IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 3, 2021, accepted November 12, 2021, date of publication November 23, 2021,
date of current version December 6, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3130471

Provable Secure Group Key Establishment Scheme for Fog Computing

WEN-CHIN CHEN, YIN-TZU HUANG, AND SHENG-DE WANG, (Member, IEEE)
Electrical Engineering Department, National Taiwan University, Taipei 10617, Taiwan
Corresponding author: Sheng-De Wang (sdwang@ntu.edu.tw)

This work was supported in part by the Ministry of Science and Technology, Taiwan, under Grant 109-2221-E-002-145-MY2.

ABSTRACT In the fog computing paradigms, fog nodes are closer to terminal devices and can extend services to the edge of the network, thereby reducing the impact of high latency and restricted networks in the Internet of Things (IoTs). Fog computing applications usually organize the terminal devices in groups and require some form of security protection. Previous studies on the establishment of group keys for fog computing architectures have high communication costs and cannot verify the authenticity of each entity. Therefore, in this paper, we propose a mutual authentication group key establishment scheme for the fog computing architecture by using elliptic curve cryptography. After mutual authentication, the cloud server can transfer the computing overhead to the fog node, which will be responsible for authenticating the device group and distributing the established group session key. The group session key consists of the private key of each entity and some random and temporarily stored values. We prove that the established group session key is protected by the Canetti-Krawczyk (CK) adversary model. Finally, we evaluate performance based on calculation and communication costs. Compared with previous studies, the proposed scheme is lightweight and effective because it only involves elliptic curve operations and symmetric cryptographic operations.

INDEX TERMS Fog computing, elliptic curve cryptography, mutual authentication, Canetti–Krawczyk adversary model.

I. INTRODUCTION

To deal with the problems of high latency and constrained networking in Internet of things (IoTs), the concept of fog computing was introduced. The components of fog computing can in general be grouped into three layers: cloud layers, fog layers and end device layers. The cloud layer comprises cloud servers with high computing capability and databases for storage, providing data analysis capabilities security services for authentication. The fog layer consists of routers, gateways, base stations, switches, etc. It decentralizes the computing requirement of the cloud server and extends the services to the edge of the network, and thus enable real time data processing [1]. In the fog layer, the fog node has certain computing and storage capacity and is responsible for processing, aggregating and transmitting the data to the cloud layer. The end device layer contains a large number of IoTs devices. These devices can be sensor devices or mobile phones and in general send the sensor data to the fog layer via short range communication such as Bluetooth, Zigbee and Wi-Fi.

The associate editor coordinating the review of this manuscript and approving it for publication was Pedro R. M. Inácio.

There are emerging applications of fog computing, such as the intelligent transportation system (ITS) using vehicular ad-hoc networks (VANETs), healthcare system and mobile networks. As mentioned in [2], in ITS, vehicles are equipped with onboard units (OBUs) to communicate to the fog nodes via the nearby roadside units (RSUs). The information of the vehicles will be sent to the intelligent transportation center (ITC) via the fog nodes, and ITC will collect the information and make decisions according to the traffic. In healthcare systems, the sensor devices will collect the physiological data and transmit them to the fog node for preprocessing [3]. The cloud server then conducts the data analysis to predict the condition of the patient.

In the architecture of fog computing, the devices are usually organized as groups. For example, in VANETs, groups of vehicles will communicate with each other in a network. If limited security protections are provided in the network, the fog nodes and devices may be compromised. The messages transmitted also can be eavesdropped. The attacker may obtain some privacy information such as personal data, location histories and healthy conditions. To ensure the stability and reliability of the messages, there must be a group key to encrypt the transmitted messages and protect the communication over insecure networks.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

Although fog computing is considered to be a more secure architecture than cloud computing [3], it still has many security and privacy issues [4]–[6]. Authentication is an important issue for the security of fog computing. Compared with cloud computing, authentication in fog computing will face greater challenges. In fog computing, there are multiple different trust domains such as cloud, fog service providers, and users, that is, multiple participants interact in an ecosystem where multiple trust domains coexist [7].

Because the fog nodes and devices will dynamically change their location, it is not realistic to pre-share a session key between these entities [8]. Authenticated key agreement (AKA) protocols can authenticate each entity and generate a common session key at each session. Each entity in the group can encrypt messages by the group session key (GSK). Since the message is authenticated without bothering the cloud server, the transmission is efficient and the devices can assure the message authenticity [9].

Shabisha Placide, *et al.* [10] developed an authentication and key agreement mechanism for groups of IoTs devices. In the authentication and key agreement mechanism, the cloud server, the fog node and devices compute the GSK with certain points on elliptic curves (EC) by using the Lagrange interpolation. The communication cost of EC points is high, resulting in high latencies. In addition, the authentication mechanism is not imposed on all of messages and thus it cannot verify whether a device is forged or not.

As the emergence of the fifth generation wireless communication (5G), the number of connected devices will be expected to increase [9]. To reduce the authentication requests, fog nodes can be used to verify the authenticity of the devices without much involving the cloud server. This mechanism thus reduces the computation overheads of the cloud server. The previous study of establishing GSK in the fog computing paradigms [10] involves many rounds of messages exchanged, resulting a high latency. In this paper, we reduce the communication overheads between the cloud server and the fog nodes by first authenticating the fog node and then delegating the authentication tasks to the authenticated fog nodes.

Centralized group key distribution (GKD) uses a group key manager (GKM) to manage and distribute the session key [11]. In this scheme, if the GKM is compromised, the entire system will be broken. In the proposed scheme the authenticated fog nodes act like the role of GKM to distribute the GSK while the fog nodes are not centralized. Each device will interact with the nearby fog node for authentication and the GSK. Furthermore, the GSK is composed of private keys of each entity and some random and temporarily stored values, and each entity also needs its private key and some random values to accomplish the mutual authentication phase. Hence, there is no key escrow problems.

We will conduct a formal security analysis with the random oracle model [12] and the Canetti–Krawczyk (CK) threat model [13]. In the security analysis, we assume that the attacker cannot distinguish the GSK from a random value. The proposed scheme will be proved with respect to an adversary who is able to reveal session state specific information, or long-term private keys and previous GSKs. Finally, a performance analysis of the communication and computation costs will be presented. The results show that the computation and communication costs is less than existing approaches [10].

II. RELATED WORK

In this section, we will review some existing approaches on key agreement and authentication for the group of devices in fog computing and IoT applications.

A. INTERNET OF THINGS (IOTs) ARCHITECTURE

Amin *et al.* [14] and Odelu, Vanga, *et al.* [15] developed an authenticated key agreement and distribution scheme of the client and server architecture. The approach focused on distributed cloud computing environments and all users are requested to be authenticated by the cloud server to establish a common session key [14]. The operations mainly include exclusive-or, symmetric keys and hash functions and thus the communication overhead is low even for the resource-constrained IoTs devices. However, Li, Wenting, *et al.* [16] pointed out that the method cannot specifically solve the problem about user anonymity and forward secrecy since some private information is preserved by the smart card. In addition, Aman, Muhammad Naveed, *et al.* [17] and Guin, Ujjwal, *et al.* [18] proposed mutual authentication schemes with physically unclonable function (PUF) in the environment of IoTs.

B. FOG COMPUTING ARCHITECTURE

Aiming at improving the AKA protocol proposed in [19], Jia, Xiaoying, *et al.* [8] presented an AKA scheme for a fog-driven healthcare application. The approach can authenticate each entity and generate a common session key for each session by using elliptic curve and bilinear mapping operations. However, because of the high cost of the bilinear pairing operation, it is hard to apply the scheme to resource constrained devices. In addition, the scheme requires the devices to have a smart card and is subject to the vulnerabilities of side-channel attacks [20]. To improve the communication costs, Ma, Mimi, *et al.* [2] and Patonico, Simone, *et al.* [21] proposed AKA protocols for the fog computing architecture, by using elliptic curves and hash functions to achieve mutual authentication and key agreement. It is shown that the scheme offers anonymity to each entity. The CK adversary model was used to analyze key exchange protocols and the adequacy of the generated session keys. Shen *et. al.* [22] proposed a lightweight authentication and matrix-based key agreement scheme for healthcare. Gope Prosanta [9] proposed a scheme to reduce the verification steps. Fog nodes can be used to verify devices without involving the central server. Moreover, devices can still receive services even if the central server is broken. Guo *et al.* [7] designed an effective authentication scheme for fog computing, but it

does not support user mobility. Many handover authentication schemes have been designed for mobile networks. The goal of these schemes is to reduce handover delays and ensure that handover schemes are secure, especially to support user's anonymity. Fan *et al.* [23] designed a handover authentication protocol called ReHand based on symmetric cryptography. In Fan *et al.*'s scheme, the long-term key of each UE is shared with the base stations of the same region, and a region key is shared between eNB and HeNBs in the same region. Symmetric key-based approaches can significantly reduce the communication cost and calculation cost of handover authentication protocols, but due to the need to pre-allocate shared keys, this makes key management complicated, has poor scalability and is difficult to resist mobile device captured attacks [24].

C. GROUP KEY AGREEMENT AND ESTABLISHMENT

Zhu, Hongfeng [25] developed an authenticated key agreement protocol with chaotic maps that can apply to a group of users without involving any central server. Recently, the approaches [10], [26], [27], [28] proposed group key establishment schemes based on elliptic curve cryptography. In [17], the GSK is established using point operation and exclusive-or operation. However, this protocol is vulnerable to several attacks and does not provide anonymity [10]. Moreover, the high computational costs make the scheme unsuitable for IoTs applications [29]. The group key exchange protocols based on point operation and bilinear mapping are discussed in [27] and [28]. To improve the key escrow problem in [27], Sun, Hung-Min, *et al.* [28] combines the advantages of public key infrastructure (PKI) and ID-based public key cryptography. They proposed a mechanism for mobile environments and used a certificate-less signature algorithm for user authentication, which not only avoids the key escrow problem in ID-based key exchange protocols but also reduces the cost of certificate management. However, the computation load of [28] is too high to exploit in IoTs environments. The mechanism [10] requires end devices to exchange information with the fog node and the cloud server, then uses Lagrange interpolation to derive the GSK after receiving certain points from other entities. However, in this protocol some points needs to be transmitted to group of devices, which will result in high latency during the transmission [30]. We will extend the concept of [9], [10] to offload some computational overheads of the cloud server, and to improve communication and computation costs. Amor *et al.* [31] proposed a mutual authentication scheme where the fog and server authenticate each other using a public-key cryptosystem. This scheme is based on pseudonym-based cryptography and does not reveal a users real identity, maintaining the anonymity of end users and provides the main security requirements but it doesn't protect the session key exchanged between users and fog servers. The computation cost of this protocol is heavy and hence, it may not be suitable for resource-oriented devices such as IoTs devices [32].

III. PRELIMINARIES

We first review Elliptic Curve Cryptography (ECC). Then, the detail of Elliptic Curve Qu-Vanstone (ECQV) certificate scheme is elaborated.

A. ELLIPTIC CURVE CRYPTOSYSTEM (ECC)

In public key cryptography, ECC can offer improved security with reduced computational requirements with respect to RSA and ElGamal encryption.

An elliptic curve [33] is a plane curve satisfying the equation $y^2 = x^3 + Ax + B \bmod p$ where $p \geq 5$ is a prime and $A, B \in Z_p$ are constants with $4A^3+27B^2 \neq 0 \bmod p$. Let $\hat{E}(Z_p)$ denotes the set of pairs $(x, y) \in Z_p \times Z_p$ satisfying the above equation. Define $E\left(Z_p\right) \stackrel{def}{=} \hat{E}\left(Z_p\right) \cup \{O\}$. $E\left(Z_p\right)$ are called points on the elliptic curve, and O is the unique point at infinity. The addition rule of $E\left(Z_p\right)$ makes $E\left(Z_p\right)$ an abelian group. The scalar multiplication is defined to be $kG = G + G + \ldots + G$, where the number of G is k, and G is the generator. If k is a selected integer in the interval $[1, n-1]$, (k, kG) can be an elliptic key pair [34]. Moreover, the security of ECC is based on the following computational problems.

(1) Elliptic curve discrete logarithm problem (ECDLP)

Given two points $G \in E\left(Z_p\right)$ and $xG \in E\left(Z_p\right)$ of an additive group N, it is computationally hard for a polynomial time bound algorithm to calculate $x \in Z_p$.

(2) Elliptic curve computational Diffie–Hellman problem (ECCDH)

Given three points $G \in E\left(Z_p\right)$, $xG \in E\left(Z_p\right)$ and $yG \in E\left(Z_p\right)$, it is computationally hard for a polynomial time bound algorithm to calculate xyG, where x and y are two unknown parameter $sx, y \in Z_p$.

(3) Elliptic curve decisional Diffie–Hellman problem (ECDDH)

Given four points G, $X = xG$, $Y = yG$ and $Z = zG$ in $E\left(Z_p\right)$, where x, y and z are unknown parameters and $x, y \in Z_p$, it is difficult to decide if Z is equal to xyG.

B. ELLIPTIC CURVE QU-VANSTONE (ECQV) CERTIFICATE SCHEME

ECQV is now the most commonly used implicit certificate, without the need of a secure channel between the certificate authority (CA) and the entity and provides a more efficient alternative to traditional certificates. Figure 1 presents the ECQV scheme [35], which is defined on an elliptic curve of generator G of order n. In this scheme, the encoding function *Encode (ID,* γ*)* will encode the arguments, and the decoding function $Decode_\gamma(\cdot)$ will extract γ from an encoding, and a hash function $H_n(\cdot)$ which hashes the input value and yields a hash value as an integer in the range [0, n-1]. The CA has private key c and public key $P_{CA} = cG$. The user with an identity ID_A first generates a random integer α and computes $A = \alpha G$. Then the user sends $\mathcal{A}$ and ID_A to CA. After receiving the information, CA chooses a random integer k from [1, n-1] and computes kG and $\gamma = A + kG$. Then

CA computes the certificate by encoding γ and ID_A, then computes $e = H_n(Cert)$ and $s = ek + c \bmod n$. The value of s and certificate $Cert$ are sent to the user. Utilizing the certificate and s, the user compute $e' = H_n(Cert)$ and his private key $a = e'\alpha + s(modn)$. Then the user extracts γ' by $Decode_\gamma(Cert)$ and computes his public key with $P_A = e'\gamma' + P_{CA}$. Finally, the user verifies if the certificate is valid by comparing the multiplication of the private key and the generator with his public key.

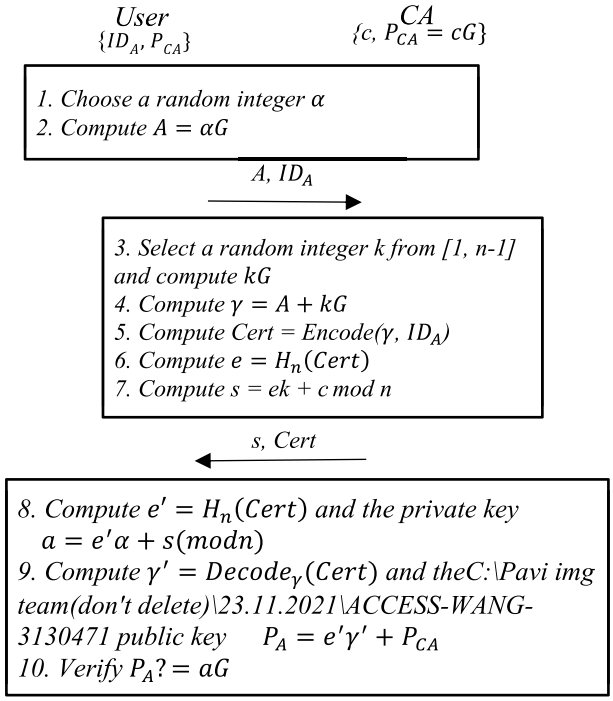

FIGURE 1. ECQV certificate scheme.

C. THREAT MODEL

The CK-adversary model is considered as a threat model [13]. In this model, the adversary can eavesdrop on the channel or modify the transmitted message. Moreover, the adversary can also reveal session state specific information, session keys, or long-term private keys.

IV. PROPOSED SCHEME

In the proposed approach, three different entities are involved: cloud Server, fog nodes and group of devices

During the registration phase, each entity needs to generate its own identity and send it to the cloud server. Next, the cloud server will compute the verifiers for fog nodes and store the identity of each device for further authentication. Because the cloud server will allocate and distribute the responsibilities to the fog nodes; the cloud server has to authenticate the fog nodes. After mutual authentication, the fog nodes and the cloud server will agree on the GSK. Once the authenticated fog node verifies the group of devices by using their identities, the GSK will be distributed to each fog node. Finally, the cloud server, fog nodes and group of devices can use the GSK to communicate with each other.

A. SYSTEM SETUP PHASE

During the system setup stage, the *CA* first chooses an elliptic curve of generator G in Z_p with generator point G of order q, and a random value c is set as the private key of the *CA*. Next, the *CA* computes the corresponding public key P_{CA}, and determines the hash function H. Also, the *CA* selects a symmetric key algorithm for encryption and decryption. The *CA* publishes the elliptic curve, together with the public key of the *CA*, the hash function and the symmetric key algorithm, and each entity will get the key pair by using ECQV mechanism. The notation used in the paper is summarized in Table 1.

TABLE 1. Notation.

Notation	Description
ID_i	Identity of entity i
$H(M)$	The hash value of M
$\|\|$	Concatenation operator
$\{M\}_K$	Message M is encrypted by the symmetric key K
(d_i, P_i)	Private and public key pair of i, where $P_i = d_i \cdot G$
T	Timestamp
GSK	The group session key
R_f	The elliptic curve point computed by fog nodes
$cred$	The credential, which is used to compute the hash value.

B. DEVICE REGISTRATION PHASE

After the setup phase, device i gets the key pair (d_{di}, P_{di}) and the cloud server gets the key pair (d_c, P_c). Before communicating with each other, the device has to generate a random variable r_{di} and a random identity ID_{di} which will change at each session. Then the device computes the elliptic curve points $R_{di} = r_{di} \cdot G$, and sends the identity ID_{di}, together with the elliptic curve points R_{di} and its public key P_{di} to the cloud server. Receiving the values, cloud server will reply an acknowledgement (Ack) message to the device and store the three items, as illustrated in Fig. 2. Note that related values and messages are transmitted through a secure channel such as DTLS (Datagram Transport Layer Security) channel in CoAP (Constrained Application Protocol). DTLS is a communications protocol providing security to datagram-based applications by allowing them to communicate in a way

designed to prevent eavesdropping, tampering, or message forgery. The CoAP is a specialized web transfer protocol for use with constrained nodes and constrained networks in the Internet of Things. In Fig. 2, parameters listed below the device and the cloud server are the values that have been stored, respectively, in each entity in the system setup phase.

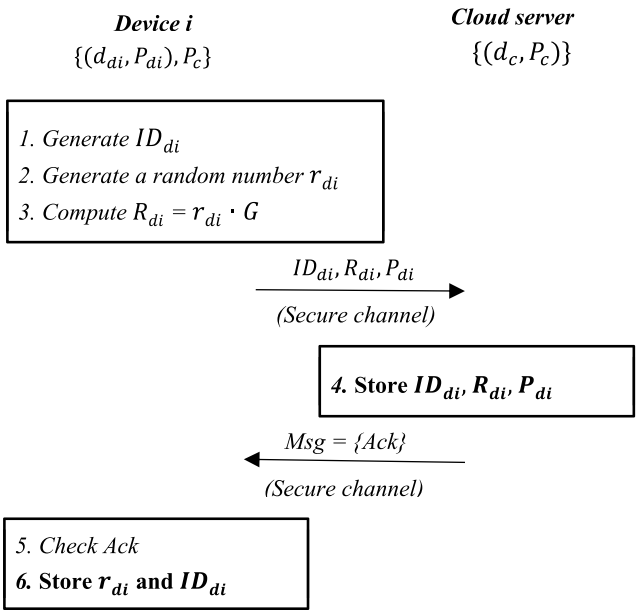

FIGURE 2. Device registration phase.

C. FOG REGISTRATION PHASE

To deploy a device, it has to register with the cloud server and obtain the GSK through the nearby fog node. The fog node will generate a random identity ID_f and send public key P_f along with the random identity ID_f to the cloud server securely. After receiving these values, the cloud server first chooses a random number r_c and computes the verifier with the ID_f, the generated random number r_c and the private key of its own. Then the cloud server sends the verifier to the fog node for future authentication. The steps for fog node registration are presented in Fig. 3.

Parameters listed below the fog node and the cloud server are the values that have been stored, respectively, in the system setup phase.

D. MUTUAL AUTHENTICATION AND GROUP KEY ESTABLISHMENT PHASE

In this stage, after the cloud server authenticating the fog node, a GSK shared between the fog node and the cloud server has been established. The GSK consists of the information of the devices, the fog node and the cloud server. Next, the fog node will authenticate joined devices and distribute the established GSK to them. The process can be divided into two parts. The first part is the mutual authentication and group key agreement process between the fog node and the cloud server. The second part is the mutual authentication and group

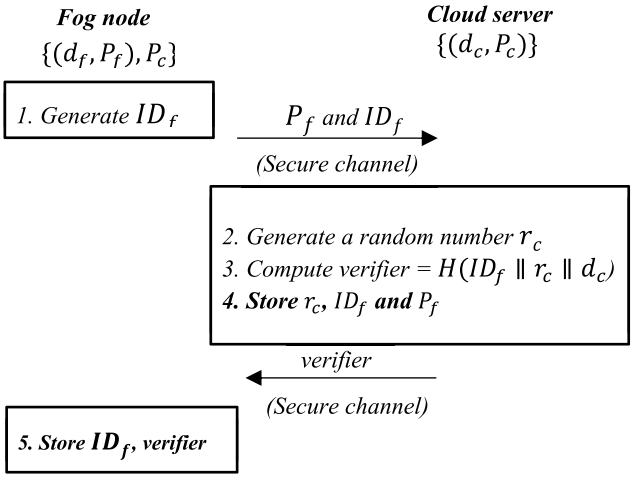

FIGURE 3. Fog node registration phase.

key distribution between the fog node and each device. The steps are represented in Fig. 4 and Fig. 5. Parameters listed below the device, the fog node and the cloud server are the values that have been stored, respectively, in each entity in the system setup phase, while the parameters of the second line are values that have been stored in the previous phase.

1) MUTUAL AUTHENTICATION AND GROUP KEY AGREEMENT PROCESS BETWEEN THE FOG NODE AND THE CLOUD SERVER (PART I)

In this part, the fog node needs to be checked by the cloud server to ensure the authenticity. If positive, the cloud server will continue to share a GSK with the fog node. Next, the fog node will conduct the following process.

(1) First, the fog node generates a random number r_f and computes the elliptic curve point $R_f = r_f \cdot G$ and the timestamp T_1. Next, to prove the identity, the fog node also needs to compute the credential $cred_f = H(d_f \cdot P_c)$. Finally, the hash value $A = H(cred_f \| verifier \| T_1 \| R_f)$ is computed, and the message $M_1 = \{T_1, A, R_f\}$ is sent to the cloud server.

(2) After receiving the message M_1 from the fog node, the cloud server first checks the timestamp T_1, then searches the database to verify the authenticity of the fog node. Then the cloud server will recover the $verifier = H(ID_f \| r_c \| d_c))$ and the credential $cred_f = H(P_f \cdot d_c)$. After deriving the recovered $verifier$, the credential $cred_f$, timestamp T_1, and the elliptic curve point R_f, the cloud server will check the hash value $\mathcal{A}$, in order to guarantee the integrity of message M_1. If the integrity of the hash value $\mathcal{A}$ is held, the process continues. The cloud server then generates a random number r_{c1} and computes $R_{c1} = r_{c1} \cdot G$, $GSK_{ds} = \sum_{i=1}^{n} r_{c1} \cdot P_{di} + d_c \cdot R_{di}$ and $GK = GK_{ds} + d_c \cdot R_f + r_{c1} \cdot P_f + d_c \cdot P_f = (g_x, g_y)$. The group session key GSK is computed as the x-coordinate of the GK which is $H(g_x)$.

Next, by generating a new random value random number r_{c2}, the cloud server compute $R_{c2} = (r_{c2} + d_c) \cdot G$,

the common key $K_1 = H\left((r_{c2} + d_c) \cdot P_f\right)$ with the fog node, $R = GK_{dc} \oplus H\left(ID_f\right)$, and $GSK_{fdi} = GSK \oplus H\left(ID_{di}\right) \oplus H\left((r_{c2} + d_c) \cdot (P_{di} + R_{di})\right)$, where the last two values represent a masked version of GK_{dc} *and* GSK, respectively. To achieve anonymity, the cloud server also compute the values $PID_{di} = ID_{di} \oplus H\left(R_{di}\right)$ $(i = 1, \ldots, n)$, which represents masked identities of the devices. To prove that the cloud server has its own private key, the cloud server computes the credential $cred_{cs} = H\left(d_c \cdot P_f\right)$. Finally, the cloud server encrypts the values of $R, P_{d1}, P_{d2}, \ldots, P_{dn}, R_{c1}, GSK_{fd1}, GSK_{fd2}, \ldots, GSK_{fdn}$ and $PID_{d1}, PID_{d2}, \ldots, PID_{dn}$ in the ciphertext S with the common key K_1 and computes timestamp T_2, $B_1 = H(S \parallel T_2 \parallel R_{c2} \parallel cred_{cs})$ and $B_2 = H\left(GSK\right)$. The message $M_2 = \{S, R_{c2}, T_2, B_1, B_2\}$ is sent to the fog node.

(3) Upon receiving the message M_2 from the cloud server, the fog node checks the timestamp T_2 and recovers the credential $cred_{cs} = H(d_f \cdot P_c)$. Then the fog node verifies the integrity of M_2 with the computed $cred_{cs}$, the received message and the hash value B_1.

To decrypts the ciphertext S, the fog node computes the common key $K_1 = H\left(R_{c2} \cdot d_f\right)$, and gets $R, P_{d1}, P_{d2}, \ldots, P_{dn}, R_{c1}, GSK_{fd1}, GSK_{fd2}, \ldots, GSK_{fdn}$ and $PID_{d1}, PID_{d2}, \ldots, PID_{dn}$. Finally, the fog node gets GK_{dc} by $R \oplus H\left(ID_f\right)$, then computes $GK = GK_{dc} + P_c \cdot r_f + R_{c1} \cdot d_f + P_c \cdot d_f = \left(g_x, g_y\right)$ and gets the $GSK = H\left(g_x\right)$. By checking if the hash value of the compute GSK equals to the received hash value B_2, the fog node can verify the authenticity of the GSK and the cloud server. Therefore, the cloud server and the fog node can achieve mutual authentication. To authenticate group of devices in the next part, the fog node stores the values $P_{d2}, \ldots, P_{dn}, R_{c1}, GSK_{fd1}, GSK_{fd2}, \ldots, GSK_{fdn}$ and $PID_{d1}, PID_{d2}, \ldots, PID_{dn}, R_{c2}$ and GSK.

2) MUTUAL AUTHENTICATION AND GROUP KEY DISTRIBUTION BETWEEN THE FOG NODE AND EACH DEVICE (PART II)

In this part, the fog node will verify joining devices to make sure the devices have already registered and have not been forged. Next, the fog node will send the masked GSK to the device and broadcast the start message as well as its public key P_f to group of devices.

(1) Upon receiving the start message, by using the private key, the device can derive the credential $cred_{di} = H(d_{di} \cdot P_f)$, $PID_{di} = ID_{di} \oplus H(R_{di})$ and the timestamp T_3. After combining these values in the hash value $C = H(cred_{di} \parallel PID_{di} \parallel T_3)$, the device stores the public key of the fog node and sends the message $M_3 = \{T_3, C\}$ to the fog node.

(2) After receiving the message, the fog node first checks the validity of the timestamp T_3 and then recovers the credential $cred_{di} = H(P_{di} \cdot d_f)$. Next, to authenticate the device, the integrity of the hash value C need to be checked by using the recovered $cred_{di}$ and the stored PID_{di}. In order to distribute the GSK to the device securely, the fog node computes the common key $K_2 = H(r_f \cdot P_{di})$ and encrypts the GSK_{fdi} in the ciphertext P with the common key K_2. Finally, the fog node uses the credential $cred'_f = H(d_f \cdot P_{di})$ and the timestamp T_4 to prove the authenticity of itself. The hash values $D_1 = H(P \parallel T_4 \parallel R_f \parallel cred'_f)$ and $D_2 = H\left(GSK\right)$ are computed, and the message $M_4 = \{R_f, P, R_{c2}, T_4, D_1, D_2\}$ is sent to the device.

(3) When the message M_4 arrives, the device first verifies the timestamp T_4. Then the device recovers the credential $cred'_f = H(P_f \cdot d_{di})$ and checks the hash value D_1 from the received message. To get the GSK, the common key $K_2 = H(R_f \cdot d_{di})$ are derived by the device. Furthermore, the device also needs to decrypt P and $\oplus H(ID_{di}) \oplus H(R_{c2} \cdot (d_{di} + r_{di}))$. After getting the GSK, the device checks the correctness of the GSK by comparing the hash value of the GSK with the received hash value D_2. If hash value of the GSK is the same as D_2, the authenticity of the fog node is verified, since only the entity authenticated by the cloud server has the masked identity PID of the devices. The verification of the hash value D_1 also illustrates that the fog node is not forged. Finally, the device stores the GSK for encrypting the transmitted message, and the group of devices can achieve mutual communication and transmit data to the cloud server through the fog node by using the GSK.

V. FORMAL PROOF OF SECURITY

In this section, we will prove the semantic security of the proposed protocol under the CK adversary model [13] and the random oracle model [12]. Because the registration phase is conducted through the secure channel, the proof of security will only include the mutual authentication and the group key establishment phase [36]. In this approach, four different entities are involved: the cloud server CS, the fog node FN, the devices D_i $(i = 1, \ldots, n)$ and a random oracle O. By assuming the adversary $\mathcal{A}$ will send the following queries to the oracle O, we then prove the Theorem 1 through a sequence of games based on the difference lemma [37].

- *Hash query.* $H(m)$. If m has been found in the list L_H, the hash value $H(m)$ will be returned. Otherwise, the random oracle will return a random number and add this value to the list L_H.
- *Send query.* This query simulates an active attack because the adversary can modify the message. The random oracle will simulate the participant and reply the query with the corresponding message. Six send queries are defined as follows:
  Send (Start, FN): $M_1 = \{T_1, A, R_f\}$
  Send (M_1, CS): $M_2 = \{S, R_{c2}, T_2, B_1, B_2\}$
  Send (M_2, FN): $Msg = \{start, P_f\}$
  Send (Msg, D_i*)* : $M_3 = \{T_3, C\}$
  Send (M_3, FN): $M_4 = \{R_f, P, R_{c2}, T_4, D_1, D_2\}$
  Send (M_4, D_i) : the messages M_1, M_2, M_3, M_4 and Msg will be added to the list L_s.
- *Execute query.* While receiving this query, the random oracle will execute the process by starting Send queries and answer all transmitted messages which are stored in the list L_s, that is, M_1, M_2, M_3, M_4 and Msg. This query

models the passive attacks that the adversary can only eavesdrop the communication.

- *Corrupt query.* In this query, the random oracle will return the long-term private key of a certain entity. The following results are queries returned from the random oracle.
  Corrupt (CS): d_c, P_c
  Corrupt (FN): d_f, P_f
  Corrupt $(D_i)(i \in \{1, \ldots, n\}) : d_{di}, P_{di}$
  Corrupt $(D_j)(j \in \{1, \ldots, n\} \cap j \neq i) : d_{dj}, P_{dj}$
- *Session specific state reveal queries (SSReveal).* In this query, the random oracle will reveal the session specific state information that is the information of local state in each session to $\mathcal{A}$. The following are the outputs returned from the random oracle.
  SSReveal(CS):

$$rc, r_{c1}, R_{c1}, ID_{di}\ (i=1, \ldots, n), ID_f, R_{di}\ (i = 1, \ldots, n), r_{c2}, T_2, verifier, GSK_{fdi}\ (i = 1, \ldots, n), GSK_{dc}, PID_{di}\ (i = 1, \ldots, n), cred_f, cred_{cs}$$

  SSReveal(FN):

$$ID_f, verifier, R_{c1}, rf, T_1, PID_{di}\ (i = 1, \ldots, n), GSK_{fdi}\ (i=1, \ldots, n), T_4, cred_f, cred_{cs}, cred_{di}, cred'_f$$

  SSReveal $(D_i)(i \in \{1, \ldots, n\}) : ID_{di}, (r_{di}, R_{di}), T_3, PID_{di}, cred_{di}, cred'_f$
  SSReveal $(D_j)(j \in \{1, \ldots, n \cap j \neq i) : ID_{dj}, (r_{dj}, R_{dj}), T_3, PID_{dj}, cred_{dj}, cred_f 1, \ldots, n \cap j \neq i) : ID_{dj}, (r_{dj}, R_{dj}), T_3, PID_{dj}, cred_{dj}, cred'_f$
- *Session key reveal query (SKReveal).* Upon receiving the query, the random oracle will return the *GSK* if the *GSK* has been generated.
- *Test query.* In this query, the random oracle flips a coin c. If the value $c = 1$, the random oracle will return the *GSK,* while the random oracle will return a random value which has the same length as the *GSK* if the value $c = 0$. This query can only be made when *SKReveal* or *Corrupt* queries has not been executed.

The goal of the adversary $\mathcal{A}$ is to distinguish the real session key from a random number, which occurs in the *Test* query. Let *Pr(S)* be the probability that $\mathcal{A}$ succeeds in predicting the results of the *Test* query. The advantage of $\mathcal{A}$ in breaking the semantic security of the proposed scheme equals to $Adv(\mathcal{A}) = |2Pr[S] - 1|$. If the advantage for $\mathcal{A}$ winning the games satisfies $Adv(\mathcal{A}) \leq \varepsilon$, for any sufficiently small value $\varepsilon > 0$, our scheme offers semantic security under the *CK* adversary and the random oracle model.

To prove the semantic security of the proposed scheme, the following definitions are given.

A. PARTNER

The two participants are partners if they have authenticated each other and share a common session key.

B. FRESHNESS

The session is called fresh if that session has not been exposed by *SKReveal* or *Corrupt* queries.

Lemma 1 (Difference Lemma): Let E_1, E_2, F be events defined in some probability distribution where F is defined as the failure event. The two events E_1 and E_2 proceed identically unless the failure event F occurs. That is, $E_1 \wedge \neg F \Leftrightarrow E_2 \wedge \neg F$. Both $Pr[E_1]$ *and* $Pr[E_2]$ have values between 0 and $Pr[F]$; therefore $|Pr[E_1] - Pr[E_2]| \leq Pr[F]$.

Theorem 1: Suppose there is a probabilistic polynomial time adversary $\mathcal{A}$ against the semantic security and wins the games with advantage $Adv(\mathcal{A})$. The advantage of $\mathcal{A}$ is bounded by $Adv(\mathcal{A}) \leq \frac{q_h^2}{2q} + \frac{(q_s+q_e)^2}{2q} + \frac{q_s^2}{2q} + Adv_{ECCDH}(A) \cdot q_h$, *where* q_h, q_s, q_e, q denote the times of the Hash, Send, Execute and the total random oracle queries respectively.

Proof: Several games will be defined and sorted according to the probability of being attacked. We will show that the advantage in attacking the proposed scheme will be bounded above by a certain probability according to the results of the difference lemma described in Lemma 1. Five games {Game 0, Game 1, Game 2, Game 3, Game 4} are defined as following. Let S_i to be the event that $\mathcal{A}$ wins the Game i, where $i \in \{0, 1, 2, 3, 4\}$, and E_i represents the i_{th} event.

1) GAME 0

This is the original attacking game described in the semantic security framework. By the definition in the semantic security framework, advantage of $\mathcal{A}$ is $Adv(\mathcal{A}) = |2Pr[S_0] - 1|$.

2) GAME 1

In this game, the oracles described above are simulated, and the results of the queries are stored in the lists L_H, L_S and L_A which are the results of the Hash query, Send query and other queries. Thus, the probability that $\mathcal{A}$ succeeds is

$$Pr[S_1] = Pr[S_0]. \tag{1}$$

3) GAME 2

In this game, all the queries are simulated except the following events.

Event 1 (E_1): The collision of the output of the hash queries among different sessions.

Event 2 (E_2): The collision of the generated random numbers among different sessions.

According to the birthday paradox, the probability that E_1 happens is

$$Pr[E_1] \leq \frac{q_h^2}{2q}$$

Since the random numbers will only be generated in Send and Execute queries, the probability that E_2 happens is

$$Pr[E_2] \leq \frac{(q_s + q_e)^2}{2q}$$

Consequently, according to the Difference lemma, we have

$$|Pr[S_2 - Pr[S_1]| \leq \frac{q_h^2}{2q} + \frac{(q_s + q_e)^2}{2q} \quad (2)$$

4) GAME 3

In this game, $\mathcal{A}$ is able to guess the hash value A, B_1, B_2, C, D_1, D_2 successfully without querying the random oracles. If the guess is correct, the scheme will be aborted. This situation will only happen when the cloud server rejects the hash value $\mathcal{A}$, the fog node rejects the hash values B_1, B_2 and C, and the device rejects the hash value D_1 and D_2 in the Send query. Thus, the polynomial is formulated by

$$|Pr[S_3] - Pr[S_2]| \leq \frac{q_s^2}{2q} \quad (3)$$

5) GAME 4

In this game, the model of the *CK* adversary is considered. $\mathcal{A}$ is able to reveal either the session specific state information or the long-term private variables of each participant. Aiming to acquire the *GSK*, $\mathcal{A}$ will perform the Execute and Hash queries. Additionally, the transcripts between every participant are available to $\mathcal{A}$. The following twelve cases are possible combinations. In each case, we first search for GSK_{fdi} $(i = 1, \ldots, n)$ in *SSReveal(CS)* and *SSReveal(FN)* queries. Next, GSK_{fdi} $(i = 1, \ldots, n)$ is derived to reconstruct the *GSK*. Moreover, to get the *GSK,* we need to decrypt S and $\oplus H(ID_f)$ to derive GK_{dc} first. For simplification, the notation i and j of the following cases are all $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, n \cap j \neq i$.

Case 1 (SSReveal (FN), SSReveal (CS), SSReveal ($\mathbf{D_i}$) and SSReveal ($\mathbf{D_j}$)): In case 1, $\mathcal{A}$ gets the $GSK_{fdi}(i = 1, \ldots, n)$ and the GK_{dc}. There are $ID_{di}(i = 1, \ldots, n)$, $R_{di}(r_{di})$ $(i = 1, \ldots, n)$, R_{c2} and P_c, but lacking of $P_{di}(d_{di})(i = 1, \ldots, n)$ to get the *GSK* from the $GSK_{fdi}(i = 1, \ldots, n)$. Besides, $\mathcal{A}$ will try to recover part of the *GSK* with P_f, r_{c1}, P_c, r_f and GK_{dc}. However, based on the difficulty of solving the ECCDH, P_c and P_f cannot be used to recover $d_c \cdot P_f$.

Case 2 (SSReveal (FN), SSReveal (CS), SSReveal ($\mathbf{D_i}$) and Corrupt ($\mathbf{D_j}$)): $\mathcal{A}$ gets GSK_{fdi} $(i = 1, \ldots, n)$ and GK_{dc}. Because of the lack of r_{di} and d_{di} of the same device, $\mathcal{A}$ cannot get *GSK* from GSK_{fdi} $(i = 1, \ldots, n)$. In another way, with P_f, r_{c1}, P_c and r_f, $\mathcal{A}$ can recover part of *GSK* with GK_{dc}. However, P_c and P_f are not sufficient to recover $d_c \cdot P_f$ unless solving the ECCDH problem.

Case 3 (SSReveal (FN), SSReveal (CS), Corrupt ($\mathbf{D_i}$) and Corrupt ($\mathbf{D_j}$)): $\mathcal{A}$ gets GSK_{fdi} $(i = 1, \ldots, n)$ and GK_{dc}. $\mathcal{A}$ tries to get part of *GSK* with GSK_{fdi} $(i = 1, \ldots, n)$ by $\oplus H(ID_{di})$, but there are only $R_{c2}(r_{c2})$, P_{di} (d_{di}) $(i = 1, \ldots, n)$, R_{di} $(i = 1, \ldots, n)$ and P_c. As a consequence, $\mathcal{A}$ is not able to obtain GSK. Next $\mathcal{A}$ may try to recover part of *GSK* with P_f, r_{c1}, P_c, r_f *and* GK_{dc}. However, to recover $d_c \cdot P_f$ by using P_c and P_f, the attacker needs to solve the ECCDH problem first.

Case 4 (SSReveal (FN), Corrupt (CS), SSReveal ($\mathbf{D_i}$) and SSReveal ($\mathbf{D_j}$)): $\mathcal{A}$ gets GSK_{fdi} $(i = 1, \ldots, n)$ from *SSReveal (FN).* $\mathcal{A}$ may use ID_{di} $(i = 1, \ldots, n)$ to eliminate $\oplus H(ID_{di})$ $(i = 1, \ldots, n)$, which is part of GSK_{fdi} $(i = 1, \ldots, n)$ to get *GSK*. However, in this case, the attacker cannot derive *GSK* with R_{c2}, P_{di} $(i = 1, \ldots, n)$ and R_{di} $(i = 1, \ldots, n)$ are available.

Case 5 (SSReveal (FN), Corrupt (CS), SSReveal ($\mathbf{D_i}$) and Corrupt ($\mathbf{D_j}$)): $\mathcal{A}$ gets GSK_{fdi} $(i = 1, \ldots, n)$ from *SSReveal (FN).* Since the attacker cannot derive relevant parameters r_{di} and d_{di} of the same device, ID_{di} of all devices; therefore, $\mathcal{A}$ is not able to derive *GSK* from GSK_{fdi} $(i = 1, \ldots, n)$.

Case 6 (SSReveal (FN), Corrupt (CS), Corrupt ($\mathbf{D_i}$) and Corrupt ($\mathbf{D_j}$)): $\mathcal{A}$ gets GSK_{fdi} $(i = 1, \ldots, n)$ from *SSReveal (FN).* There are several parameters P_{di} $(i = 1, \ldots, n)$, R_{c2} and P_c which can be used to get *GSK*, but due to the lack of ID_{di} $(i = 1, \ldots, n)$ and $R_{di}(r_{di})$ $(i = 1, \ldots, n)$, $\mathcal{A}$ is unable to obtain GSK.

Case 7 (Corrupt (FN), SSReveal (CS), SSReveal ($\mathbf{D_i}$) and SSReveal ($\mathbf{D_j}$)): $\mathcal{A}$ gets GSK_{fdi} $(i = 1, \ldots, n)$ and GK_{dc}. $\mathcal{A}$ may try to use ID_{di} $(i = 1, \ldots, n)$ to get *GSK*. However, P_{di} $(i = 1, \ldots, n)$, R_{di} $(i = 1, \ldots, n)$, P_c and $R_{c2}(r_{c2})$ are not sufficient to eliminate the remaining part of GSK_{fdi} $(i = 1, \ldots, n)$ and get *GSK*. Trying to recover part of *GSK* with GK_{dc} by adding $r_{c1} \cdot P_f + P_c \cdot d_f$ may be another way for $\mathcal{A}$. However with only P_c and R_f are available, $\mathcal{A}$ needs to solve the ECCDH problem first, in order to derive *GSK*.

Case 8 (Corrupt (FN), SSReveal (CS), SSReveal ($\mathbf{D_i}$) and Corrupt ($\mathbf{D_j}$)): $\mathcal{A}$ gets GSK_{fdi} $(i = 1, \ldots, n)$ and GK_{dc}. $\mathcal{A}$ can use ID_{di} $(i = 1, \ldots, n)$ to eliminate $\oplus H(ID_{di})$ $(i = 1, \ldots, n)$ part of GSK_{fdi} $(i = 1, \ldots, n)$ and obtain the *GSK*. However, P_{di} $(i = 1, \ldots, n)$, R_{di} $(i = 1, \ldots, n)$ and $R_{c2}(r_{c2})$ are parameters that $\mathcal{A}$ can derive; therefore, $\mathcal{A}$ cannot acquire the *GSK*.

Case 9 (Corrupt (FN), SSReveal (CS), Corrupt ($\mathbf{D_i}$) and Corrupt ($\mathbf{D_j}$): $\mathcal{A}$ gets GSK_{fdi} $(i = 1, \ldots, n)$ and GK_{dc}. $\mathcal{A}$ can use ID_{di} $(i = 1, \ldots, n)$ to eliminate $\oplus H(ID_{di})$ $(i = 1, \ldots, n)$ part of GSK_{fdi} $(i = 1, \ldots, n)$ and to obtain the *GSK*. However, parameters P_{di} $(i = 1, \ldots, n)$, R_{di} $(i = 1, \ldots, n)$, P_c and $R_{c2}(r_{c2})$ are not sufficient to get the *GSK*. Additionally, $\mathcal{A}$ can recover part of *GSK* with GK_{dc} by adding $r_{c1} \cdot P_f + P_c \cdot d_f$. To recover the remaining part $d_c \cdot R_f$ of the *GSK*, $\mathcal{A}$ need to solve ECCDH problem first, since there are only P_c and R_f can be used.

Case 10 (Corrupt (FN), Corrupt (CS), SSReveal ($\mathbf{D_i}$) and SSReveal ($\mathbf{D_j}$): $\mathcal{A}$ may try to decrypt S and acquire the GSK_{fdi} $(i = 1, \ldots, n)$. However, it is not sufficient to acquire the *GSK* from GSK_{fdi} $(i = 1, \ldots, n)$ by using ID_{di} $(i = 1, \ldots, n)$, P_{di} $(i = 1, \ldots, n)$, $R_{di}(r_{di})$ $(i = 1, \ldots, n)$ and R_{c2}.

Case 11 (Corrupt (FN), Corrupt (CS), SSReveal ($\mathbf{D_i}$) and Corrupt ($\mathbf{D_j}$)): $\mathcal{A}$ gets GSK_{fdi} $(i = 1, \ldots, n)$ by decrypting S. However, without ID_{di}, $P_{di}(d_{di})$ *and* $R_{di}(r_{di})$ of the device, $\mathcal{A}$ cannot eliminate $\oplus H((r_{c2} + d_c) \cdot (P_{di} + R_{di}))$ to get *GSK*.

Case 12 (Corrupt (FN), Corrupt (CS), Corrupt ($\mathbf{D_i}$) and Corrupt ($\mathbf{D_j}$)): $\mathcal{A}$ gets GSK_{fdi} $(i = 1, \ldots, n)$ by

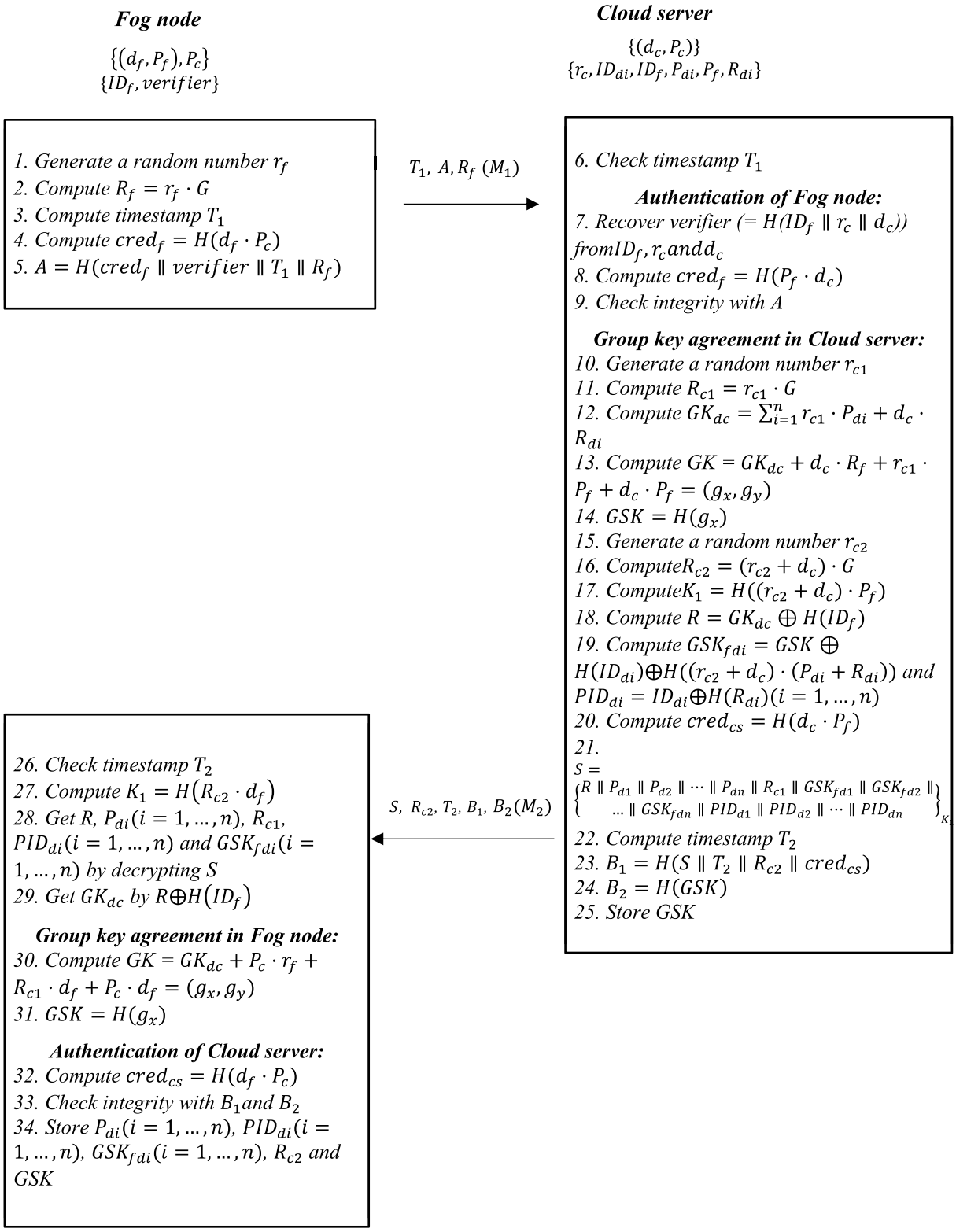

FIGURE 4. Mutual authentication and group key agreement process.

decrypting S, but it is hard for $\mathcal{A}$ to derive the GSK from GSK_{fdi} $(i = 1, \ldots, n)$ without having the ID_{di} $(i = 1, \ldots, n)$ and $R_{di}(r_{di})$ $(i = 1, \ldots, n)$.

The above cases indicate that $\mathcal{A}$ does not have sufficient information to reconstruct GSK unless solving the ECCDH problem. The difference between Game 3 and Game 4 is

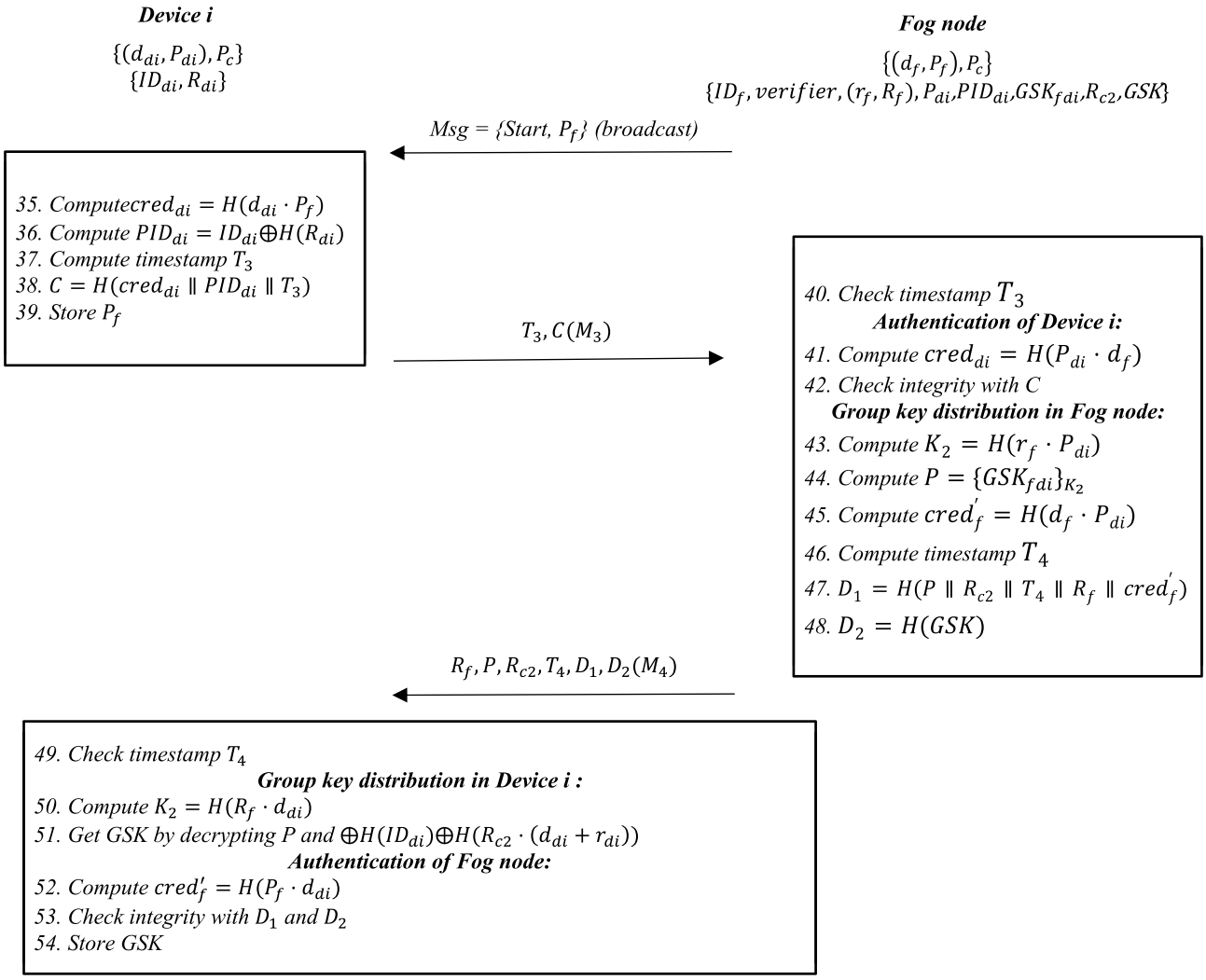

FIGURE 5. Mutual authentication and group key distribution.

ECCDH assumption. Thus,

$$|Pr[S_4] - Pr[S_3]| \leq Adv_{ECCDH}(\mathcal{A}) \cdot q_h \tag{4}$$

From Game 0 to Game 4, Lemma 1 and eq. (1) to (4), we can prove Theorem 1, that is, the advantage of $Adv(\mathcal{A})$ is bounded.

$$Adv(\mathcal{A}) \leq \frac{q_h^2}{2q} + \frac{(q_s + q_e)^2}{2q} + \frac{q_s^2}{2q} + Adv_{ECCDH}(A) \cdot q_h$$

VI. SECURITY ANALYSIS

This section demonstrates that the proposed approach is secure against several attacks.

A. ANONYMITY AND UN-TRACEABILITY

In the proposed approach, an adversary cannot retrieve the identities of other devices while transmitting messages during the mutual authentication and key establishment process. Since the whole process is combined with hash values and ciphertext. The identities are hidden in the ciphertexts

$$S = \left\{ \begin{array}{l} R \| P_{d1} \| P_{d2} \| \ldots \| P_{dn} \| R_{c1} \| GSK_{fd1} \| \\ GSK_{fd2} \| \ldots \| GSK_{fdn} \| PID_{d1} \| PID_{d2} \\ \| \ldots \| PID_{dn} \end{array} \right\}_{K_1}$$

and hash values $A = H_4\left(cred_f \| verifier \| T_1 \| R_f\right)$, $C = H_8(cred_{di} \| PID_{di} \| T_3)$. In addition, since the ciphertext is encrypted with the private key, only the authenticated devices, fog node and cloud server can recover the identities. Moreover, the entities will be authenticated at each session.

B. MAN IN-THE-MIDDLE ATTACK

Even if the transcripts are intercepted and modified by the attacker, the attacker still cannot forge each of the entity. The transcripts are composed of both the long-term variables and the session state specific information of the entity. According to the CK adversary model, the attacker cannot get both the long-term variables and the session state specific information of the same entity. If the attacker modifies the transcripts,

the forged messages will not be authenticated by all entities. Hence, the proposed scheme is against man in-the-middle attacks.

C. REPLAY ATTACK

In the proposed approach, the transmitted messages contain timestamps. Only the message with the newest time will be accepted. Thus, all the entities will reject the message without the correct timestamps.

D. PERFECT FORWARD SECRECY/GROUP FORWARD SECRECY

If the long-term private keys of all entities are compromised, the previous establish *GSK* will still be secure. Since the *GSK* is composed of both the long-term private keys and the random numbers, and the ciphertexts are encrypted with both long-term private key and state specific information, so the attacker cannot compute the *GSK* and cannot recover the *GSK* from the ciphertext

$$S = \left\{ \begin{array}{l} R \parallel P_{d1} \parallel P_{d2} \parallel \ldots \parallel P_{dn} \parallel R_{c1} \parallel GSK_{fd1} \parallel \\ GSK_{fd2} \parallel \ldots \parallel GSK_{fdn} \parallel PID_{d1} \parallel PID_{d2} \\ \parallel \ldots \parallel PID_{dn} \end{array} \right\}_{K_1}$$

and $P = \{GSK_{fdi}\}_{K_2}$ with only long-term private keys of all entities.

E. BACKWARD CONFIDENTIALITY

A device joining a new session cannot get the former GSK from the present session. As described in Perfect forward secrecy/ Group forward secrecy, the GSK is composed of the long-term private keys and the random numbers. The state specific information and the random numbers will change at each session; therefore, the new device cannot acquire the former GSK by using the wrong key.

F. UN-LINKABILITY AND MOBILITY

The GSK will change at each session according to the fog node. If one of the fog nodes is compromised, the device can refuse the connection. With un-linkability, the proposed scheme can be applied to other scenarios such as mobile devices.

The analysis of security features for our proposed scheme in comparison with the recent protocols has been presented in Table2. As it can be observed, our suggested protocol is secure against all mentioned attacks and is able to provide security requirements such as anonymity and un-traceability. Hence, our proposed scheme is able to provide a high level of security, compared to other existing authentication schemes.

VII. PERFORMANCE

In this section, we compare our approach with the approach of [10], and further discuss the computational cost and communication cost of the proposed scheme.

TABLE 2. Comparison of functionality features.

	[38]	[31]	[39]	[40]	Proposed Method
Resist Man In-The Middle Attack	X	V	X	X	V
Resist Replay Attack	X	V	X	X	V
KeyCompromise Impersonation	X	X	X	V	V
Perfect Forward Secrecy	X	V	X	X	V
Privacy Preserving	X	V	X	X	V

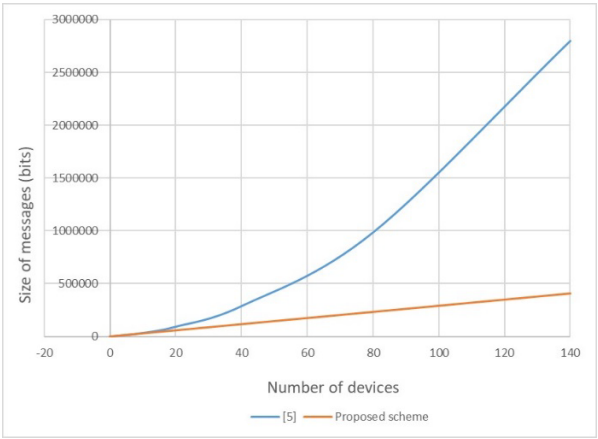

FIGURE 6. Comparison of message sizes in different scheme.

A. COMPUTATIONAL COSTS

The computational cost is measured by counting the most computationally-intensive operations such as point multiplication, point addition, hash operation and symmetric encryption and decryption. Noted that the time for point multiplication, point addition, hash operation, symmetric encryption and symmetric decryption are T_m, T_a, T_h, T_e and T_d. In this study, we used a personal computer with a 1.8GHz to 2.0GHz CPU. This computer is an Intel®Core™i7-8550U with 8192MB RAM, running Windows 10 operating system as a cloud server, and used Raspberry Pi 3 Model B with 1.2 GHz CPU as the fog node. An Arduino Uno with 32 KB flash memory and 2 KB static random access memory (SRAM) was used as the device. Table 3 provides the computational time of each operation. During the experiment, we used the Arduino Crypto library [41], which can implement these operations under the terms of the MIT license. In addition, the elliptic curve operation is implemented by Curve25519 that provides a 128-bit security level [42]. We also use AES with ECB mode to perform symmetric encryption/decryption operations and the key size is considered with 128 bits. We choose SHA256 as the secure hash algorithm of the hash function.

Table 3 shows the results of comparing the operands proposed in this study with the operands in [10].

TABLE 3. Computed time of each operation in different platforms.

Platform	EC points multiplication (μs)	EC points addition (μs)	AES_ECB encryption (μs per byte)	AES_ECB decryption (μs per byte)	SHA256 (μs per byte)
Arduino Uno	6634856	13104	35.39	67.45	43.65
Personal Computer	3	1	0.000125	0.0002125	0.00003125
Raspberry Pi 3 Model B	22375	61	0.744038	1.1499	0.152844

TABLE 4. Computational costs of the proposed Scheme.

	Device	Fog node	Cloud server
Registration	$1T_m$	0	$1T_h$
Group key establishment	$4T_m + 9T_h + 1T_d$	$(3n+7)T_m + 3T_a + (6n+8)T_h + nT_e + 1T_d$	$(3n+8)T_m + (3n+2)T_a + (3n+9)T_h + 1T_e$
[10] Initialization	$5T_m + 3T_h + 1T_e$	$1T_m + 2T_h + 1T_e$	$4T_m + 1T_a + 4T_h + 2T_e$
[10] Group key agreement	$7T_m + 3T_a + 5T_h + 1T_e$	$(4n+7)T_m + (2n+2)T_a + (3n+2)T_h + 2nT_e$	$(8n+6)T_m + (3n+2)T_a + (6n+4)T_h + 2nT_e$

TABLE 5. Communication costs of the proposed scheme.

	Size of the proposed scheme (bits)	**Size of [10] (bits)**
Registration/ Initialization	1127	1925
Group key establishment/ agreement	545 (M_1) 1058 + 1283n (M_2) 288n (M_3) 1314n (M_4)	805 (M_1) 1286 (M_2) 514 (M_3) $128n^2$ (M_4) 770n (M_5) 770n (M_6)
Total (excluding registration)	1603 + 2885n	2091 + 2054n + $128n^2$

The parameter n represents the number of devices. Since the time of multiplication and addition of elliptic curve points is longer than that of symmetric encryption/decryption and hashing, we can consider point multiplication and addition the most. Compared with [10], the calculation computational cost of this scheme is less than [10], because the coefficient of n is smaller.

B. COMMUNICATION COSTS

For communication cost, we measure the amount of the transmitted messages which are sent between different entities. We consider the security level to be 128 bits. Because elliptic curve points can be compressed to just one coordinate plus 1 bit, the size of public key is 257 bits. Next, the size of the SHA256 hash function is 256 bits. The sizes of the timestamp, Ack and Start messages, and identity are assumed to be 32 bits, 4 bits and 32 bits, respectively. Table 5 shows the size of the transmitted messages in the whole process. We also compare the proposed scheme with that of [10]. In the approach [10], n elliptic curve points need be transmitted for computing the GSK, and these n points should be transmitted to n devices as well. Thus, in, the communication cost in [10] is proportional to the square of the number of devices n while in the proposed scheme is linear with respect to n, as shown Table 5.

VIII. CONCLUSION

A mutual authentication group key establishment scheme is proposed to form a safe and effective environment for fog computing architecture. We have shown the semantic security

of the scheme in the random oracle model under the CK adversary model. In the proposed scheme, fog nodes can be used to verify the authenticity of the device with reduced overheads of the cloud server. The result also shows that the computing and communication costs are less than the existing method. The proposed method can be deployed in practical IoTs applications with fog nodes being off-loading the cloud server.

REFERENCES

[1] F. Bonomi, "Fog computing and its role in the Internet of Things," in *Proc. 1st MCC Workshop Mobile Cloud Comput.*, 2012, pp. 13–16.
[2] M. Ma, D. He, H. Wang, N. Kumar, and K.-K.-R. Choo, "An efficient and provably secure authenticated key agreement protocol for fog-based vehicular ad-hoc networks," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8065–8075, Oct. 2019.
[3] J. Ni, K. Zhang, X. Lin, and X. Shen, "Securing fog computing for Internet of Things applications: Challenges and solutions," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 601–628, 1st Quart., 2018.
[4] I. Stojmenovic, S. Wen, X. Huang, and H. Luan, "An overview of fog computing and its security issues," *Concurrency Comput., Pract. Exper.*, vol. 28, no. 10, pp. 2991–3005, 2016.
[5] A. Alrawais, A. Alhothaily, C. Hu, and X. Cheng, "Fog computing for the Internet of Things: Security and privacy issues," *IEEE Internet Comput.*, vol. 21, no. 2, pp. 34–42, Mar./Apr. 2017.
[6] K. Lee, D. Kim, D. Ha, U. Rajput, and H. Oh, "On security and privacy issues of fog computing supported Internet of Things environment," in *Proc. 6th Int. Conf. Netw. Future (NOF)*, Sep. 2015, pp. 1–3.
[7] Y. Guo, Z. Zhang, and Y. Guo, "Fog-centric authenticated key agreement scheme without trusted parties," *IEEE Syst. J.*, early access, Oct. 26, 2020, doi: 10.1109/JSYST.2020.3022244.
[8] X. Jia, D. He, N. Kumar, and K.-K. R. Choo, "Authenticated key agreement scheme for fog-driven IoT healthcare system," *Wireless Netw.*, vol. 25, pp. 4737–4750, May 2018.
[9] P. Gope, "LAAP: Lightweight anonymous authentication protocol for D2D-aided fog computing paradigm," *Comput. Secur.*, vol. 86, pp. 223–237, Sep. 2019.
[10] P. Shabisha, A. Braeken, P. Kumar, and K. Steenhaut, "Fog-orchestrated and server-controlled anonymous group authentication and key agreement," *IEEE Access*, vol. 7, pp. 150247–150261, 2019.
[11] H. Xiong and Y. Z. Wu Lu, "A survey of group key agreement protocols with constant rounds," *ACM Comput. Surv.*, vol. 52, no. 3, pp. 1–32, 2019.
[12] M. Bellare and D. P. P. Rogaway, "Authenticated key exchange secure against dictionary attacks," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, Bruges, Belgium, May 2000, pp. 139–155.
[13] R. Canetti and H. Krawczyk, "Analysis of key-exchange protocols and their use for building secure channels," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, Innsbruck, Austria, May 2001, pp. 453–474.
[14] R. Amin, N. Kumar, G. Biswas, R. Iqbal, and V. Chang, "A light weight authentication protocol for IoT-enabled devices in distributed Cloud Computing environment," *Future Gener. Comput. Syst.*, vol. 78, pp. 1005–1019, Jan. 2018.
[15] V. Odelu, A. K. Das, M. Wazid, and M. Conti, "Provably secure authenticated key agreement scheme for smart grid," *IEEE Trans. Smart Grid*, vol. 9, no. 3, pp. 1900–1910, May 2018.
[16] W. Li, B. Li, Y. Zhao, P. Wang, and F. Wei, "Cryptanalysis and security enhancement of three authentication schemes in wireless sensor networks," *Wireless Commun. Mobile Comput.*, vol. 2018, pp. 1–11, Jul. 2018.
[17] M. N. Aman, K. C. Chua, and B. Sikdar, "Mutual authentication in IoT systems using physical unclonable functions," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1327–1340, Oct. 2017.
[18] U. Guin, A. Singh, M. Alam, J. Canedo, and A. Skjellum, "A secure low-cost edge device authentication scheme for the Internet of Things," in *Proc. 31st Int. Conf. VLSI Design 17th Int. Conf. Embedded Syst. (VLSID)*, Jan. 2018, pp. 85–90.
[19] H. A. Al Hamid, S. M. M. Rahman, M. S. Hossain, A. Almogren, and A. Alamri, "A security model for preserving the privacy of medical big data in a healthcare cloud using a fog computing facility with pairing-based cryptography," *IEEE Access*, vol. 5, pp. 22313–22328, 2017.
[20] D. Wang and P. Wang, "On the anonymity of two-factor authentication schemes for wireless sensor networks: Attacks, principle and solutions," *Comput. Netw.*, vol. 73, pp. 41–57, Jul. 2014.
[21] S. Patonico, A. Braeken, and K. Steenhaut, "Identity-based and anonymous key agreement protocol for fog computing resistant in the Canetti-Krawczyk security model," *Wireless Netw.*, pp. 1–13, Jul. 2019, doi: 10.1007/s11276-019-02084-6.
[22] J. Shen, H. Yang, A. Wang, T. Zhou, and C. Wang, "Lightweight authentication and matrix-based key agreement scheme for healthcare in fog computing," *Peer Peer Netw. Appl.*, vol. 12, no. 4, pp. 924–933, Jul. 2019.
[23] C.-I. Fan, J.-J. Huang, M.-Z. Zhong, R.-H. Hsu, W.-T. Chen, and J. Lee, "ReHand: Secure region-based fast handover with user anonymity for small cell networks in mobile communications," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 927–942, 2020.
[24] Y. Guo and Y. Guo, "FogHA: An efficient handover authentication for mobile devices in fog computing," *Comput. Secur.*, vol. 108, Sep. 2021, Art. no. 102358.
[25] H. Zhu, "Secure chaotic maps-based group key agreement scheme with privacy preserving," *Int. J. Netw. Secur.*, vol. 18, no. 6, pp. 1001–1009, Nov. 2016.
[26] P. Porambage, A. Braeken, C. Schmitt, A. Gurtov, M. Ylianttila, and B. Stiller, "Group key establishment for enabling secure multicast communication in wireless sensor networks deployed for IoT applications," *IEEE Access*, vol. 3, pp. 1503–1511, 2015.
[27] T. Y. Wu, Y. M. Tseng, and T. T. Tsai, "A revocable ID-based authenticated group key exchange protocol with resistant to malicious participants," *Comput. Netw.*, vol. 56, no. 12, pp. 2994–3006, 2012.
[28] H.-M. Sun, B.-Z. He, C.-M. Chen, T.-Y. Wu, C.-H. Lin, and H. Wang, "A provable authenticated group key agreement protocol for mobile environment," *Inf. Sci.*, vol. 321, pp. 224–237, Nov. 2015.
[29] A. S. Sani, D. Yuan, P. L. Yeoh, W. Bao, S. Chen, and B. Vucetic, "A lightweight security and privacy-enhancing key establishment for Internet of Things applications," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.
[30] K. Y. Choi, J. Y. Hwang, and D. H. Lee, "Efficient ID-based group key agreement with bilinear maps," in *Proc. Int. Workshop Public Key Cryptogr.*, Singapore, Mar. 2004, pp. 130–144.
[31] A. B. Amor, M. Abid, and A. Meddeb, "A privacy-preserving authentication scheme in an edge-fog environment," in *Proc. IEEE/ACS 14th Int. Conf. Comput. Syst. Appl. (AICCSA)*, Oct. 2017, pp. 1225–1231.
[32] R. Kalaria, A. S. M. Kayes, W. Rahayu, and E. Pardede, "A secure mutual authentication approach to fog computing environment," *Comput. Secur.*, vol. 111, Dec. 2021, Art. no. 102483.
[33] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*. Boca Raton, FL, USA: CRC Press, 2014.
[34] D. Brown, "Standards for efficient cryptography, SEC 1: Elliptic curve cryptography," *Released Standard Version*, May 2009.
[35] M. Campagna, "SEC 4: Elliptic curve Qu-Vanstone implicit certificate scheme (ECQV)," Certicom Res., Toronto, ON, Canada, Tech. Rep., Jan. 2013.
[36] D. R. Brown and R. S. A. Gallant Vanstone, "Provably secure implicit certificate schemes," in *Proc. Int. Conf. Financial Cryptogr.*, Grand Cayman, British West Indies, Feb. 2001, pp. 156–165.
[37] V. Shoup, "Sequences of games: A tool for taming complexity in security proofs," *IACR Cryptol. ePrint Arch.*, pp. 332–365, 2004.
[38] M. S. Farash, "Security analysis and enhancements of an improved authentication for session initiation protocol with provable security," *Peer Peer Netw. Appl.*, vol. 9, no. 1, pp. 82–91, Jan. 2016.
[39] S. Qiu, G. Xu, H. Ahmad, and Y. Guo, "An enhanced password authentication scheme for session initiation protocol with perfect forward secrecy," *PLoS ONE*, vol. 13, no. 3, Mar. 2018, Art. no. e0194072.
[40] M. Nikooghadam and H. Amintoosi, "A secure and robust elliptic curve cryptographybased mutual authentication scheme for session initiation protocol," *Secur. Privacy*, vol. 3, no. 1, p. e92, 2020.
[41] R. Weatherley. (2018). *Arduino Cryptography Library. Source Code*. [Online]. Available: http://github.com/rweather/arduinolibs
[42] A. Langley and M. S. H. Turner, *Elliptic Curves for Security*, document RFC 7748, 2016.