Wireless Networks (2019) 25:4737–4750
https://doi.org/10.1007/s11276-018-1759-3

Authenticated key agreement scheme for fog-driven IoT healthcare system

Xiaoying Jia[1] · Debiao He[2] · Neeraj Kumar[3] · Kim-Kwang Raymond Choo[4,5]

Published online: 29 May 2018
© Springer Science+Business Media, LLC, part of Springer Nature 2018

Abstract
The convergence of cloud computing and Internet of Things (IoT) is partially due to the pragmatic need for delivering extended services to a broader user base in diverse situations. However, cloud computing has its limitation for applications requiring low-latency and high mobility, particularly in adversarial settings (e.g. battlefields). To some extent, such limitations can be mitigated in a fog computing paradigm since the latter bridges the gap between remote cloud data center and the end devices (via some fog nodes). However, fog nodes are often deployed in remote and unprotected places. This necessitates the design of security solutions for a fog-based environment. In this paper, we investigate the fog-driven IoT healthcare system, focusing only on authentication and key agreement. Specifically, we propose a three-party authenticated key agreement protocol from bilinear pairings. We introduce the security model and present the formal security proof, as well as security analysis against common attacks. We then evaluate its performance, in terms of communication and computation costs.

Keywords Fog computing · Cloud computing · Internet-of-Things (IoT) · Healthcare · Authenticated key agreement

1 Introduction

Cloud computing is relatively mature and has been utilized in a number of applications, including those involving Internet of Things (IoT) devices. IoT devices are Internet connected devices (also referred to as objects or things) designed to collect data (e.g. sense environmental data such as moisture and air temperature) prior to sending the data to a processing center (e.g. the cloud) for storage, processing, analysis, etc. In other words, the bulk of the processing is undertaken at a remote data center site that may be physically located in another country. Such a deployment model may not be suitable for applications that have specific requirements [4], such as the following:

– *Latency/delay sensitive applications* Latency/delay sensitive applications such as video conferencing and industrial automation may demand an extremely short latency in order to maintain a high quality of (user) experience (e.g. Quality of Service—QoS, and Quality of Experience—QoE). Other latency sensitive applications such as battlefields, smart traffic lights and emergency response services may require an even

✉ Debiao He
hedebiao@163.com

Xiaoying Jia
xiaoyin.jia@mail.scuec.edu.cn

Neeraj Kumar
neeraj.kumar@thapar.edu

Kim-Kwang Raymond Choo
raymond.choo@fulbrightmail.org

1 School of Mathematics and Statistics, South-Central University for Nationalities, Wuhan, China

2 Key Laboratory of Aerospace Information Security and Trusted Computing Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan, China

3 Department of Computer Science and Engineering, Thapar University, Patiala, India

4 Department of Information Systems and Cyber Security, The University of Texas at San Antonio, San Antonio, TX 78249, USA

5 Department of Electrical and Computer Engineering, The University of Texas at San Antonio, San Antonio, TX 78249, USA

shorter latency as any delay can have real-world consequences (e.g. fatalities).

- *Network connectivity constrained applications* In a cloud computing model, all data and requests are transmitted to and processed at the cloud server. The significant increase in the number of IoT devices also results in a corresponding increase in the amount of data to be transmitted, processed, stored, etc. However, IoT devices typically have limited network (and computing) capacities. Thus, it is challenging to deliver continuous and reliable service in a constrained network environment.
- *Geographically distributed applications* IoT applications can be geographically distributed, for example, in smart grids, railways, and pipeline monitoring, and the distance between IoT devices and remote cloud center affects latency and consequently the quality of service.
- *Real-time mobile applications* Cloud servers are deployed in a static location, but IoT applications such as those deployed in smart transportation and environmental monitoring are dynamic and have high mobility.

1.1 Fog computing

To mitigate some of the above challenges, a more recent cloud-related trend is to move computing capabilities (e.g. limited processing and storage) closer to the users and their devices. In such a paradigm (also referred to as fog computing), fog nodes (routers, gateways, switchers, access points, etc) are deployed at the edge of the network, geographically close to the end devices [15, 22]. By extending the cloud services to the edge of network, fog computing turns a cloud data center into a distributed platform, while retaining cloud services for users; thus, minimizing latency and improving QoS and/or QoE. Furthermore, fog computing provides better support for mobile and geographically distributed devices and large-scale sensor networks.

Typically, fog computing has a hierarchical architecture, comprising the cloud layer, the fog layer, and the end device layer (see Fig. 1).

Healthcare system is one of several potential industry IoT (IIoT) applications, where IoT devices can be deployed in venues as big as a large healthcare center (e.g. University of Texas Health System) or venues as small as a private specialist clinic. In addition, the popularity of wearable smart devices has changed the healthcare system from hospital-centric to patient-centric, by allowing citizens to access healthcare services anytime, anywhere without any geographical limitations. For example, patients can upload their health conditions to healthcare professionals using their embedded (e.g. pacemakers), wearable

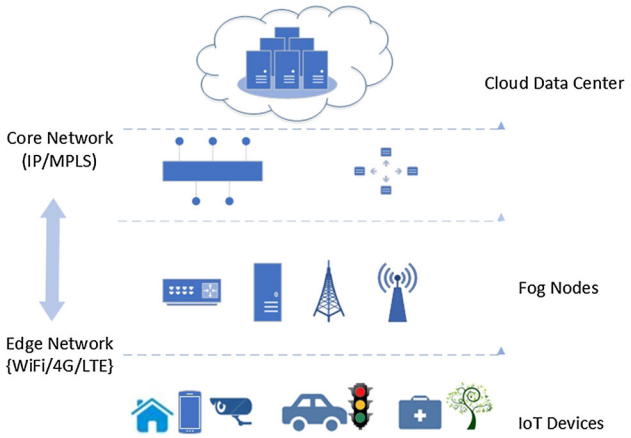

Fig. 1 An example hierarchical fog computing architecture

(e.g. smart watches) and other IoT devices, without the need to physically visit a hospital.

Healthcare is one particular industry that requires latency to be minimized. For example, a minute delay could have catastrophic consequences on an individual suffering from a heart attack (e.g. fatality). Thus, we posit a fog-driven solution in this context. For example, a fog-driven IoT healthcare system described in Fig. 1 comprises three layers, namely: the IoT healthcare devices layer, the IoT healthcare fog layer, and the IoT healthcare cloud layer [8].

- *IoT healthcare device layer* This layer consists of a large number of distributed IoT devices and sensors, which are used to monitor the physical objects, collect data, and transmit data to the fog nodes. For example, embedded and wearable devices facilitate the collection of a user's health data (e.g. blood pressure, heartbeat and body temperature), which can be used by the responding healthcare professionals to formulate medical delivery strategy when the user arrives at the hospital. These devices connect to the nearest fog nodes through various communication protocols, either via a wired or a wireless network.
- *IoT Healthcare fog layer* This layer consists of fog nodes. The fog nodes can be deployed in appropriate sites. Fog nodes have the capability to process, transmit, and store the received data. They receive data and commands from both the user and the cloud, and are responsible for filtering the raw data, and sending the aggregated data to the cloud for long-term storage or further analysis. These fog nodes can also provide real-time response on time-sensitive applications. To accommodate different kinds of devices, the fog nodes must be equipped with different interfaces and have the ability to communicate with different communication protocols. Fog nodes and the cloud

server are connected by a backbone network, such as an IP/MPLS network.

– *IoT Healthcare cloud layer* This layer consists of high performance servers and storage devices. It provides computing, networking and storage services for large amount of data. Unlike conventional cloud computing, cloud servers are only responsible for work that requires high computing power and has low latency requirements. Cloud server also provides long-term storage for critical or sensitive data such as medical records of patients. Big data analytics is also an important function of the cloud server. By analyzing the aggregated medical data or stored / archival data, the cloud layer can provide suggestions about the health status, disease predicting, etc. The cloud server also provides an interface for the patients and the healthcare professionals (e.g. medical doctors) to monitor and manage the connected IoT devices. It also enables the users to read or share their medical data.

Figure 2 shows our proposed fog-driven IoT personal health monitoring system.

Unlike a centralized cloud computing system, fog nodes are usually deployed in an environment without adequate physical security measures. In other words, end devices and fog nodes may be easier to compromise (e.g. data transmitted between devices and devices, devices and fog nodes, fog nodes and cloud servers can be eavesdropped, modified and replayed). This could result in the leakage of user privacy information, such as identity, location, health status, and medical records [1, 14].

In order to establish trust and prevent impersonation, each user or fog node in the system should be uniquely identified and authenticated. Furthermore, to ensure the security and privacy of data transmitted over the public channel and stored in fog nodes or cloud server, the data need to be encrypted. However, it is not realistic to pre-share session keys between fog nodes and end devices due to the dynamic and mobility nature of end devices and fog nodes. Thus, authenticated key agreement (AKA) protocol is a good solution to authenticate users or nodes and produce common session keys [7]. Unfortunately, there are relatively few AKA schemes designed specifically for a fog computing environment. In the settings of most existing AKA protocols designed for IoT applications, a user and an IoT device authenticate each other with the help of a gateway node, which has no ability to process data and does not have access to a common session key. Also, usually there is no role for a cloud server in these protocols. In the fog computing environment, however, fog nodes are more than gateway nodes. Fog nodes need to preprocess data collected from IoT devices, and receive instructions from the user or the cloud.

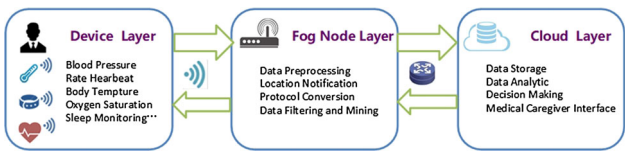

Fig. 2 Proposed fog-driven IoT personal health monitoring system

In this paper, we mainly focus on AKA protocols designed specifically for a fog-driven IoT personal health monitoring system. Our key contributions are listed as follows:

– The proposed efficient three party AKA protocol, based on bilinear pairings. By execution the protocol, the IoT device and fog node are authenticated by the cloud server, and a common session key shared by the three entities is generated. The protocol execution only requires one round communication to achieve mutual authentication and key agreement.
– The proposed protocol is then proved secure formally in the BRP security model. Informal security analysis also demonstrates that the protocol preserves user anonymity and un-traceability and is immune to various attacks.
– The protocol's performance is then evaluated, which demonstrates that our protocol outperforms Hamid et al.'s protocol [11] in terms of computation and communication costs.

In the next section, we will describe extant literature on IoT healthcare and three-party AKA protocols. Sections 3 and 4 introduce the mathematical assumptions used in the proposed protocol, and the security model and security definition for the protocol. The proposed AKA protocol is then presented in Sect. 5, whose security analysis is presented in Sect. 6. The performance evaluation of the proposed protocol is then presented in Sect. 7. Section 8 concludes this paper.

2 Related literature

Bonomi et al. [5] introduced the concept of fog computing and described its characteristics and potential application in IoT services. Since then, fog computing has been the subject of ongoing research [22]. For example, similar to Huang, Lu and Choo [15], Sookhak et al. [24] presented a fog vehicular computing (FVC) infrastructure for smart transportation. Farahani et al. [8] discussed the application of IoT in healthcare and medicine and presented a fog-based IoT eHealth ecosystem. Similar to this paper, Gia et al. [10] and Rahmani et al. [23] presented fog-based healthcare systems. More recently in 2017, Hu et al. [14]

summarized the architecture, key technologies, applications and open issues of fog computing.

There has also been studies dedicated to the security and privacy of fog computing. For example, Stojmenovic et al. [25], for example, focused on man-in-the-middle attack in fog computing. Authentication and authorization were presented as viable solutions in their follow-up work [26]. Yi et al. [31] described security and privacy issues such as trust and authentication, network security, secure data storage, secure and private data computation, privacy, access control and intrusion detection. In [17], a number of security issues and possible solutions in fog computing were also presented. However, there was no discussion of a specific practical solution for these security challenges.

In the fog computing setting, three entities (data user, fog node and cloud server) need to authenticate each other and negotiate a common session key to protect the confidentiality, integrity and authenticity of the transmitted data. There are a large number of three-party AKA protocols proposed for wireless sensor networks (WSNs), which can be adapted for IoT applications. Turkanović et al. [27] proposed an efficient AKA protocol for the heterogeneous WSN. In their protocol, an IoT user can authenticate with a sensor node without having to communicate with a gateway node. However, Farash et al. [9] found that the scheme does not provide user untraceability and sensor node anonymity, as claimed. It was also found to be vulnerable to stolen smart card attack. More recently in 2018, Amin et al. [2] presented a lightweight AKA protocol for IoT enabled devices in the distributed cloud computing environment. In their protocol, the user, the service provider server and the control server achieve mutual authentication and a common session key is shared between the user and the service provider.

The above mentioned schemes only used symmetric cryptosystem to achieve high efficiency. As pointed out by Wang et al. [28], anonymous authentication cannot be achieved by symmetric cryptosystem alone. Thus, there has also been focus on designing asymmetric-based AKA schemes. Hayajneh et al. [12] proposed a lightweight authentication protocol based on Rabin signature for remote patient monitoring with wireless medical sensor networks and implemented the protocol on different hardware settings. Yeh et al. [30] proposed the first AKA scheme for WSNs based on elliptic curve cryptography (ECC). Since then, many more ECC-based AKA protocols have been designed and presented in the literature [6, 20, 21, 29].

Despite the many AKA protocols designed for IoT applications, few such protocols are suitable for direct deployment in a fog computing environment. Hamid et al. [11] proposed a three party one-round AKA protocol with bilinear pairings to preserve the privacy of medical big data in a fog-based healthcare system. However, the session key produced by their protocol is static and does not provide forward privacy. As their key exchange mechanism is based on the tripartite Diffie-Hellman key exchange algorithm of Joux [16], it is also vulnerable to man-in-the-middle attacks launched by an active adversary.

In this paper, we will present an efficient and secure AKA protocol for fog-driven healthcare IoT system, which does not suffer from the drawbacks in Hamid et al.'s scheme [11].

3 Complexity assumptions

ECC has been widely used in designing public key cryptographic protocols, since it can achieve equivalent security level with relatively short key size compared to non-ECC public key cryptosystems (e.g. Diffie–Hellman-based and RSA-based cryptosystems). Let E be an elliptic curve over a finite field F_p defined by the following equation:

$$y^2 = x^3 + ax + b(\text{mod p}),$$

where $x, y, a, b \in F_p$ and $(4a^3 + 27b^2) \bmod p \neq 0$. E/F_p denotes the set of points on E. $G = E/F_p \cup O$, where O is a "point at infinity". G is an additive cyclic group under the point addition operation. Scalar multiplication is denoted as $kP = P + P + \ldots + P$ (k times), where $k \in Z_n$ and n is the order of group G.

Next, we introduce the bilinear pairings defined on elliptic curve groups, and the related complexity assumption which underpins the security of the proposed protocol.

G_1 is an additive cyclic subgroup of E/F_p with an order q, where q is a large prime number. P is a generator of group G_1, which we call a base point. Let G_2 be a multiplicative cyclic group of the same order q. A bilinear pairing is a map $e : G_1 \times G_1 \rightarrow G_2$, which satisfies the following three properties.

- *Computability* Given two points $P_1, P_2 \in G_1$, there is a polynomial time algorithm to compute $e(P_1, P_2)$.
- *Non-degeneracy* If P is an arbitrary generator of G_1, then $e(P, P) \neq 1$.
- *Bilinearity* Given $P_1, P_2 \in G_1$ and $a, b \in Z_n^*$, there is $e(aP_1, bP_2) = e(P_1, P_2)^{ab}$.

A bilinear map satisfies the above properties is a non-degenerate admissible bilinear map, which can be obtained from the Weil, Tate or Ate pairings over super-singular elliptic curves.

BDH Assumption. Let P be a base point of group G_1. The Bilinear Diffie-Hellman (BDH) assumption says that any given $xP, yP, zP \in G_1$ for some unknown $x, y, z \in Z_q^*$, it is computationally hard to compute $e(P, P)^{xyz} \in G_2$.

DBDH Assumption. Let P be a base point of group G_1. The Decisional Bilinear Diffie-Hellman assumption says that any given $xP, yP, zP \in G_1$ for some unknown $x, y, z \in Z_q^*$ and $h \in G_2$, it is computationally hard to decide if $h = e(P,P)^{xyz}$ holds.

4 Security model

We slightly modify the three-party BRP model [3] to define the security of the proposed protocol.

Let P be the protocol, and U is a participant of the protocol. In our protocol, the participants include the client C, the fog nodes FN and the cloud service provider S. Π_U^i denotes oracle machine of the i-th instance of participant U. The adversary $\mathcal{A}$ and the protocol participants execute the protocol interactively through various oracle queries. Each type of oracle query models the adversary's ability to attack the protocol. The oracle queries are described as follows.

- $Send(U^i, M)$: This query models an active attack against the instance U^i. $\mathcal{A}$ sends message M to the oracle Π_U^i. Π_U^i responses with the message that would be output by the instance U^i in a true execution of the protocol. $\mathcal{A}$ may start the protocol by issuing a $Send(U^i, START)$ query.
- $Reveal(U^i)$: This query models the leakage of the session key. If Π_U^i has already produced a session key SK_U^i, then it returns SK_U^i to $\mathcal{A}$; otherwise, it outputs a $\perp$.
- $Corrupt(U)$: This query simulates the perfect forward privacy. $\mathcal{C}$ returns the U's long-term key. In our protocol, it could be the smart card or the password of the client.
- $Execute(A^i, B^j, S^k)$: This query models the passive eavesdropping on the public channel. The protocol is executed among the client instances Π_A^i, Π_B^j and the server instance Π_S^k. The oracle machine returns all messages exchanged in the execution of the protocol.
- $Test(U^i)$: This query is used to define the semantic security of the session key. The adversary is allowed to issue this query only once. The *Test* oracle flips a coin and obtains a bit $b \in \{0,1\}$. If $b = 1$, then the adversary is returned a true session key SK_U^i. If $b = 0$, the adversary is returned a random value of the same size.

Definition 1 (*Partnership*) Two instances Π_A^i and Π_B^j are called *partners* if, and only if, the following conditions are satisfied:

1. Π_A^i and Π_B^j exchange messages directly;
2. Π_A^i and Π_B^j accept the common session key SK; and
3. There is no other instance who would accept SK, except for Π_A^i and Π_B^j.

Definition 2 (*Freshness*) An instance Π_U^i is *fresh* if the following conditions hold:

- Π_U^i has accepted the session key SK;
- *Reveal* oracle has not been queried before Π_U^i being accepted; and
- *Corrupt* oracle has never been queried on Π_U^i and its partners (see Definition 1).

Suppose Π_A^i and Π_B^j are partners and have established a common session key SK. SK is said to be *fresh* if, and only if, both Π_A^i and Π_B^j are fresh.

Definition 3 (*AKA-security*) For any adversary $\mathcal{A}$, $Succ(\mathcal{A})$ denotes the event that $\mathcal{A}$ issues a $Test(U^i)$ query some accepted fresh instances, and outputs the correct b. The advantage of $\mathcal{A}$ in attacking the semantic security of protocol P is defined as

$$Adv_P^{AKA}(\mathcal{A}) = |2Pr[Succ(\mathcal{A})] - 1|.$$

A protocol P is said to be AKA-secure if for any probabilistic polynomial time adversary $\mathcal{A}$, $Adv_P^{AKA}(\mathcal{A})$ is negligible.

5 Proposed scheme

There are three entities in our protocol, namely: the user, the fog note and the cloud service provider (CSP). The user controls one or more IoT devices connected to the fog node and is able to gain access to the system using a password and a smart card.

In our authentication model, fog nodes are regarded as entities that are not trustworthy. Thus, these nodes will not maintain/store any private authentication information of users. They are not responsible for authenticating users. They only transfer public authentication messages between the user and the cloud server, and they provide a share of the session key.

We establish a verifier-based mechanism to authenticate the user and fog nodes. A verifier-based AKA scheme is a variant of a password-based AKA scheme, in which the server only preserves a verifier instead of an image of the password. There have been a number of verifier-based three-party AKA protocols in the literature [13, 18, 19].

Each fog node or end device has a unique identity and must register with the cloud server. The cloud server will keep a verifier of that node or device for future

authentication. The verifier is derived from the identity and the CSP's master key. The protocol generates a common session key shared by an end device, the fog node and the cloud server. The fog node does not preserve the private information (password, verifier) of any user. If a fog node is compromised, then user's data on other nodes will not be affected even if they use the same password and smart card.

Now, we present the details of the proposed AKA scheme.

5.1 System setup

The CSP chooses a non-singular elliptic curve over finite field F_p, where p is a large prime number, and $l = log_2 p$ is the security parameter. Let G be a cyclic group of order n generated by a base point P. *CSP* then chooses a random $s \in Z_n^*$, and calculates $P_{pub} = sP$. (G, P, P_{pub}) are published as the public system parameters, while s is kept secret. *CSP* selects 6 secure hash functions $\{h_0, h_1, h_2, h_3, h_4, h_5\}$, where $h_0 : G_1 \rightarrow \{0.1\}^*$, $h_1 : \{0,1\}^* \times \{0,1\}^* \rightarrow Z_p^*$, $h_2 : \{0,1\}^* \times Z_n^* \times Z_p^* \rightarrow Z_p^*$, $h_3 : G_1 \times Z_p^* \times G_1 \times \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^* \rightarrow Z_p^*$, $h_4 : G_1 \times G_1 \times G_1 \times G_1 \times \{0,1\}^* \times \{0,1\}^* \rightarrow Z_p^*$, $h_5 : G_2 \times G_1 \times G_1 \times G_1 \rightarrow Z_p^*$. *CSP* also maintains a database to record the registered users and fog nodes. We assume that *CSP* is trustworthy.

5.2 User registration phase

In this phase, each user sends a registration request to the trusted cloud, and receives a smart card.

1. U_i randomly chooses $r_i \in Z_p^*$, inputs the password PW_i and the identity ID_i and computes $RID_i = h_1(ID_i || PW_i) \oplus r_i$. U_i sends (ID_i, RID_i) to *CSP* securely.
2. After receiving U_i's request, *CSP* randomly chooses $x_i \in Z_p^*$ and computes $R_i = h_2(ID_i || s || x_i) \oplus RID_i$. *CSP* stores R_i on a smart card, and sends it to U_i via a secure channel. *CSP* also records (ID_i, x_i) in its own database.
3. After receiving the smart card, U_i computes $R_i^* = R_i \oplus r_i$ and replaces R_i on the card with R_i^*.

Here, we emphasize that the registration process must be carried out through a secure channel and the user must be authenticated by the *CSP*. The process is depicted in Fig. 3.

5.3 Fog node registration phase

Each fog node FN_j must register with the *CSP* before it is deployed. FN_j sends its identity ID_j to the *CSP*. *CSP* randomly chooses $y_j \in Z_p^*$ and computes $R_j = h_2(ID_j || s || y_j)$. *CSP* sends R_j to the fog node in a secure way and stores

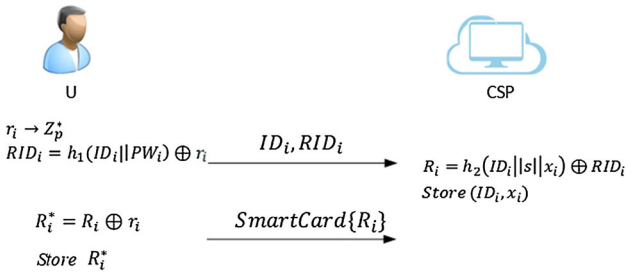

Fig. 3 User Registration Phase

(ID_j, y_j) in its own database. The process is depicted in Fig. 4.

5.4 Authentication and key agreement phase

In this phase, the user, the fog node and the cloud server authenticate each other and generate a common session key. The process is depicted in Fig. 5.

1. U_i randomly chooses $a \in Z_n^*$, and computes $A = aP$, $\bar{A} = aP_{pub}$, $PID_i = ID_i \oplus h_0(\bar{A})$, $M_i = h_1(ID || PW_i) \oplus R^*$, $N_i = h_3(\bar{A} || M_i || A || ID_i || ID_j || T_u)$, where T_u is the current timestamp. U_i sends $Msg_1 = (A, PID_i, N_i, T_u)$ to FN_j.
2. Upon receiving the authentication request from U_i, FN_j first checks the freshness of the timestamp T_u, then randomly chooses $b \in Z_n^*$, and computes $B = bP$, $\bar{B} = bP_{pub}$, $PID_j = ID_j \oplus h_0(\bar{B})$, $L_j = h_3(\bar{B} || R_j || A || PID_i || ID_j || T_f)$, where T_f is the current timestamp. FN_j forwards $Msg_2 = (A, B, PID_i, PID_j, N_i, L_j, T_u, T_f)$ to the *CSP*.
3. After receiving FN_j's authentication request, *CSP* first checks the validity of two timestamps T_u, T_f, and then proceeds as follows.
   - *CSP* computes $\bar{A}' = sA$, $\bar{B}' = sB$, $ID_i' = PID_i \oplus h_0(\bar{A}')$, $ID_j' = PID_j \oplus h_0(\bar{B}')$.
   - *CSP* searches its database to find the entries (ID_i', x_i), (ID_j', y_j). If the entries are not found, then *CSP* rejects the request and aborts the session.

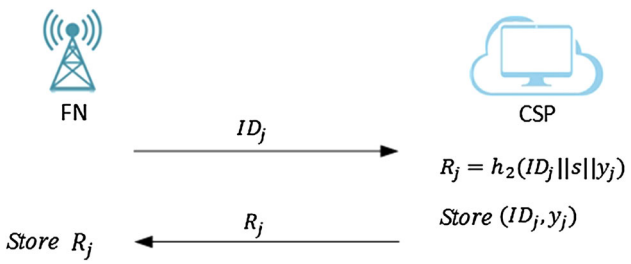

Fig. 4 Fog Node Registration Phase

$a \leftarrow Z_n^*, T_u$
$A = aP, \bar{A} = aP_{pub}$
$PID_i = ID_i \oplus h_0(\bar{A})$
$M_i = h_1(ID_i||PW_i) \oplus R^*$
$N_i = h_3(\bar{A}||M_i||A||ID_i||ID_j||T_u)$

A, PID_i, N_i, T_u ----------►

$b \leftarrow Z_n^*, T_f$
$B = bP, \bar{B} = bP_{pub}$
$PID_j = ID_j \oplus h_0(\bar{B})$
$L_j = h_3(\bar{B}||R_j||A||PID_j||ID_j||T_f)$

$A, B, PID_i, PID_j, N_i, L_j, T_u, T_f$ ----------►

$\bar{A}' = sA, \bar{B}' = sB$
$ID_i' = PID_i \oplus h_0(\bar{A}')$
$ID_j' = PID_j \oplus h_0(\bar{B}')$
$M_i' = h_2(ID_i'||s||x_i)$
$R_j' = h_2(ID_j'||s||y_i)$
$N_i' = h_3(\bar{A}'||M_i'||A||ID_i'||ID_j'||T_u)$
$L_j' = h_3(\bar{B}'||R_j'||A||PID_i||ID_j'||T_f)$
$N_i = N_i'?\, L_j' = L_j?$
$c \leftarrow Z_n^*, C = cP, T_c$
$Auth_i = h_4(A||B||C||\bar{A}'||ID_i'||T_c)$
$Auth_j = h_4(A||B||C||\bar{B}'||ID_j'||T_c)$
$K_c = e(A,B)^c$
$SK_c = h_5(K_c||A||B||C)$

$C, Auth_i, Auth_j, T_c$ ◄----------

$Auth_j{}' = h_4(A||B||C||\bar{B}||ID_j||T_c)$
$Auth_j' = Auth_j?$
$K_f = e(A,C)^b$
$SK_f = h_5(K_f||A||B||C)$

$B, C, Auth_i, T_c$ ◄----------

$Auth_i{}' = h_4(A||B||C||\bar{A}||ID_i||T_c)$
$Auth_i' = Auth_i?$
$K_u = e(B,C)^a$
$SK_u = h_5(K_u||A||B||C)$

Fig. 5 Authentication and Key Agreement Phase

– *CSP* continues to compute $M_i' = h_2(ID_i'||s||x_i)$, $R_j' = h_2(ID_j'||s||y_j)$, $N_i' = h_3(\bar{A}'||M_i'||A||ID_i'||\ ID_j'||T_u)$, $L_j' = h_3(\bar{B}'||R_j'||A||PID_i||ID_j'||T_f)$.

– *CSP* checks if $N_i = N_i'$ and $L_j = L_j'$. If either one does not hold, then *CSP* rejects the request and aborts.

- *CSP* randomly chooses $c \in Z_n^*$, and computes $C = cP$, $Auth_i = h_4(A||B||C||\bar{A}'||ID_i'||T_c)$, $Auth_j = h_4(A||B||C||\bar{B}'||ID_j'||T_c)$, $K_c = e(A,B)^c$, $SK_c = h_5(K_c||A||B||C)$, where T_c is the current timestamp.
- *CSP* sends $Msg_3 = (C, Auth_i, Auth_j, T_c)$ to FN_j.

4. Upon receiving the response from *CSP*, FN_j checks the freshness of T_c, and verifies if $Auth_j = h_4(A||B||C||\bar{B}||ID_j||T_c)$. If not, then FN_j aborts the session. Otherwise, FN_j computes $K_f = e(A,C)^b$, $SK_f = h_5(K_f||\ A||B||C)$ and forwards $Msg_4 = (B, C, Auth_i, T_c)$ to U_i.
5. Upon receiving the response from the fog node, U_i checks the freshness of T_c and verifies if $Auth_i = h_4(A||B||C||\bar{A}||ID_i||T_c)$. If not, then U_i aborts the session. Otherwise, U_i computes $K_u = e(B,C)^a$, $SK_u = h_5(K_u||A||B||C)$.

We claim that if the protocol proceeds correctly, then U_i, FN_j and *CSP* achieve mutual authentication and share a common session key at the end of the authentication process. The correctness is guaranteed by the equation $e(B,C)^a = e(A,C)^b = e(A,B)^c = e(P,P)^{abc}$. Then, we have $K_u = K_f = K_c$ and therefore, $SK_u = SK_f = SK_c$.

5.5 Password update phase

To prevent password guessing attacks, users are advised to change their password regularly. In the proposed scheme, a user U_i can update his/her password locally, as frequent as required. U_i executes the following steps.

1. U_i inserts the smart card and issues an instruction to modify the password.
2. U_i is required to input the old password PW_u and the new password PW_u^{new}.
3. The smart card computes $(R^*)^{new} = h_1(ID_i||PW_u) \oplus h_1(ID_i||PW_u^{new}) \oplus R^*$, and replaces R^* with $(R^*)^{new}$.

5.6 User revocation and re-registration

In some cases, users need to revoke their accounts from the system. For example, if a user U_i's smart card is lost or stolen, then he/she should revoke the old account in such event. U_i sends a revocation request to the *CSP*. After verifying the U_i's identity (based on the old password or other identity information), *CSP* deletes the entry (ID_i, x_i) from its database and thereafter the login requests issued by the old smart card will be rejected.

U_i can re-register with the *CSP* using the same identity and a new password following the registration procedure described in Sect. 5.2. *CSP* then chooses a new random number x_i^{new} and stores (ID_i, x_i^{new}) in its database.

5.7 Fog node revocation

If a fog node FN_j is damaged or compromised, then *CSP* will revoke the node by deleting the record (ID_j, y_j) from its database. Subsequently, any access to the fog node will be declined, since all authentication requests issued by FN_j cannot successfully pass the verification without the random number y_j.

6 Security analysis

In this section, we present the formal security proof of the proposed scheme, as well as the informal security analysis against possible attacks.

6.1 Formal security proof

We prove that that the proposed scheme is AKA-secure and achieves mutual authentication under the security model described in Sect. 4.

Theorem 1 *Let P be the proposed protocol. If there is a probabilistic polynomial time adversary $\mathcal{A}$ who wins the AKA attack game with advantage $Adv_P^{AKA}(\mathcal{A})$, then there must be a probabilistic polynomial time algorithm that can solve the BDH problem with advantage*

$$Adv_{G_1,G_2}^{DBDH} \geq \frac{1}{q_s} Adv_P^{AKA}(\mathcal{A}) - \frac{\sum_{i=0}^{i=4} q_{h_i}^2 + (q_s + q_e)^2}{2pq_s} + \frac{q_{h_5}}{pq_s} + \frac{2q_{h_0}q_{h_3}}{p^2 q_s}),$$

where q_{h_i} $(i = 0, 1, 2, 3, 4, 5)$, q_s, q_e, q_r denotes the times of the hash, Send, Execution and Reveal oracle queries respectively.

Proof Let $\mathcal{A}$ be an adversary who attacks the protocol and wins the game with an advantage of ϵ. We construct a challenger $\mathcal{C}$ who makes use of $\mathcal{A}$'s ability to solve a DBDH problem instance (P, xP, yP, zP, h) for some unknown $x, y, z \in Z_n^*$ and $h \in G_2$, namely, to decide if $h = e(P,P)^{xyz}$ holds. The hash functions in the protocol are simulated as random oracles. $\mathcal{C}$ maintains a hash list L_h, which is initialized to be empty. $Msg_i (i = 1, 2, 3, 4)$ denotes the corresponding messages transmitted among the entities during the execution of the protocol. $\mathcal{C}$ randomly picks a random $s \in Z_n^*$, and $P_{pub} = sP$ publishes (G, P, P_{pub}) as the public parameters, and keeps s secret. $\mathcal{C}$ also sets identity and password (ID_i, PW_i) and the smart

card information R_i^* for each user, as well as (ID_j, R_j^*) for each fog. □

$\mathcal{C}$ simulates the protocol and answers $\mathcal{A}$'s oracle queries as follows.

- *Send query* $\mathcal{A}$ issues an active attack and sends messages adaptively constructed by itself through different *Send* queries. According to the execution process of the protocol, there are four different *Send* queries available to $\mathcal{A}$.
  - $Send(C^i, (FN, START))$: Upon receiving this query, $\mathcal{C}$ starts a new session and returns the login message generated by the user. Specifically, $\mathcal{C}$ randomly chooses $a \in Z_n^*$ and computes $A = aP$, $\bar{A} = aP_{pub}$. $\mathcal{C}$ then computes $PID_i = ID_u \oplus h_0(\bar{A})$, $M_i = h_1(ID_u || PW_u) \oplus R^*$, $N_i = h_3(\bar{A}||M_i \quad ||A||ID_i||ID_j||T_u)$. $\mathcal{C}$ returns (A, PID_i, N_i, T_u) to $\mathcal{A}$, and the instance Π_C^i was set to an expecting state.
  - $Send(FN^j, Msg_1)$: Upon receiving this query, $\mathcal{C}$ first divides Msg_1 into (A, PID_i, N_i, T_u), then randomly picks $b \in Z_n^*$ and computes $B, \bar{B}$, $PID_j = ID_j, L_j$ as described in Sect. 5.4. $\mathcal{C}$ returns $(A, B, PID_i, PID_j, N_i, L_j)$ to $\mathcal{A}$ and sets the instance Π_{FN}^j to an expecting state.
  - $Send(S^k, Msg_2)$: Upon receiving this query, $\mathcal{C}$ divides Msg_2 into $(A, B, PID_i, PID_j, N_i, L_j)$, and computes (N_i', L_j') as described in Sect. 5.4. $\mathcal{A}$ verifies if $N_i = N_i'$ and $L_j = L_j'$, if not, $\mathcal{C}$ rejects $\mathcal{A}$'s query and returns nothing. Otherwise, $\mathcal{C}$ picks a random $c \in Z_q^*$, and computes $(C, Auth_i, Auth_j, K_c, SK_c)$ as in Sect. 5.4. $\mathcal{C}$ returns $(C, Auth_i, Auth_j)$ to $\mathcal{A}$, and the CSP instance terminates.
  - $Send(FN^j, Msg_3)$: Upon receiving this query, assuming that Π_{FN}^j is in the expecting state, $\mathcal{C}$ first divides Msg_3 into $(C, Auth_i, Auth_j)$, and verifies if $Auth_j = h_4(A||B||C||\bar{B}||ID_j)$ holds. If not, then $\mathcal{C}$ rejects $\mathcal{A}$'s query and returns nothing. Otherwise, $\mathcal{C}$ returns $(B, C, Auth_i)$ to $\mathcal{A}$ and the fog instance terminates.
  - $Send(C^i, Msg_4)$: Upon receiving this query, assuming that Π_C^i is in the expecting state, $\mathcal{C}$ first divides Msg_4 into $(B, C, Auth_i)$ and verifies if $Auth_i = h_4(A||B||C||\bar{A}||ID_i)$. If not, then $\mathcal{C}$ rejects $\mathcal{A}$'s query and returns nothing. Otherwise, $\mathcal{C}$ computes $K_u = e(B, C)^a$, $SK = h_5(K_u||A||B||C)$, and the client instance accepts and terminates. $(Msg_1, Msg_2, Msg_3, Msg_4)$ is added to the list Γ.
- *Corrupt query*. The adversary issues this query to obtain one of the user's long term key.
  - On receiving $Corrupt(C, PW)$, returns the client C's password PW_u.
  - On receiving $Corrupt(C, SC)$, returns the information R_u^* stored on the smart card.
- $Execute(U^i, FN^j, S^k)$. On receiving this query, $\mathcal{C}$ simulates the execution process of the protocol by issuing the following *Send* queries.

  $Msg_1 \leftarrow Send(C^i, (FN, START))$;

  $Msg_2 \leftarrow Send(FN^j, Msg_1)$;

  $Msg_3 \leftarrow Send(S^k, Msg_2)$;

  $Msg_4 \leftarrow Send(FN^j, Msg_3)$;

  $\mathcal{C}$ returns $(Msg_1, Msg_2, Msg_3, Msg_4)$ to $\mathcal{A}$.
- $Reveal(\Pi_P^i)$. On receiving this query, responses with the session key SK if the session instance Π_P^i is accepted, else returns $\perp$.
- *Test query*. On receiving a query $Test(U^i)$, $\mathcal{C}$ randomly picks $\tau \in \{0, 1\}$. If $\tau = 1$, then returns the true session key SK; otherwise, returns a random value with the same size.

The proof consists of a sequence of games: $G_0, G_1, \ldots G_5$. Let S_i be the event that $\mathcal{A}$ outputs the correct τ in game G_i $(i = 1, 2, 3, 4, 5)$.

Game G_0. G_0 is the original attacking game. In this game, $\mathcal{C}$ simulates the oracle queries as a real player would do, as listed above. Hash functions are modeled as random oracles. Thus, the probability of success in this game is equal to the probability that $\mathcal{A}$ succeeds in attacking the real protocol. By definition, we have

$$\epsilon = |2Pr[S_0] - 1|. \tag{1}$$

Game G_1. G_1 is as same as G_0, except that $\mathcal{C}$ maintains hash value lists $L_0 - L_5$. When a hash oracle is queried, $\mathcal{C}$ first searches the corresponding list and if there is an entry already, then returns the same, else $\mathcal{C}$ returns a random chosen value and adds the result to the corresponding list L_i. From the properties of random oracles, it is easy to see that G_1 is indistinguishable from G_0, so

$$Pr[S_1] = Pr[S_0]. \tag{2}$$

Game G_2. This game simulates all kinds of queries just like in game G_1, except that the simulation will be terminated if the following two events happen:

- Event E_1: Collisions on the output of hash queries.
- Event E_2: Collisions on the copy of the messages $(Msg_1, Msg_2, Msg_3, Msg_4)$.

According to the birthday paradox, we have

$$Pr[E_1] \leq \sum_{i=0}^{i=4} q_{h_i}^2/(2p). \quad (3)$$

Since a, b, c are randomly chosen, the probability that E_2 happens is $Pr[E_2] \leq (q_s + q_e)^2/(2p)$, where q_s and q_e are the upper bound of *Send* and *Execute* queries, respectively. Thus,

$$|Pr[S_2] - Pr[S_1]| \leq (\sum_{i=0}^{i=4} q_{h_i}^2 + (q_s + q_e)^2)/(2p). \quad (4)$$

Game G_3. In this game, we modify the *Send* query. $\mathcal{C}$ randomly picks a matched instance (C^i, FN^j, S^k) and answers $\mathcal{A}$'s *Send* queries as follows.

- When $\mathcal{A}$ issues a $Send(C^i, (FN, START))$ query, $\mathcal{C}$ sets $A = xP$, $\bar{A} = sA$, and generates PID_i, M_i, N_i, T_u as it does in G_2. $\mathcal{C}$ returns $Msg_1 = (A, PID_i, N_i, T_u)$ to $\mathcal{A}$.
- When $\mathcal{A}$ issues a $Send(FN^j, Msg_1)$ query, $\mathcal{C}$ sets $B = yP$, $\bar{B} = sB$, and generates PID_j, L_j, T_f as it does in G_2. $\mathcal{C}$ returns $Msg_2 = (A, B, PID_i, PID_j, N_i, L_j)$ to $\mathcal{A}$.
- When $\mathcal{A}$ issues a $Send(S^k, Msg_2)$ query, $\mathcal{C}$ sets $c = zP$, $\bar{C} = sC$, and generates $Auth_i, Auth_j$ as it does in G_2. $\mathcal{C}$ sets $K_c = h$ and calculates $SK_c = h_5(K_c||A||\ B||C)$. $\mathcal{C}$ returns $Msg_3 = (C, Auth_i, Auth_j)$ to $\mathcal{A}$.
- When $\mathcal{A}$ issues a $Send(FN^j, Msg_3)$ query, $\mathcal{C}$ sets $K_f = h$, $SK_f = h_5(K_f||A||B||C) and returns\ Msg_4 = (B, C, Auth_i)$ to $\mathcal{A}$.
- When $\mathcal{A}$ issues a $Send(C^i, Msg_4)$ query, $\mathcal{C}$ sets $K_u = h$, $SK_u = h_5(K_u||A||B||C)$ and terminates the instance.

We demonstrates that if the *DBDH* assumption holds, then the difference between game G_2 and G_3 is negligible, just as the following equation shows.

$$|Pr[S_3] - Pr[S_2]| \leq q_s Adv_{G_1,G_2}^{DBDH}. \quad (5)$$

Suppose there is a differentiator that can successfully distinguish game G_2 and G_3, $\mathcal{C}$ can make use of this differentiator to solve the *DBDH* problem. The differentiator selects an instance with probability of $1/q_s$. From the above description, we can see that $\mathcal{C}$ simulates all the queries without knowing x, y, z. If $h = e(P,P)^{xyz}$ holds, then the differentiator actually interacts with game G_2. Otherwise, the differentiator interacts with G_3. If the differentiator decides it is interacting with G_2, then $\mathcal{C}$ outputs 1, or else outputs 0. So we have $Adv_{G_1,G_2}^{DBDH} \geq (1/q_s)|Pr[S_3] - Pr[S_2]|$, and Equation (5) holds.

In game G_3, $K_u = K_f = K_c = h$ is a random value independent of the password and the number x, y, z. The adversary may distinguish a true session key and a random one, if the following events happen.

- Event E_3: $\mathcal{A}$ has queried h_5 oracle on input (h, A, B, C), which has a probability of

$$Pr[E_3] = q_{h_5}/p. \quad (6)$$

- Event E_4: $\mathcal{A}$ successfully impersonates the user and forges a $Msg_1 = (A, PID_i, N_i, T_u)$, which passes the verification executed by the *CSP*. To achieve this, $\mathcal{A}$ has to calculate a valid PID_i and N_i. $\mathcal{A}$ can issue the $Corrupt(C)$ query to achieve one of the long term keys but not both. Since x is an unknown value, $\mathcal{A}$ cannot compute $\bar{A}$, and the probability that $\mathcal{A}$ outputs a valid Msg_1 is:

$$Pr[E_4] \leq \frac{q_{h_0}}{p} \cdot \frac{q_{h_1}}{p} = \frac{q_{h_0} q_{h_3}}{p^2}. \quad (7)$$

- Event E_5: $\mathcal{A}$ successfully impersonates the fog and forges a Msg_2, which passes the verification executed by the *CSP*. Similar to event E_4, the probability that $\mathcal{A}$ outputs a valid Msg_2 is:

$$Pr[E_5] \leq \frac{q_{h_0}}{p} \cdot \frac{q_{h_3}}{p} = \frac{q_{h_0} q_{h_3}}{p^2}. \quad (8)$$

In summary, we have

$$Pr[S_3] \leq \frac{1}{2} + \frac{q_{h_5}}{p} + \frac{2 q_{h_0} q_{h_3}}{p^2}. \quad (9)$$

From Equation (1)-(5) and Equation (9), we have

$$Adv_P^{AKA}(\mathcal{A}) \leq \frac{\sum_{i=0}^{i=4} q_{h_i}^2 + (q_s + q_e)^2}{2p} + \frac{q_{h_5}}{p} + \frac{2 q_{h_0} q_{h_3}}{p^2} + q_s Adv_{G_1,G_2}^{DBDH}.$$

Hence,

$$Adv_{G_1,G_2}^{DBDH} \geq \frac{1}{q_s} Adv_P^{AKA}(\mathcal{A}) - \frac{\sum_{i=0}^{i=4} q_{h_i}^2 + (q_s + q_e)^2}{2 p q_s} - \frac{q_{h_5}}{p q_s} - \frac{2 q_{h_0} q_{h_3}}{p^2 q_s}.$$

Now, we have concluded the proof of Theorem 1. In other words, as long as the BDH problem is hard, it is computationally challenging for any polynomial time adversary to break the AKA-security of the proposed protocol.

6.2 Security analysis

- *User anonymity and un-traceability* In the proposed protocol, an adversary (e.g. an inside user, a fog node, or an external attacker) cannot extract the real identity of the user, even if it intercepts all messages $Msg_i (i = 1, 2, 3, 4)$ transferred over the public channel during the authentication and key agreement process. The user's real identity is hidden by a hash value that can only be calculated by the *CSP*, namely, $PID_i = ID_i \oplus H_0(\bar{A})$,

instead of being transmitted in plaintext over the public channel. What's more, a, b, c are random numbers and the timestamps are dynamic and unique for each session.

- *Perfect forward privacy* Perfect forward privacy requires that even if the user's long-time key (here, long-term key include password and smart card) are compromised, previous session keys established by these keys are still secure. Namely, the session keys are independent of the user's long-term key. In the proposed protocol, even if PW_i and the smart card are compromised at the same time, $\mathcal{A}$ still cannot break the security of the previously generated session keys. This is because $SK = h(K||A||B||C)$ where $K = e(P,P)^{a,b,c}$, $A = aP, B = bP, C = cP$, the generation of each session key needs the random numbers a, b, c chosen by all the three entities, which is independent of the user's long-term key. The perfect forward privacy is assured by the randomness of the session key and the collision resistance of the underpinning hash function.
- *Offline dictionary attack* In the proposed scheme, neither the CSP nor the fog node store any user's password. The only possible way to get some information about a user's password is from the smart card. Suppose an adversary $\mathcal{A}$ has tried to obtain a user's smart card and extract the information R^* on it. We can see that the user U_i's password is protected by a secure hash and masked by the server's private key and a random number, which is unknown to $\mathcal{A}$. Namely, $R^* = h(ID_i||PW_i) \oplus h(ID_i||s||\ x_i)$. Thus, it is infeasible for $\mathcal{A}$ to verify his/her guess. The protocol is secure against offline dictionary attack.
- *Stolen-verifier attack* We can see that in the proposed protocol, the *CSP* only stores the user's identity ID_i and a randomly chosen x_i instead of the real verifier $h(ID_i||s||x_i)$ directly. Suppose there is an adversary $\mathcal{A}$ who has stolen the information stored in the *CSP*, namely, (ID_i, x_i). A still cannot calculate the real verifier $h(ID_i||s||x_i)$ without the secret key s. Thus, it is infeasible for $\mathcal{A}$ to impersonate U_i even if $\mathcal{A}$ obtains the information stored in the *CSP*. Similarly, $\mathcal{A}$ cannot impersonate a fog node FN_j by stealing FN_j's verifier (ID_j, y_j) either.
- *Stolen smart card attack* Suppose there is an adversary $\mathcal{A}$ who has stolen the smart card of the user U_i and extracted the authentication factor R^* stored in the smart card and attempted to produce a legitimate login message Msg_1. Suppose $\mathcal{A}$ also knows the user's identity ID_i, the identity of fog node ID_j. $\mathcal{A}$ can even choose a random number $a' \in Z_n^*$, a fresh timestamp T_a, and compute $\bar{A}' = a'P_{pub}$. However, $\mathcal{A}$ has no idea of PW_i, which is necessary in calculating $M_i = h(ID_i||PW_i) \oplus R_i^*$ and $N_i = h(\bar{A}||M_i||ID_i||I \quad D_j||T_u)$. Thus, $\mathcal{A}$ cannot produce a legitimate login message. The protocol is vulnerable to stolen smart card attack.
- *Known session key attack* In the proposed protocol, a session key $SK = h(K||A||B||C)$, where $K = e(P,P)^{abc}$, is established after mutual authentication. Notice that a, b, c are dynamic numbers randomly chosen by different entities in each session. In other words, session keys in different sessions are independent from each other. Therefore, the leakage of one session key has no effect on the privacy of the others.
- *Man-in-the-middle attack* We show that even if $\mathcal{A}$ intercepts and modifies all the messages transferred over the public channel, it is still infeasible for $\mathcal{A}$ to produce a legitimate message $Msg_i (i = 1, 2, 3, 4)$ in the name of any entity in the protocol. Suppose that $\mathcal{A}$ knows the user's identity ID_i and the fog node's ID_j. To produce a valid Msg_1, $\mathcal{A}$ can choose its own random number $a \in Z_n^*$ and the timestamp T_a, and computes $A, \bar{A}, PID_i$ as the legitimate user does, but it cannot calculate M_i, N_i without R^* and PW_i. Thus, the adversary can have both values as long as he/she does not get the password and the smart card at the same time. In addition, if $\mathcal{A}$ uses R^* and PW_i of its own choice, it will be easily detected by the CSP. Similarly, $\mathcal{A}$ cannot generate a Msg_2 without the registered verifier R_j. Furthermore, without the *CSP*'s private key s, $\mathcal{A}$ cannot forge Msg_3 and Msg_4 either. In summary, the proposed scheme is secure against man-in-the-middle attack.
- *Replay attack* In the proposed protocol, timestamps are involved in each step of the authentication process. As long as the time interval to assure the freshness of the timestamps is sufficiently small and the clock remains synchronized, it is challenging for $\mathcal{A}$ to replay a legitimate message.

7 Performance evaluation

In this section, we evaluate the performance of the proposed protocol, in terms of computation and communication costs.

In the evaluation, we used a super singular elliptic curve E/F_p, and the Ate pairing $e : G_1 \times G_1 \rightarrow G_2$ generated by a point on E/F_p. Both G_1 and G_2 are groups of prime order q, and p and q are large prime numbers with a length of 512 and 160 bits respectively.

The evaluation was realized by MIRACL library on two platforms. One is the elastic compute service (ECS) host provided by the Alibaba Cloud platform, which was used to simulate the cloud server and fog node. The host's

Table 1 Computation time of basic operations (ms)

Operation	Description	ECS host	Google Nexus One
$TG_{\hat{e}}$:	Bilinear pairing	5.275	48.66
TG_m :	Scala multiplication	1.970	19.919
T_h :	Hash function	0.009	0.029
TG_a :	Point addition	0.027	0.176

operating system is Ubuntu 14.04 for 64 bit with an Intel(R) Xeon(R) CPU E5-2630 0 @ 2.30 GHz, and equipped with 1 GB RAM. Another one is a Google Nexus One smart phone with ARM CPU armeabi-v7a 2 GHz, 300 MiB RAM, and Android 4.4.2 operation system. Table 1 lists the computational cost in these two platforms for basic operations used in the protocol.

Table 2 lists the computation costs of three entities in the different phases of the protocol. From the table, we observed that the computation cost in the registration phase is very small and has little impact on the overall performance. In the authentication and key agreement phase, the time cost on the user is $2TG_m + 5T_h + TG_e$ (88.643 ms), time cost on the fog node is $2TG_m + 5T_h + TG_e$ (9.26 ms), and time cost on the cloud server is $3TG_m + 9T_h + TG_e$ (11.266 ms). It can also be observed from the table that the time cost is mainly in the calculation of bilinear pairings. Thus, we could reduce the time costs by designing a protocol without bilinear pairings.

To evaluate the communication costs, we denote the length of a point in group G_1 as $|G_1|$, which is 1024 bit. The output of hash functions h_0, h_1, h_2, h_3, h_4 is in Z_p^*, which has a length of 160 bit, denoted as $|q|$. The output length of h_5 is equal to the session key size k, and we assume it to be 256 bit. We also assume that timestamp has a length of 32 bit, denoted as T. The communication costs in each phase of the protocol are listed in Table 3. From the table, we observed that the communication costs on the user and cloud are the same, namely, $|G_1| + 2|q| + |T|$ (1376 bit). Communication on the fog node side is relatively higher at $3|G_1| + 6|q| + 3|T|$ (4128 bit).

8 Conclusion

Fog computing has applications in a wide range of applications, ranging from civilian (e.g. healthcare settings such as the context in this paper) to military (e.g. battlefields in fog-of-battlefields). Thus, the capability to ensure security and privacy of a fog-driven deployment will be increasingly important.

In this paper, we proposed a three-party AKA protocol with bilinear pairings, and proved its security in the random oracle model. The performance evaluation was also presented, which demonstrated its potential to be deployed in a real-world healthcare organization.

Future work includes exploring ways to improve the efficiency of the scheme, in order to be more suited for other lightweight applications.

Table 2 Computation cost at each entity (ms)

Phase	User	Fog Note	Cloud
User registration	T_h	–	T_h
(ms)	0.009	–	0.009
Fog node registration	–	–	T_h
(ms)	–	–	0.009
Authenticaiton and key agreement	$2TG_m + 5T_h + TG_e$	$2TG_m + 4T_h + TG_e$	$3TG_m + 9T_h + TG_e$
(ms)	88.643	9.251	11.266
Total	$2TG_m + 6T_h + TG_e$	$2TG_m + 4T_h + TG_e$	$3TG_m + 11T_h + TG_e$)
(ms)	88.652	9.251	11.284

Table 3 Communication costs at each entity (bits)

Phase	User	Fog Note	Cloud
User registration	$2\|q\|$	–	$\|q\|$
(bits)	320	–	160
Fog node registration	–	$\|q\|$	$\|q\|$
(bits)	–	160	160
Authenticaiton and key agreement	$\|G_1\| + 2\|q\| + \|T\|$	$3\|G_1\| + 6\|q\| + 3\|T\|$	$\|G_1\| + 2\|q\| + \|T\|$
(bits)	1376	4128	1376

Acknowledgements The work was supported in part by the National Natural Science Foundation of China (Nos. 61501333, 61572379, U1536204) and the National High-Tech Research and Development Program of China (863 Program) (No. 2015AA016004) and in part by the Fundamental Research Funds for the Central Universities under Grant CZY18034.

References

1. Alrawais, A., Alhothaily, A., Hu, C., & Cheng, X. (2017). Fog computing for the internet of things: Security and privacy issues. *IEEE Internet Computing*, *21*(2), 34–42.
2. Amin, R., Kumar, N., Biswas, G., Iqbal, R., & Chang, V. (2018). A light weight authentication protocol for iot-enabled devices in distributed cloud computing environment. *Future Generation Computer Systems*, *78*, 1005–1019.
3. Bellare, M., Pointcheval, D., & Rogaway, P. (2000). Authenticated key exchange secure against dictionary attacks. *Tecnologia Electronica E Informatica*, *1807*, 139–155.
4. Bonomi, F., Milito, R., Natarajan, P., & Zhu, J. (2014). Fog computing: A platform for internet of things and analytics. In *Big data and internet of things: A roadmap for smart environments* (pp. 169–186). Cham: Springer.
5. Bonomi, F., Milito, R., Zhu, J., & Addepalli, S. (2012, August). Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing* (pp. 13–16). ACM.
6. Chaudhry, S. A., Naqvi, H., Mahmood, K., Ahmad, H. F., & Khan, M. K. (2017). An improved remote user authentication scheme using elliptic curve cryptography. *Wireless Personal Communications*, *96*(4), 5355–5373.
7. Choo, K. K. R. (2009). *Secure key establishment, advances in information security* (Vol. 41). Berlin: Springer.
8. Farahani, B., Firouzi, F., Chang, V., Badaroglu, M., Constant, N., & Mankodiya, K. (2018). Towards fog-driven IoT ehealth: Promises and challenges of IoT in medicine and healthcare. *Future Generation Computer Systems, 78*, 659–676.
9. Farash, M. S., Turkanović, M., Kumari, S., & Hölbl, M. (2016). An efficient user authentication and key agreement scheme for heterogeneous wireless sensor network tailored for the internet of things environment. *Ad Hoc Networks*, *36*, 152–176.
10. Gia, T. N., Jiang, M., Rahmani, A. M., Westerlund, T., Liljeberg, P., & Tenhunen, H. (2015, October). Fog computing in healthcare internet of things: A case study on ecg feature extraction. In *IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM)* (pp. 356–363). IEEE.
11. Hamid, H. A. A., Rahman, S. M. M., Hossain, M. S., Almogren, A., & Alamri, A. (2017). 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*, *5*, 22313–22328.
12. Hayajneh, T., Mohd, B. J., Imran, M., Almashaqbeh, G., & Vasilakos, A. V. (2016). Secure authentication for remote patient monitoring with wireless medical sensor networks. *Sensors*, *16*(4), 424.
13. He, D., & Wang, D. (2015). Robust biometrics-based authentication scheme for multiserver environment. *IEEE Systems Journal*, *9*(3), 816–823.
14. Hu, P., Dhelim, S., Ning, H., & Qiu, T. (2017). Survey on fog computing: Architecture, key technologies, applications and open issues. *Journal of Network & Computer Applications*, *98*, 27–42.
15. Huang, C., Lu, R., & Choo, K. K. R. (2017). Vehicular fog computing: Architecture, use case, and security and forensic challenges. *IEEE Communications Magazine*, *55*(11), 105–111.
16. Joux, A. (2004). A one round protocol for tripartite diffie-hellman. *Journal of Cryptology*, *17*(4), 263–276.
17. Khan, S., Parkinson, S., & Qin, Y. (2017). Fog computing security: A review of current applications and security solutions. *Journal of Cloud Computing*, *6*(1), 19.
18. Kwon, J. O., Jeong, I. R., Sakurai, K., & Dong, H. L. (2007). Efficient verifier-based password-authenticated key exchange in the three-party setting. *Computer Standards & Interfaces*, *29*(5), 513–520.
19. Lee, T. F., Liu, J. L., Sung, M. J., Yang, S. B., & Chen, C. M. (2009). Communication-efficient three-party protocols for authentication and key agreement. *Computers & Mathematics with Applications*, *58*(4), 641–648.
20. Li, C. T., Wu, T. Y., Chen, C. L., Lee, C. C., & Chen, C. M. (2017). An efficient user authentication and user anonymity scheme with provably security for IoT-based medical care system. *Sensors*, *17*(7), 1482.
21. Liu, C. H., & Chung, Y. F. (2017). Secure user authentication scheme for wireless healthcare sensor networks. *Computers & Electrical Engineering*, *59*, 250–261.
22. Osanaiye, O. A., Chen, S., Zheng Yan, R. L., Choo, K. K. R., & Dlodlo, M. E. (2017). From cloud to fog computing: A review and a conceptual live vm migration framework. *IEEE Access*, *5*, 8284–8300.
23. Rahmani, A. M., Gia, T. N., Negash, B., Anzanpour, A., Azimi, I., Jiang, M., et al. (2018). Exploiting smart e-health gateways at the edge of healthcare internet-of-things: A fog computing approach. *Future Generation Computer Systems*, *78*, 641–658.
24. Sookhak, M., Yu, R., He, Y., Talebian, H., Safa, N. S., Zhao, N., et al. (2017). Fog vehicular computing: Augmentation of fog computing using vehicular cloud computing. *IEEE Vehicular Technology Magazine*, *PP*(99), 1–1.
25. Stojmenovic, I., & Wen, S. (2014, September). The fog computing paradigm: Scenarios and security issues. In *Computer Science and Information Systems (FedCSIS)*, 2014 Federated Conference on (pp. 1–8). IEEE.
26. Stojmenovic, I., Wen, S., Huang, X., & Luan, H. (2016). An overview of fog computing and its security issues. *Concurrency & Computation Practice & Experience*, *28*(10), 2991–3005.
27. Turkanović, M., Brumen, B., & Hölbl, M. (2014). A novel user authentication and key agreement scheme for heterogeneous ad hoc wireless sensor networks, based on the internet of things notion. *Ad Hoc Networks*, *20*, 96–112.
28. Wang, D., & Wang, P. (2014). On the anonymity of two-factor authentication schemes for wireless sensor networks: Attacks, principle and solutions. *Computer Networks*, *73*, 41–57.
29. Xie, Q., Wong, D. S., Wang, G., Tan, X., Chen, K., & Fang, L. (2017). Provably secure dynamic id-based anonymous two-factor authenticated key exchange protocol with extended security model. *IEEE Transactions on Information Forensics and Security*, *12*(6), 1382–1392.
30. Yeh, H. L., Chen, T. H., Liu, P. C., Kim, T. H., & Wei, H. W. (2011). A secured authentication protocol for wireless sensor networks using elliptic curves cryptography. *Sensors*, *11*(5), 4767–4779.
31. Yi, S., Qin, Z., & Li, Q. (2015). Security and privacy issues of fog computing: A survey. In *International conference on wireless algorithms, systems, and applications* (pp. 685–695). Springer.

Xiaoying Jia received her Ph.D. degree from the Stake Key Laboratory of Information Security, Graduate University of Chinese Academy of Sciences, in 2012. She is currently a lecturer in South-Central University for Nationalities. Her research interests include applied cryptography, cloud computing, and network security.

Debiao He received his Ph.D. degree in applied mathematics from School of Mathematics and Statistics, Wuhan University in 2009. He is currently a Professor of the School of Cyber Science and Engineering, Wuhan University. His main research interests include cryptography and information security, in particular, cryptographic protocols.

Neeraj Kumar received his Ph.D. in CSE from Shri Mata Vaishno Devi University, Katra, India. He is now an Associate Professor in the Department of Computer Science and Engineering, Thapar University, Patiala, Punjab (India). He is a member of IEEE. His research is focused on mobile computing, parallel/distributed computing, multi-agent systems, service oriented computing, routing and security issues in mobile ad hoc, sensor and mesh networks. He has more than 100 technical research papers in leading journals such as-IEEE TII, IEEE TIE, IEEE TDSC, IEEE ITS, IEEE TWPS, IEEE SJ,IEEE ComMag, IEEE WCMag, IEEE NetMag and conferences. His research is supported from DST, TCS and UGC. He has guided many students leading to M.E. and Ph.D.

Kim-Kwang Raymond Choo received the Ph.D. in Information Security in 2006 from Queensland University of Technology, Australia. He currently holds the Cloud Technology Endowed Professorship at The University of Texas at San Antonio (UTSA), and has a courtesy appointment at the University of South Australia. In 2016, he was named the Cybersecurity Educator of the Year—APAC (Cybersecurity Excellence Awards are produced in cooperation with the Information Security Community on LinkedIn), and in 2015 he and his team won the Digital Forensics Research Challenge organized by Germany's University of Erlangen-Nuremberg. He is the recipient of the 2018 UTSA College of Business Col. Jean Piccione and Lt. Col. Philip Piccione Endowed Research Award for Tenured Faculty, ESORICS 2015 Best Paper Award, 2014 Highly Commended Award by the Australia New Zealand Policing Advisory Agency, Fulbright Scholarship in 2009, 2008 Australia Day Achievement Medallion, and British Computer Society's Wilkes Award in 2008. He is also a Fellow of the Australian Computer Society, and an IEEE Senior Member.