Authenticated Key Agreement Scheme for Fog Computing in a Health-Care Environment

Publisher: IEEE Cite This PDF

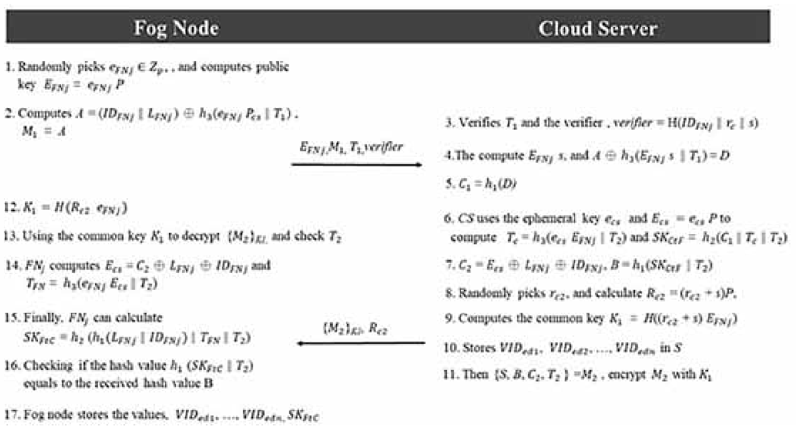
Yin-Tzu Huang ; Tzer-Shyong Chen ; Sheng-De Wang **All Authors**

5 Cites in Papers **1299** Full Text Views

PDF Help

Open Access Comment(s)

Abstract:
The application of the Internet of Things has been greatly expanded; meanwhile, real-time and efficient communication has become an important feature of the Internet of Things. However, the centralized characteristics of cloud computing cannot meet the needs of low latency and high computing efficiency. To solve these issues, we utilized fog computing which is a new distributed computing paradigm that extends cloud services to the edge of the network, with mobility and low latency. Nevertheless, fog computing also brings new security issues, especially identity authentication. Authentication and key exchange are significant challenges that need to be taken into consideration in fog computing. Therefore, in this research, we proposed the architecture of the mutual authentication key establishment scheme based on elliptic curve cryptography for fog computing. After mutual authentication, the cloud server can transfer the remaining verification work to fog nodes, and fog nodes will be responsible for authenticating the device and distributing the established session key, thereby reducing the computational cost of the cloud server. Moreover, to evaluate the security of the proposed scheme, we not only used the random oracle model and the extended Canetti-Krawczyk (eCK) threat model to conduct a detailed analysis, but also proved that the security of the proposed scheme is strong enough against several attacks.



Fog Node

1. Randomly picks $e_{FNj} \in Z_{p^*}$, and computes public key $E_{FNj} = e_{FNj} P$
2. Computes $A = (ID_{FNj} \| L_{FNj}) \oplus h_3(e_{FNj} P_{cs} \| T_1)$, $M_1 = A$

Cloud Server

3. Verifies T_1 and the verifier, $verifier = H(ID_{FNj} \| r_c \| s)$
4. The compute $E_{FNj}\ s$, and $A \oplus h_3(E_{FNj}\ s \| T_1) = D$
5. $C_1 = h_1(D)$
6. CS uses the ephemeral key e_{cs} and $E_{cs} = e_{cs} P$ to compute $T_c = h_3(e_{cs}\ E_{FNj} \| T_2)$ and $SK_{CtF} = h_2(C_1 \| T_c \| T_2)$
7. $C_2 = E_{cs} \oplus L_{FNj} \oplus ID_{FNj}$, $B = h_1(SK_{CtF} \| T_2)$
8. Randomly picks r_{c2}, and calculate $R_{c2} = (r_{c2} + s)P$.
9. Computes the common key $K_1 = H((r_{c2} + s)\ E_{FNj})$
10. Stores $VID_{ed1}, VID_{ed2}, \dots, VID_{edn}$ in S
11. Then $\{S, B, C_2, T_2\} = M_2$, encrypt M_2 with K_1

Fog Node

12. $K_1 = H(R_{c2}\ e_{FNj})$
13. Using the common key K_1 to decrypt $\{M_2\}_{K1}$ and check T_2
14. FN_j computes $E_{cs} = C_2 \oplus L_{FNj} \oplus ID_{FNj}$ and $T_{FN} = h_3(e_{FNj}\ E_{cs} \| T_2)$
15. Finally, FN_j can calculate $SK_{FtC} = h_2(h_1(L_{FNj} \| ID_{FNj}) \| T_{FN} \| T_2)$
16. Checking if the hash value $h_1(SK_{FtC} \| T_2)$ equals to the received hash value B
17. Fog node stores the values, $VID_{ed1}, \dots, VID_{edn}, SK_{FtC}$

The interaction between fog nodes and the cloud server during the mutual authentication phase.

Published in: IEEE Access (Volume: 11)

Page(s): 46871 - 46881

DOI: 10.1109/ACCESS.2023.3275017

Date of Publication: 10 May 2023

Publisher: IEEE

Electronic ISSN: 2169-3536

Funding Agency:

SECTION I. Introduction

With the rise of 5G, the amount of data generated by healthcare IoT devices has increased significantly. Besides, cloud computing has greatly expanded the potential applications of wearable medical sensors (WMS)-based systems due to its high storage capacity and flexible processing services [1]. Consequently, the storage and security of such extensive data have become major concerns [2]. Researchers and institutions around the world have been working on prototypes to leverage WMS technology and services offered by tl cloud. The benefit of keeping medical data in a centralized cloud environment is that the PHR can be shar easily [3]; however, cloud computing still faces several issues for sensitive applications, such as:

PDF

Help

(1) Data retrieval times for urgent situations are unreasonably long.

(2) Sending data to the cloud for calculations frequently requires lots of energy consumption and associated costs, especially given the volume of data produced by sensors.

(3) A typical cloud service has a severe delay and low sustained performance compared to a distributed computing architecture with numerous computing nodes in various locations.

To address the challenges of cloud computing mentioned above, a new computing paradigm called fog computing has emerged. This approach involves inserting an intermediary layer between IoT devices and cloud computing [4]. Fog-based solutions shift data processing closer to the network edge which can bring, faster response times; therefore, data can be processed on fog nodes and servers near users, as shown in Figure 1.

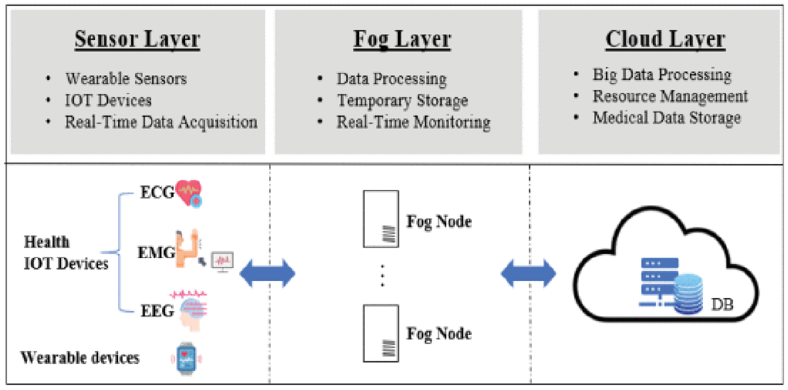


FIGURE 1.
Example of a three-layer smart healthcare architecture.

A. IoT Healthcare Device Layer

A lot of distributed IoT devices and sensors are included in this layer. They are used to monitor physical objects, to gather data, and to send data to fog nodes. For instance, embedded and wearable devices enable the gathering of user's personal health information, including blood glucose, heart rate, body pressure values, and so on. It can be utilized to build medical plans by corresponding specialists.

B. IoT Healthcare Fog Layer

Fog nodes collect data from devices and the cloud, and are responsible for transmitting collected data to the cloud for long-term storage or further analysis. In the time-sensitive scenario, these nodes can also response immediately.

C. IoT Healthcare Cloud Layer

The cloud server can handle operations which require high computational power. A cloud server also plays a role as a user interface for patients and health-care professionals (such as physicians) to monitor and to operate connected IoT devices [5].

Smart e-Health gateways process sensed information and direct it to either cloud computing or fog computing, depending on the situation. In urgent cases, fog nodes are used to gather information quickly, enabling experts to respond with minimal delay.

Although fog computing is regarded as a more efficient architecture than cloud computing [6]; it also brings new security issues, especially identity authentication [7], [8], [9], [10]. In order to rectify this problem, authentication and key exchange are significant challenges that need to be taken into consideration in fog computing.

Recently, many studies have proposed Authenticated Key Agreement (AKA) in fog computing architecture Authenticated key agreement (AKA) protocols can be utilized to authenticate each entity and produce a shared session key for each session. In [11], they proposed a method to decrease the frequency of authentication requests. In the scheme, fog nodes is utilized to check device authenticity without heavily engaging the cloud server, and reducing the computational cost of the cloud server. A message is verified without the involvement of a cloud server; however, numerous rounds of the message-exchange are used, which results in excessive delay. Placide et al. [12] created an AKA mechanism for groups of IoT devices. In this approach, the cloud server, fog nodes, and devices use Lagrange interpolation to generate the group session key based on specific points on elliptic curves (ECs); however, due to the communication of EC points, it might result in high latency. Furthermore, the authentication process is not applied to all communications, and an unauthorized device cannot always be recognized. Therefore, in this paper, we propose a mutual authentication key establishment scheme for the fog computing architecture by using elliptic curve cryptography. During the mutual authentication phase(fog node to cloud and device to fog node), the cloud server first authenticates fog nodes. After that, fog nodes will validate devices, thereby reducing the computational cost of the cloud server.

By using this method, the cloud server can transfer the remaining verification work to fog nodes, and fog nodes will deal with authenticating the device and distributing the established session key; therefore, we increase the security for preserving the privacy of medical data in a fog-based healthcare system.

This research focuses on AKA protocols designed for a health monitoring system based on fog-based IoT. Our main contributions are summarized as follows:

(1) The proposed method reduces communication costs between the cloud server and fog nodes by verifying the fog nodes through the cloud server and then having the fog nodes validate devices. This reduces the computational load on the cloud server.

(2) The proposed protocol has been proven secure in the eCK security model, and an informal security analysis has also shown that it maintains user anonymity and un-traceability while being resistant to various types of attacks.

(3) Compared with several recent schemes, our enhanced protocol outperforms others in terms of security features and communication efficiency.

The paper is organized into the following sections. In sections II and III, we review the related research and the mathematical backgrounds. In Section IV, the proposed approach, system model, and design goals are introduced. We used the random oracle model [13] and the extended Canetti-Krawczyk (eCK) threat model to conduct a detailed analysis; moreover, we also proved that the proposed scheme is secure enough against several attacks in Section V. Finally, Section VI concludes the paper.

SECTION II.
Related Work

In this section, we review a few existing approaches for key agreement and authentication in fog computing in health-care environments.

A. Fog Computing in Health-Care IoT Systems

Recent improvements in wireless sensor networks have made it possible to implement ubiquitous IoT networks in a variety of applications ranging from transportation to manufacturing and health care. The Healthcare 4.0 system is patient-driven, it utilizes a lots of IoT-based medical devices, including wearables, sensors, and smartphones, to monitor patients' real-time health status. This information can be captured by any personal computer or cell phone in real-time and securely matched with the cloud eHealth platform [14]. Regardless of the domain, IoT devices create data 24 hours a day, seven days a week, causing major performance challenges for the well-established cloud platform [15]. The traditional cloud model, which involves collecting and analyzing patients' sensitive real-time data, bio-signals, and vital signs across a wide geographical area, is a time-consuming process that is not compatible with Healthcare 4.0. The most effective method to tackle these problems is to use fog computing to extend the services to another level [16]. A few intermediary nodes, known as fog nodes, are inserted between the cloud and end users in fog computing (EUs). These nodes provide computing and storage services to EUs, improving service quality.

Several research [17], [18], [19] have concentrated on deploying fog and edge computing in health-care systems. In [19], deep learning was implemented in an integrated IoT-fog computing environment to create HealthFog, an SHS. This system was created with the goal of achieving low latency and energy-efficient data processing. HealthFog can automatically diagnose heart problems and manage heart patients' data effectively. Li et al. [20] developed a secure fog computing platform based on software-defined networking for IoT-enabled health-care systems. Despite the benefits of resource sharing in intelligent health-care systems, the lack of cloud access control methods leads to data leakage and illegal access to health-care data. Because health-care data is outsourced, it is difficult to guarantee data confidentiality and query privacy.

B. Privacy Preservation in Fog Computing

In the implementation and deployment of Healthcare 4.0, security and privacy are significant considerations, and it is crucial that unauthorized entities cannot get access to users' private information. Many authentication approaches have been developed in the field of mobile networks. The purpose of those schemes is to decrease handover durations while still ensuring that data can be protected, user anonymity particularly. Data integrity is a critical concern in sensor-based systems. Ara et al. [21] created a secure data aggregation solution by constructing the ElGamal cryptosystem for data authentication. Sun et al. [22] introduced a privacy-protected emergency response strategy for an e-healthcare framework. The bilinear pairing approach, in particular, was used to verify data integrity. In [23], the Boneh-Goh-Nissim (BGN) encryption approach is used to resist several attacks and to maintain data integrity.

Guo et al. [24] presented a fog computing authentication approach; nevertheless, their scheme does not provide user mobility. Fan et al. [25] developed the ReHand symmetric cryptography-based handover authentication. In [25], each EU's long-term key is shared among base stations in the same area. The symmetric key-based methods can remarkably devaluate the computation costs of handover authentication protocols, yet, managing keys will be difficult. Moreover, it will cause limited scalability due to the requirement for preallocating shared keys.

Saurabh et al. [10] proposed a mutual AKA protocol for fog computing. They used bilinear paring to provide mutual authentication and secure session keys. Their system, however, lacks forward security and user anonymity. Amor et al. [26] presented a mutual authentication protocol in which the fog and server authenticate each other. Their proposed method is based on pseudonym-based encryption and does not expose users' true identities, preserving EU anonymity and meeting the primary security requirements. Nevertheless, the session key exchange process between EUs and fog servers is not secure. Besides, the developed protocol requires high computation costs, making it unsuitable for resource-constrained equipment like IoT devices.

Previous research on the formation of group keys for fog computing systems has great communication costs; besides, it is also difficult for those studies to verify the validity of each entity. Chen et al. [27] suggested an elliptic curve cryptographic mutual authentication group key setup approach for the fog computing architecture. Following mutual authentication, the cloud server can shift the processing overhead to the fog node; then, the fog node needs to authenticate the device group and distribute the formed group session key. The group session key is made up of each entity's private key as well as certain random and temporarily stored variables. In their study, the scheme is proven secure under the CK security model. However, compared to Canetti-Krawczyk (CK) model, the extended Canetti-Krawczyk (eCK) model is widely used to provide security arguments for AKA protocols. The CK model introduced state information being revealed, whereas, in the eCK model, a new query to the ephemeral key is used, which is claimed to cover almost all 'session-specific secret' information. Therefore, the assumptions of the eCK model are stronger than the CK model.

SECTION III. Preliminaries

A. Elliptic Curve Cryptosystem (ECC)

In comparison to RSA and ElGamal, public-key cryptography ECC uses shorter key lengths to provide the equivalent level of encryption strength as RSA. An elliptic curve [28] is a plane curve equation that is defi as $y^2 = x^3 + Ax + B \bmod p$, where $p \geq 5$ is a prime number and $A, B \in Z_p$ are constants with the requirement $4A^3 + 27B^2 \neq 0 \bmod p$. Let (Z_p) represent the set of $(x, y) \in Z_p \times Z_p$ pairings that fulfill elliptic-curve equation. Furthermore, $E(Z_p) \overset{\text{def}}{=} (Z_p) \cup \{O\}$, where $E(Z_p)$ represents elliptic curve points and O is a unique point at infinity. It is an abelian group due to the addition rule of $E(Z_p)$. The formula for scalar multiplication is $kG = G + G + \ldots + G$, where k is the number of generators (G) . Let k be a chosen integer in the range [1, $n-1$], then (k , kG) can be an elliptic key pair [29]. Additionally, ECC creates keys that are more difficult to crack. These computational difficulties are listed below.

1) Elliptic-Curve Discrete Logarithm Problem

Given two points G and xG , where $G, xG \in E(Z_p)$ of an additive group N , calculating $x \in Z_p$ by using a polynomial-time-bound algorithm is computationally hard.

2) Elliptic-Curve Computational Diffie–Hellman Problem

Given three distinct points $G \in E(Z_p)$, $xG \in E(Z_p)$ and $yG \in E(Z_p)$, calculating xyG by using a polynomial-time-bound algorithm is computationally challenging; for some unknown variables *x, y*, where $x, y \in Z_p$.

3) Elliptic-Curve Decisional Diffie–Hellman Problem

Given the following points G , $X = xG$, $Y = yG$, and $Z = zG$ in $E(Z_p)$, for some unknown parameters *x, y,* and z , where $x, y, z \in Z_p$. To determine if Z is equal to xyG is quite difficult.

B. Extended Canetti-Krawczyk Adversary Model

To build a public-session key for communication between parties, AKA protocols are widely used. Most of the proptocals are based on the user's permanent, private, and ephemeral keys for simultaneous mutual authentication. Bellare and Rogaway [30] proposed the first secure adversarial model based on the AKA protocol. Canetti and Krawczyk [31] extended the model of Bellare and Rogaway to create the Canetti–Krawczyk (CK) model. La Macchia et al. [32] proposed an extended CK (eCK) model by accounting for more powerful adversaries. Theoretically, the eCK model can compromise the permanent private or ephemeral keys and has been extensively used to demonstrate the security of the AKA protocol [33], [34].

SECTION IV. Proposed Scheme

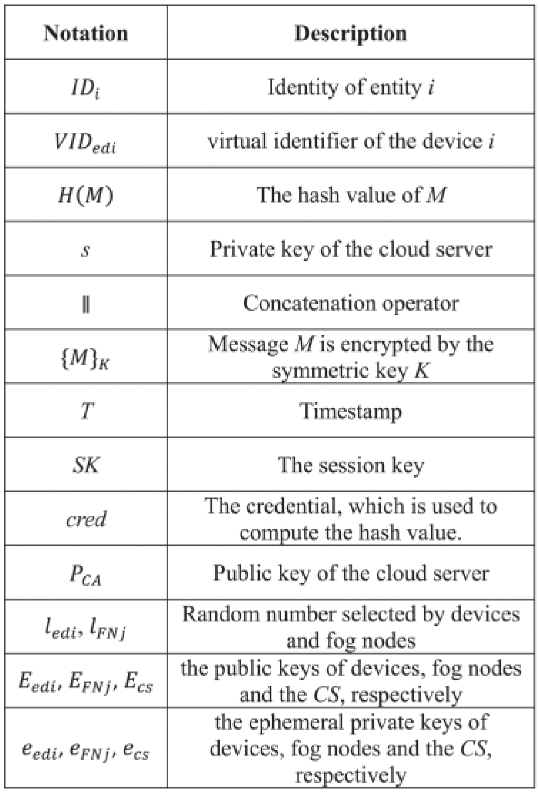
Fog nodes are usually distributed in an environment with inadequate physical security procedures when compared to a centralized cloud computing system. Namely, devices and fog nodes might be easily compromised. This situation threatens information security; therefore, users or fog nodes in the system ought to be individually recognized and verified to mitigate these potential risks. The proposed scheme consists of three distinct entities: a cloud server, fog nodes, and groups of devices.

Each fog node or end device generates a specific ID when registering with the cloud server. The cloud server also maintains a database to record the registered users and fog nodes. There will be a table that can store the id of the device after using the hash function. If the id of the device after using the hash function cannot be found in the database, it means that the device has not been registered; similarly, the id of the fog node during the registration phase will also be stored in the cloud server. Before generating the session key, the cloud server verifies the legitimacy of the node or device. During mutual authentication, the fog node is authenticated first by the cloud server. Then, the fog node and cloud server agree on the session key. Finally, the authenticated fog node is responsible for verifying the validity of the device. After mutual authentication is completed, the cloud server, fog nodes, and devices can communicate with each other.

A. Initialization Phase

First of all, Certificate Authority will choose the generator P on the Elliptic curve in Z_p with generator point P of order q , and also decide the hash function H . Then, secret key s is randomly selected by the cloud server, where $s \in Z_{p*}$. After that, the cloud server computes the public key $P_{CS} = s \cdot P$ accordingly and defines the three hash functions h_1 , h_2 , h_3 , where h_1 is defined from $\{0, 1\}^*$ to Z_{*p} ; h_2 is defined from $\{0, 1\}^*$ to $\{0, 1\}^l$, l is the length of session key; h_3 is defined from $\{0, 1\}^*$ to $\{0, 1\}^{2\lambda}$. Then public parameters { E, P, P_{CS}, h_1 , h_2 , h_3 } will be published while keeping s secret. Table 1 presents the notations mentioned i the proposed protocol.

TABLE 1 The Notation used in the Secret Key Agreement



Notation	Description
ID_i	Identity of entity i
VID_{edi}	virtual identifier of the device i
$H(M)$	The hash value of M
s	Private key of the cloud server
$\|$	Concatenation operator
$\{M\}_K$	Message M is encrypted by the symmetric key K
T	Timestamp
SK	The session key
$cred$	The credential, which is used to compute the hash value.
P_{CA}	Public key of the cloud server
l_{edi}, l_{FNj}	Random number selected by devices and fog nodes
E_{edi}, E_{FNj}, E_{cs}	the public keys of devices, fog nodes and the CS, respectively
e_{edi}, e_{FNj}, e_{cs}	the ephemeral private keys of devices, fog nodes and the CS, respectively

B. Device Registration

(1) The device should initially choose its identity ID_{edi} and need to register to the Cloud Server (*CS*). First of all, the device $Device_i$ sends a hashed personal identifier for verification $h(ID_{edi})$ and encrypts the information with the public key of the *CS* $E_{Pcs}(h\ (ID_{\ edi})\ )$, which we utilize DTLS to transmit. DTLS stands for Datagram Security Transport Protocol, which can prevent messages from being eavesdropped, tampered with, and forged. Therefore, only the *CS* can decrypt it and verify the existence of $h(ID_{edi})$. If *CS* cannot find $h(ID_{edi})$ in the database, it means the device has not registered. To examines whether ID_{edi} has been registered, the *CS* uses its private key to decrypt the message received. If it has not been registered, then the cloud server applies the Pseudo Random Number Generator (PRNG) mechanism to generate a pseudonym virtual identifier (*VID*)to hide the anonymity of the device.

(2) Then the $Device_i$ randomly selects l_{edi} in Z_{p*} and computes $H_1 = h_1\ (VID_{edi} \| l_{edi}\ )$. After that, the cloud server stores the device's hashed personal identifier H_1 , and its corresponding virtual identifier (*VID*) in the database for future use and acknowledgment.

(3) Furthermore, *CS* also needs to complete the calculation of $L_{edi} = (sH_1)P$, in which s is the private key of the cloud server. After the calculation, the *CS* will send L_{edi} to $Device_i$.

(4) After receiving L_{edi} sending from the *CS*, $Device_i$ will store $\{L_{edi}$, H_1 } and make it the long-term key, and also delete the random number l_{edi} which is chosen in the second step. As shown in Figure 2.

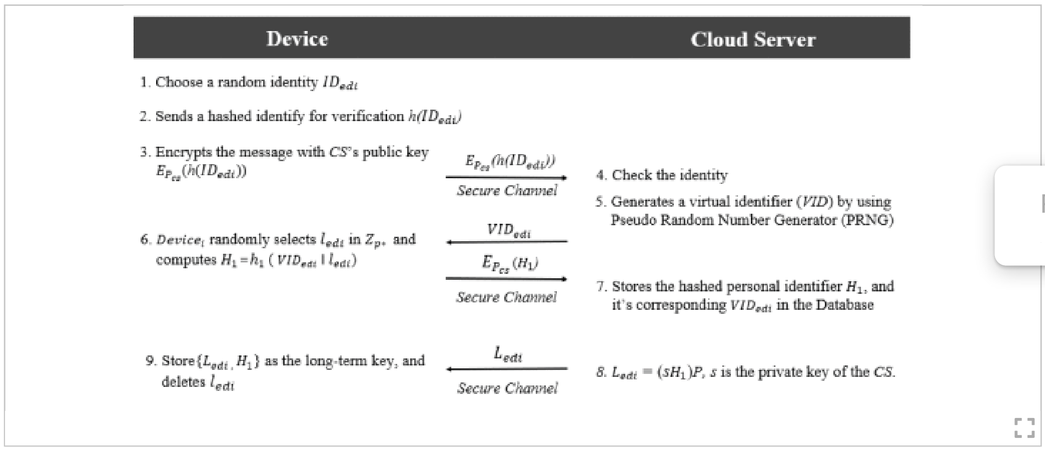

Device
Cloud Server
1. Choose a random identity ID_{edi}
2. Sends a hashed identify for verification $h(ID_{edi})$
3. Encrypts the message with CS's public key $E_{P_{cs}}(h(ID_{edi}))$
$E_{P_{cs}}(h(ID_{edi}))$
Secure Channel
4. Check the identity
5. Generates a virtual identifier (VID) by using Pseudo Random Number Generator (PRNG)
6. $Device_i$ randomly selects l_{edi} in Z_{p*} and computes $H_1 = h_1(VID_{edi} \| l_{edi})$
VID_{edi}
$E_{P_{cs}}(H_1)$
Secure Channel
7. Stores the hashed personal identifier H_1, and it's corresponding VID_{edi} in the Database
9. Store $\{L_{edi}, H_1\}$ as the long-term key, and deletes l_{edi}
L_{edi}
Secure Channel
8. $L_{edi} = (sH_1)P$, s is the private key of the CS.

FIGURE 2.
The device registration phase.

C. Fog Registration

Before authenticating a device, the fog node has to register with the cloud server.

(1) First of all, FN_j randomly selects l_{FNj} in Z_{p*} and computes $P_{FNj} = l_{FNj}$ *P*, and then sends identity ID_{FNj} and P_{FNj} to the *CS*, in a secure channel via DTLS.

(2) Secondly, the *CS* verifies if the ID has been registered before; if not, then the *CS* will calculate $H_2 = h_1(ID_{FNj}\|P_{FNj})$ and the Long-term key of FN_j , in which $L_{FNj} = (sH_2)$ *P*. Then randomly choose a parameter r_c to calculate the *verifier* $=H$ $(ID_{FNj}||r_c||s$). Finally, the *CS* will send L_{FNj} and the verifier to FN_j .

(3) After receiving, the fog node will store $\{L_{FNj}$, l_{FNj} } and the *verifier* securely for future authentication and then publish P_{FNj} .

Figure 3 demonstrates the process being carried out when the *FN* wants to register with the *CS*.

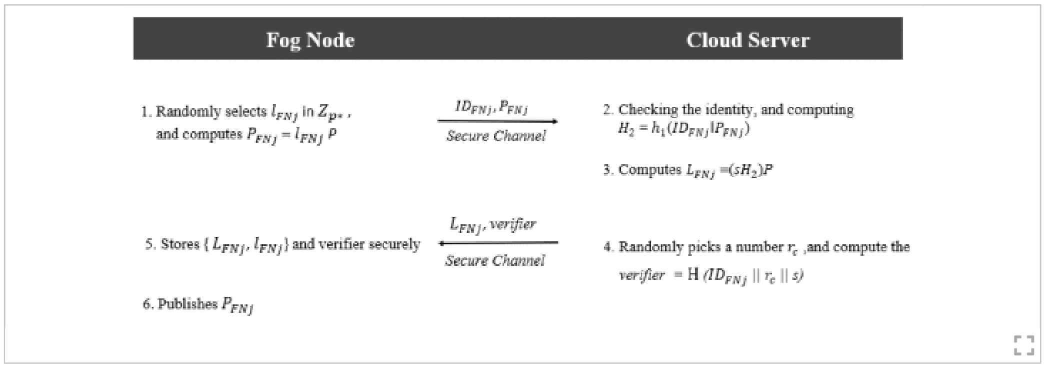
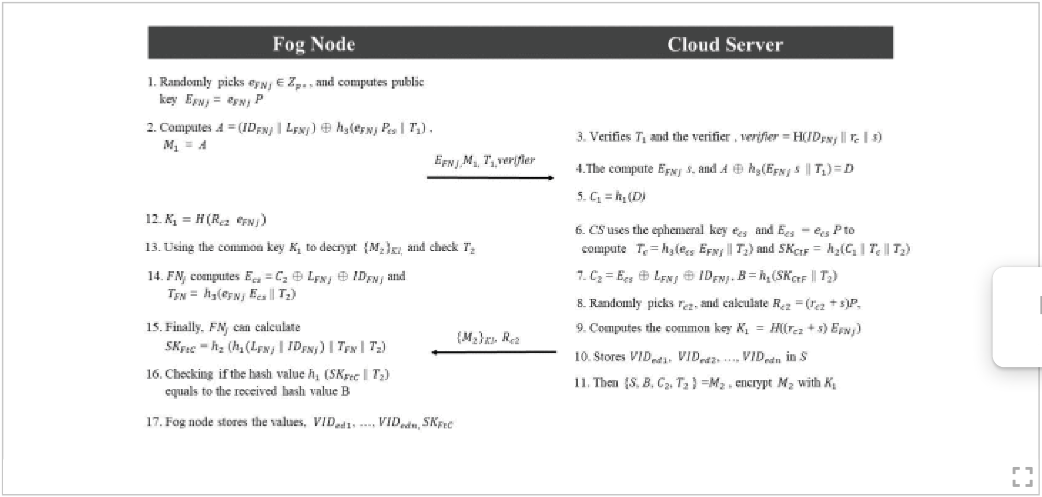

Fog Node
Cloud Server
1. Randomly selects l_{FNj} in Z_{p*}, and computes $P_{FNj} = l_{FNj} P$
ID_{FNj}, P_{FNj}
Secure Channel
2. Checking the identity, and computing $H_2 = h_1(ID_{FNj} \| P_{FNj})$
3. Computes $L_{FNj} = (sH_2)P$
L_{FNj}, verifier
Secure Channel
5. Stores $\{L_{FNj}, l_{FNj}\}$ and verifier securely
4. Randomly picks a number r_c, and compute the verifier = H $(ID_{FNj} \| r_c \| s)$
6. Publishes P_{FNj}

FIGURE 3.
The fog node registration phase.

D. Mutual Authentication and Key Establish Phase

The mutual authentication phase consists of two parts. As for the first part, the fog node needs to be validated by the *CS;* then, a session key will be established. After that, fog nodes are responsible for verifying the legitimacy of end devices and further generating another *SK* with end devices. Figure 4 and Figure 5 depict the authentication procedure.


Fog Node

1. Randomly picks $e_{FNj} \in Z_{p*}$, and computes public key $E_{FNj} = e_{FNj}P$
2. Computes $A = (ID_{FNj} \| L_{FNj}) \oplus h_3(e_{FNj}P_{cs} \| T_1)$, $M_1 = A$

$\xrightarrow{E_{FNj}, M_1, T_1, verifier}$

Cloud Server

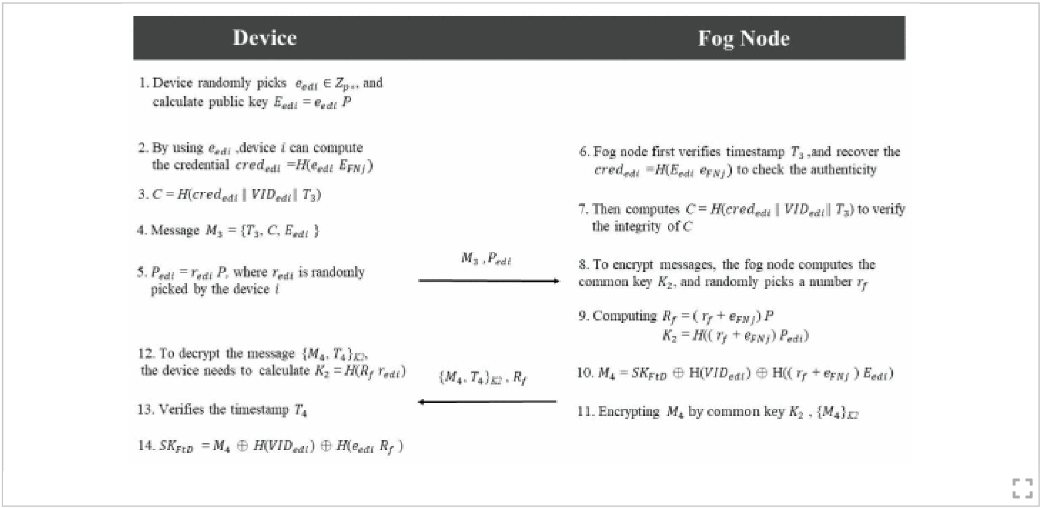
3. Verifies T_1 and the verifier, $verifier = H(ID_{FNj} \| r_c \| s)$
4. The compute $E_{FNj}s$, and $A \oplus h_3(E_{FNj}s \| T_1) = D$
5. $C_1 = h_1(D)$
6. CS uses the ephemeral key e_{cs} and $E_{cs} = e_{cs}P$ to compute $T_c = h_3(e_{cs}E_{FNj} \| T_2)$ and $SK_{CtF} = h_2(C_1 \| T_c \| T_2)$
7. $C_2 = E_{cs} \oplus L_{FNj} \oplus ID_{FNj}$, $B = h_1(SK_{CtF} \| T_2)$
8. Randomly picks r_{c2}, and calculate $R_{c2} = (r_{c2} + s)P$,
9. Computes the common key $K_1 = H((r_{c2} + s)E_{FNj})$
10. Stores $VID_{ed1}, VID_{ed2}, \ldots, VID_{edn}$ in S
11. Then $\{S, B, C_2, T_2\} = M_2$, encrypt M_2 with K_1

$\xleftarrow{\{M_2\}_{K1}, R_{c2}}$

Fog Node

12. $K_1 = H(R_{c2}e_{FNj})$
13. Using the common key K_1 to decrypt $\{M_2\}_{K1}$, and check T_2
14. FN_j computes $E_{cs} = C_2 \oplus L_{FNj} \oplus ID_{FNj}$ and $T_{FN} = h_3(e_{FNj}E_{cs} \| T_2)$
15. Finally, FN_j can calculate $SK_{FtC} = h_2(h_1(L_{FNj} \| ID_{FNj}) \| T_{FN} \| T_2)$
16. Checking if the hash value $h_1(SK_{FtC} \| T_2)$ equals to the received hash value B
17. Fog node stores the values, $VID_{ed1}, \ldots, VID_{edn}, SK_{FtC}$

FIGURE 4.
The interaction between fog nodes and the cloud server during the mutual authentication phase.


Device

1. Device randomly picks $e_{edi} \in Z_{p*}$, and calculate public key $E_{edi} = e_{edi}P$
2. By using e_{edi}, device i can compute the credential $cred_{edi} = H(e_{edi}E_{FNj})$
3. $C = H(cred_{edi} \| VID_{edi} \| T_3)$
4. Message $M_3 = \{T_3, C, E_{edi}\}$
5. $P_{edi} = r_{edi}P$, where r_{edi} is randomly picked by the device i

$\xrightarrow{M_3, P_{edi}}$

Fog Node

6. Fog node first verifies timestamp T_3, and recover the $cred_{edi} = H(E_{edi}e_{FNj})$ to check the authenticity
7. Then computes $C = H(cred_{edi} \| VID_{edi} \| T_3)$ to verify the integrity of C
8. To encrypt messages, the fog node computes the common key K_2, and randomly picks a number r_f
9. Computing $R_f = (r_f + e_{FNj})P$
$K_2 = H((r_f + e_{FNj})P_{edi})$
10. $M_4 = SK_{FtD} \oplus H(VID_{edi}) \oplus H((r_f + e_{FNj})E_{edi})$
11. Encrypting M_4 by common key K_2, $\{M_4\}_{K2}$

$\xleftarrow{\{M_4, T_4\}_{K2}, R_f}$

Device

12. To decrypt the message $\{M_4, T_4\}_{K2}$, the device needs to calculate $K_2 = H(R_f r_{edi})$
13. Verifies the timestamp T_4
14. $SK_{FtD} = M_4 \oplus H(VID_{edi}) \oplus H(e_{edi}R_f)$

FIGURE 5.
Processes between fog nodes and devices during the mutual authentication phase.

1) Key Agreement Between Fog Nodes and the Cloud Server

(1) Initially, the authentication of the fog node should be confirmed by the cloud server. Then the cloud server will share a *SK* with the fog node. After that, the fog node will perform procedures as follows. First, the fog node FN_j randomly chooses $e_{FNj} \in Z_{p*}$, and then calculates the corresponding public key $E_{FNj} = e_{FNj}P$. Next, the fog node FN_j must also calculate A = ($ID_{FNj}||L_{FNj}) \oplus h_3(e_{FNj}P_{cs}||T_1)$ and sets $M_1 = A$. Subsequently, transmits the calculated parameters E_{FNj} , timestamp T_1 , and the *verifier* to the *CS* as a request for service.

(2) When the message arrives, the cloud server will examine the accuracy of the timestamp T_1 and check out the database to verify its authenticity. After that, to confirm the identity of the fog nod, the *CS* will recalculate the *verifier* = $H(ID_{FNj}||r_c||s)$. Then calculate $E_{FNj}s$, and $A \oplus h_3(E_{FNj}s||T_1) = D$. If the *CS* can successfully get the result, then the legitimacy of the cloud can be confirmed. Then the cloud server can get $ID_{FNj}||L_{FNj}$ and then use it to calculate $C_1 = h_1$ (D). Next, the *CS* uses its ephemeral key e_{cs} and $E_{cs} = e_{cs}P$, $T_c = h_3(e_{cs}E_{FNj}||T_2)$ to get the session key $SK_{CtF} = h_2(C_1||T_c||T_2)$. Then, the *CS* calculates $C_2 = E_{cs} \oplus L_{FNj} \oplus ID_{FNj}$, $B = h_1(SK_{Dts}||T_2)$. To encrypt the critical information,

the *CS* generates a random number r_{c2} again to calculate R_{c2} = $(r_{c2}$ + $s)P$, and uses the public key E_{FNj} of the fog node to calculate the common key K_1 = H $((r_{c2}$ + $s)E_{FNj}$). To send the virtual identifier of devices to the fog node, the *CS puts* $VID_{d1}, VID_{d2}, \ldots, VID_{dn}$ into S . Moreover, the cloud server will put { *S, B,* C_2 , T_2 } into messages M_2 , in which M_2 ={ *S, B,* C_2 , T_2 }, and use the common key K_1 to encrypt the message. Finally, send $\{M_2\}_{K1}$ and R_{c2} to the fog node.

(3) To decrypt cipher text $\{M_2\}_{K1}$, first, the fog node has to compute the common key K_1 = H $(e_{FNj} R_{c2}$). After decrypting the message M_2 , the fog node will examine the timestamp T_2 first and receive VID_{ed1} , $VID_{ed2}, \ldots, VID_{edn}$ which are stored in S . Next, FN_j calculates E_{cs} = $C_2 \oplus L_{FNj} \oplus ID_{FNj}$ and T_{FN} = $h_3(e_{FNj} E_{cs}||T_2)$ and session key $SK_{FtC} = h_2(h_1(L_{FNj} \mid\mid ID_{FNj}) \mid\mid T_{FN} \mid\mid T_2)$. Finally, FN_j calculates h_1 $(SK_{FtC}||T_2$) = B to examine whether the result is the same as parameter B in M_2 to check the correctness of *SK*. By calculating T_{FN} = $h_3(e_{FNj}\ E_{cs} \mid\mid T_2) = h_3(e_{cs}\ e_{FNj}\ P|| \ T_2) = h_3(e_{cs}\ E_{FNj} \mid\mid T_2)$ = T_c , and the following equation, it can be confirmed that the results of SK_{CtF} and SK_{FtC} calculated by the *CS* a the fog node, respectively, are the same. SK_{CtF} = $h_2(C_1||T_c||T_2)$ = $h_2(h_1(L_{FNj}||ID_{FNj})||T_{FN}||T_2$ $h_2(C_1||T_{FN}||T_2)$ = SK_{FtC}

Finally, because the fog node needs to verify devices, it will store the related parameters

$$VID_{ed1}, VID_{ed2}, \ldots, VID_{edn}, R_{c2} \quad \text{and} \quad SK_{FtC}.$$

View Source

2) Key Agreement Between Fog Nodes and Devices

In this part, fog nodes have to validate the joining device. Fog nodes are responsible for verifying the legitimacy of devices, and making sure the devices have not been forged.

(1) Upon receiving the start message, the device randomly chooses an ephemeral private key $e_{edi} \in Z_{p*}$, and calculates the corresponding public key E_{edi} = $e_{edi}P$. After that, the device can calculate the credential $cred_{edi}$ = $H(e_{edi} E_{FNj})$ with the private key. After combining $cred_{edi}$, VID_{edi} , T_3 and putting them into a hash function $H(cred_{edi}||VID_{edi}||T_3)$ = C , the device then calculates P_{edi} = $r_{edi}P$, where r_{edi} is a number randomly picked by the device *i*. Next, *t* he device sends message M_3 ={T_3 , C , E_{edi} } and P_{edi} to the fog node.

(2) Upon receiving the message M_3 from the device, the timestamp T_3 is first checked by the fog node. Then the fog node recovers the $cred_{edi}$ = $H(E_{edi}e_{FNj})$. Additionally, to confirm the legitimacy of the device, it is necessary to use $cred_{edi}$ and the VID_{edi} stored in the fog node to verify the integrity of C (C = H $(cred_{edi}||VID_{edi}||T_3)$). After verifying the device's authenticity, the fog node will generate the common key K_2 . First of all, the fog node randomly generates a number r_f to calculate R_f = $(r_f$ + $e_{FNj})P$ and then use the public key E_{edi} of the device to calculate the common key K_2 = H $((r_f$ + $e_{FNj})P_{edi}$). Then the message M_4 =$SK_{FtD} \oplus H(VID_{di}) \oplus H$ $((r_f$ + e_{FNj}) E_{edi}) is calculated. Finally, the message M_4 will be encrypted by the common key K_2 .

(3) To decrypt the message $\{M_4$, $T_4\}_{K2}$, the device must calculate the common key K_2 before it can derive any information. Therefore, the formula K_2 = $H(r_{edi}R_f)$ needs to be calculated. After that, the fog node checks the freshness of the timestamp T_4 . Then the device can get the session key SK_{FtD} = $M_4 \oplus H(VID_{edi}) \oplus H$ $((e_{edi}R_f$).

E. Intra-Fog Authentication

If the device wants to move to another fog node in the same network, parameters will be exchanged between the preceding and succeeding fog nodes. First of all, the device will broadcast a message; then the succeeding fog node will check if the VID_{edi} exists in its list and make sure the verification has been checked. Secondly, the preceding fog node will transmit the VID_{edi} to the succeeding fog node to avoid duplication. After that, mutual authentication between the fog node and the device will be repeated to establish communication.

SECTION V.
Derailed Analysis

A. Anonymity and Un-Traceability

To prevent an adversary from obtaining the identities of other devices when the message is transmitting, hash values and ciphertext are combined; therefore, the devices' identities are hidden in the ciphertexts.

During the device registration phase, the public key of the *CS* is used to encrypt the message $E_{Pcs}h(ID_{edi})$; hence the identities can only be recovered by the authenticated *CS*. Furthermore, to achieve anonymity, the masked identities of the devices $H_1 = h_1\ (VID_{edi}||l_{edi}$) are also computed.

B. Replay Attack

While transmitting messages, each message contains a timestamp; therefore, only the message within the legal time interval will be admitted.

PDF

Help

C. Man-In-The-Middle-Attack

In the device authentication phase, the device puts $cred_{edi}$, VID_{di} , T_3 into a hash function C = $H(cred_{edi}||VID_{di}||T_3)$, then sends the message M_3 ={T_3 , C , E_{edi} } to the fog node. As mentioned previously, only the authenticated cloud server and the fog node can obtain VID_{di} . Furthermore, the ephemeral private key e_{edi} of the device is needed if the attacker wants to compute the credential $cred_{edi}$ = H ($e_{edi}E_{FNj}$). However, to derive the e_{edi} , one must solve the *ECDLH* problem.

D. Perfect Forward Secrecy

In this research, every communication session generates a unique encryption key and only lasts for the duration of the session. If an attacker compromised one of the user's special key, the conversations would stay encrypted and secure. Even if one of these session keys is revealed, data that come from any other session will not be influenced. Since the *SK* consists of long-term private keys and random numbers, in addition, the ciphertext is encrypted by both the long-term private key and the state-specific information; therefore, past sessions and information are protected from attacks, and the attacker cannot derive the *SK* from the ciphertext.

During the authentication phase between fog nodes and the cloud server, the SK_{CtF} is composed of the ephemeral key of the for node e_{FNj} , which will change at a different session.

$$\begin{aligned} T_{FN} &= h_3(e_{FNj}E_{cs}||T_2) \\ SK_{CtF} &= h_2(C_1||T_c||T_2) = h_2(h_1(L_{FNj}||ID_{FNj})||T_{FN}||T_2) \\ &= h_2(C_1||T_{FN}||T_2) = SK_{FtC} \end{aligned}$$

View Source

Furthermore, the common key used to encrypt the message M_2 also contains a random number picked by the cloud server.

$$K_1 = H((r_{c2} + s)E_{FNj})$$

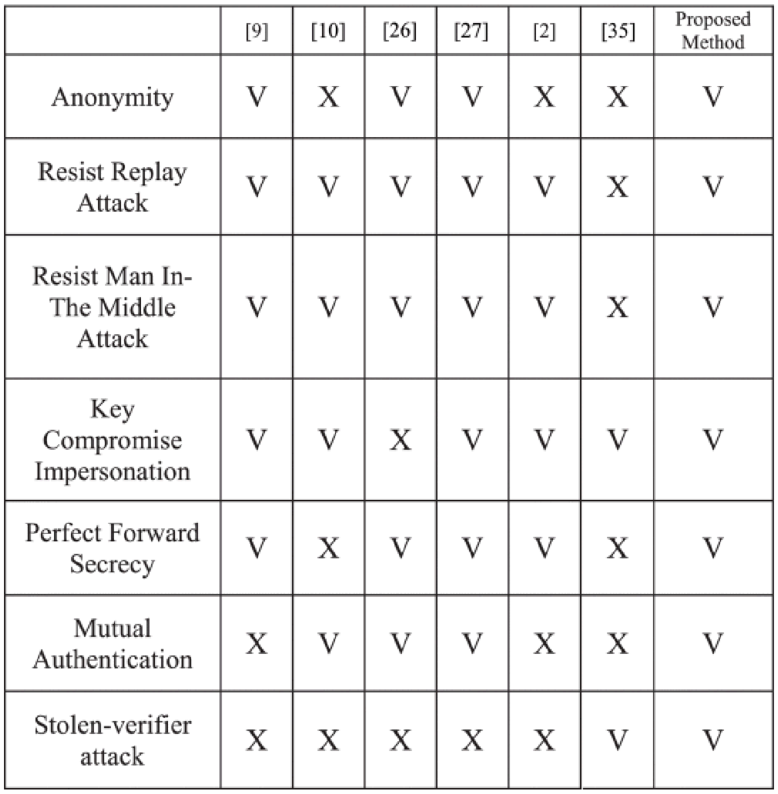
View Source

E. Stolen-Verifier Attack

In the proposed protocol, the *CS* only keeps the identity of the fog node ID_{FNj} , rather than the actual verifier. If an adversary $\mathcal{A}$ were to steal the ID_{FNj} stored in the *CS*, they would not be able to calculate the real verifier = $H(ID_{FNj}||r_c||$ s) without the *CS*'s secret key s . Therefore, even if the adversary $\mathcal{A}$ gains access to the information stored in the *CS*, it would still be impossible for them to impersonate the fog node.

Table 2 presents the comparison of the proposed scheme with other research. As Table 2 shows, the proposed scheme can resist the attacks mentioned above, and it is more secure than the aforementioned methods.

TABLE 2 Comparison of Functionality Features



	[9]	[10]	[26]	[27]	[2]	[35]	Proposed Method
Anonymity	V	X	V	V	X	X	V
Resist Replay Attack	V	V	V	V	V	X	V
Resist Man In-The Middle Attack	V	V	V	V	V	X	V
Key Compromise Impersonation	V	V	X	V	V	V	V
Perfect Forward Secrecy	V	X	V	V	V	X	V
Mutual Authentication	X	V	V	V	X	X	V
Stolen-verifier attack	X	X	X	X	X	V	V

F. Formal Proof of Security

In this section, the security of the proposed scheme under the eCK security model is discussed.

Theorem 1: Parameters q_{h_1} , q_{h_2} , q_{h_3} denotes queries that the random oracle sends. If a probabilistic polynomial time adversary $\mathcal{A}$ against the proposed scheme making at most q_s query, then there must be a probabilistic polynomial time algorithm that can solve the ECCDH problem with the advantage

$$Adv(\mathcal{A}) \leq \frac{q_s}{2^{\lambda-2}} + \frac{q_s}{2^{2\lambda-2}} + \frac{2^{\lambda}\cdot q_{h_1}^2 + q_{h_3}^2}{2^{2\lambda}} + \frac{q_{h_2}^2}{2^l} + 2q_{h_2}q_s^2 + Adv^{ECCDH}(ş)$$

View Source

Adv^{ECCDH} (ş) means the success probability of solving an instance of ECCDH problem by an algorithm ş.

Proof: The following games demonstrate the possibility of being cracked. The experiments prove that the advantage of cracking the proposed scheme will be limited by an inevitable possibility.

Experiment 0: This experiment represents the circumstance of the attacks against the actual protocols in the random oracle model.

Thereby, the possibility of success is equal to that of an actual protocol. In accordance with the definition, we have $Adv(\mathcal{A}) = |2\Pr[S_0] - 1|$

Experiment 1: The simulation described above is implemented, and the following lists L_H , L_S and $L_{\mathcal{A}}$ represents the query result. L_H , L_S , $L_{\mathcal{A}}$ are denoted as the result of Hash, Send, and other queries. Therefore, the chance for the adversary $\mathcal{A}$ to succeed is

$$Pr[S_1] = Pr[S_0]$$

View Source

Experiment 2: This experiment simulates oracles just like Experiment 1. However, if there is a collision, the simulation will be terminated. Based on the birthday paradox theory, the probability that the collision happens in the outcome of the oracle h_1 is at most $\frac{q_{h_1}^2}{2^{\lambda+1}}$, where q_{h_1} represents the maximum times of querying h_1 . The same deduction can be applied to h_2 and h_3 . Thus $Pr[S_2]-Pr[S_1]\leq\frac{q_{h_1}^2}{2^{\lambda+1}}+\frac{q_{h_3}^2}{2^{2\lambda+1}}+\frac{q_{h_2}^2}{2^{l+1}}$.

Experiment 3: As for this experiment, the protocol will not stop unless the adversary $\mathcal{A}$ makes a guess of C_1 or T_c (T_{FN}) without querying h_1 or h_3 ; therefore, we have $Pr[S_3]-Pr[S_2]\leq 2\cdot\frac{q_s}{2^\lambda}+2\cdot\frac{q_s}{2^{2\lambda}}$.

Experiment 4: In this experiment, suppose that the adversary $\mathcal{A}$ chooses a random session as the test session. Furthermore, a random key selected from the key set is used to compute the test session key. Thereby, the difference between Experiment 3 and Experiment 4 is whether $\mathcal{A}$ queries the tuple ($C_1||T_{FN}||T_2$) or ($C_1||T_c||T_2$) to h_2 in the test session. To further explain the difference, three cases are described as follows.

(1) If both the long-term key of the fog node L_{FNj} and the ephemeral key of the cloud server e_{CS} is obtained, A can be calculated by L_{FNj} during the mutual authentication stage. However, because the secret key s of the cloud server cannot be derived, C_1 cannot be calculated, and therefore the attacker $\mathcal{A}$ will fail to get the session key SK_{CtF} .

(2) In the event that the ephemeral key e_{FNj} of FN and the long-term key s of the cloud server are obtained, L_{FNj} can be calculated; therefore, A and C_1 may be derived during the mutual authentication stage. However, when computing SK_{CtF} , the $\mathcal{A}$ needs to use the ephemera key of the cloud server e_{CS} to get T_c . In addition, SK_{CtF} is encrypted by the common key K_1 , and the operation of K_1 requires the random parameter r_{c2} as well.

(3) In case the ephemeral key e_{FNj} of the FN and the ephemeral key e_{edi} of the device are obtained, the attacker $\mathcal{A}$ still needs to obtain the random parameter r_f of the fog node or the r_{edi} of the device to calculate the common key K_2 , and then decrypt M_4 to obtain the Session key. However, since the random parameter r_f is a session parameter and changes constantly, it is difficult for $\mathcal{A}$ to obtain the correct r_f .

(4) Even if attackers obtain both the ephemeral key of the device e_{edi} and the random parameter of the fog node r_f , they still cannot correctly calculate the common key K_2 . This is because e_{FNj} is the ephemeral key of the fog node, and solving the ECDLP problem is required to obtain the e_{FNj} . Therefore, without e_{FNj} , one cannot derive the common key K_2 and decrypt the message M_4 .

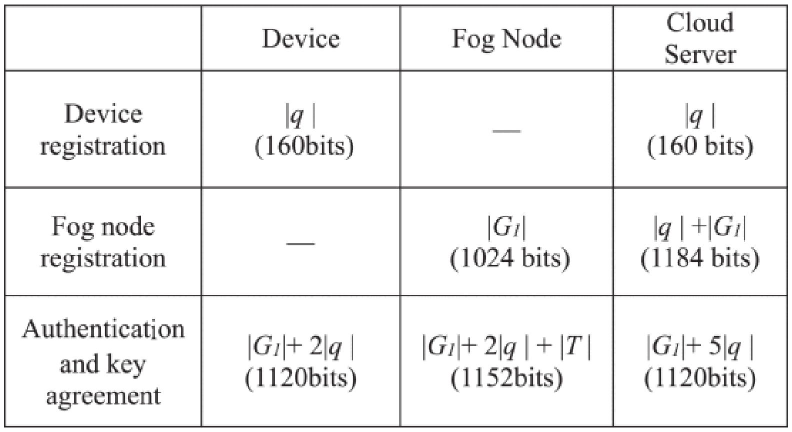
(5) If malicious attackers intercept the common key K_2 generated by the fog node and the credential $cred_{edi}$ of the device, they can use the common key K_2 to decrypt the message M_4 during the mutual authentication process between the fog node and the device. Besides, to calculate the session key SK_{CtF} , attackers also need to derive the device's ephemeral key e_{edi}. However, to deduce the ephemeral key e_{edi} from the device's credential $cred_{edi}$, one must first solve the ECDLP problem.

If any of these three cases happens, then constructing an algorithm ȿ and solving the ECCDH problem will be possible, and there exists $Pr[S_4]-Pr[S_3]\leq q_{h_2}q_s^2Adv^{ECCDH}$ (ȿ).

ECC can achieve the same level of security with a shorter key size than non-ECC public-key cryptosystems; therefore, it is greatly used in the design of public-key cryptographic protocols. G_1 and G_2 are groups of prime order q , where p and q are large prime numbers. The length of p and q are 512 and 160 bits, respectively.

To evaluate the performance, the length of a point in group G_1 is indicated as $|G_1|$, which is 1024 bits. The output of hash functions h_1 , h_2 , and h_3 has a length of 160 bits, expressed as $|q|$, and the timestamp has a length of 32 bits, denoted as T . Table 3 presents the computational costs of the proposed method.

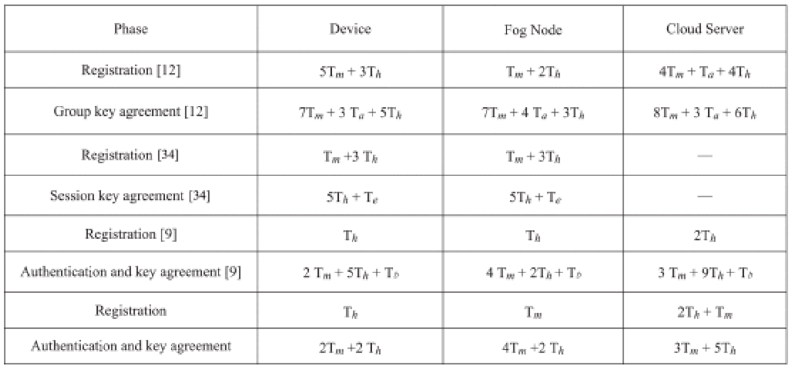
TABLE 3 Computational Costs of the Proposed Scheme



	Device	Fog Node	Cloud Server
Device registration	$\lvert q \rvert$ (160bits)	—	$\lvert q \rvert$ (160 bits)
Fog node registration	—	$\lvert G_1 \rvert$ (1024 bits)	$\lvert q \rvert + \lvert G_1 \rvert$ (1184 bits)
Authentication and key agreement	$\lvert G_1 \rvert + 2\lvert q \rvert$ (1120bits)	$\lvert G_1 \rvert + 2\lvert q \rvert + \lvert T \rvert$ (1152bits)	$\lvert G_1 \rvert + 5\lvert q \rvert$ (1120bits)

Table 4 lists the computation costs of three entities during different phases of the protocol. Before presenting the analysis, we define the notations used in estimating computation efficiency. $\mathrm{T}h$, $\mathrm{T}m$, $\mathrm{T}a$, T *b*, $\mathrm{T}e$ represent the time for hash function, point multiplication in elliptic curve group, point addition in elliptic curve group, bilinear map, and an exponentiation in cyclic group, respectively. The method proposed by [12] consumes a large number of calculations during the registration and group key agreement phases for both the device and the fog node. Additionally, in [9], during the authentication and key agreement stage, their proposed method requires additional computation of bilinear pairings compared to our method. In [34], mutual authentication mechanism was not used. Therefore, although the amount of calculation required is less than other studies, it may cause security problems during communication. In comparison to the aforementioned methods, the method proposed in this research is a faster and more secure mechanism. It minimizes communication costs between the cloud server and the fog nodes. First, fog nodes need to be verified by the cloud server. Second, fog nodes validate devices, which can help reduce the computing costs of the cloud server, as each device communicates with nearby fog nodes for identity authentication and to obtain a session key before communication. In addition, the proposed method has the session key composed of the private keys of each entity and some randomly generated values. For mutual authentication, each entity needs its private key and these randomly generated values, which eliminates the key escrow issue.

TABLE 4 Comparison of the Communication Efficiency



Phase	Device	Fog Node	Cloud Server
Registration [12]	$5T_m + 3T_h$	$T_m + 2T_h$	$4T_m + T_a + 4T_h$
Group key agreement [12]	$7T_m + 3\ T_a + 5T_h$	$7T_m + 4\ T_a + 3T_h$	$8T_m + 3\ T_a + 6T_h$
Registration [34]	$T_m +3\ T_h$	$T_m + 3T_h$	—
Session key agreement [34]	$5T_h + T_e$	$5T_h + T_e$	—
Registration [9]	T_h	T_h	$2T_h$
Authentication and key agreement [9]	$2\ T_m + 5T_h + T_b$	$4\ T_m + 2T_h + T_b$	$3\ T_m + 9T_h + T_b$
Registration	T_h	T_m	$2T_h + T_m$
Authentication and key agreement	$2T_m +2\ T_h$	$4T_m +2\ T_h$	$3T_m + 5T_h$

SECTION VI. Conclusion

Fog computing helps doctors to make decisions during an emergency for time-critical Healthcare applications. It also helps to protect sensitive data with reduced delay in comparison to the standalone cloud-based application. However, data privacy and system compatibility are essential challenges that must be addressed when considering how medical records can be delivered. In the case of inadequate network security, fog nodes, and devices can be hacked, and communication can be intercepted. The mutual authentication key establishment proposed herein is a safe and efficient information security mechanism for fog computing architecture in a healthcare environment. In this method, fog nodes can be applied to validate the device's authenticity, thereby reducing the computational cost of the cloud server. Furthermore, we used the random oracle model and the extended Canetti-Krawczyk (eCK) threat model to conduct a detailed analysis, and proved that the proposed scheme is secure against several attacks as well. Finally, based on the comparisons made with previous schemes, it can be concluded that the proposed protocol is more secure than the aforementioned methods. The development of this mechanism aids in the establishment of secure and stable data transmission, which, in turn, can improve medical services and can provide benefits for the medical industry.

PDF
Help

Authors

Figures

References

Citations

Keywords

Metrics