applied sciences

Article

Novel Security Models for IoT–Fog–Cloud Architectures in a Real-World Environment

Mohammed A. Aleisa [1,2,*], Abdullah Abuhussein [3,*], Faisal S. Alsubaei [4] and Frederick T. Sheldon [1]

1 Department of Computer Science, The University of Idaho, Moscow, ID 83844, USA; sheldon@uidaho.edu
2 Department of Computer Science, Majmaah University, Al-Majmaah 11952, Saudi Arabia
3 Information Systems Department, St. Cloud State University, St. Cloud, MN 56301, USA
4 Department of Cybersecurity, University of Jeddah, Jeddah 23890, Saudi Arabia; fsalsubaei@uj.edu.sa
* Correspondence: alei3598@vandals.uidaho.edu or m.aleisa@mu.edu.sa (M.A.A.); aabuhussein@stcloudstate.edu (A.A.)

Abstract: With the rise of the Internet of Things (IoT), there is a demand for computation at network edges because of the limited processing capacity of IoT devices. Fog computing is a middle layer that has appeared to address the latency issues between the Internet of things (IoT) and the cloud. Fog computing is becoming more important as companies face increasing challenges in collecting and sending data from IoT devices to the cloud. However, this has led to new security and privacy issues as a result of the large number of sensors in IoT environments as well as the massive amount of data that must be analyzed in real time. To overcome the security challenges between the IoT layer and fog layer and, thus, meet the security requirements, this paper proposes a fine-grained data access control model based on the attribute-based encryption of the IoT–Fog–Cloud architecture to limit the access to sensor data and meet the authorization requirements. In addition, this paper proposes a blockchain-based certificate model for the IoT–Fog–Cloud architecture to authenticate IoT devices to fog devices and meet the authentication requirements. We evaluated the performance of the two proposed security models to determine their efficiency in real-life experiments of the IoT–Fog–Cloud architecture. The results demonstrate that the performance of the IoT–Fog–Cloud architecture with and without the blockchain-based certificate model was the same when using one, two, or three IoT devices. However, the performance of the IoT–Fog–Cloud architecture without the access control model was slightly better than that of the architecture with the model when using one, two, or three IoT devices.

Keywords: blockchain; AWS cloud metrics; fog computing; access control; cloud computing; authentication; Internet of Things; authorization

check for updates

Citation: Aleisa, M.A.; Abuhussein, A.; Alsubaei, F.S.; Sheldon, F.T. Novel Security Models for IoT–Fog–Cloud Architectures in a Real-World Environment. *Appl. Sci.* **2022**, *12*, 4837. https://doi.org/10.3390/app12104837

Academic Editor: Eui-Nam Huh

Received: 31 March 2022
Accepted: 28 April 2022
Published: 10 May 2022

1. Introduction

The Internet of Things (IoT) ecosystem includes multitudinous devices connected to the Internet [1] with a variety of capabilities, such as sensing, processing, and communicating. In 2025, the number of IoT devices is estimated to exceed 75 billion [2,3], driving a parallel rise in the already massive amount of data that must be locally processed at the edges of networks to reduce latency and save network bandwidth. Cloud computing provides a lot of processing and storage power for thousands of IoT devices [4]. However, due to the geographic centralization of cloud computing data centers, the large volume of data generated by the distributed IoT devices will not be processed in a timely manner, which will increase the latency from IoT devices to the cloud especially as the number of IoT devices continues to grow. To overcome these challenges, fog computing has emerged to deal with the high processing demand and temporary storage. Fog computing serves as a middle layer between IoT devices and the cloud [5–7], solving the data transmission latency between them.

Despite the benefits of fog computing for IoT devices and the cloud, there are several security issues between the IoT, fog, and cloud layers. For example, the Dyn cyberattack (21 October 2016) disrupted Internet service across Europe and the US [8] through a series of distributed denial-of-service (DDoS) attacks that targeted IoT-enabled devices such as cameras, residential gateways, and baby monitors. Many services were affected by this cyberattack, including businesses such as Amazon, Comcast, PayPal, and Netflix and news networks such as Fox News and CNN.

Further significant threats to IoT devices include eavesdropping [1,4,9,10] and unauthorized access [9,11,12], which can lead to device failure. Because there is no human interaction involved in the communication between these devices and because they have extensive operating times, it is difficult to monitor and detect their security issues. Therefore, it is essential to build a security model that meets the security requirements of the IoT–Fog–Cloud architecture by applying authentication and authorization between the IoT, fog, and cloud layers.

Because there are not enough real-life implementations of cloud-based IoT environments, as mentioned in our previous work [5], we proposed two architectures of cloud-based IoT environments and three analysis methods using a real-world environment [6,7]. We utilized a fog layer between IoT devices and the cloud in the first architecture, while in the second architecture, IoT devices sent data directly to the cloud [6,7]. We conducted several experiments and evaluated our results of the methodologies and the three analysis methods [6], finding that the first architecture (IoT–Fog–Cloud) outperforms the second architecture (IoT–Cloud) in terms of performance. This is because all of the IoT devices in the second architecture (IoT–Cloud) need to have a certificate to be authenticated to the AWS cloud. However, only one certificate which is placed on the fog device of the first architecture (IoT–Fog–Cloud) is required to be authenticated to the AWS cloud. Therefore, the communication between the IoT layer and the fog layer, in particular, was left without adequate authentication and authorization mechanisms [6].

To fill the gap in security requirements between the IoT layer and fog layer and overcome the limitations and challenges presented by these security issues [5–7], this paper makes the following contributions:

- We propose a fine-grained data access control model based on the attribute-based encryption (ABE) of the IoT–Fog–Cloud architecture to limit access to sensor data to meet the authorization aim.
- We propose a blockchain-based certificate model of the IoT–Fog–Cloud architecture to authenticate IoT devices to fog devices to meet the authentication aim.
- We evaluate the performance of the security model (fine-grained data access control and blockchain-based certificate) using AWS message broker metrics for a real-life scenario of the IoT–Fog–Cloud architecture.
- We compare the performance of the IoT–Fog–Cloud architecture with and without our security model using AWS message broker metrics and present its efficiency and feasibility.

The remainder of the paper is structured as follows: In Section 2, we provide an overview of the IoT–Fog–Cloud architecture. In Section 3, we explain the authentication model, a blockchain-based certificate model, and the authorization model, a fine-grained data access control model based on the ABE of the IoT–Fog–Cloud architecture. In Section 4, we detail the setup of the IoT–Fog–Cloud architecture experiments and describe the evaluation metrics that were used to evaluate the IoT–Fog–Cloud architecture performance with the two security models. In Section 5, we present the evaluation methods used to evaluate the IoT–Fog–Cloud architecture with the two security models. In Section 6, we evaluate the experiment results based on the evaluation methods of the IoT–Fog–Cloud architecture. In Section 7, we conclude the paper.

2. Overview of IoT–Fog–Cloud Architecture

We describe our IoT–Fog–Cloud architecture in detail in this section. IoT refers to devices that generate data, communicate with other devices in a real-world scenario, and have storage for configuration. Since DHT11 sensors are not equipped with network capabilities and storage for configuration, one DHT11 sensor will be linked to only one Raspberry Pi board in the IoT–Fog–Cloud architecture using GPIO pins. The DHT11 sensor connected to the Raspberry Pi will be considered an IoT device. We used the Raspberry Pi to provide storage for configuration and enable the Wi-Fi connectivity of the DHT11 sensor. Each IoT device will be linked to other Raspberry Pi devices through Wi-Fi, which served as a fog device in the fog layer.

The MQTT protocol was utilized to communicate between IoT devices, fog nodes, and the cloud because the MQTT protocol is extensively used and supported by all IoT platforms and commercial sensors. The MQTT broker named Eclipse Mosquitto [13] was installed in the Raspberry Pi which served as a fog device. The Mosquitto MQTT broker is a server that receives all messages from IoT devices and publishes them to other devices using the MQTT protocol. Some advantages of using an MQTT broker include (1) facilitating scalability with a large number of IoT devices, (2) handling authentication credentials and certificates, (3) decreasing cellular network strain without compromising security, and (4) preventing unsafe and vulnerable devices from being connected. The Mosquitto MQTT broker used the subscribe–publish strategy described in [5] to exchange messages and filtered them based on the topics. The topics are UTF-8 strings used by the broker to identify the sensor data type (temperature degree or humidity degree). In our experiments, each data type (temperature degree or humidity degree) was considered a separate topic to provide consistent and accurate results. The three DHT11 sensor data gathered by the three IoT devices were sent through Wi-Fi to the Raspberry Pi that served as the fog device. The Raspberry Pi that acts as the fog device with the MQTT broker Mosquitto was connected through the Internet to the AWS cloud. The Internet served as the communication between the three layers.

We conducted several experiments based on the three analysis methods in our previous work [6], and we found that the performance of the IoT–Fog–Cloud architecture outperforms the performance of the IoT–Cloud architecture. However, we found that the IoT–Fog–Cloud architecture does not meet the security requirements. Therefore, to fill the gap of security requirements between the IoT layer and fog layer and meet the authentication and authorization aims, first, we propose a blockchain-based certificate model of the IoT–Fog–Cloud architecture to fill the gap of the authentication problem. Second, we propose a fine-grained data access control model based on the ABE of the IoT–Fog–Cloud architecture to fill the gap of the authorization problem. Figure 1 shows an overview of the IoT–Fog–Cloud architecture and where the two security models are applied.

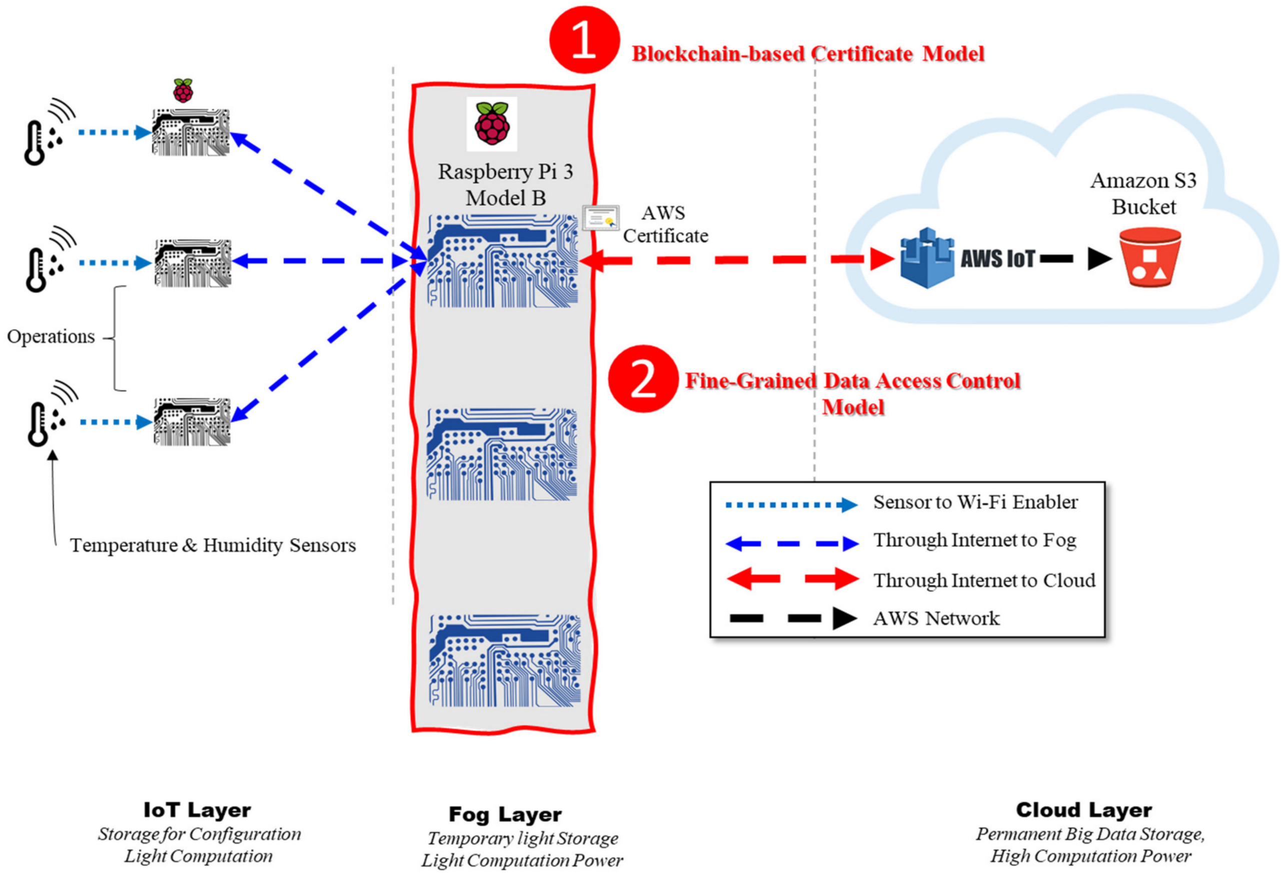

Figure 1. Overview of the IoT–Fog–Cloud architecture.

3. Authentication Model and Authorization Model

3.1. Proposed Authentication Model: Blockchain-Based Certificate

To fill the gap of the security requirements between the IoT layer and fog layer [5–7], we propose a blockchain-based certificate model of the IoT–Fog–Cloud architecture to authenticate the IoT devices to fog devices and achieve the authentication aim of this study. Figure 2 presents the operations comprising the model, which are as follows: (1) the IoT devices make a connection request to the fog devices; (2) the fog devices distribute a valid certificate to the IoT devices; (3) the handshake mechanism using the TLS cryptographic protocol is established between IoT devices and fog devices; (4) encrypted communication is established between the IoT devices and fog devices; (5) because the fog devices are expected to be limited, the blockchain technology is applied to a set of fog devices within their geographical location; and (6) each fog device inside the blockchain has a copy of transactions, such as the distributed IoT device certificates. Figure 2 shows how the blockchain-based certificate model was applied to the IoT–Fog–Cloud architecture using a real-life environment.

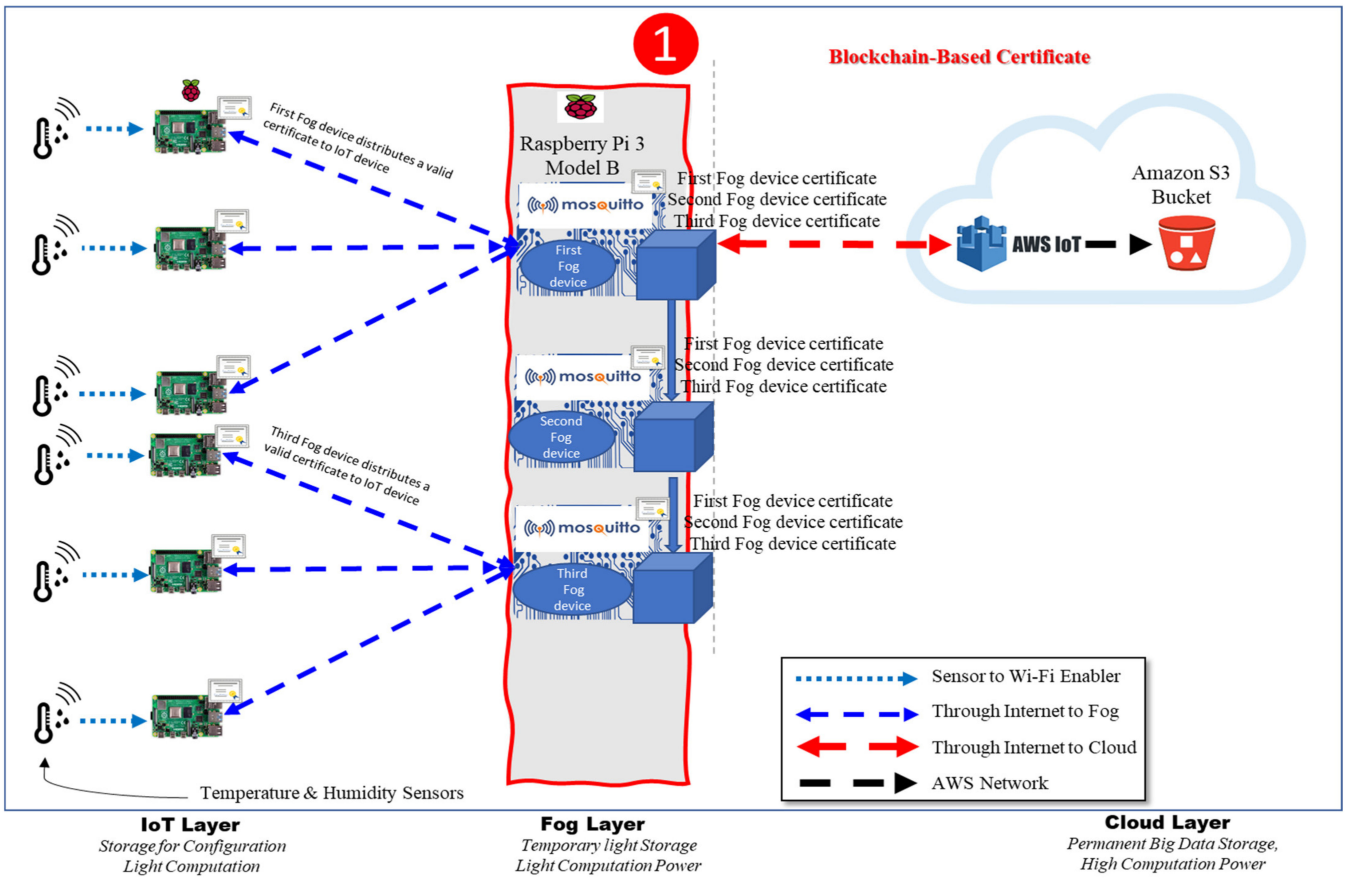
1
Blockchain-Based Certificate
First Fog device distributes a valid certificate to IoT device
Raspberry Pi 3 Model B
mosquitto
First Fog device
Second Fog device
Third Fog device
First Fog device certificate
Second Fog device certificate
Third Fog device certificate
Amazon S3 Bucket
AWS IoT
Third Fog device distributes a valid certificate to IoT device
Temperature & Humidity Sensors
Sensor to Wi-Fi Enabler
Through Internet to Fog
Through Internet to Cloud
AWS Network
IoT Layer
Storage for Configuration
Light Computation
Fog Layer
Temporary light Storage
Light Computation Power
Cloud Layer
Permanent Big Data Storage,
High Computation Power

Figure 2. Blockchain-based certificate model applied to the IoT–Fog–Cloud architecture.

3.2. Proposed Authorization Model: Attribute-Based Encryption for Access Control

To fill the gap in the security requirements between the IoT and fog layers [5–7], we propose a fine-grained data access control model based on the ABE of the IoT–Fog–Cloud architecture to limit access to sensor data and achieve the authorization aim of this study.

Figure 3 illustrates the several operations comprising the model, which are as follows: (1) attributes are generated for each sensor data type; (2) keys containing a set of attributes or corresponding to attributes are generated; (3) the generated sensor data type values are encrypted with the corresponding key that contains their attributes; (4) the encrypted message is published to the fog device using a secure communication channel; (5) the access tree that specifies the policy of the set of attributes is generated; (6) the ciphertext is decrypted if the key containing a set of attributes satisfies the access policy tree; and (7) the decrypted message is published to the AWS cloud. The model is designed such that each data type in an IoT device is associated with attributes, which represent the topic of each sensor data type. For example, the DHT11 sensor attached to the Raspberry Pi is considered an IoT device [6,7] and generates two types of data: (A) temperature degree and (B) humidity degree. The temperature value is encrypted according to the key that contains a set of attributes and then published to the fog device.

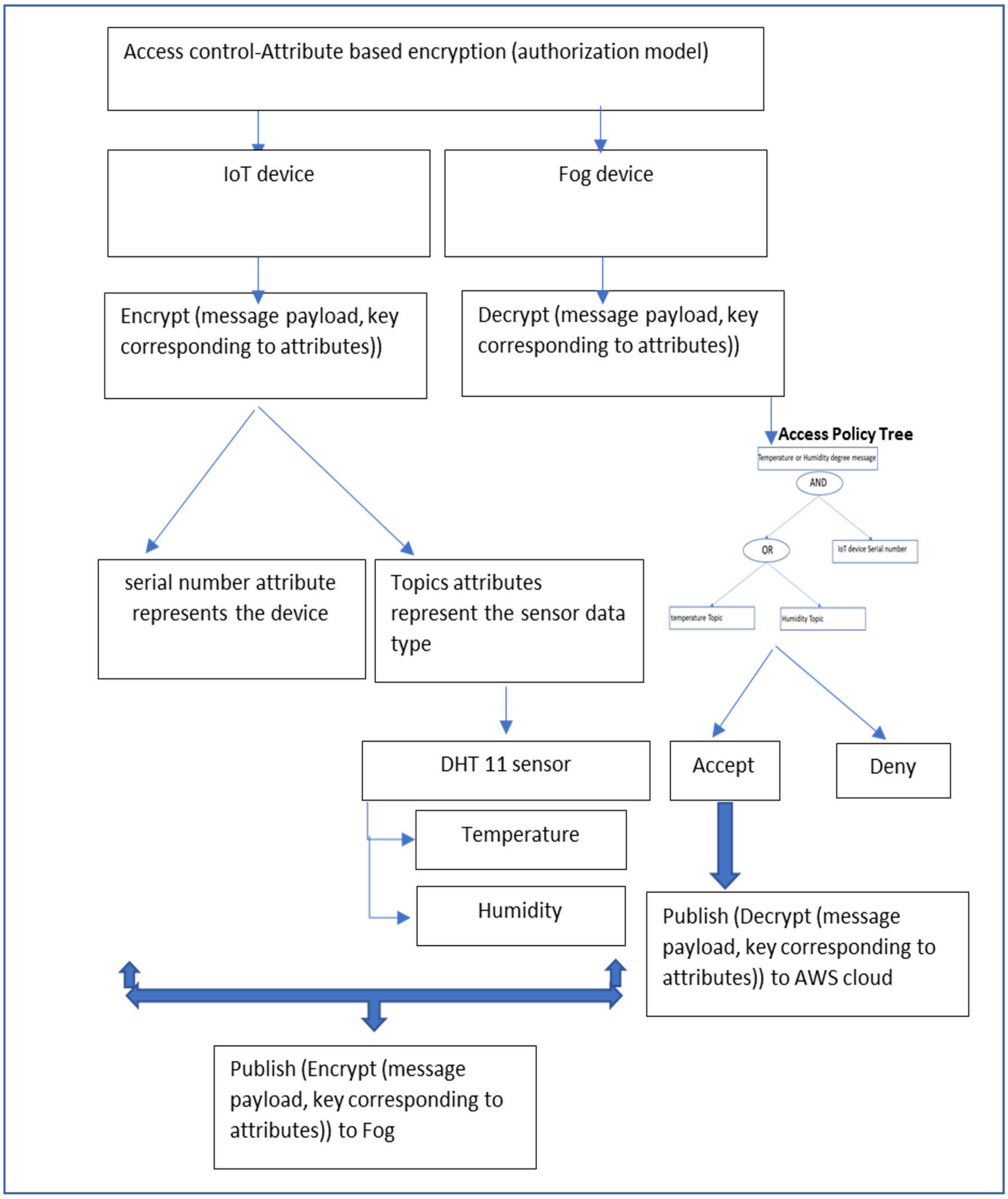
Access control-Attribute based encryption (authorization model)
IoT device
Fog device
Encrypt (message payload, key corresponding to attributes))
Decrypt (message payload, key corresponding to attributes))
Access Policy Tree
Temperature or Humidity degree message
AND
OR
IoT device Serial number
temperature Topic
Humidity Topic
serial number attribute represents the device
Topics attributes represent the sensor data type
DHT 11 sensor
Accept
Deny
Temperature
Humidity
Publish (Decrypt (message payload, key corresponding to attributes)) to AWS cloud
Publish (Encrypt (message payload, key corresponding to attributes)) to Fog

Figure 3. Access control model for the IoT–Fog–Cloud architecture.

The fog device then generates an access policy tree according to the attributes of each data type in the IoT device, an example of which is presented in Figure 4. Once the fog device receives the ciphertext from the IoT device, it decrypts it if the ciphertext key that contains the attributes satisfies the access policy tree. Otherwise, it will decline the decryption request. Then, the temperature degree value will be published to the AWS cloud. Figure 5 shows how the ABE for the access control model was applied to the IoT–Fog–Cloud architecture.

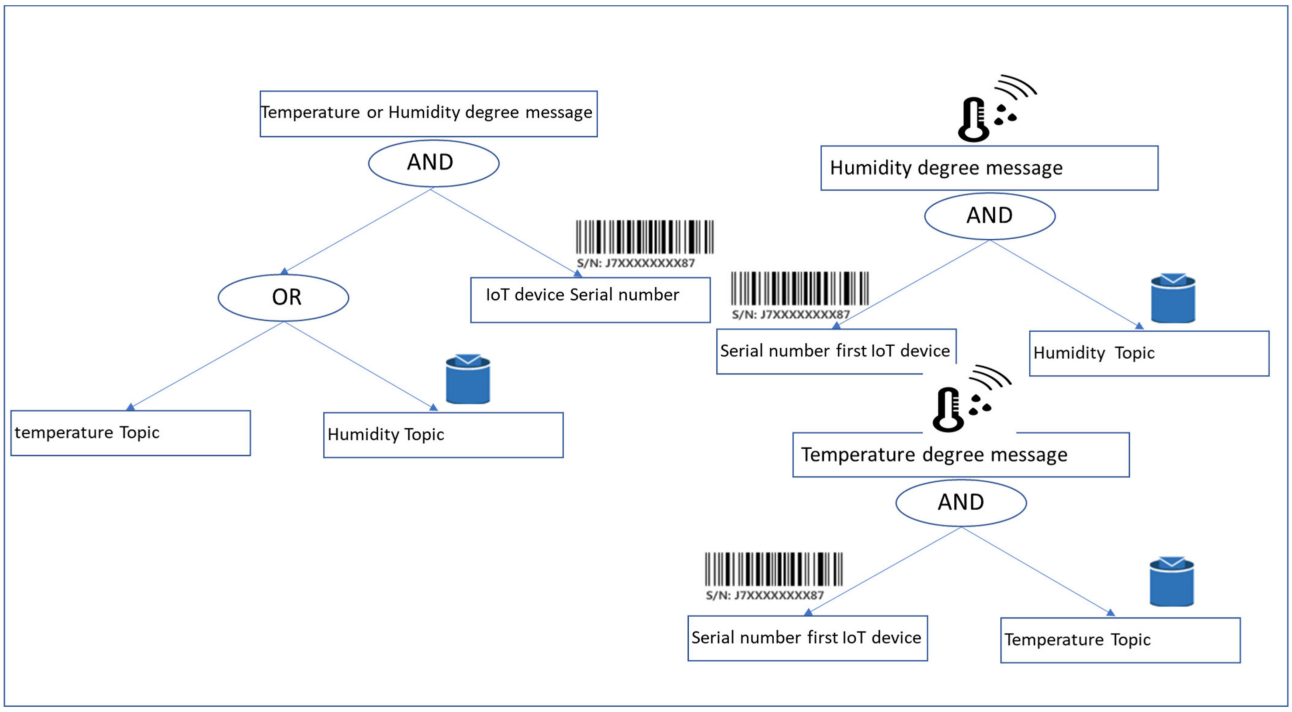

Figure 4. Access policy tree for the access control model for the IoT–Fog–Cloud architecture.

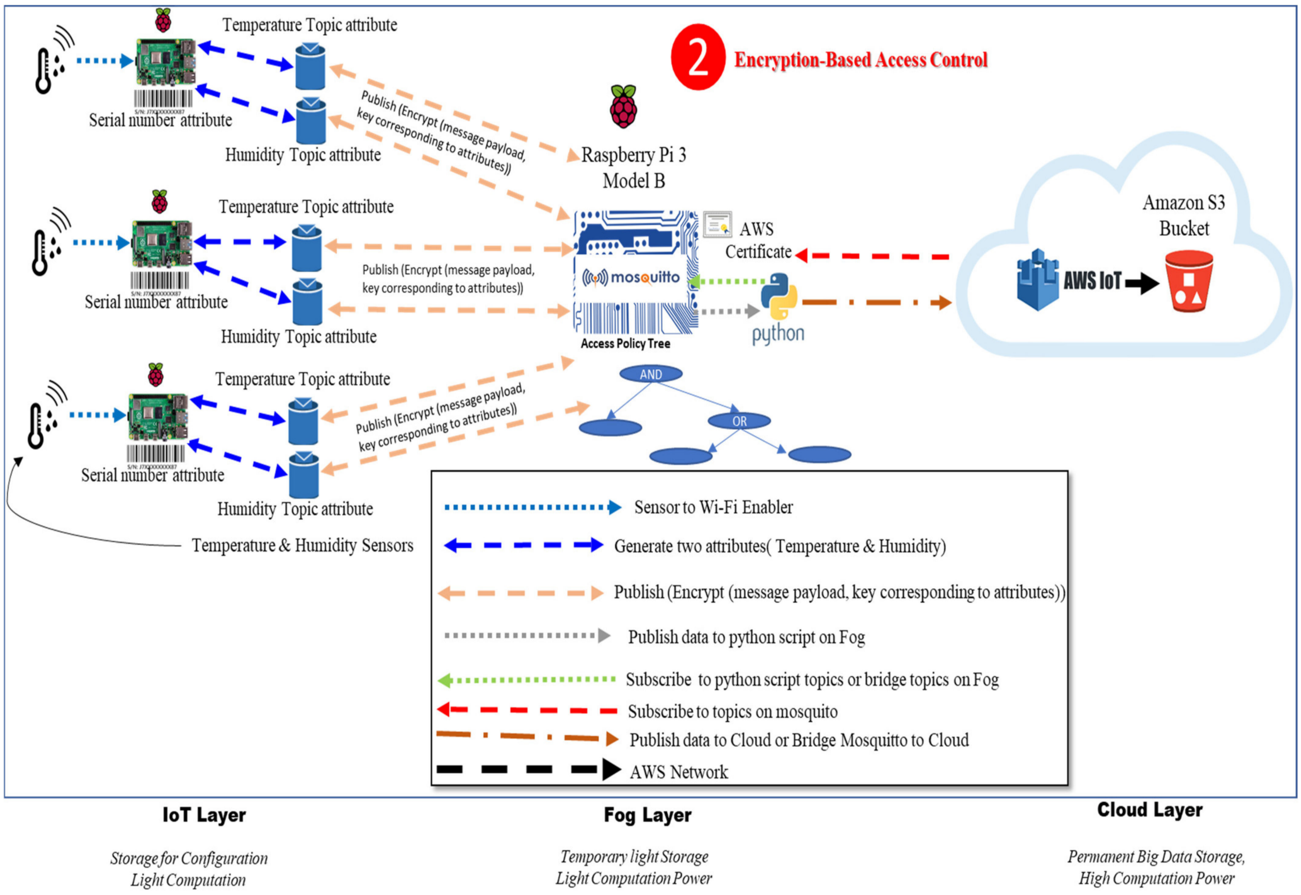
Temperature Topic attribute
Serial number attribute
Humidity Topic attribute
2 Encryption-Based Access Control
Publish (Encrypt (message payload, key corresponding to attributes))
Raspberry Pi 3 Model B
AWS Certificate
mosquitto
Access Policy Tree
python
Amazon S3 Bucket
AWS IoT
AND
OR
Temperature & Humidity Sensors
Sensor to Wi-Fi Enabler
Generate two attributes(Temperature & Humidity)
Publish (Encrypt (message payload, key corresponding to attributes))
Publish data to python script on Fog
Subscribe to python script topics or bridge topics on Fog
Subscribe to topics on mosquitto
Publish data to Cloud or Bridge Mosquitto to Cloud
AWS Network
IoT Layer
Storage for Configuration
Light Computation
Fog Layer
Temporary light Storage
Light Computation Power
Cloud Layer
Permanent Big Data Storage,
High Computation Power

Figure 5. Access control model applied to the IoT–Fog–Cloud architecture.

4. Experiment Setup and Evaluation Metrics

In this paper, we propose two security models of the IoT–Fog–Cloud architecture [6] to meet the security requirements [4,9]. The following subsections present the hardware, software configurations, and evaluation metrics used in our IoT–Fog–Cloud architecture experiments.

4.1. Hardware

In this section, the experiments are based on the first architecture of IoT–Fog–Cloud which was proposed in [6], as shown in Figure 1. First, we describe, in detail, the types of devices that were used in the IoT–Fog–Cloud architecture as shown in Figure 6. The devices used to perform the experiments in the IoT–Fog–Cloud architecture were a DHT11 sensor [14] and the Raspberry Pi 3 Model B [15]. The DHT11 sensor is a low-cost device that produces real-time data for measuring the temperature degree and humidity degree of the surrounding air. The Raspberry Pi is a single-board computer with integrated Wi-Fi and computing capabilities that is used in a variety of applications, including smart health care, weather monitoring, and smart homes. The Raspberry Pi was used in our work to provide light computational capabilities for data generated by the DHT11 sensor. In addition, the Raspberry Pi was used to provide light storage for the configuration performed by the DHT11 sensor. One of the advantages of using the Raspberry Pi is that it can be readily transferred from one location to another. Table 1 shows a complete list of the hardware utilized in the IoT–Fog–Cloud architecture. Figure 6 shows the hardware used in the IoT–Fog–Cloud architecture.

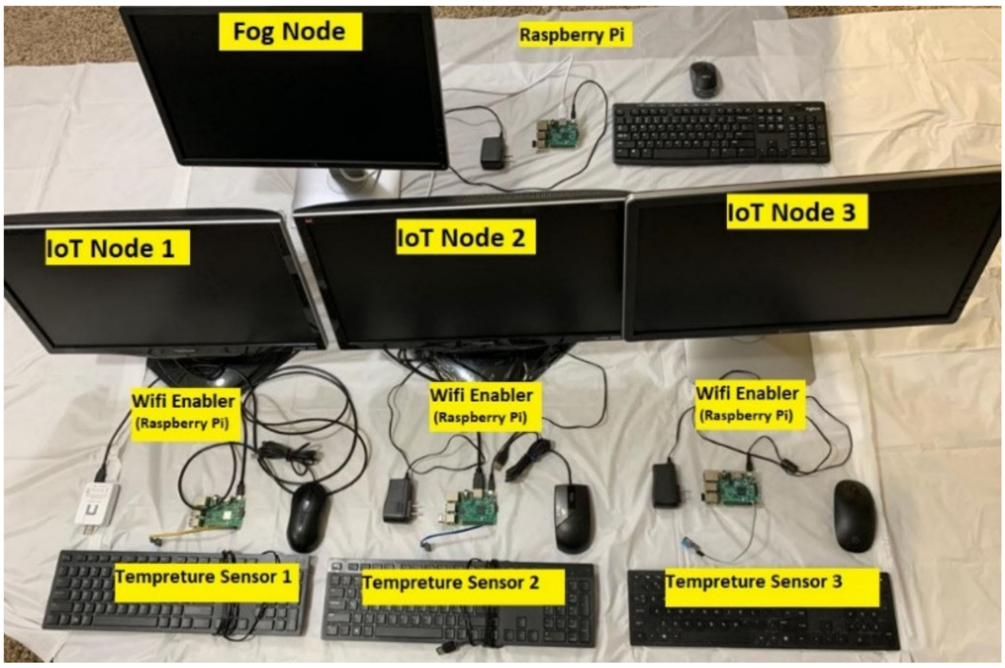

Figure 6. Hardware used in the IoT–Fog–Cloud architecture.

Table 1. Complete list of the hardware utilized in the IoT–Fog–Cloud architecture.

Equipment Name	Equipment Type	Quantity	Purpose
DHT11	Sensor	3	Generates temperature degree and humidity degree
Raspberry Pi	Version 3 (Model B)	4	Provides WiFi service and processing and storage capabilities
Micro SD card	32 GB of storage	4	Operating system storage
Monitor	HP	4	Monitors the experiments
Keyboards and mice	HP	4	Make it easier to work on a Raspberry Pi
Power supply/adapter	CanaKit	4	Provides the Raspberry Pi with power
HDMI cable	onn	4	Provides the connection between the Raspberry Pi and a monitor

4.2. Software

We installed Python on the three IoT devices (DHT sensor + Raspberry Pi) and the fog device. Then, we installed the Circuit Python DHT Library on the three IoT devices to allow communication between the DHT11 sensor and Raspberry Pi. Next, we installed the cryptography library on the three IoT devices and fog device to perform the security operations. Algorithm 1 illustrates the authentication and authorization operations from IoT devices (DHT11 sensor + Raspberry Pi) to the fog device, and Algorithm 2 illustrates the authentication and authorization operations from the fog device to the IoT devices (DHT11 sensor + Raspberry Pi) and from the fog device to the AWS cloud.

4.3. Evaluation Metrics: AWS Cloud Metrics

The performance of the IoT–Fog–Cloud architecture can be evaluated using a variety of metrics from the cloud side. Since we chose AWS as our cloud service provider, we used Amazon CloudWatch to evaluate the IoT–Fog–Cloud architecture performance [6,7,13,16,17]. We used Amazon CloudWatch to evaluate the IoT–Fog–Cloud architecture performance since it collects and evaluates data at the same time and gives the following metric, as shown in Table 2.

Table 2. Descriptions of AWS cloud metrics.

AWS Cloud Metrics	Descriptions
connect.success	This is used to count how many successful connections our fog devices made with the AWS cloud.
ping.success	This is used to count how many ping messages our fog devices sent to the AWS cloud in the IoT–Fog–Cloud architecture.
publishin.success	This is used to count how many publish requests were processed by the AWS cloud.
publishout.success	This is used to count how many publish requests were made by the AWS cloud to the fog devices in the IoT–Fog–Cloud architecture.
subscribe.success	This is used to count how many subscribe requests were processed by the AWS cloud.
unsubscribe.success	This is used to count how many unsubscribe requests were processed by the AWS cloud.

Algorithm 1: Collect data from the IoT device and send it to the fog device–IoT–Fog–Cloud architecture. This algorithm provides authentication and authorization operations from IoT devices (DHT11 sensor + Raspberry Pi) to fog devices.

```
1:  Import board
2:  Import adafruit_dht
3:  Import paho.mqtt.client as mqtt
4:  From Crypto.Cipher import ABS
5:  Import base64
6:  Define the DHT sensor type (DHT11)
7   Define the Raspberry Pi's input/output pins to which the DHT11 is connected
8:  Define a Python library (adafruit_dht:DHT11) to read the DHT series of humidity and temperature sensors on a Raspberry
    Pi with one argument, DHT pin connected
9:  Define the key length which must be either 16, 24, or 32 bytes long
10: Define humidity topic variable and temperature topic variable for each IoT device (DHT11 attached to Raspberry Pi) in
    each experiment
11: Define MQTT broker variable
12: Define the variable of MQTT port
13: While True do
14:     Define a connection function
15:         Connect to Internet
16:         If (the connection is established) then
17:             Print "connected"
18:         Else (the connection is not established) then
19:             Try reconnecting to Internet
20:     Define a message function
21:         Read humidity degree from Raspberry Pi serial port using (dhtDevice.humidity)
22:         Read temperature degree from Raspberry Pi serial port using (dhtDevice.temperature)
23:         Print humidity degree
24:         Print temperature degree
25:         Generate keys containing a set of attributes for each sensor data type in each IoT device
26:         Generate a first key containing a set of attributes for temperature sensor in each IoT device
27:         Generate a second key containing a set of attributes for humidity sensor in each IoT device
28:         Create the cipher config for first key (temperature sensor)
29:         Create the cipher config for second key (humidity sensor)
30:         Use the cipher of the first key to encrypt the humidity degree message using cipher.encrypt
31:         Use the cipher of second key to encrypt the temperature degree message using
    cipher.encrypt
32:         Encode the cipher and humidity degree message using the base64 module
33:         Encode the cipher and temperature degree message using the base64 module
34:         Print the encrypted message
35:         Publish humidity topic with its encrypted message to fog device
36:         Publish temperature topic with its encrypted message to fog device
37: End while
38: Create a client to connect to fog device
39: Make the client run connect, and message function
40: Enable the transport layer security using fog device certificates and MQTT protocol version
41: Connect the client to the MQTT broker using fog device's IP address and MQTT port 1883
42: Call a loop_start() method for the client connection
```

Algorithm 2: Collect data received from the fog device and send it to the cloud–IoT–Fog–Cloud architecture. This algorithm provides authentication and authorization operations from fog devices to IoT devices (DHT11 sensor + Raspberry Pi) and from fog devices to the AWS cloud.

```
1:  Import sys
2:  Import ssl
3:  Import adafruit_dht
4:  Import paho.mqtt.client as mqtt
5:  From Crypto.Cipher import ABS
6:  Import base64
7:  Define the key length which must be either 16, 24, or 32 bytes long
8:  Define the MQTT broker variable
9:  Define the MQTT port variable
10: While True do
11:         Define a connection function
12:                 Subscribe for all topics in each IoT devices
13:                 Connect to Internet
14:                 If (the connection is established) then
15:                     Print "connected"
16:                 Else (the connection is not established) then
17:                     Try reconnecting to Internet
18:         Define a message function
19:                 Define keys containing a set of attributes for each sensor data type in each IoT device based on access policy
20:                 Define a first key containing a set of attributes for temperature sensor in each IoT device
21:                 Define a second key containing a set of attributes for humidity sensor in each IoT device
22:                 Create the cipher config for first key (temperature sensor)
23:                 Create the cipher config for second key (humidity sensor)
24:                 Decode the encrypted message using the base64 module
25:                 Use the cipher of the first key to decrypt the humidity degree message using cipher.decrypt
26:                 Use the cipher of second key to decrypt the temperature degree message using cipher.decrypt
27:                 Print the decrypted message
28:                 Publish humidity topic with its decrypted message to AWS cloud
29:                 Publish temperature topic with its decrypted message to AWS cloud
30: end while
31: Create two clients, the first client used for the MQTT broker, and the second client used for the AWS broker
32: Make the first client run connect, and message function
33: Connect the first client to MQTT broker using fog device's IP address and MQTT port
34: Call a loop_start() method for the first client connection
35: Enable the transport layer security for the second client using the AWS certificates paths and MQTT protocol version
36: Connect the second client to AWS broker using AWS Endpoint and AWS port
37: Call a loop_start () method for the second client connection
```

5. Evaluation Methods

AWS cloud metrics were used to evaluate the performance of the blockchain-based certificate model and the fine-grained data access control model based on the ABE for the IoT–Fog–Cloud architecture proposed in our previous papers [6,7]. We defined the subscribe and publish request number as two for each IoT device because each IoT device generated two types of data: (A) temperature degree and (B) humidity degree. Therefore, when the number of IoT devices increases, the numbers of subscribe and publish requests will also increase. This produces accurate and real results about the performance of the blockchain-based certificate model and the fine-grained data access control model based on the ABE for the IoT–Fog–Cloud architecture. In this section, we provide the methods used to evaluate the performance of the blockchain-based certificate model and the fine-grained data access control model based on the ABE for the IoT–Fog–Cloud architecture.

5.1. IoT–Fog–Cloud Architecture with Blockchain-Based Certificate Model versus without Blockchain-Based Certificate Model

The performance of the IoT–Fog–Cloud architecture with the blockchain-based certificate model was analyzed and compared with the architecture without the blockchain-based certificate model, which was presented in the previous work [6], using AWS metrics. The experiment was conducted using different numbers of IoT devices (one, two, or three) two times, once with the blockchain-based certificate model and once without. The results were compared and analyzed to show the impacts of our blockchain-based certificate model on the proposed IoT–Fog–Cloud architecture [6,7]. Because the AWS cloud provider requires a certificate to authenticate any device, and because the fog device is the next layer, the certificate was placed in the Raspberry Pi that served as the fog device. The communication between the IoT device and the fog device was left without an authentication model [6]; this gap was filled by the proposed security model. The fog device distributed the certificates to the IoT devices after a connection request was made by the IoT devices. The fog device certificate was placed in each IoT device to allow them to be authenticated to the fog device. The objective of this method was to illustrate the impact of the blockchain-based certificate model on the IoT–Fog–Cloud architecture and that the performance of the model remains identical when using different numbers of IoT devices.

5.2. IoT–Fog–Cloud Architecture with Access Control Model versus without Access Control Model

The performance of the IoT–Fog–Cloud architecture with the access control model was evaluated by comparison with the architecture without the model [6]. The experiment was performed using one, two, and three IoT devices. The experiment was performed twice, once with the access control model and once without. The results were compared and evaluated to show the impact of our access control model on the proposed IoT–Fog–Cloud architecture [6]. Since the IoT devices were authenticated to the fog device using a blockchain-based certificate model, the sensor data needed to be unavailable to the other IoT devices and have limited access. Therefore, the access control model proposed in this paper fills the authorization requirement gap between the IoT layer and fog layer. The objective of the analysis method was to show the impact of the access control model on the IoT–Fog–Cloud architecture and how the performance changed using different numbers of IoT devices.

6. Evaluation of Results

6.1. IoT–Fog–Cloud Architecture with Blockchain-Based Certificate Model versus without Blockchain-Based Certificate Model

In this subsection, we evaluate the IoT–Fog–Cloud architecture with our blockchain-based certificate model using AWS cloud metrics, as shown in Table 3.

The first experiment of the IoT–Fog–Cloud architecture was performed using one IoT device. We ran the first experiment twice simultaneously, one without our blockchain-based certificate model and the other with the model. We used two subscribes and two publishes because the IoT device generated two types of data: (A) temperature degree and (B) humidity degree. The results show that the numbers of subscribes and publishes (i.e., subscribe.success and connect.success) for the IoT–Fog–Cloud architecture with and without the blockchain-based certificate model were the same and reflect the defined number of subscribes and publishes for one IoT device. This is because the connection of neither experiment (with versus without the security model) was disconnected and, thus, the subscribe request was not lost. Although the first experiment with the security model had a certificate in the authentication process of the IoT–fog–cloud layers, it did not affect the number of subscribes and publishes. Furthermore, the number of published messages (publishout.success and publishin.success) for the IoT–Fog–Cloud architecture with the blockchain-based certificate model remained the same as that of the architecture without the model.

Table 3. AWS cloud metric results on N. Virginia datacenter for the IoT–Fog–Cloud architecture with the blockchain-based certificate model versus without the blockchain-based certificate model.

AWS Cloud Metrics (N. Virginia Datacenter)—Python Script—IoT–Fog–Cloud Architecture <u>without</u> Blockchain-Based Certificate Model						**AWS Cloud Metrics (N. Virginia Datacenter)—Python Script—IoT–Fog–Cloud Architecture <u>with</u> Blockchain-Based Certificate Model**					
IoT devices	1					IoT devices	1				
Subscribe and publish requests	2					Subscribe and publish requests	2				
AWS cloud metrics (minutes)	0.5	1	5	15	60	AWS cloud metrics (minutes)	0.5	1	5	15	60
connect.success	2	2	2	2	2	connect.success	2	2	2	2	2
ping.success	2	2	8	29	120	ping.success	2	2	8	29	120
publishin.success	44	44	206	586	2360	publishin.success	44	44	206	586	2360
publishout.success	44	44	206	586	2360	publishout.success	44	44	206	586	2360
subscribe.success	2	2	2	2	2	subscribe.success	2	2	2	2	2
unsubscribe.success	2	2	2	2	2	unsubscribe.success	2	2	2	2	2
IoT devices			2			IoT devices			2		
Subscribe and publish requests			4			Subscribe and publish requests			4		
AWS cloud metrics (minutes)	0.5	1	5	15	60	AWS cloud metrics (minutes)	0.5	1	5	15	60
connect.success	4	4	4	4	4	connect.success	4	4	4	4	4
ping.success	2	2	8	29	120	ping.success	2	2	8	29	120
publishin.success	70	70	350	1110	4590	publishin.success	70	70	350	1110	4590
publishout.success	70	70	350	1110	4590	publishout.success	70	70	350	1110	4590
subscribe.success	4	4	4	4	4	subscribe.success	4	4	4	4	4
unsubscribe.success	4	4	4	4	4	unsubscribe.success	4	4	4	4	4
IoT devices			3			IoT devices			3		
Subscribe and publish requests			6			Subscribe and publish requests			6		
AWS cloud metrics (minutes)	0.5	1	5	15	60	AWS cloud metrics (minutes)	0.5	1	5	15	60
connect.success	6	6	6	6	6	connect.success	6	6	6	6	6
ping.success	2	2	9	29	119	ping.success	2	2	9	29	119
publishin.success	115	115	548	1660	6510	publishin.success	115	115	548	1660	6510
publishout.success	115	115	548	1660	6510	publishout.success	115	115	548	1660	6510
subscribe.success	6	6	6	6	6	subscribe.success	6	6	6	6	6
unsubscribe.success	6	6	6	6	6	unsubscribe.success	6	6	6	6	6

The second and third experiments of the IoT–Fog–Cloud architecture were performed using two and three IoT devices with four and six subscribes and publishes, respectively. The results show that the numbers of subscribes and publishes (i.e., subscribe.success and connect.success) were the same for the IoT–Fog–Cloud architecture with and without the blockchain-based certificate model and reflect the defined number of subscribes and publishes for two or three IoT devices. This is because when the fog device distributed the certificates to the two or three IoT devices, those devices were authenticated simultaneously to the fog device. Therefore, there was no sign of failure in the number of connects and subscribes because the two or three IoT devices remained authenticated to the fog device and started publishing messages. Moreover, the number of published messages (publishout.success and publishin.success) for the IoT–Fog–Cloud architecture with and without the blockchain-based certificate model also remained identical using two or three IoT devices; there was no loss in the number of published messages, as there was no sign of failure in the number of connects and subscribes (connect.success and subscribe.success) because the two or three IoT devices remain authenticated to the fog device and start publishing messages at the same time.

Overall, we found that the performance of the IoT–Fog–Cloud architecture with and without the blockchain-based certificate model was the same when using one, two, or three IoT devices. Thus, there was no delay in the number of published messages for the IoT–Fog–Cloud architecture with the blockchain-based certificate model, as shown in Table 3. This is because the first layer of security requirements, authentication, was proposed and added to the IoT–Fog–Cloud architecture, and it did not affect its performance. This means that the IoT–Fog–Cloud architecture had improved performance and security simultaneously.

6.2. IoT–Fog–Cloud Architecture with Access Control Model versus without Access Control Model

In this subsection, we evaluate the IoT–Fog–Cloud architecture with our access control model using AWS cloud metrics, as shown in Table 4.

The first experiment of the IoT–Fog–Cloud architecture was conducted using one IoT device. We ran this experiment twice simultaneously, one instance without our access control model and the other with the model. We used two subscribes and two publishes because the IoT device generated two types of data: (A) temperature degree and (B) humidity degree. The results show that the numbers of subscribes and publishes (i.e., subscribe.success and connect.success) for the IoT–Fog–Cloud architecture with and without the access control model were the same and reflect the defined number of subscribes and publishes for one IoT device. This is because the connection of neither experiment (with versus without the access control model) was disconnected, so the subscribe request was not lost. Although the first experiment with the security model had a certificate in the authentication process of the IoT–fog–cloud layers, it did not affect the number of subscribes and publishes. In contrast, the number of published messages (publishout.success andpublishin.success) for the IoT–Fog–Cloud architecture with the access control model was slightly less than that of the architecture without the model. This is because the IoT device performed some security operations that took one second for each sensor data type. Each sensor data type (i.e., temperature data and humidity data) took one second to generate keys containing a set of attributes and encrypt the generated sensor data type value with the corresponding key containing its attribute.

The second experiment of the IoT–Fog–Cloud architecture was performed by connecting two IoT devices and making four subscribe and publish requests. This experiment was also run twice simultaneously, with and without our access control model. The results show that the numbers of subscribes and publishes (i.e., subscribe.success and connect.success) for the IoT–Fog–Cloud architecture with and without the access control model were the same for two IoT devices and match the defined number of subscribe and publish requests for one IoT device. However, the number of published messages (publishout.success and publishin.success) for the architecture with the access control model was slightly less than that of the architecture without the model. This is because each of the two IoT devices performed security operations, which took one second for each sensor data type (i.e., temperature data or humidity data) for each of the two IoT devices. Each sensor data type (i.e., temperature data or humidity data) of each of the two IoT devices took one second to generate a key containing a set of attributes and encrypt the generated sensor data type value with the corresponding key containing its attribute.

Overall, we found that the performance of the IoT–Fog–Cloud architecture without the access control model was slightly better than that of the architecture with the model when using one, two, or three IoT devices. However, when using one, two, or three IoT devices, the number of published messages was delayed two seconds when using the access control model, as shown in Table 4. This is because the second layer of security requirements, authorization, was proposed and added to the IoT–Fog–Cloud architecture. Therefore, this gives the IoT–Fog–Cloud architecture a better performance and security at the same time.

Table 4. AWS cloud metric results on N. Virginia datacenter for the IoT–Fog–Cloud architecture with the access control model versus without the access control model.

AWS Cloud Metrics (N. Virginia Datacenter)—Python Script—IoT–Fog–Cloud Architecture without Access Control Model						**AWS Cloud Metrics (N. Virginia Datacenter)—Python Script—IoT–Fog–Cloud Architecture with Access Control Model**					
IoT devices	1					IoT devices	1				
Subscribe and publish requests	2					Subscribe and publish requests	2				
AWS cloud metrics (minutes)	0.5	1	5	15	60	AWS cloud metrics (minutes)	0.5	1	5	15	60
connect.success	2	2	2	2	2	connect.success	2	2	2	2	2
ping.success	2	2	8	29	120	ping.success	2	2	8	29	120
publishin.success	44	44	206	586	2360	publishin.success	42	42	204	584	2358
publishout.success	44	44	206	586	2360	publishout.success	42	42	204	584	2358
subscribe.success	2	2	2	2	2	subscribe.success	2	2	2	2	2
unsubscribe.success	2	2	2	2	2	unsubscribe.success	2	2	2	2	2
IoT devices			2			IoT devices			2		
Subscribe and publish requests			4			Subscribe and publish requests			4		
AWS cloud metrics (minutes)	0.5	1	5	15	60	AWS cloud metrics (minutes)	0.5	1	5	15	60
connect.success	4	4	4	4	4	connect.success	4	4	4	4	4
ping.success	2	2	8	29	120	ping.success	2	2	8	29	120
publishin.success	70	70	350	1110	4590	publishin.success	68	68	348	1108	4588
publishout.success	70	70	350	1110	4590	publishout.success	68	68	348	1108	4588
subscribe.success	4	4	4	4	4	subscribe.success	4	4	4	4	4
unsubscribe.success	4	4	4	4	4	unsubscribe.success	4	4	4	4	4
IoT devices			3			IoT devices			3		
Subscribe and publish requests			6			Subscribe and publish requests			6		
AWS cloud metrics (minutes)	0.5	1	5	15	60	AWS cloud metrics (minutes)	0.5	1	5	15	60
connect.success	6	6	6	6	6	connect.success	6	6	6	6	6
ping.success	2	2	9	29	119	ping.success	2	2	9	29	119
publishin.success	115	115	548	1660	6510	publishin.success	113	113	546	1658	6508
publishout.success	115	115	548	1660	6510	publishout.success	113	113	546	1658	6508
subscribe.success	6	6	6	6	6	subscribe.success	6	6	6	6	6
unsubscribe.success	6	6	6	6	6	unsubscribe.success	6	6	6	6	6

7. Conclusions

In this paper, we proposed a blockchain-based certificate model and a fine-grained data access control model based on the ABE for the IoT–Fog–Cloud architecture using a real environment. The two proposed models meet the authentication and authorization security requirements of the architecture. We ran a number of experiments and increased the numbers of IoT devices, subscribes, and publishes in each experiment. We used AWS cloud metrics to evaluate the performance of the models based on the evaluation methods. First, we evaluated the IoT–Fog–Cloud architecture performance with and without the blockchain-based certificate model. Second, we evaluated the IoT–Fog–Cloud architecture performance with and without the access control model. The results indicate that the performance of the IoT–Fog–Cloud architecture with and without the blockchain-based certificate model was the same when using one, two, or three IoT devices. Furthermore, the performance of the IoT–Fog–Cloud architecture without the access control model was slightly better than that of the architecture with the model when using one, two, or three IoT devices. This work aimed to improve the performance and security of the IoT–Fog–Cloud architecture.

Author Contributions: Conceptualization, M.A.A., A.A. and F.S.A.; methodology, M.A.A.; writing, M.A.A.; validation, M.A.A., A.A., F.S.A. and F.T.S.; supervision, M.A.A., A.A., F.S.A. and F.T.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the University of Jeddah, Jeddah, Saudi Arabia, under grant No. (UJ-21-ICI-6). The authors, therefore, acknowledge with thanks the University of Jeddah for its technical and financial support.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Nebbione, G.; Calzarossa, M.C. Security of IoT Application Layer Protocols: Challenges and Findings. *Future Internet* **2020**, *12*, 55. [CrossRef]
2. Yunana, K.; Alfa, A.A.; Misra, S.; Damasevicius, R.; Maskeliunas, R.; Oluranti, J. Internet of Things: Applications, Adoptions and Components—A Conceptual Overview. In Proceedings of the Hybrid Intelligent Systems; Abraham, A., Hanne, T., Castillo, O., Gandhi, N., Nogueira Rios, T., Hong, T.-P., Eds.; Springer International Publishing: Cham, Germany, 2021; pp. 494–504.
3. Zhou, W.; Jia, Y.; Peng, A.; Zhang, Y.; Liu, P. The Effect of IoT New Features on Security and Privacy: New Threats, Existing Solutions, and Challenges Yet to Be Solved. *IEEE Internet Things J.* **2019**, *6*, 1606–1616. [CrossRef]
4. Alzoubi, Y.I.; Osmanaj, V.H.; Jaradat, A.; Al-Ahmad, A. Fog Computing Security and Privacy for the Internet of Thing Applications: State-of-the-Art. *Secur. Priv.* **2021**, *4*, e145. [CrossRef]
5. Aleisa, M.A.; Abuhussein, A.; Sheldon, F.T. Access Control in Fog Computing: Challenges and Research Agenda. *IEEE Access* **2020**, *8*, 83986–83999. [CrossRef]
6. Aleisa, M.A.; Abuhussein, A.; Alsubaei, F.S.; Sheldon, F.T. Examining the Performance of Fog-Aided, Cloud-Centered IoT in a Real-World Environment. *Sensors* **2021**, *21*, 6950. [CrossRef] [PubMed]
7. Aleisa, M.; Hussein, A.A.; Alsubaei, F.; Sheldon, F.T. Performance Analysis of Two Cloud-Based IoT Implementations: Empirical Study. In Proceedings of the 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, 1–3 August 2020; IEEE: New York, NY, USA, August, 2020; pp. 276–280.
8. The DDoS Attack on Dyn's DNS Infrastructure. Available online: https://www.thousandeyes.com/blog/dyn-dns-ddos-attack/ (accessed on 13 February 2022).
9. Alrawais, A.; Alhothaily, A.; Hu, C.; Xing, X.; Cheng, X. An Attribute-Based Encryption Scheme to Secure Fog Communications. *IEEE Access* **2017**, *5*, 9131–9138. [CrossRef]
10. Khan, S.; Parkinson, S.; Qin, Y. Fog Computing Security: A Review of Current Applications and Security Solutions. *J. Cloud Comp.* **2017**, *6*, 19. [CrossRef]
11. A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications. Available online: https://ieeexplore.ieee.org/abstract/document/7879243/ (accessed on 13 February 2022).
12. Patwary, A.A.-N.; Fu, A.; Naha, R.K.; Battula, S.K.; Garg, S.; Patwary, M.A.K.; Aghasian, E. Authentication, Access Control, Privacy, Threats and Trust Management Towards Securing Fog Computing Environments: A Review. *arXiv* **2020**, arXiv:2003.00395.
13. Eclipse Mosquitto. Available online: https://mosquitto.org/ (accessed on 13 February 2022).
14. Industries, A. DHT11 Basic Temperature-Humidity Sensor + Extras. Available online: https://www.adafruit.com/product/386 (accessed on 20 November 2020).
15. Foundation, T.R.P. Buy a Raspberry Pi 3 Model B. Available online: https://www.raspberrypi.com/products/raspberry-pi-3-model-b/ (accessed on 13 February 2022).
16. Amazon CloudWatch Documentation. Available online: https://docs.aws.amazon.com/cloudwatch/index.html (accessed on 13 February 2022).
17. AWS IoT Core Documentation. Available online: https://docs.aws.amazon.com/iot/ (accessed on 13 February 2022).