads the encrypted SK and the index-hash (e.g. File-ID) to the smart contract of the FF. This approach would not allow the FNs to download the Encrypted-File (EF) from the SS (e.g. IPFS in this approach) unless being verified by the smart contract to get the EF's hash-index to be used to download the EF. In addition, when the SC verifies the FN (requestor) as legitimate, it would return the EF's SK that would be used by the FN for decryption process upon downloading the EF from the IPFS. This means use of Ethereum Blockcahin Smart contract concept boosts data confidentiality in this approach as it protects the On-chain Tracking-Table from being tampred and acting as a protection wall against rogue FNs from access encrypted data stored in the IPFS.

5.3.2 Integrity

In addition to providing the confidentiality to the stored data, this approach maintains the integrity of the data stored in the SS. Fog-Nodes (FNs) calculate the hash of the files

prior to the encryption process, then they post the encrypted files' hashes to the Smart Contract (SC) along with the index-hashes returned by the SS (e.g. IPFS storage). When an authorized FN retrieves an Encrypted-File (EF) from the SS using the hash-index, it is able to check the EF integrity post decryption process. The mismatched hash(es) will verify the damaged of the EF, and accordingly the FN will post that to the SC to make it noticeable to other FNs in the same FF that this EF has been somehow modified. The SC would mark this EF as damaged.

5.3.3 Availability

Fog computing emerged with important features to be added to the current computing paradigm. Fog computing is considered as middle level between cloud computing and end-users' devices such as IoT and mobile devices because it filters and pre-process data before being outsourced to the cloud[]. It is one of the Fog-Federation (FF) concept motivation to keep data available to end-users even though the Fog-Node providing services to end-devices in a specific location is down or ousted for what so ever reason. The end-users still could communicate with other FNs in same FF to get the same services provided with the ousted FN. It is undoubted that the Fog-Federation concept will greatly improve the data and nodes management in the fog computing paradigm.

5.4 Proposed Approach and Assumptions

5.4.1 Overview of the Blockchain-Encrypted-Based Approach

In this section, we describe the architecture of the proposed Blockchain-Encrypted-Based Approach based on smart contracts. By referring to Fig. 5.1, we first describe the entities comprise our approach. This approach comprises of the following entities:

- Shared-Storage (SS): In this approach, we used the InterPlanetary File System (IPFS)

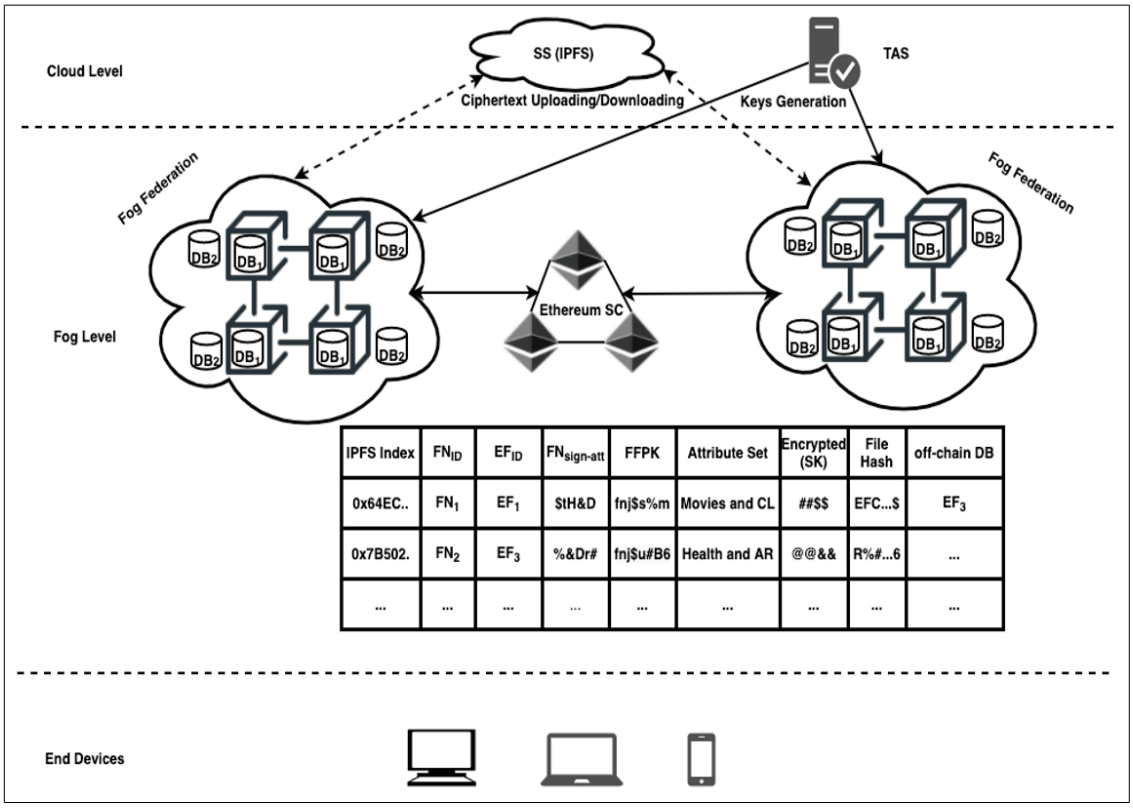
as a Shared-Storage (SS) to store the Encrypted-Files (EFs) uploaded by the FNs. The IPFS returns the index-hash upon receiving the uploaded files.

- Fog-Nodes (FNs): FNs are the entities that provides services to end-users in the fog computing paradigm. FNs with same attributes form a Fog Federation (FF) which is considered as a private blockchain. This means that if the Fog Node (FN) providing services to end users goes down/rogue, the end user can still obtains the same services from other FNs in the same FF. FNs in same FF can verify each other by invoking the smart contact, i.e. smart contract can check whether FN requesting an access to an encrypted file posses the FF's legitimate attributes or not.

- Trusted-Authority-Server (TAS): This sever is assumed to be fully trusted in this approach. We used it to set up the system and to generate the required keys to operate the system. It is responsible of posting FFPK and FF's attributes to the smart contract. It is a pre-verified authorized node on the Ethereum blockchain. It assigns attributes to FFs, and adding new FNs to FF.

- Smart Contract: Ethereum blockchain smart contract was exploited to enforce trust among FNs in smae FF. Also it is used to track the On-Chain-Tracking-Table to prevent any unauthorized access to the encrypted data stored in the SS. It functions as miners to validate the requests generated by FN in same FF. Through this chapter, we use fog federation and private blockchain interchangeably.

Data delivery from cloud storage (e.g. SS in this approach) to end-users' devices is hindered by time latency and communication overhead. Fog computing has emerged to address these issues. However, FNs are exposed to malicious attacks; consequently, the end users' data security may be violated. To maintain confidentiality, integrity and availability for end-users' data travelling through FNs to the SS, we exploit and integrate blockchain



IPFS Index	FN_{ID}	EF_{ID}	$FN_{sign-att}$	FFPK	Attribute Set	Encrypted (SK)	File Hash	off-chain DB
0x64EC..	FN_1	EF_1	$tH&D	fnj$s%m	Movies and CL	##$$	EFC...$	EF_3
0x7B502.	FN_2	EF_3	%&Dr#	fnj$u#B6	Health and AR	@@&&	R%#...6	...
...	...	...	...	...	...	...	...	...

Figure 5.1: The system model.

with the CP-ABE algorithm, an advanced cryptographic primitive that enforces fine-grained data access control to secure communication between FNs and the CSP. Using blockchain, FNs can perform the authorization process in a distributed manner. To solve the problem of high communication overhead, we deploy an access control policies table (on-chain files tracking table) on Ethereum smart contract ,as shown in Table 5.1, which must be protected from tampering conducted by malicious (rogue) FNs. Blockchain plays a vital role here, as it is tamper-proof by nature.

In this approach, after a TAS validates a FN for possessing a set attributes, the TAS then will issue an ID for the FN and assign it to the corresponding FF based on its attributes, namely location and services. This validation is carried out in the context of smart contract.

Each FN in the same FF maintains an on-chain files tracking table to provide fine-grained access control by using a smart contract. This allows other FNs in the same FF to

access the encrypted data on the IPFS when their attributes satisfy the stored predicates in the on-chain files' tracking table.

When a FN sends a query to the SC to access a certain EF, the SC checks whether any FN in the FF has the required file stored in its off-chain database (DB). If so, it refers the requestor to get the file from the FN currently in possession of the file instead of sending the request to the IPFS. When the FN (data requester) requests an EF from another FN in the same FF, the receiver would verify the requestor legitimacy using the smart contract. This feature is a benefit of this approach, as it minimizes the communication overhead between the Shared Storage (SS) and the FF. When the FN obtains the index-hash of the EF stored in the IPFS, it could retrieve the desired encrypted file.

In fog computing, FNs collect data from end devices. They process those data and upload the results to the cloud for further processing and storage. We equip every FN with two databases (DBs), an off-chain and an on-chain DB. The off-chain DB stores the encrypted data most frequently accessed by end-users, which reduces the communication overhead between the SS and the FFs. FN can retrieve data encrypted by FNs in the same FF. This feature helps maintaining data availability when ousting an FN from the FF near to end-users' devices. Also the end-users would still obtain same services provided by the FN close to them, if this FN got ousted, from another FN in the FF. The on-chain database is considered as digital ledger and stored on the blockchain. It stores an on-chain file tracking table as shown in Table-5.1 to verify FF's members identifications in context of smart contract. For the sake of clarity, we will consider the following example:

Suppose FN_1 encrypts file F_1 using a random secret key (SK). Next, it uploads the encrypted file (EF_1) to the IPFS and obtain a index-hash from IPFS. After that the FN_1 query the smart contract to obtain the FFPK to use it in encrypting the SK. Then, FN_1 post the index-hash, encrypted SK, and the file hash to the Ethereum blockchain smart contract.

Now, suppose FN_2 which is in same FF as FN_1 seeks to retrieve an EF_1, there are three scenarios we have studied in this chapter:

- First scenario: FN_2 intends to retrieve the EF_1 in which no other FN in same FF has an updated copy of the file in its off-chain DB. In this case, FN_1 will query the smart contract in order to obtain the index-hash, file hash and the encrypted SK. If FN_1 is successfully verified by the smart contact, it will receive the index-hash, file hash and the encrypted SK of the desired EF . Next, FN_2 decrypts the SK using its own FNSK. Next, it retrieves the encrypted file from the shared storage using the index-hash.

- Second Scenario: FN_2 seeks retrieving EF_1 in which FN_3 in same FF has updated copy of the file. In this case, FN_2 queries the SC to retrieve EF_1. SC returns the required file credentials to FN_2 plus FN_3's address. Further, FN_2 will communicate FN_3 to get EF_1. FN_3 will validate FN_2 identity by invoking the smart contract. If the smart contract returns that FN_2 is a member of the same FF, FN_3 will return the file to FN_2. This feature gives our approach a credit in minimizing the communication overhead between the SS and the FFs. This is possible with the fog federation idea. For more details, read the approach's description.

- Third scenario: FN_2 is not verified by the smart contract, so it will not pass the authorization process to get enough information to retrieve the encrypted data from the SS. Accordingly, the FN_2 (data requestor) will be reported as a rogue FN and deleted from the FF.

An FN leaves the system for whatsoever reason, or it could go rogue after being compromised by a malicious attack. To ensure users' data security, namely data confidentiality, data integrity and availability, there has been a demand to take precautions against rogue FNs. This approach efficiently acts as a protection layer against rogue FNs to prevent them

from accessing the encrypted data stored on the SS. On-chain file tracking table is stored on the Ethereum blockchain. Any FN query the blockchain with false information will be

Table 5.1: On-Chain Tracking Table

IPFS Index	**FN**$_{ID}$	**EF**$_{ID}$	**FN**$_{sign-att}$	**FFPK**	**Attributes Set**	**Encrypted(SK)**	**File Hash**	**off-chain DB**
0x64EC...	FN_1	EF_1	6*R!@MN	fNJk3u	Movies and CL	!!@3DSx	*EFC!...	EF_3
0x7B502...	FN_2	EF_3	5*R*!WE	KLJk3J@	Health and AR	@@!!NB4	MNC!...	0
	...	...	...	...	...	...	...	...
07x0B62...	FN_n	EF_7	9TR!!NQ	@3EFJJL@	Education and AR	ms!!uB*	!67jK*...	0

detected based on the data pre-stored on the tracking table. We utilize the blockchain in this approach to track the FNs in each FF and prevent unauthorized and ousted FNs from accessing the data stored in the SS.

If an FN issues an update and it is not passed by most of the SC, the issuer would be considered as a rogue FN and then excluded from the FF. When a rogue FN is detected in the FF, it is decoupled from the FF. Accordingly, this feature enhances the approach's ability to protect against rogue FNs.

5.4.2 Threat Model

5.4.2.1 Rogue Fog Nodes

In this approach, we define rogue fog node as: a fog-node that retrieves encrypted data stored in the Shared-Storage (SS) for malicious intention. It also could , maliciously modifies the encrypted data. These incidents would violate the security of the end-users' data stored in the Shared-Storage (SS). For instance, when a rogue FN becomes able to download encrypted data from the SS or from other FF's members, the data confidentiality and data integrity of enc-users' secret data would be compromised.

5.4.2.2 Man-in-the-Middle (MITM) Attack

In our approach, the MIMT attack is defined as a third party intercepts the encrypted data travelling between the FNs and the SS, or among the FNs in same FF to maliciously violate credential data. If this malicious third party manages to modify the travelled data, FNs would not be able to detect this modification unless counting on a robust method to prevent and detect that. In addition, if MITM attacker manages to intercept the encrypted data, that would cause the end-users' data confidentiality to be violated.

5.4.2.3 Curious Cloud Storage Service Provider

In any developed system meant to be secure, it is not a smart way to consider the third party providing the storage service fully trusted. End-users care about their outsourced confidential data to the fog nodes to be protected while being stored in the Shared-Storage (SS). If the SS service provider discovers the uploaded encrypted-data, that would violate the data confidentiality.

5.5 Design of the A Blockchain-Encryption-Based Approach Algorithms

In this section, we present the description of our approach based on the integration of the CP-ABE algorithm and Ethereum blockchain smart contract. We exploited the access tree model presented in [22] as an access structure A, as shown in Fig-5.2. To satisfy the access tree conditions, we follow the same methods in [22][20]. For more information about the access tree model, read through [22][20]. Next we provide algorithms and smart contracts needed for our model. Algorithms 5.1, 5.2, 5.3 and 5.4 are implemented on external environment from the Ethereum blockchain. The rest of the algorithms in this approach are smart contracts we implemented in solidity.

Algorithm 5.1 is based on [22]. It is executed by TAS to start the system. It receives

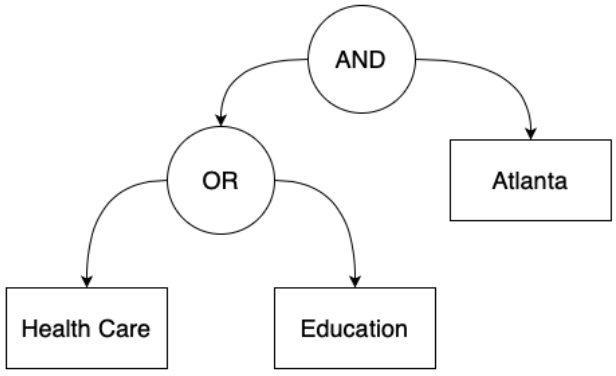


Figure 5.2: Example of an Access Structure.

a security parameter as an input, then it outputs a System Master Key SMK and a System Public Key SPK. These keys are used to generate keys for FFs and for FNs. It builds a universal set of attributes (USAT), which FF classification is based on in our approach; USAT = FN's location, FN's services. Algorithm 5.2 is executed by an FN that aims to join the system to provide a service for end-users. It contacts the TAS, then the TAS checks the pre-stored Fog-Federation-List (FFL). The FFL is a list of FFs along with FNs in them. The TAN accordingly assigns the FN to the corresponding FF if its attributes are verified. If the FN is verified but no FF_{att} matches its attributes, the TAS creates a new FF and adds the new FN to it. After that, the TAS invokes (adding FN to the FF smart contract) to add the new FN's credentials to the smart contract. However, if the FN is not verified, the connection will be refused. The details of how to verify and authenticate the FN's attributes are out of our scope for now.

Algorithm 5.3 is also executed by the TAS. It generates a Fog Federation-Public-Key (FFPK) and a Fog-Federation-Master-Key (FFMK). The FFPK is public to all FNs in the FF, whereas the FFMK is private to the TAS. It also generates a distinct Fog-Node-Secret-Key (FNSK) for each FN in the FF. Then, the TAS securely shares the FFPK and FNSK with the new FN. TAS posts the FFPK to the smart contract and map the FFPK to FF's attributes. This would be accomplished by invoking the Fog-Federation-Creation Smart Contract.

Algorithm 5.1 Setup(λ)

1: Choose bilinear cyclic group G_1 of prime order p with generator g_1;
2: Randomly generates exponents $\alpha, \beta \in G_p^*$;
3: Random Oracle H: $USAT \rightarrow \{0,1\}^*$ // to map each att in USAT into random element $\in G_1$;
4: compute SMK and SPK as:
5: $SMK = \{\beta, g_1^{\alpha}\}$
6: $SPK = \{G_1, g_1, g_1^{\beta}, e(g_1, g_1)^{\alpha}\}$;
7: The SPK is public to all FNs, but SMK is private for TAS

Algorithm 5.4 is executed by authorized FNs in a given FF. This algorithm was executed separately from the Ethereum smart contract as the current Ethereum platform does not support ABE algorithms. The FN intending to encrypt a file generates a Secret-Key (SK) and uses it to encrypt data. Next, FN encrypts the SK using the FFPK to enable every FN in the same FF to decrypt the encrypted SK when needing to retrieve encrypted file from the SS. Finally, the FN (data owner) upload the Encrypted-File's (EF) to the Shared-Storage (SS) which we use here the IPFS. Then the data owner upload the file hash, encrypted file index-hash returned by the IPFS, and the encrypted SK to the smart contract by invoking adding-File and Encrypted Secret-Key(SK) smart contract.

Algorithm 5.5 This procedure is executed by the Trusted-Authority-Server (TAS) to deploy the smart contract to the Ethereum blockchain. The deployment process returns the contract address in which the Fog-Nodes in Fog-Federations contact to be served. The TAS sends the contract-address to every new FN upon joining the Fog-Federations groups. The basic function of this smart contract is to keep tracking which FNs have which files in the off-chain DB. This means less communication overhead between the Shared-Storage (SS) and the FFs. No FN can retrieve the EF or perform any operation unless it gets authorized by the smart contract. This is attributed to the blockchain tamper-proof feature. When

Algorithm 5.2 Join (FN_{att})

New FN contacts TAS to join the system
TAN verifies the FN (requestor)
if not verified **then**
    Declined
else
    Check the FFL
    if FF_{att} == new-FN_{att} **then**
        Assign new-FN_{att} to the corresponding FF_{ID}, and assign ID to the new FN
        TAN posts the FN_{ID}, and attributes by invoking adding FN to the FF Smart Contract
    end if
end if

Algorithm 5.3 Generating $FNSK\ (SPK, SMK, S, FN_{ID})$

TAS gets request from new FN to join the system
Choose bilinear cyclic group G_1 of prime order p with generator g_1;
Randomly generates exponents $\alpha_1, \beta_1 \in G_p^*$;
Random Oracle H: $USAT \rightarrow \{0,1\}^*$ // to map each att in USAT into random element $\in G_1$;
compute $FFPK$ and $FFMK$ as:
$FFPK = \{\beta_1, g_1^{\alpha_1}\}$
$FFMK = \{G_1, g_1, g_1^{\beta_1}, e(g_1, g_1)^{\alpha_1}\}$
post the FFPK to the smart contract by invoking addFogFederationMember() algorithm in smart contract
generate random $r \in Z_p^*$
for every $att_i \in S$ **do**
    generate random $r_i \in Z_P^*$
end for
for each new $\text{FN}_i \in FF$ **do**
    compute $FNSK$ as follows
    $FNSK = (D = g^{(\alpha_1+r)/\beta_1}, \forall i \in USA : D_i = g^r.H(FN_{ID})^{\alpha_{1ver}}.H(i)^{r_i}, D_i{}' = g^{r_i}.H(FN_{ID})^{\alpha_{1ver}})$
    send $FNSK$ to FN_i
end for

an FN intends to retrieve an encrypted file from the Shared-Storage (SS) or from another FN in same FF, FNs could authenticate the FN (requestor) using the smart contract. This approach prevents FNs in the FF from falsifying information about files' tracking table or accessing the EF in the CSP. This feature strengthen our approach robustness.

Algorithm 5.6 is a smart contract created by the Trusted-Authority-Server (TAS) and deployed on Ethereum. TAS uses this part of the smart contract to issue Fog-Federations (FFs) and map these FFs to FFPKs and $\text{FF}_{attributes}$. It takes as inputs the Fog-Federation-Public-Key (FFPK), $\text{FF}_{attributes}$ (e.g. Location and services), members (e.g. FNs to be added to the FF).

Algorithm 5.7 is a smart contract to add FNs to the FF and deployed on Ethereum. It is invoked upon new FN joining the FF group. This smart contract maps the new FN to the FF to the FFPK, and FF's attributes. This smart contract also map the FN to the

Algorithm 5.4 Data Encryption (M, T, SPK, $FFPK$)

SK = gen.DEC-Key
Encrypted-File (EF) = Encrypt (M, SK) //M is the message to encrypt
A is set of atts represented by monotone access structure tree T
for each node x in T **do**
  set a polynomial
  if node x is root node in T **then**
    set $q_{(0)R}$ = s // s is random value $\in Z_P^*$
  else
    $q(0)_x = q_{parent(x)}(index(x))$
    then choose $d_x = k_x$ - 1 as polynomial degree to interpolate with q_x
  end if
end for
for each y $\in$ Y **do** //Y is the set of leaf nodes in T
  $C_y = g^{q_y(0)}$
  $C_y^{'} = H(att(y))^{q_{y(0)}}$
end for
Compute: $C = g^{\beta s}$
for each $SK_i \in SK$ **do**
  Compute $C_{ver_i} = SK_i.e(g,g)^{\alpha s}$
end for
Upload [EF] to the Shared Storage (IPFS)
post the encrypted-File-Hash, index-hash and the encrypted SK to smart contract by invoking addFile() function in the smart contract

Algorithm 5.5 Smart Contract Deployment

TAS deploy the smart contract into the Ethereum blockchain
    return $\text{contract}_{address}$
TAN securely save the $\text{contract}_{addrees}$ to be distributed to the FF members

encrypted files to the FF this FN belongs to. The FN accordingly can retrieve the encrypted files credentials stored on the smart contract. FNs that are not part of the smart contract cannot access the encrypted data stored in the Shared-Storage (SS). We developed a smart contract to enable FNs in the same FF of adding encrypted files credentials and encrypted secret key (SK) to the smart contract as shown in Algorithm 5.8. This smart contract allow FNs to add encrypted file's credentials to the smart contract as follow: the file hash, index-hash and encrypted (SK). The smart contract verifies whether the FN's (requestor) credentials matches the FF's credentials or not. The FN is incapable of adding encrypted

Algorithm 5.6 Fog Federation Creation Smart Contract

```
1: struct FogFederation{
2:       string FFPK;
3:       string attribute1;
4:       string attribute2;
5:       string[ ] members;
6:       string[ ] fogFederationFiles;
7:       }
8: mapping(string ⇒ string) public fogNodeFFPK; // map FN to FFPK , FN ⇒ FFPK
9: mapping(string ⇒ bool) public setFFPK; // check for uniqueness of FFPK, if FFPK1
   is given then set to true,
10: string[] fogFederationsList; // list of fog federations
11: mapping(string ⇒ FogFederation) public mapFFPK; // FFPK ⇒ FF
12: event Fog-Federation-Event(string FFPK, string attribute1, string attribute2, string[]
   members, string[ ] files);
13: function createFogFederation (string ffpk, string att1, string att2) public returns(string
   FFPK, string attribute1, string attribute2, string[] members, string[] files){
14:           require(msg.sender == TAN_address, "You are not allowed");
15:           require(setFFPK[ffpk] == false keccak256(abi.encodePacked
   (mapFFPK[ffpk].FFPK)) ==keccak256(abi.encodePacked("")), "FFPK already have at-
   tributes");
16:           setFFPK[ffpk] = true;
17:           mapFFPK[ffpk].FFPK = ffpk;
18:           mapFFPK[ffpk].attribute1 = att1;
19:           mapFFPK[ffpk].attribute2 = att2;
20:           fogFederationsList.push(ffpk);
21:           emit GroupEvent(ffpk,mapFFPK[ffpk].attribute1,mapFFPK[ffpk].attribute2,
   mapFFPK[ffpk].members,mapFFPK[ffpk].fogFederationFiles);
22:           return(ffpk, mapFFPK[ffpk].attribute1, mapFFPK[ffpk].attribute2,
   mapFFPK[ffpk].members, mapFFPK[ffpk].groupFiles);
```

files credentials to the smart contract in case the FN credentials do not match the FF's credentials. In addition, this smart contract connects the encrypted file index-hash in the Shared-Storage (SS), encrypted (SK) and the encrypted file's hash with the Fog-Federation's Attributes.

Algorithm 5.9 introduces the File-Retrieving smart contract. When an FN seeks retrieving an encrypted-file from the Shared-Storage (SS). The FN starts by invoking the File-Retrieving smart contract. This approach has introduced two scenarios for encrypted

data retrieving. First scenario: an FN intends to retrieve an encrypted file while there is another FN in the same FF has an updated copy of the encrypted file. The FN (requestor) sends a request to the SC to get the EF cridentials. The SC will return the FN currently in possession of the encrypted file. Then, the FN queries the EF from the returned FN, next the request receiver verifies the data requestor by executing the smart contract. If the smart contract returns that the FN (requestor) is a legitimate member of the FF, the FN(receiver) will sends the encrypted file to the data requestor. Second scenario: the encrypted-file is in the SS. The FN (requestor) invokes the smart contract to retrieve the following encrypted-file credentials: file Hash, file index-hash, and the encrypted SK. The smart contract verifies whether the FN (requestor) identity is authentic or not. If the FN becomes authorized, the smart contract will return the corresponding file Hash, file index-hash, and the encrypted SK of the encrypted file.

One of the important contribution of this research is to prevent rogue FNs from accessing the encrypted data stored either in the SS or in other FN in the same FF. Algorithm 5.10 shows the smart contract to remove the rogue FN from the FF which it can not retrieve the encrypted file credentials from the smart contract.

5.6 Performance Evaluation and Security Analysis

In this section, the experiment design and the experimental evaluation are provided. We evaluate the cost of the smart contracts creation followed by the execution cost of their functions to achieve the blockchain-encrypted-based approach proposed objectives. In particular, we exploit Ethereum's smart contract technology to oust and prevent the rogue FNs from accessing encrypted data stored in the Shared-Storage (SS) or in other FN's off-chain storage. We start with providing a security analysis of our approach followed by the implementation details.

Algorithm 5.7 adding FN to the FF Smart Contract

event FFmemberEvent(stirng FFPK, string attribute1, string attribute2, string[ ] members, string[ ] files)
function addFN (string memory FFPK, string memory FN_{ID}) public returns(string memory FFPK, string memory attribute1, string memory attribute2, string[ ] memory members, string[ ] memory files) {
    require(msg.sender == $TAN_{address}$, "You are not allowed");
    require(setFFPK[ffpk] == true keccak256(abi.encodePacked(mapFFPK[ffpk].ffpk)) == keccak256(abi.encodePacked(ffpk)), "FFPK not available");
    require(keccak256(abi.encodePacked(memberFFPK[FNid]))== keccak256(abi.encodePacked("")), "Member already added in FF");
    mapFFPK[ffpk].members.push(FNId);
    memberFFPK[FNid] = FFPK;
    emit MemberEvent(ffpk, mapFFPK[ffpk].attribute1, mapFFPK[ffpk].attribute2, mapFFPK[ffpk].members, mapFFPK[FFPK].groupFiles);
    return(ffpk, mapFFPK[ffpk].attribute1, mapFFPK[ffpk].attribute2, mapFFPK[ffpk].members, mapFFPK[ffpk].groupFiles); }

Algorithm 5.8 addingFile and Encrypted Secret-Key(SK)

event UploadEvent (stirng FFPK, string attribute1, string attribute2, string[ ] members, string[ ] files)
function uploadFileHashAndSK (string FFPK, string $sender_{ID}$,string filehash, string fileIndex, string EncryptedSK) public returns (string FFPK, string attribute1, string attribute2,string [ ] members, string [] files){
    require (memberFFPK [$sender_{ID}$] == FFPK);
    require (fileInFF [fileIndex][filehash] == " ");
    require (fileMapHash [fileIndex][filehash] == " ");
    require (fileMapHashMapMember [filIndex][EncryptedSK] == " ");
    fileInFF[filehash] = FFPK;
    mapFFPK[FFPK].groupFiles.push(fileHash);
    fileMapHash[fileHash][fileIndex] = EncryptedSK;
    memberFiles[$sender_{ID}$].push(fileHash);
    memberFiles[$sender_{ID}$].push(fileIndex);
    fileMapHashMember[fileHash][EncryptedSK]=$sender_{ID}$;
    emit UploadEvent (FFPK,mapFFPK [FFPK].attribute1,
mapFFPK [FFPK].attribute2, mapFFPK [FFPK].members,
mapFFPK[FFPK].groupFiles);
    return(FFPK, mapFFPK[FFPK].attribute1, mapFFPK[FFPK].attribute2, mapFFPK[FFPK].members, mapFFPK[FFPK].groupFiles);
}

Algorithm 5.9 File Retrieving Smart Contract

Case 1: Another FN ∈ same FF has copy of the Encrypted-File (EF)
FN receives address of the FN has the EF in its off-chain DB, and the file index-hash plus encrypted SK from SC
FN (requestor) sends a request to the FN holding the EF;
FN (reciever) invokes the samrt contract to verify the FN (requestor) identity;
if FN(requestor) is not verified **then**
    decline;
else
    send the EF to the requestor
end if
        Case 2: Encrypted File is stored in the Shared-Storage (SS)
FN (requestor) invoke the samart contract to get the file-hash and the encrypted SK
if FN (requestor) attributes and ID are not verified **then**
    decline;
else
    return (file-hash in the SS and the encrypted SK)
end if
FN downloads the EF from the SS using the file-hash
FN decrypt the encrypted SK using its FNSK, then decrypt the retrieved EF

Algorithm 5.10 Rogue Fog Node Removing

```
function removeFN(String FFPK, string FN) public{
        require(msg.sender == TAN_address);
        require(memberFFPK[FN]==FFPK);
        memberFFPK[FN] = " ";
emit removeFN(FFPK, FN_ID); }
```

5.6.1 Security Analysis

In this section, we state the security issues that the Blockchain-Encryption-Based Approach provides solution for. The approach proposed solutions relatively focused on: rogue FNs, an MIMM attack, and a curious storage service provider. Based on our proposed approach in 5.4.1, we discuss each of these issues as follow:

5.6.1.1 Security against Rogue Fog Nodes

In this approach we consider the Fog-Node (FN) a rogue one whenever it submits a false credentials to the Smart Contract, and accordingly the SC removes the FN from the FF by decoupling it from the SC. Then, the FN cannot be authorized by the SC. When an FN encrypts a file, it encrypts it using a random generated Secret-Key (SK), then it uploads the encrypted file to the Shared-Storage (SS) (e.g. IPFS in this approach). IPFS returns an idex-hash of the location of the file in the IPFS. Then, the FN encrypts the SK using the FFPK already posted on the Smart Contract (SC) and uploaded it along with the index-hash of the file to the SC. Meaning no FN can download this file unless providing the correct index-hash. Any FN has to be verified using the SC to be able to get the Encrypted-File (EF) credential to be able to retrieve it from the SS. This means that the FN would not be able to obtain the index-hash, encrypted SK, and the hash of the encrypted file stored in the Shared-Storage (SS). This feature has been accomplished using the Smart Contracts as shown in Algorithms 8 and 9 respectively. If the FN fails to obtain authorization through the SC, it would be reported as a rogue FN and consequently be deleted from the FF, as shown SC (Algorithm 10). These features make our approach efficient in protecting end users' data confidentiality.

5.6.1.2 Security against MITM Attack

In this approach, the FN computes the hash of the file prior to encryption process, and the hash is uploaded to the SC along the encrypted file and encrypted SK. All of these are bonded to the FFPK and the FF's attributes. Later in future, if another FN goes through the authorization process via the SC and downloads the Encrypted-File (EF) from the Shared-Storage (SS), the FN (data requestor) would be capable of verifying the Encrypted-File (EF) integrity after decryption process. In addition, every FN in the FF has a FNSK different

from others but related to the FFPK. In this mean, any party not authorized member of the FF would not have a FNSK paired to the FFPK. Therefore, they will not be able to put hand of the encrypted data, but if it happens the FNs that retrieve the encrypted data would be able to discover the damaged files by checking their hashes. In short, this approach is subtle regarding the damaged/corrupted files detection.

5.6.1.3 Security against a Curious Shared-Storage Provider

In this solution, the FN (e.g. considered data owner) does not need third party to distribute the encrypted (SK) used to encrypt data to the other FNs in the FF. It also does not need to outsource the SK with the Encrypted-Files (EF) to the Shared-Storage (SS) provider. FNs encrypt SK of each EF with the FFPK and upload it to the blockchain smart contract; therefore, the SS provider cannot obtain the SK. Moreover, the SS service provider does not have a copy of the FFPK as it is only public to the FF members. It is obvious that as long as the Ethereum blockchain smart contract and the FNs' credentials are safe, this approach is secure against the curiosity of the storage service provider.

5.6.2 Implementation Details

This section presents the implementation details. Our proposed approach is a combination of the Ciphertext-Attribute-Based-Encryption (CP-ABE) technique and Ethereum blockchain to fulfill the proposed approach's objectives. Ethereum blockchain enables developers to program their smart contracts and deploy them on Ethereum platform.

5.6.2.1 Blockchain-Encryption-Based Experiment Design

This experiment aims to evaluate the proposed smart contracts' feasibility and performance in terms of the amount of gas consumed by the smart contracts' functions. In

the Blockchain-Encryption-Based Approach experiment design, we classify the Fog-Nodes (FNs) into Fog-Federations (FFs) based on their attributes namely locations and services they provide to end-users. FNs that are in same FF can access and share encrypted data stored on the Shared-Storage (SS) by invoking the smart contract. We used the Ethereum Blockchain smart contract for the following reasons: 1) To enable FNs in the same FF to carry on a distributed authorization process. 2) To store an on-chain access control table, called Encrypted-Files Tracking-Table, which must be protected from being tampered with malicious/rogue FNs. As this approach aims to minimize the communication overhead between the SS and FNs, when a FN seeks retrieving an encrypted data from the SS, it must query the smart contract first, the smart contract would check whether there is a FN in same FF having an updated copy of the requested file or not. If it is the case, the smart contract directs the FN (requestor) to obtain the required encrypted file from the FN currently in possession of the EF. In this case, exploiting the Ethereum smart contract technology here makes our model robust against falsifying the Encrypted-Files Tracking-Table.

We implemented a prototype of this approach to study its feasibility and performance. The experimental environment's configuration is: macOS Mojave 10.14 with an Intel Core i7 CPU 2.2 GHz processor. The RAM is 16 GB. We used the Solidity programming language to write the smart contracts, and Java programming language to execute the CP-ABE program. This library provides pairing-based cryptosystems in java.

At the time this research Ethereum solidity does not support Attribute-Based-Encryption (ABE) algorithms; therefore, we used the the smart contract to monitor the encrypted files tracking table and executed the algorithms related to the ABE algorithm externally in java programming language. Algorithms: 1, 2, 3, and 4 are implemented on java but do nothing with the smart contracts. In the CP-ABE program, we used the Java Pairing-Based Cryptography (JPBC) library used in []. Algorithms: 5-10 are the smart contracts of this

approach which we developed using Solidity programming language.

We used a lightweight java library called web3j [] for clients integration on the Ethereum network. Developers use web3j to evaluate the time takes to publish the the smart contract.

We also used Ganache which is a blockchain based emulator used to simulate a personal Ethereum Blockchain. It is used for testing solidity smart contracts[].

5.6.2.2 The Blockchain-Encryption Approach Setup

The InterPlanetary File System (IPFS) is used as Shared-Storage (SS). The FNs are simulated as clients which grouped into FFs by their attributes. One of these FNs nominated as a Trusted Authority Node (TAN) in each FF. Algorithms: 1, 2, 3, and 4 are simulated on java. We developed Algorithms 5-10 as smart contracts and deployed on a private ethereum blockchain.

The smart contract stores the following credentials of each Fog-Federation (FF): Fog-Node-Identity (FN_{ID}), EF_{ID}, attribute set, $FN_{att-sig}$, and off-chain database content of each FN. It also maintains the cryptographic keys of each FF, such as Fog-Federation-Public-Key (FFPK). It also maintains a set of encrypted files FF's members are able to access.

In addition, the smart contract comprises a set of credentials about each Encrypted-File (EF) upon being uploaded to the SS. These credentials are: the returned hash address of the SS, the hash of the EF before being encrypted, and the secret key used in the encryption process by the FN in an encrypted format. All of these information are protected from being tempered with by being stored on the Ethereum smart contract.

In this research, we try to minimize the communication overhead between the Shared-Storage (SS) and the FNs. Therefore, we equipped the smart contract with a list of FNs and what EF they currently have in their off-ChainDB. The smart would check the list upon

receiving a request from a FN to retrieve a certain EF. The smart returns the EF's credentials and the FN currently in possession of the EF. The smart contract considers the EF in the off-chainDB valid only for period of time. This is because the approach cannot guarantee the FNs from going rogue and then poisoning the EF they currently in possession of. On the other hand, if the smart contract verifies the FN (requestor) but no FN in the same FF currently has an updated copy of the requested EF, the smart contract is programmed to return the EF's credentials as the FN (requestor) can download the required EF from the SS. However, as the Fog Computing was introduced to minimize the time latency and communication overhead between the cloud and end-devices, this approach satisfies less communication between the SS and FNs by construction.

When a FN (data requestor) does not pass the smart contract verification process, it is considered a rogue FN and then the smart contract decouples this FN from the FF as it becomes incapable to accessing the encrypted files anymore. FNs that belong to other FFs or provide mistaken request input to the smart contract will not be able to circumvent the blockchain smart contract.

5.6.2.3 The Blockchain-Encryption Approach Experimental Evaluation

At the time of simulating this approach's Smart Contracts (SC), 1 ether price was equal to 191$. The gas price was equal 2Gwei which 1Gwei = 10^{-9} ether. The cost takes our smart contracts to execute the functions are given in Table 5.2. The transaction cost refers to the required amount of gas to post the data on the blockchain network. The execution cost refers to the function execution computational fee.

The Ethereum platform does not support Attribute-Based-Encryption (ABE) algorithms at the time of this research. Therefor, we executed Algorithms 5.1, 5.2, 5.3, and 5.4 on external environment. In this approach, we merely use the smart contract for managing

encrypted data access policies among FNs in same FF.

The smart contract is deployed by deploying Algorithm 5.5 to the Ethereum blockchain. The deployment process returns the smart contract address which authorized FNs have to use whenever to query the smart contract to perform any function. As shown in Table 5.2, the SC Deployment operation costed 0.1661432 ether which equals to 57.81 USD.

For each FF, we externally generated the following: Fog-Federation-Public-Key (FFPK), and Attributes (e.g. FNs locations and services provided by the FN in the FF). Then, the TAN invokes the Fog-Federation Creation Smart Contract to perform makeFF() function that posts the FF credentials to the blockchain. In this work, we grouped the FFs by their attributes. Each set of attributes is paired to a FFPK. When a TAN invokes the smart contract to making a FF, this performed function costed 0.0779918 ether which equals to 27.14 USD.

When a new FN joins the system, the TAN would add it to the corresponding FF by invoking adding FN to the FF Smart Contract. This smart contract performs an addFN() function. In this step, the smart contract stores the FN_{ID}, the FF this FN belongs to, the FFPK, and the FF attributes. This function costed 0.0338856 ether which equals to 11.79 USD.

This approach is basically invented to prevent rogue FNs from accessing the encrypted data stored in the Shared-Storage (SS). Suppose a FN has a file to be encrypted and uploaded to the SS. Let's consider this FN got ousted for technical or malicious reasons. In this case, we are not benefiting from the fog computing features. Therefore, the Fog-Federation idea plays a great role here by maintaining data availability to the end-users. When a FN seeks encrypting a file and uploads it the SS, it follows the following steps: 1) Performing the hashing and encryption process externally by executing Algorithm 5.4. Then, the FN encrypts the Secret-Key (SK) that was used to encrypt the file using the FFPK. Next, it

uploads the Encrypted-File (EF) to the SS which is the IPFS in this research. The IPFS returns the index-hash of the uploaded EF. Consequently, the FN invokes the addingFile and Encrypted Secret-Key(SK) Smart Contract (SS) to upload the encrypted file credentials to the blockchain as: (EF_{ID}, Encrypted (SK), File hash, Index-hash). This smart contract executes the upload File Index-Hash And SK function. This function costed 0.0854764 ether which equals to 29.74 USD. In addition, it will add the file hash. This would be carried out by executing add file hash function, and this function costed 0.081512 ether which equals to 28.36 USD.

When a FN in the FF seeks retrieving an Encrypted File (EF) from the SS, FN (requestor) invokes the File-Retrieving Smart Contract. This SC checks whether another FN in the same FF has a copy or not. If the FN (requestor) passes the SC identity validation check, and another FN in the same FF has an updated copy of the required EF, the SS returns the EF credentials that are the encrypted (SK), EF-Hash, and the idex-hash plus the identity of the FN currently has the EF in its off-chain-DB. When the FN (requestor) sends the requestor the FN, in which has a copy of the EF, the FN (receiver) would query the SC to validate the FN (requestor) identity. On the other hand, if the required EF is not available in any of the FF's members off-chain database, the FN (requestor) will download the EF from the SS, which is the IPFS in this model) using the index-hash received from the SC. The SC accomplished this by performing the Getting SK and index-hash function which costed 0.0370896 ether which equals to 12.90 USD. In addition, when any FN (requestor) is verified by the system, the verifying-FN function is provoked and it costed 0.0409558 ether which equals to 14.25 USD.

In addition to the ability of this approach to verify FNs (data requestors) before putting hands on the encrypted data, it is capable to ousting the rogue/malicious FNs upon receiving incorrect credentials from the FN upon provoking any of the SC's functions. When

Table 5.2: Smart Contract and Functions Test Cost)

SC/Function	Gas Used	Cost in (ether)	USD($)
SC Deployment	4153581	0.1661432	57.81
Fog Federation Creation	1949796	0.0779918	27.14
Adding FN to FF	847139	0.0338856	11.79
Add File Index and SK	2136909	0.0854764	29.74
Add File Hash	2037800	0.081512	28.36
Getting SK and file index	927240	0.0370896	12.90
Remove FN from FF	806318	0.0322527	11.22
Verifying FN by attribute	952460	0.0409558	14.25

the FN does not pass the validation process (verifying FN function), the SC instantaneously provokes the Rogue Fog Node Removing Smart Contract which decouples the rogue FN from the FF smart contract which become inaccessible to the rogue FN. So that, rogue FNs, that are not authorized and authenticated by the SC, requests will be discarded the SC. This would be accomplished by performing removeFN() function that costed 0.0322527 ether which equals to 11.22 USD.

5.7 Discussion

The security analysis in subsection 5.6.1 shows our approach ability to provide secure environment for end-users' data travelled among Fog-Nodes (FNs) in same FF, and between FNs and Shared-Storage (SS) providers. In particular, this approach provides the main objectives of information security that are: data confidentiality, data integrity, and data availability. In this approach, we exploited the Ethereum blockchain smart contract technology to store the On-chain File Tracking Table 5.1 on the smart contract and prevent register and prevent any modification of it performed by FN unless being authenticated by the smart contract. In addition, the FNs in same FF authenticate and verify each other using the smart contract and this is considered as a decentralization as there is no third

party we need to trust. The meant of this approach was not to develop a high performance approach, but rather it was to show its feasibility and applicability by showing its ability in protecting data security by ousting rogue FNs. In addition, as this approach focus on the channel between the cloud storage and fog nodes, we considered the channels between end-devices and the fog nodes secured. Besides, in this approach, if a FN provides false information when querying the smart contract, it is considered a rogue one and accordingly it is removed from the smart contract and that would lead to its incapability of accessing the encrypted data to the FF.

5.8 Applications

This section illustrates some of the applications in which this approach could be exploited to protect end users data security. We provide one example as follow:

This approach could be exploited in hospitals as follow. Suppose every hospital has multiple Fog-Nodes (FNs) (e.g. Fog-Node(FN) for each clinic) characterized by attributes: health-service, hospital's name. Those FNs considered as a Fog-Federation (FF). Patients records would be stored in the Shared-Storage (SS). Doctors can outsource patients information to the FN of the clinic for processing and further outsource to the SS for storage. FN holds the Encrypted-File (EF) for a specific time,after outsourcing it to the SS, in which it could be requested by another FN in same FF. Every FN in the hospital is registered to the Ethereum blockchain smart contract. When a certain doctor requested a certain private data from the clinic FN, the FN would query the smart contract for obtaining the required file address in the SS and its hash, and the encrypted Secret-Key (SK). If the smart contract verifies the FN, it will return the EF credentials to the FN. If another department FN in the same hospital has an updated copy of the EF, the smart contract return the address of the FN having the updated copy of the required file. Otherwise, the FN will get the EF from the

SS. Exploiting the smart contract concept would help FNs in same hospital to authenticate each other with tracking the actions provided by each FN. To illustrate, posting all actions on the blockchain smart contract would help in tracing any malicious event taking by a FN. We believe that applying this approach into hospitals would increase the level of patients' data security independently from trusted third party.

5.9 Conclusion

As an effort to allow Fog-Nodes (FNs) in same Fog-Federation (FF) to carry on a distributed authorization process, and to provide fine-grained data access control with no depending on a third party. We deployed an On-chain File Tracking Table to monitor the actions of FNs in same FF for a purpose of preventing rogue FNs from accessing the encrypted data stored either in another FN in same FF or in the Shared-Storage. This approach provides data security by providing analysis for that, namely: confidentiality, integrity, and availability. Functions that are important to fulfill this approach were

6 Conclusion

In this chapter, we start by showing our approaches' contributions against the current literature approaches. Then, we conclude this dissertation.

6.1 Approaches' Features Comparison

Table.6.1 shows the contributions of our approaches against literature available approaches in terms of provided features by each approach.

Table 6.1: Comparison between Our Approaches and Approaches from Literature

	Features							
Schemes	FF	FFSG	Confidentiality	Integrity	Availability	Authentication Process	Access Control	Rogue Fog Node Detection and Ousting
Encryption-Based	✓	×	✓	✓	✓	Digital Signature	Fine-grained	Ousting
Multi-Level	✓	✓	✓	✓	✓	Digital Signature	Fine-grained	Ousting
Encryption-Blockchain-Based	✓	×	✓	✓	✓	Decentralized using Ethereum Smart Contract	Fine-grained	Detection and Ousting
[96]	×	×	✓	×	×	Digital Signature	Fine-grained	×
[97]	×	×	✓	×	×	Authenticate nodes by hashing request message	Using Access Key	×
[44]	×	×	×	×	✓	Challenge-response Mechanism	×	×
[45]	×	×	✓	×	✓	Digital signature	Fine-Grained	Revoking Users' Attributes
[76]	×	×	×	×	×	Challenge response technique and PKI method	×	Detecting rogue vehicle
[79]	×	×	×	×	×	×	×	Detecting rogue AP using round trip time
[71]	×	×	✓	✓	×	×	×	×
[53]	×	×	✓	×	×	Certificate	Fine-grained	Revoking users' attributes
[54]	×	×	✓	×	✓	ABS	Fine-grained	×
[98]	×	×	✓	×	×	Multi-authority CP-ABE method	Fine-grained	×

6.2 Conclusion

This dissertation discussed the services that fog computing provides to individuals and organizations. Besides, it discussed the reasons behind the emergence of the fog computing paradigm. The dissertation then pointed out the security issues that could face individuals' and organizations' confidential data while being stored in fog nodes or the cloud or transit between the fog computing nodes and the cloud service providers. We summarized the malicious actors' impact on these confidential data. This dissertation focused on the issues related to rogue fog nodes, Man-in-the-Middle interceptors, and curious cloud service providers in the context of fog computing. We provided the potential methods in which those actors would violate individuals' and organizations' security. Through this dissertation, we focused on securing data in terms of confidentiality, integrity, and availability.

This dissertation introduced three main contributions: Encryption-Based Approach, Multi-Level Approach, and a Blockchain-Encryption-Based Approach. First, Encryption-based Approach is a system that introduced the idea of Fog-Federation (FF) by grouping fog nodes that share the same attributed (location and services they provide) into fog federations. This approach manages to oust the rogue Fog-Nodes (FNs), so they become incapable of accessing the encrypted data in the cloud or other FNs in the same FF. It also manages to allow other FNs in the same FF to access the re-encrypted data with no latency. Encryption-Based Approach splits the file to be encrypted into blocks to reduce the communication time and communication overhead between the Cloud-Service-Provider (CSP) and Fog-Nodes (FNs) in FF. This approach allows the CSP to track what each fog node currently has/needs. Our experiment showed that splitting files to be encrypted into blocks is statistically faster than for whole files. This method helped in minimizing the communication overhead between the CSP and FFs.

Second, the Multi-level Approach utilized a CP-ABE and hashing cryptographic

primitives to reduce data breaching by rogue fog nodes. This approach designed the Fog-Federation-Subgroups (FFSG) of Fog-Nodes (FNs) in the same FF. Further, this scheme splits files to be encrypted into blocks based on data sensitivity. Then, FNs are placed on FFSGs based on their access rights given to them. This means, if one fog node goes rogue, as no fog node in the FFSGs can access all blocks of the encrypted file, the amount of data in danger of being breached by this rogue node is minimized. FN cannot access the encrypted data of other FFSG unless satisfying the attributes of the FF and FFSG. In this scheme, the CSP maintains a progress list called Encrypted Files Tracking Table to demonstrate each FN's possession in the Fog-Federation (FF). We designed algorithms to fulfill this approach's objectives. Then we provided a correctness proof of our encryption and decryption algorithms. Besides, we provided a security analysis followed by our experiments. Our experiments showed that the idea of Fog-Federation-Subgroups (FFSGs) helps in minimizing the data that are in danger from being breached by rogue fog nodes.

Third, a Blockchain-Encryption-Based Approach is an approach designed to oust rogue fog node and minimize the communication overhead between the CSP and FNs by integrating the Ciphertext-Policy Attribute-Based-Encryption (CP-ABE) algorithm and blockchain technology. We adopted blockchain in this approach as a medium to store an on-chain tracking table to verify the identity of each fog node in the fog federation using the smart contract before they could access the encrypted data on the CSP. In this approach, blockchain helps prevent the fog nodes from maliciously changing the on-chain tracking table; if such a change detected, the FN that issues the request is reported as a rogue fog node. In this approach, we used the IPFS as shared storage, and we simulated the blockchain using the Ethereum smart contract. In our experiment, we showed the feasibility and applicability of this approach by simulation. Then, we showed that this approach satisfied the proposed objectives because the consumption gas takes the smart contract to perform the approach's

functions should be affordable to FNs.

Our approaches in this dissertation focused on the cloud and fog computing levels. In our future research, we will focus on providing a holistic approach to serve all levels (e.g., end-users devices-fog-cloud).

Bibliography

[1] A. Michael, F. Armando, G. Rean, D. J. Anthony, K. Randy, K. Andy, L. Gunho, P. David, R. Ariel, S. Ion *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[2] A. D. JoSEP, R. KAtz, A. KonWinSKi, L. Gunho, D. PAttERSon, and A. RABKin, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, 2010.

[3] B. Hayes, "Cloud computing," 2008.

[4] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[5] R. L. K. R. D. Vines and R. Krutz, *Cloud security: A comprehensive guide to secure cloud computing*. Wiley Publishing, Inc, 2010.

[6] A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and I. Stoica, "Above the clouds: A berkeley view of cloud computing," *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS*, vol. 28, no. 13, p. 2009, 2009.

[7] Q. Duan, Y. Yan, and A. V. Vasilakos, "A survey on service-oriented network virtualization toward convergence of networking and cloud computing," *IEEE Transactions on Network and Service Management*, vol. 9, no. 4, pp. 373–392, 2012.

[8] A. S. Prasad and S. Rao, "A mechanism design approach to resource procurement in cloud computing," *IEEE Transactions on Computers*, vol. 63, no. 1, pp. 17–30, 2013.

[9] R. Deng, R. Lu, C. Lai, and T. H. Luan, "Towards power consumption-delay tradeoff by workload allocation in cloud-fog computing," in *2015 IEEE International Conference on Communications (ICC)*. IEEE, 2015, pp. 3909–3914.

[10] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 13–16.

[11] B. Varghese and R. Buyya, "Next generation cloud computing: New trends and research directions," *Future Generation Computer Systems*, vol. 79, pp. 849–861, 2018.

[12] G. Anthes, "Security in the cloud,‖ in acm communications (2010), vol. 53," *Issue11*, pp. 16–18.

[13] H. Takabi, J. B. Joshi, and G.-J. Ahn, "Security and privacy challenges in cloud computing environments," *IEEE Security & Privacy*, vol. 8, no. 6, pp. 24–31, 2010.

[14] D. Zissis and D. Lekkas, "Addressing cloud computing security issues," *Future Generation computer systems*, vol. 28, no. 3, pp. 583–592, 2012.

[15] J. Li, J. Jin, D. Yuan, M. Palaniswami, and K. Moessner, "Ehopes: Data-centered fog platform for smart living," in *2015 International Telecommunication Networks and Applications Conference (ITNAC)*. IEEE, 2015, pp. 308–313.

[16] S. Yi, C. Li, and Q. Li, "A survey of fog computing: concepts, applications and issues," in *Proceedings of the 2015 workshop on mobile big data*. ACM, 2015, pp. 37–42.

[17] M. Mukherjee, R. Matam, L. Shu, L. Maglaras, M. A. Ferrag, N. Choudhury, and V. Kumar, "Security and privacy in fog computing: Challenges," *IEEE Access*, vol. 5, pp. 19 293–19 304, 2017.

[18] I. Stojmenovic, S. Wen, X. Huang, and H. Luan, "An overview of fog computing and its security issues," *Concurrency and Computation: Practice and Experience*, vol. 28, no. 10, pp. 2991–3005, 2016.

[19] I. Stojmenovic and S. Wen, "The fog computing paradigm: Scenarios and security issues," in *2014 Federated Conference on Computer Science and Information Systems*. IEEE, 2014, pp. 1–8.

[20] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Annual international cryptology conference*. Springer, 2001, pp. 213–229.

[21] M. Green, S. Hohenberger, B. Waters *et al.*, "Outsourcing the decryption of abe ciphertexts." in *USENIX Security Symposium*, vol. 2011, no. 3, 2011.

[22] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *2007 IEEE symposium on security and privacy (SP'07)*. IEEE, 2007, pp. 321–334.

[23] C. Wang and J. Luo, "An efficient key-policy attribute-based encryption scheme with constant ciphertext length," *Mathematical Problems in Engineering*, vol. 2013, 2013.

[24] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Security and Privacy, 2007. SP'07. IEEE Symposium on*. IEEE, 2007, pp. 321–334.

[25] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM conference on Computer and communications security*. Acm, 2006, pp. 89–98.

[26] A. Beimel *et al.*, *Secure schemes for secret sharing and key distribution*. Technion-Israel Institute of technology, Faculty of computer science, 1996.

[27] A. Hughes, A. Park, J. Kietzmann, and C. Archer-Brown, "Beyond bitcoin: What blockchain and distributed ledger technologies mean for firms," *Business Horizons*, vol. 62, no. 3, pp. 273–281, 2019.

[28] F. Tschorsch and B. Scheuermann, "Bitcoin and beyond: A technical survey on decentralized digital currencies," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 2084–2123, 2016.

[29] X. Xu, I. Weber, M. Staples, L. Zhu, J. Bosch, L. Bass, C. Pautasso, and P. Rimba, "A taxonomy of blockchain-based systems for architecture design," in *2017 IEEE International Conference on Software Architecture (ICSA)*. IEEE, 2017, pp. 243–252.

[30] S. Nakamoto and A. Bitcoin, "A peer-to-peer electronic cash system," *Bitcoin.–URL: https://bitcoin. org/bitcoin. pdf*, 2008.

[31] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, "Bitcoin-ng: A scalable blockchain protocol," in *13th {USENIX} symposium on networked systems design and implementation ({NSDI} 16)*, 2016, pp. 45–59.

[32] M. Crosby, P. Pattanayak, S. Verma, V. Kalyanaraman *et al.*, "Blockchain technology: Beyond bitcoin," *Applied Innovation*, vol. 2, no. 6-10, p. 71, 2016.

[33] A. Singh, R. M. Parizi, Q. Zhang, K.-K. R. Choo, and A. Dehghantanha, "Blockchain smart contracts formalization: Approaches and challenges to address vulnerabilities," *Computers & Security*, vol. 88, p. 101654, 2020.

[34] T. Salman, M. Zolanvari, A. Erbad, R. Jain, and M. Samaka, "Security services using blockchains: A state of the art survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 858–880, 2018.

[35] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," *white paper*, vol. 3, no. 37, 2014.

[36] F. Tao, M. Zhang, and A. Y. C. Nee, *Digital twin driven smart manufacturing*. Academic Press, 2019.

[37] J. Acharya and S. Gaur, "Edge compression of gps data for mobile iot," in *2017 IEEE Fog World Congress (FWC)*. IEEE, 2017, pp. 1–6.

[38] S. Kunal, A. Saha, and R. Amin, "An overview of cloud-fog computing: Architectures, applications with security challenges," *Security and Privacy*, vol. 2, no. 4, p. e72, 2019.

[39] S. Yi, Z. Hao, Z. Qin, and Q. Li, "Fog computing: Platform and applications," in *2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*. IEEE, 2015, pp. 73–78.

[40] S. Parasuraman and A. K. Sangaiah, "Fog–driven healthcare framework for security analysis," in *Computational Intelligence for Multimedia Big Data on the Cloud With Engineering Applications*. Elsevier, 2018, pp. 253–270.

[41] J. Ni, A. Zhang, X. Lin, and X. S. Shen, "Security, privacy, and fairness in fog-based vehicular crowdsensing," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 146–152, 2017.

[42] S. Yi, Z. Qin, and Q. Li, "Security and privacy issues of fog computing: A survey," in *International conference on wireless algorithms, systems, and applications*. Springer, 2015, pp. 685–695.

[43] H. Wang, Q. Wu, B. Qin, and J. Domingo-Ferrer, "Identity-based remote data possession checking in public clouds," *IET Information Security*, vol. 8, no. 2, pp. 114–121, 2014.

[44] H. Wang, "Identity-based distributed provable data possession in multicloud storage," *IEEE Transactions on Services Computing*, no. 2, pp. 328–340, 2015.

[45] S. Yu, C. Wang, K. Ren, and W. Lou, "Attribute based data sharing with attribute revocation," in *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*. ACM, 2010, pp. 261–270.

[46] Z. Liu, Z. L. Jiang, X. Wang, and S.-M. Yiu, "Practical attribute-based encryption: Outsourcing decryption, attribute revocation and policy updating," *Journal of Network and Computer Applications*, vol. 108, pp. 112–123, 2018.

[47] S. Wang, L. Yao, and Y. Zhang, "Attribute-based encryption scheme with multi-keyword search and supporting attribute revocation in cloud storage," *PloS one*, vol. 13, no. 10, p. e0205675, 2018.

[48] J. Li, Y. Zhang, X. Chen, and Y. Xiang, "Secure attribute-based data sharing for resource-limited users in cloud computing," *Computers & Security*, vol. 72, pp. 1–12, 2018.

[49] P. Tysowski and M. Hasan, "Hybrid attribute-based encryption and re-encryption for scalable mobile applica-tionsin clouds," *IEEE Transactions onCloudComputing*, pp. 172–186, 2013.

[50] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *Infocom, 2010 proceedings IEEE*. Ieee, 2010, pp. 1–9.

[51] J. Ning, X. Dong, Z. Cao, L. Wei, and X. Lin, "White-box traceable ciphertext-policy attribute-based encryption supporting flexible attributes," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 6, pp. 1274–1288, 2015.

[52] J. Ning, Z. Cao, X. Dong, L. Wei, and X. Lin, "Large universe ciphertext-policy attribute-based encryption with white-box traceability," in *European Symposium on Research in Computer Security*. Springer, 2014, pp. 55–72.

[53] J. Li, W. Yao, Y. Zhang, H. Qian, and J. Han, "Flexible and fine-grained attribute-based data storage in cloud computing," *IEEE Transactions on Services Computing*, vol. 10, no. 5, pp. 785–796, 2017.

[54] Q. Huang, Y. Yang, and L. Wang, "Secure data access control with ciphertext update and computation outsourcing in fog computing for internet of things," *IEEE Access*, vol. 5, pp. 12 941–12 950, 2017.

[55] S. Wang, J. Zhou, J. K. Liu, J. Yu, J. Chen, and W. Xie, "An efficient file hierarchy attribute-based encryption scheme in cloud computing," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 6, pp. 1265–1277, 2016.

[56] N. Kaaniche and M. Laurent, "Attribute based encryption for multi-level access control policies," 2017.

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

[58] Y. Jiang, W. Susilo, Y. Mu, and F. Guo, "Ciphertext-policy attribute-based encryption against key-delegation abuse in fog computing," *Future Generation Computer Systems*, vol. 78, pp. 720–729, 2018.

[59] C. Zuo, J. Shao, G. Wei, M. Xie, and M. Ji, "Cca-secure abe with outsourced decryption for fog computing," *Future Generation Computer Systems*, vol. 78, pp. 730–738, 2018.

[60] M. Xiao, J. Zhou, X. Liu, and M. Jiang, "A hybrid scheme for fine-grained search and access authorization in fog computing environment," *Sensors*, vol. 17, no. 6, p. 1423, 2017.

[61] X. Mao, J. Lai, Q. Mei, K. Chen, and J. Weng, "Generic and efficient constructions of attribute-based encryption with verifiable outsourced decryption," *IEEE Transactions on Dependable and Secure Computing*, no. 1, pp. 1–1, 2016.

[62] Y. Li, Z. Dong, K. Sha, C. Jiang, J. Wan, and Y. Wang, "Tmo: Time domain outsourcing attribute-based encryption scheme for data acquisition in edge computing," *IEEE Access*, vol. 7, pp. 40 240–40 257, 2019.

[63] Q. Xu, C. Tan, Z. Fan, W. Zhu, Y. Xiao, and F. Cheng, "Secure data access control for fog computing based on multi-authority attribute-based signcryption with computation outsourcing and attribute revocation," *Sensors*, vol. 18, no. 5, p. 1609, 2018.

[64] C. Dsouza, G.-J. Ahn, and M. Taguinod, "Policy-driven security management for fog computing: Preliminary framework and a case study," in *Proceedings of the 2014 IEEE 15th International Conference on Information Reuse and Integration (IEEE IRI 2014)*. IEEE, 2014, pp. 16–23.

[65] S. Alharbi, P. Rodriguez, R. Maharaja, P. Iyer, N. Bose, and Z. Ye, "Focus: A fog computing-based security system for the internet of things," in *2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2018, pp. 1–5.

[66] Y. Imine, D. E. Kouicem, A. Bouabdallah, and L. Ahmed, "Masfog: An efficient mutual authentication scheme for fog computing architecture," in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 2018, pp. 608–613.

[67] A. Alotaibi, A. Barnawi, and M. Buhari, "Attribute-based secure data sharing with efficient revocation in fog computing," *Journal of Information Security*, vol. 8, no. 03, p. 203, 2017.

[68] 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," *Future Generation Computer Systems*, vol. 78, pp. 753–762, 2018.

[69] S. Salonikias, I. Mavridis, and D. Gritzalis, "Access control issues in utilizing fog computing for transport infrastructure," in *International Conference on Critical Information Infrastructures Security*. Springer, 2015, pp. 15–26.

[70] F. Li, Y. Rahulamathavan, M. Conti, and M. Rajarajan, "Robust access control framework for mobile cloud computing network," *Computer Communications*, vol. 68, pp. 61–72, 2015.

[71] M. B. Mollah, M. A. K. Azad, and A. Vasilakos, "Secure data sharing and searching at the edge of cloud-assisted internet of things," *IEEE Cloud Computing*, vol. 4, no. 1, pp. 34–42, 2017.

[72] M. Ali, M.-R. Sadeghi, and X. Liu, "Lightweight revocable hierarchical attribute-based encryption for internet of things," *IEEE Access*, vol. 8, pp. 23 951–23 964, 2020.

[73] R. Guo, C. Zhuang, H. Shi, Y. Zhang, and D. Zheng, "A lightweight verifiable outsourced decryption of attribute-based encryption scheme for blockchain-enabled wireless body area network in fog computing," *International Journal of Distributed Sensor Networks*, vol. 16, no. 2, p. 1550147720906796, 2020.

[74] A. Alrawais, A. Alhothaily, C. Hu, and X. Cheng, "Fog computing for the internet of things: Security and privacy issues," *IEEE Internet Computing*, vol. 21, no. 2, pp. 34–42, 2017.

[75] M. H. Ibrahim, "Octopus: An edge-fog mutual authentication scheme." *IJ Network Security*, vol. 18, no. 6, pp. 1089–1101, 2016.

[76] B. Al-Otaibi, N. Al-Nabhan, and Y. Tian, "Privacy-preserving vehicular rogue node detection scheme for fog computing," *Sensors*, vol. 19, no. 4, p. 965, 2019.

[77] L. Chen, S.-L. Ng, and G. Wang, "Threshold anonymous announcement in vanets," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 3, pp. 605–615, 2011.

[78] M. Conti, N. Dragoni, and V. Lesyk, "A survey of man in the middle attacks," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 2027–2051, 2016.

[79] H. Han, B. Sheng, C. C. Tan, Q. Li, and S. Lu, "A measurement based rogue ap detection scheme," in *IEEE INFOCOM 2009*. IEEE, 2009, pp. 1593–1601.

[80] A. S. Sohal, R. Sandhu, S. K. Sood, and V. Chang, "A cybersecurity framework to identify malicious edge device in fog computing and cloud-of-things environments," *Computers & Security*, vol. 74, pp. 340–354, 2018.

[81] G. Shivaraj, M. Song, and S. Shetty, "A hidden markov model based approach to detect rogue access points," in *MILCOM 2008-2008 IEEE Military Communications Conference*. IEEE, 2008, pp. 1–7.

[82] M. McCormick, "Data theft: a prototypical insider threat," in *Insider Attack and Cyber Security*. Springer, 2008, pp. 53–68.

[83] P. Jyothi, R. Anuradha, and D. Y. Vijayalata, "Minimizing internal data theft in cloud through disinformation attacks," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 2, no. 9, 2013.

[84] M. Alshehri and B. Panda, "An encryption-based approach to protect fog federations from rogue nodes," in *International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage*. Springer, 2019, pp. 225–243.

[85] M. Sookhak, F. R. Yu, M. K. Khan, Y. Xiang, and R. Buyya, "Attribute-based data access control in mobile cloud computing: Taxonomy and open issues," *Future Generation Computer Systems*, vol. 72, pp. 273–287, 2017.

[86] S. Ruj, A. Nayak, and I. Stojmenovic, "Dacc: Distributed access control in clouds," in *2011IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications*. IEEE, 2011, pp. 91–98.

[87] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[88] G. D. Knott, "Hashing functions," *The Computer Journal*, vol. 18, no. 3, pp. 265–278, 1975.

[89] A. Caro and V. Iovino, "Java pairing-based cryptography library," 2013.

[90] B. Lynn, "The pairing-based cryptography (pbc) library," 2010.

[91] M. Alshehri and B. Panda, "Minimizing data breach by a malicious fog node within a fog federation," in *2020 7th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2020 6th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, 2020, pp. 36–43.

[92] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[93] B. Varghese, N. Wang, S. Barbhuiya, P. Kilpatrick, and D. S. Nikolopoulos, "Challenges and opportunities in edge computing," in *2016 IEEE International Conference on Smart Cloud (SmartCloud)*. IEEE, 2016, pp. 20–26.

[94] M. Alshehri and B. Panda, "A blockchain-encryption-based approach to protect fog federations from rogue nodes," in *2019 3rd Cyber Security in Networking Conference (CSNet)*, 2019, pp. 6–13.

[95] S. Nakamoto *et al.*, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[96] A. Alrawais, A. Alhothaily, C. Hu, X. Xing, and X. Cheng, "An attribute-based encryption scheme to secure fog communications," *IEEE access*, vol. 5, pp. 9131–9138, 2017.

[97] M. Shamseddine, W. Itani, A. Al-Dulaimy, and J. Taheri, "Mitigating rogue node attacks in edge computing," in *2019 2nd IEEE Middle East and North Africa COMMunications Conference (MENACOMM)*, 2019, pp. 1–6.

[98] K. Vohra and M. Dave, "Multi-authority attribute based data access control in fog computing," *Procedia computer science*, vol. 132, pp. 1449–1457, 2018.

7 Appendix

7.1 Publications Published, Submitted, and Planned

- Alshehri, Mohammed, and Brajendra Panda. "An encryption-based approach to protect fog federations from rogue nodes." International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage. Springer, Cham, 2019. **(Published)**

- M. Alshehri and B. Panda, "A Blockchain-Encryption-Based Approach to Protect Fog Federations from Rogue Nodes," 2019 3rd Cyber Security in Networking Conference (CSNet), Quito, Ecuador, 2019, pp. 6-13, doi: 10.1109/CSNet47905.2019.9108975. **(Published)**

- M. Alshehri and B. Panda, "Minimizing Data Breach by a Malicious Fog Node within a Fog Federation," 2020 7th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2020 6th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom), New York, NY, USA, 2020, pp. 36-43, doi: 10.1109/CSCloud-EdgeCom49738.2020.00016.**(Published)**

7.2 Reprint Permissions

This dissertation was based on three published papers as mentioned in 7.1. The three papers were used verbatim in 3, 4, and 5. The following reprint permissions were obtained from the publishers.

Figure 7.1: Reprint Permissions