Development of Authentication Techniques on Fog Computing

Muhammad Kamal Nadjieb
Informatics/Computer Science
School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Bandung, Indonesia
kamal.nadjieb@gmail.com

Yudistira Dwi Wardhana Asnar
Informatics/Computer Science
School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Bandung, Indonesia
yudis@informatika.org

***Abstract*—Fog computing is a computational approach where the computing capabilities of the cloud are extended to the edge network. The purpose of fog computing is to reduce data transmission latency between end-devices and centralized computing or can also be referred to as cloud. The fog computing approach is suitable to be implemented on systems that require processing and action on a data in real time. Authentication in fog computing is a challenge because here are involved devices that have limitations in terms of computing resources. The authentication solutions that are commonly applied to the world of web and mobile cannot be directly applied to the world of fog computing because it usually requires a lot of computing resources if using these methods. This paper introduces some methods to authenticate devices and messages in fog computing that are friendly to computing resources of fog computing devices. For device authentication, authentication with encryption approach is used with password and challenge question techniques. For message authentication, HMAC is used to produce MAC that corresponded to the creator of the message. The key for message authentication is produced by key agreement process with Elliptic-Curve Diffie-Hellman method.**

Keywords—fog computing, authentication, cryptography

I. INTRODUCTION

Fog computing is a concept where the computing capability of the cloud is extended to the edge network where data is generated and action on the environment is carried out. Formally, fog computing is a layered model for enabling ubiquitous access to a shared continuum of scalable computing resources [1]. The purpose of the fog computing concept is to reduce the latency of data transmission between devices and computing services to achieve real time conditions. Real-time data processing is carried out in the fog layer while long-term data processing is carried out in the cloud layer.

Fig. 1. show a broader picture of cloud-based ecosystems that serve smart end-devices. Fog computing acts as an intermediary between centralized services and end-devices. In the layer of fog computing also occurs computing, storage, learning, and data visualization as is common in cloud computing. Diferent usecase scenarios might have different architectures based on the optimal approach to supporting end-devices functionality.

There are several characteristics that are essential in distinguishing fog computing from other computing paradigms [1].

- Contextual location awareness, and low latency.
- Geographical distribution.
- Heterogeneity.
- Interoperability and federation.
- Real-time interactions.
- Scalability and agility of federated, fog node clusters.

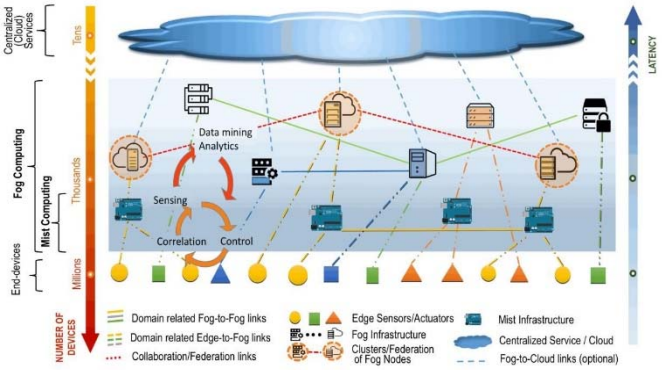

Fig. 1. Fog layer is located between end-devices and centralized (cloud) services [1]

The threats of security on fog computing are very important to be mitigated because the consequences faced are not only disclosing important information or losing money but also can be life threatening. Hardware that was not previously connected to the internet is now connected, giving rise to the possibility of new vulnerabilities and threats. Authentication is the main key in ensuring the security of fog computing that is guaranteeing that all entities involved are genuine.

Authentication in fog computing is a challenge because of several reasons. Devices that are involved in layers of things and layers of fog have limited computing capabilities. Conventional authentication techniques that exist in the web world cannot be directly applied in fog computing because they require large computing power. Authentication in fog computing itself must also maintain the main characteristics of fog computing, one of which is real-time processing.

The main contribution of this paper is to create authentication techniques that are friendly to computing resources of fog computing devices with real-time processing. Device authentication and things authentication in the fog layer and in the things layer are the main focus in this paper. To achieve this goals, three phases in our proposed techniques must be passed. The phases in order are device authentication phase, key agreement phase, and message authentication phase.

978-1-5386-9559-3/18/$31.00 ©2018 IEEE

II. Related Works

A. Authentication in the IoT – challenges and opportunities by Paul Madsen

Paul Madsen, who at the time was a Senior Technical Architect from Ping Identity, in his article discussed the authentication of IoT, the challenges of authentication on IoT, and the opportunities for the future [3]. The article begins with threats that can occur such as hacking health equipment to make readers aware of the importance of security to IoT. In a web environment, users are authenticated using passwords sent via the SSL protocol. However, the ugliness of authentication on the web has also been inherited to IoT, even worse. There is no padlock symbol commonly found on browsers on devices that do not have screens and it is not possible to enter ten digit passwords on the user's wristband.

The security token model is the most suitable approach for authentication on IoT. The actor authenticates itself by inserting a token obtained from the previous message. Tokens are used to authenticate the sender and make authorization decisions by the recipient. OAuth 2.0 and OpenID Connect 1.0 are two standards that allow authentication and authorization to be carried out. However, both implementations are only matured in the HTTPS protocol. HTTP according to many security experts is not suitable to be applied to things. There are several protocols developed for IoT including MQTT and CoAP. The exploration has begun to implement OAuth and Connect in this protocol.

There are several approaches that have been started to implement authentication on IoT. In an IoT environment that uses a smartphone, users can use it as an authentication tool. Most smartphones are considered always attached to the user so that they are quite valid to be used as an authentication tool. Biometric users can also be used as an authentication key. In this article, it is not discussed in depth about the possibility of passive authentication on IoT especially those involving devices that have limited computing capabilities.

B. Authentication and Authorization for the Internet of Things by Kim and Lee

Hokeun Kim and Edward A. Lee in a publication entitled Authentication and Authorization for the Internet of Things introduced the Auth network architecture with the concept of locally centralized and globally distributed trust management in implementing authentication and authorization [2].

Authentication and authorization between Auth and sensor and actuator devices are carried out centrally with Auth as the base station. Auth maintains important information from devices that are registered locally with the expectation that local domain experts can manage Auth and registered devices better. They assume that network granularity from local IoT devices can vary depending on the environment (Personal area network, vehicle, building, etc.). Authorization is carried out by distributing session keys for specific access activities. Auth supports TCP and UDP communication protocols via Wi-Fi and BLE and supports one-to-many communication such as broadcasting and publishing subscribe. Auth allows devices with limited resources to use longer cached session keys with lower energy-consuming cryptography.

Authentication and authorization between Auths are carried out in a distributed manner using HTTPS to communicate with certificates. The certificate is signed by other Auth who have been trusted. There is no need for a domain name like the one in the PKI.

III. Technical Specifications

The device chosen to be used as the fog node is the Raspberry Pi 3 Model B. The device chosen to be used as things is the NodeMCU DevKit Board. In the fog layer, the operating system used is Raspbian Stretch Lite. For firmware development on NodeMCU, Arduino IDE is used. The network used in communication between devices is Wi-Fi in 2.4 GHz. The transport protocol used in communication between devices is MQTT.

In the fog layer, the Python 2.7 programming language is used. In the layers of things, the C ++ programming language is used. The cryptographic library selected was cryptography for the fog layer and the Arduino Cryptography Library for the things layer. Both libraries are chosen because they are well documented and are one of the most complete cryptographic libraries in their language environment.

IV. The Authentication Phases

The general authentication scheme is divided into three phases in sequence: device authentication, key agreement for message authentication, and message authentication itself. In the first phase, both devices authenticate each other. When both devices are authenticated, the second phase is the exchange of keys for message authentication. The results of the key exchange are then used in the third phase of message authentication. The authentication process is always initiated by things. The general authentication scheme can be seen in Fig. 2.

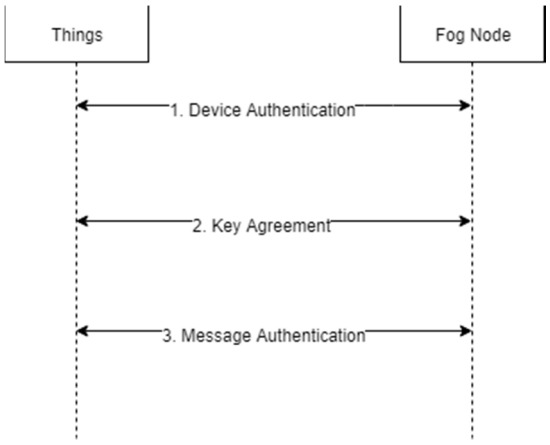

Fig. 2. Authentication general scheme

A. Device Authentication Phase

There are five alternative solutions proposed for device authentication, namely:

- PKI-RSA
- PKI-ECC
- Identity-Based Encryption
- Authentication with Encryption – Password
- Authentication with Encryption – Challenge Question

In the PKI option, a certificate is used for authentication. In PKI-RSA, the RSA asymmetric key algorithm is used to

create certificates. In PKI-ECC, an elliptic curve signature algorithm is used to create certificates.

In the IBE option, the identity of an entity is used as a public key. There is a PKG (Private Key Generator) that is responsible for generating a private key for a public key. The identity used can be either an IP address, MAC address, or an unique value specified.

In the password option, Device A sends a password that has been encrypted to Device B. Device B then decrypts the message it receives and validates it. If the password is valid then Device A is authenticated.

In the challenge question option, Device A sends a unique value that is encrypted and then sent to Device B. Device B then decrypts the message and sends it back in plain text to Device A. If the plain text is the same as the unique value generated by Device A then Device B is authenticated.

In device authentication options with passwords and challenge questions, nonce is used to avoid reply attack. Nonce is concatenated with the key in each encryption and decryption. The nonce values are unique and should not be used more than once.

Not all proposed solutions can be implemented because of several reasons. The library used in the Python environment on the fog node and Arduino on things does not support all existing cryptographic algorithms. In Table V.3. explained the results of the implementation along with the information.

TABLE I. IMPLEMENTATION RESULTS OF PROPOSED DEVICE AUTHENTICATION SOLUTIONS

Proposed Device Authentication Solution	Implementation Status	Explanation
PKI-RSA	Not Implemented	It's difficult to find an RSA implementation for Arduino. Computational resources on things are not enough to process the algorithm.
PKI-ECC	Not Implemented	The two libraries used support NIST P-521 curve but with different key size and signature size.
Identity-Based Encryption	Not Implemented	It's difficult to make PKG. The possibility of failure is also high considering that the things used are limited to computational resources.
Authentication with Encryption – Password	Implemented	Using AES algorithm with CFB method.
Authentication with Encryption – Challenge Question	Implemented	Using AES algorithm with CFB method.

B. *Key Agreement Phase*

Key agreement is needed to generate a symmetric key that is used for message authentication. The symmetric key algorithm for message authentication is chosen because the processing time and the use of computational resources from the symmetric key algorithm are lower than asymmetric key algorithms.

There are two alternative key exchange solutions proposed, namely:

- Deffie-Hellman (DH)
- Elliptic-Curve Deffie-Hellman (ECDH)

In the DH option, the power and modulo operations in arithmetic are used to exchange keys without revealing each entity's secret key. In the ECDH option, an elliptic curve equation is used to generate a private value and public value. The public value of an entity is exchanged for the public value of another entity. The result of the process of calculating private values with public values of other entities is a shared secret key.

All key agreement methods can be implemented but with some obstacles to the proposed DH solution. The parameters used for the calculation of shared secret keys in DH are in a very small range of values and are explicitly defined in the program code. Ideally the parameter values used are in a large range and are generated randomly. This cannot be done because things have limited computational resources. If you use integer data types, they have a minimum and maximum value limit that they can accommodate. If the calculation of values using string processing, it often crashes when counting secret keys together especially after getting public values from other devices.

For other alternative solutions, namely ECDH, it can be implemented properly. The two cryptographic libraries used in Python and Arduino support key concatenation with Curve25519. The keys are processed and produced far less than DH.

C. *Message Authentication Phase*

For message authentication only HMAC proposed as a solution. CBC-MAC is not included as an alternative solution because it uses an encryption-decryption algorithm that is heavier than the hash algorithm. To avoid attack reply, nonce is used which is connected with the key in each MAC making with HMAC algorithm. Nonce values are unique and should not be used more than once.

Message authentication with HMAC has been successfully implemented on fog nodes and things. Nonces

are used which is concatenated to the key to avoid reply attack. Nonce generated from fog nodes uses unix time, while things use noise generated from analog pins used.

V. TESTING AND EVALUATION

The parameters that are tested in this project are as follows:

- **Voltage consumption**. Calculated the amount of voltage consumed by things.
- **Computing time**. Calculated the time needed to carry out a process involving authentication.
- **Scalability**. Tested fog node authentication system capabilities in authenticating multiple devices simultaneously.

Voltage consumption is chosen as a parameter that is tested to determine the effect of the authentication system built on things to voltage consumption. The computation time is chosen as the parameter that is tested to find out how real time the authentication system is process. Scalability is chosen as a parameter that is tested to determine the ability of the authentication system built in the fog node to overcome the authentication process of various devices.

A. Voltage Consumption Test

There are three voltage consumption testing scenarios on things as follows:

1. Testing for device voltage consumption without authentication.
2. Testing the consumption of things voltage by authenticating the device using passwords.
3. Testing the consumption of things voltage by authenticating the device using challenge questions.

The three scenarios are tested with the following steps:

1. A 9 V battery is used as a voltage source.
2. Measured the initial voltage of the battery.
3. Batteries are used as a source of energy by things.
4. After ± 10 minutes of use, the voltage is measured again.

Here are the results of testing the consumption of stress on things that can be seen in Table 2.

TABLE II. VOLTAGE CONSUMPTION TEST RESULT

Scenario	Initial Voltage	Final Voltage	Delta
Without Authentication	9.95 V	8.47 V	1.48 V
Device Authentication with Password	9.98 V	7.84 V	2.14 V
Device Authentication with Challenge Question	9.96 V	8.36 V	1.6 V

It can be seen that the delta of the voltage on the device without authentication is less than the delta of the voltage on the device with authentication. This is because there are soime authentication processes in password authentication and challenge questions scenarios that require more energy. Power consumption testing can also be done by calculating the voltage value and the current value that flows to things. However, there are obstacles in calculating the current value that flows to things. Multimeters that are used to measure the value of current that flows do not display values. This is because the resistance on the multimeter is large and cannot be regulated by the user.

B. Computing Time Test

There are three computational time tests as follows:

1. **Testing the computation time of device authentication and key agreement on things**. In this test there are two processes that are calculated in the process time, namely the process of initiating authentication and the authentication process on the callback.
2. **Testing the computation time of device authentication and key agreement on the fog node**. In this test there are four processes that are calculated in the process time, which are the password /challenge question decryption process, key agreement, password / challenge question encryption, and public value encryption. Each process is carried out five times and then the average value and standard deviation of the computation time is calculated.
3. **Testing the computation time of message authentication on things**. Calculated the average value of computing time required in the humidity sensing process, sensing light intensity, watering action, and the action of the intensity of the lights for 100, 1000, and 10000 loops.

Each test is carried out in a process that does not involve authentication, password authentication, and challenge question authentication.

1) The Results of Computation Time Test of Device Authentication and Key Agreement on Things

The results of the average computing time of device authentication and key agreement on things can be seen in Table 3. The authentication initialization time needed in the password scenario is faster than the challenge question scenario. However, the authentication process time needed in the callback for password scenarios and challenge questions is not much different. This is because in the scenario authentication initialization process the challenge questions have random value generation so that it requires additional time compared to the password scenario which only takes the stored password value.

TABLE III. THE AVERAGE COMPUTING TIME OF DEVICE AUTHENTICATION AND KEY AGREEMENT ON THINGS

Scenario	Initialization	Callback
Password	181 ms	166 ms
Challenge Question	195 ms	167 ms

2) The Results of Computation Time Test of Device Authentication and Key Agreement on The Fog Node

The results of the average computing time of the device authentication and key agreement on the fog node can be seen in Table 4. It takes a long computing time to carry out device authentication and key agreement. When compared to the device authentication process and key agreement that occurs in things, the same process on the fog node takes a much longer time. In fact, computing resources on fog nodes are bigger than things.

TABLE IV. THE AVERAGE COMPUTING TIME OF THE DEVICE AUTHENTICATION AND KEY AGREEMENT ON THE FOG NODE

Scenario	Password / Challenge Question Decryption (in seconds)	Key Agreement (in seconds)	Password / Challenge Question Encryption (in seconds)	Public Value for Key Agreement Encryption (in seconds)
Password	20.4844	6.3476	2.8616	4.8094
Challenge Question	23.6132	4.7464	2.7044	2.2622

The results of the standard deviation of the device authentication computation time and key agreement on the fog node can be seen in Table 5. The standard deviation from the results of testing the device authentication time and key agreement is quite large. This indicates that the time required in a process varies.

TABLE V. THE STANDARD DEVIATION OF THE COMPUTING TIME OF DEVICE AUTHENTICATION AND KEY AGREEMENT ON THE FOG NODE

Scenario	Password / Challenge Question Decryption (in seconds)	Key Agreement (in seconds)	Password / Challenge Question Encryption (in seconds)	Public Value for Key Agreement Encryption (in seconds)
Password	14.6158	4.193628	4.596699	5.035066
Challenge Question	14.04308	3.388331	2.27613	2.113952

Based on the results of the process time testing that has been done, it takes a lot of computational processing time on the fog node. This is the opposite of what happens to things where the computational time needed is still acceptable.

3) The Results of Computation Time Test of Message Authentication on Things

As a comparison, in Table 6. it shows the processing time required by things in sensing the environment and taking action on the environment without authentication. The computation time needed to sense is approximately 1 ms while for action on the environment it takes approximately 0.1 ms.

In Table 7. it appears that there is an additional processing time to authenticate messages on things. In the sensing process there are some considerable increase in computing time. However, there are no significant computational time increase in the action process to the environment. Although there are additional computation time, these values can still be accepted because they do not cause the system to not run in real time. This is possible because the hash algorithm is used which is designed to produce digest quickly.

TABLE VI. THE AVERAGE TIME OF THE SENSING PROCESS AND THE ACTION TO THE ENVIRONMENT ON THINGS WITHOUT MESSAGE AUTHENTICATION

Component	10000 Loop (in miliseconds)
Soil Sensor	1.20
LDR Sensor	1.02
Watering Actuator	0.09
LED Actuator	0.09

TABLE VII. THE AVERAGE TIME OF THE SENSING PROCESS AND THE ACTION TO THE ENVIRONMENT ON THINGS WITH MESSAGE AUTHENTICATION

Component	100 Loop (miliseconds)	1000 Loop (milliseconds)
Soil Sensor	17.19	18.77
LDR Sensor	17.88	19.07
Watering Actuator	0.92	0.89
LED Actuator	0.95	0.89

C. Scalability Test

There is a fog node named `fog_node_1` which is connected to another fog node named `fog_node_2` and things named `things_1`. The `fog_node_2` device doesn't connect directly with `things_1`. Fig 3 show the schema for scalability test. There are three scalability testing scenarios as follows:

- The `fog_node_2` device do authentication with the `fog_node_1` device followed by `things_1` that do authentication with the `fog_node_1` device.
- The `things_1` device do authentication with the `fog_node_1` device followed by `fog_node_2` that do authentication with the `fog_node_1` device.
- The `things_1` device and the `fog_node_2` device do authentication with the `fog_node_1` device simultaneously.

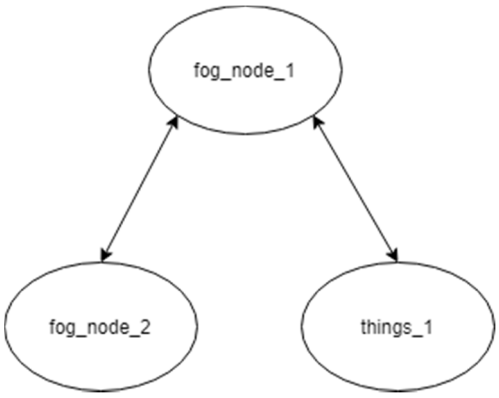

Fig. 3. Scalability test scheme

Based on the test results, the `fog_node_1` device can handle authentication requests from `fog_node_2` and `things_1` for all test scenario cases.

VI. Conclution and Future Works

Based on the research that has been done, there are some conclusiona that can be taken as follows:

1. There are two common threats to authentication on fog computing, namely device forgery and message forgery. Devices are forged so hackers can send information to authentic devices or receive data from authentic devices. Messages are forged as if the message came directly from an authentic message sender.
2. The device authentication method used in this project is authentication with encryption. Each device involved sends an encrypted message to the other. If each party can decrypt the message properly then they are authentic.
3. Cryptographic algorithms on authentication devices used in this project are symmetric key algorithms. The symmetric key algorithm is used because it requires computing capability that is not heavy and fast so that it supports the real-time nature of fog computing.
4. Key integrity integration in this project uses Elliptic-Curve Deffie-Hellman algorithm because the key agreement between devices can be done without revealing the secret key of each device with less key length compared to pure Deffie-Hellman algorithm.
5. The message authentication method used in this final project is HMAC algorithm to generate MAC. HMAC was chosen because it uses hash algorithm that is faster than cipher block algorithm.

The authentication techniques in fog computing introduced in this paper still have many things to develop. Following are some development suggestions for related research:

1. Testing in this project are only done with the MQTT application protocol. Testing with various application protocols commonly used on IoT in general and especially fog computing is required.
2. Testing on this project are only limited to Raspberry Pi and NodeMCU. Testing with various devices commonly used on IoT in general and especially fog computing is required.
3. Testing the computation time on fog nodes is better done using CPU time.

References

[1] M. Iorga, L. Feldman, R. Barton, M. J. Martin, N. Goren and C. Mahmoudi, "Fog Computing Conceptual Model," National Institute of Standards and Technology, Gaithersburg, 2018.

[2] H. Kim and E. A. Lee, "Authentication and Authorization for the Internet of Things," *Trusting the Internet of Things,* pp. 27-33, 2017.

[3] P. Madsen, "Authentication in the IoT – challenges and opportunities," 6 January 2015. [Online]. Available: https://www.secureidnews.com/news-item/authentication-in-the-iot-challenges-and-opportunities.

[4] L. Atzori, A. Iera and G. Morabito, "The Internet of Things: A survey," *Computer Networks 54,* pp. 2787-2805, 2010.

[5] N. Daswani, C. Kern and A. Kesavan, Foundations of Security: What Every Programmer Needs to Know, New York: Springer-Verlag New York, 2007.

[6] J. Gubbi, R. Buyya, S. Marusic and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems 29,* pp. 1645-1660, 2013.

[7] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," National Institute of Standards and Technology, Gaithersburg, 2011.