Huang et al. J Wireless Com Network (2021) 2021:150
https://doi.org/10.1186/s13638-021-02022-1

EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

# An efficient authentication and key agreement protocol for IoT-enabled devices in distributed cloud computing architecture

Huihui Huang[1], Siqi Lu[1,2], Zehui Wu[1*] and Qiang Wei[1]

*Correspondence: wuzehui2010@foxmail.com
[1] State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450001, Henan, China
Full list of author information is available at the end of the article

## Abstract

With the widespread use of Internet of Things and cloud computing in smart cities, various security and privacy challenges may be encountered.The most basic problem is authentication between each application, such as participating users, IoT devices, distributed servers, authentication centers, etc. In 2020, Kang et al. improved an authentication protocol for IoT-Enabled devices in a distributed cloud computing environment and its main purpose was in order to prevent counterfeiting attacks in Amin et al.' protocol, which was published in 2018. However, We found that the Kang et al.'s protocol still has a fatal vulnerability, that is, it is attacked by offline password guessing, and malicious users can easily obtain the master key of the control server. In this article, we extend their work to design a lightweight pseudonym identity based authentication and key agreement protocol using smart card. For illustrating the security of our protocol, we used the security protocol analysis tools of AVISPA and Scyther to prove that the protocol can defend against various existing attacks. We will further analyze the interaction between participants authentication path to ensure security protection from simulated attacks detailedly. In addition, based on the comparison of security functions and computing performance, our protocol is superior to the other two related protocols. As a result, the enhanced protocol will be efficient and secure in distributed cloud computing architecture for smart city.

**Keywords:** Authentication, AVISPA tool, Scyther tool, Security attack, Distributed cloud architecture

## 1 Introduction

In recent years, Internet of things (IoT) devices, such as sensor devices, RFID tags, actuators and smart objects, are increasingly being used in daily life to provide people with a convenient life. The main functions of IoT-enabled devices are interconnected and interlinked in a heterogeneous wireless environment, in which the devices can continuously monitor and analyze sensor data from multifarious applications to achieve real-time automation of smart decision-making processes in smart cities. However, as we all know, IoT devices are resource-constrained and data-intensive. Thus, there should be a standard platform that can handle efficiently large amount of heterogeneity data and devices, as the data and devices are growing exponentially [1]. To process such a large database repository generated from various IoT devices, Cloud Computing has emerged as a key technology [2–4]. In current days, there are several types of cloud services provided by the cloud provider such as Software as a Service (SaaS) cloud (Ex. IBM LotusLive), Platform as a Service (PaaS) (Ex. Google AppEngine) and Infrastructure as a Service (IaaS) (Ex. Amazon Web Services) [5]. However, there is a basic problem that how the private distributed cloud server authenticates the connected IoT devices. For example, the private information from IoT devices is stored in distributed private cloud server, so that only legitimate users are allowed to access the sensitive information. Recently, many authentication protocols integrated with IoT and distributed cloud computing have been proposed for secure access control on large-scale IoT networks [5–13]. In Amin et al. [5] proposed an authentication protocol for IoT-enabled devices in distributed cloud computing environment, which showed many security vulnerabilities of two authentication protocols proposed by Xue et al. [8] and Chuang and Cheng [9]. However, Kang et al. [10] found that Amin et al.'s [5] protocol is vulnerable to counterfeit attacks and improved the protocol. Unfortunately, by studying a large number of authentication protocols [14], we further discover an off-line password guessing attack on Kang et al.'s protocol, that is, a malicious user can easily get the secret number of the master control server. This is a fatal vulnerability to the entire system. Thus, we extend upon their work by designing a lightweight dynamic pseudonym identity based authentication and key agreement protocol using a smartcard, which is proven to be efficient and secure.

The rest of paper is organized as follows. The methods and experimental of our article are briefly introduced in Sect. 2. In Sect. 3, we review the Kang et al.'s protocol and point out the security weaknesses in detail. The enhanced protocol is proposed in Sect. 4. Results and Discussion are given in Sect. 5. Finally, the article is concluded in Sect. 6.

## 2 Methods and experimental

In this paper, we give a scenario: Assumed a cloud computing service provider has built a distributed private cloud environment covering the entire smart city. There are many IoT devices that should be interconnected to each other via the nearest private cloud service which records confidential information. Then, the distributed cloud service can realize high-speed computing and real-time communication with each IoT-enabled device to provide high-quality services [15, 16]. This scenario involves three main entities: the cloud computing provider, which is regarded as the server control *CS*, a single distributed private cloud server namely $S_m$ and each IoT-enabled device, which belong to the user $U_i$ in smart city. We briefly describe this scenario as shown in Fig. 1. Since the protocol is designed for IoT devices, which have tight computing resources and data-intensive, the protocol only uses hash functions and X-or operations.

In the experimental section, we used the security protocol analysis tools of AVISPA and Scyther to simulation of our proposed protocol for illustrating the security of the protocol. And We personally build the AVISPA (Version of 2006/02/13) and Scyther(v1.1.3) in a virtual machine of an ubuntu operating system. Then, in the security analysis, we mainly use cryptography knowledge to analyze in detail the authentication paths among $U_i$, $S_m$, and *CS* in our proposed, so as to protect against the most common attacks of impersonation attack. Finally, security functionality and computational performance are concretely compared among our protocol with the other two protocols.

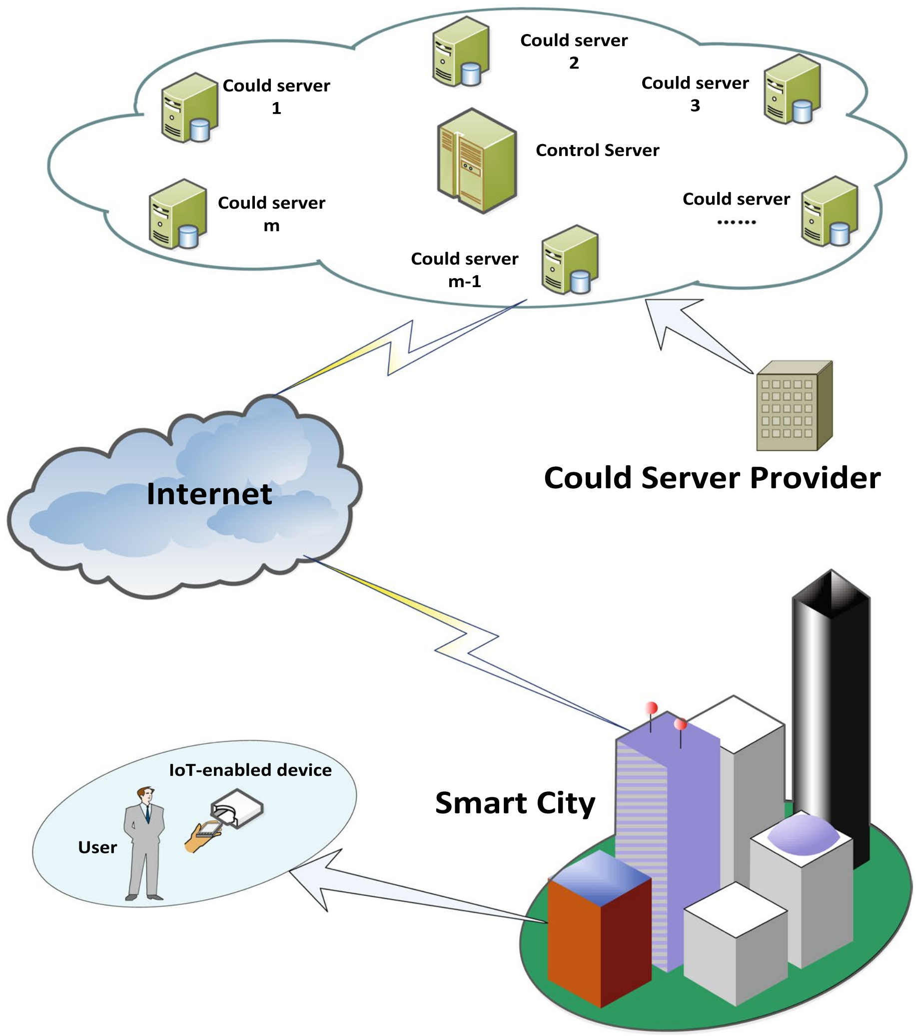



**Fig. 1** IoT-enabled distributed cloud architecture in smart city. The real scenario of the IoT-enabled distributed cloud architecture in smart city, which involves three main entities: the cloud computing provider, which is regarded as the server control *CS*, a single distributed private cloud server namely $S_m$ and each IoT-enabled device, which belong to the user $U_i$

## 3 Kang et al.'s protocol and its weaknesses

In this section, we give the overview of Kang et al.'s [10] protocol and some security drawbacks of their protocol are described carefully. In Kang et al.'s protocol, there are 3 participants: an ordinary user $U_i$, $m$th cloud providing servers $S_m$, and the control server ($CS$). The server $CS$ is a trusted third party responsible for registration and authentication of users and cloud servers. The notations used in this article are shown in Table 1.

### 3.1 Kang et al.'s protocol

In this section, we introduce the registration, login, and authentication key agreement phases of Amin et al.'s [5] protocol, as their protocol only includes three parts. To facilitate analysis, the full implementation of Kang et al.'s protocol is shown in Fig. 2.

**Table 1** Notations used in this paper

| Symbol | Description |
|---|---|
| $CS$ | The control server |
| $S_m$ | mth cloud server |
| $SID_m$ | Identity of $S_m$ |
| $d$ | Random number of $S_m$ |
| $U_i$ | ith user |
| $ID_i$ | Identity of $U_i$ |
| $B_i$ | Biometric of $U_i$ |
| $P_i$ | Password of $U_i$ |
| $b_i$ | Random number of $U_i$ |
| $CR$ | The card reader |
| $x$ | Secret key only known to $CS$ for authenticate all $U_i$ |
| $y$ | Secret key only known to $CS$ for authenticate all $S_m$ |
| $h(\bullet)$ | Hash function $(0,1)^l \rightarrow (0,1)^n$ |
| $T$ | Timestamp |
| $\Delta T$ | Estimated time delay |
| $\oplus$ | Bit-wise xor operation |
| $\|$ | Concatenate operation |

#### 3.1.1 Registration phase

During server registration, the cloud server $S_m$ sends the message $\langle BS_m, d \rangle$ to $CS$. After receiving it, $CS$ computes $PSID_m = h(SID_m \parallel d)$, $BS_m = h(PSID_m \parallel SID_m \parallel d)$ and sends $BS_m$ to $S_m$ via a secure channel. Finally, $S_m$ stores secret parameter $\langle BS_m, d \rangle$ into the memory.

In the phase of user registration, the user $U_i$ computes $A_i = P_i \oplus h(B_i)$, where $B_i$ is the biometric of $U_i$, and sends $\langle ID_i, A_i \rangle$ to the $CS$ securely. On getting it, $CS$ chooses a random number $b_i$ and calculates the following operations: $PID_i = h(ID_i \parallel b_i)$, $C_i = h(ID_i \parallel A_i)$, $D_i = h(PID_i \parallel x)$, $E_i = D_i \oplus A_i$ and $\Delta_i = h(PID_i \parallel ID_i \parallel x)$. Finally, $CS$ delivers a smart card recording the information $\langle C_i, \Omega_i, \Delta_i, E_i, h(\cdot) \rangle$ to $U_i$ in a secure channel.

#### 3.1.2 Login phase

When wanting to access the information of the cloud server $S_m$, $U_i$ provides $ID_i^*$, $P_i^*$ and $B_i^*$ to a card reader ($CR$). Then, $CR$ calculates $A_i^* = P_i^* \oplus h(B_i^*)$, $C_i^* = h(ID_i^* \parallel A_i^*)$ and checks whether $C_i^*$ is equal to $C_i$ . If $C_i^* = C_i$ , $CR$ produces a random number $N_i$ and current timestamp $TS_i$ to compute the following operations: $b_i = \Omega_i \oplus A_i$, $PID_i = h(ID_i \parallel b_i)$, $D_i = E_i \oplus A_i$, $O_i = ID_i \oplus D$, $G_i = h(ID_i \parallel SID_m \parallel N_i \parallel TS_i \parallel D_i)$, $F_i = \Delta_i \oplus N_i$ and $Z_i = SID_m \oplus h(D_i \parallel N_i)$. After that, $CR$ submits the login message $\langle G_i, F_i, Z_i, O_i, PID_i, TS_i \rangle$ to the cloud server $S_m$ over an public channel.

#### 3.1.3 Authentication key agreement phase

This phase describes mutual authentication and key agreement among the participants, which can be divided into four steps as follows.

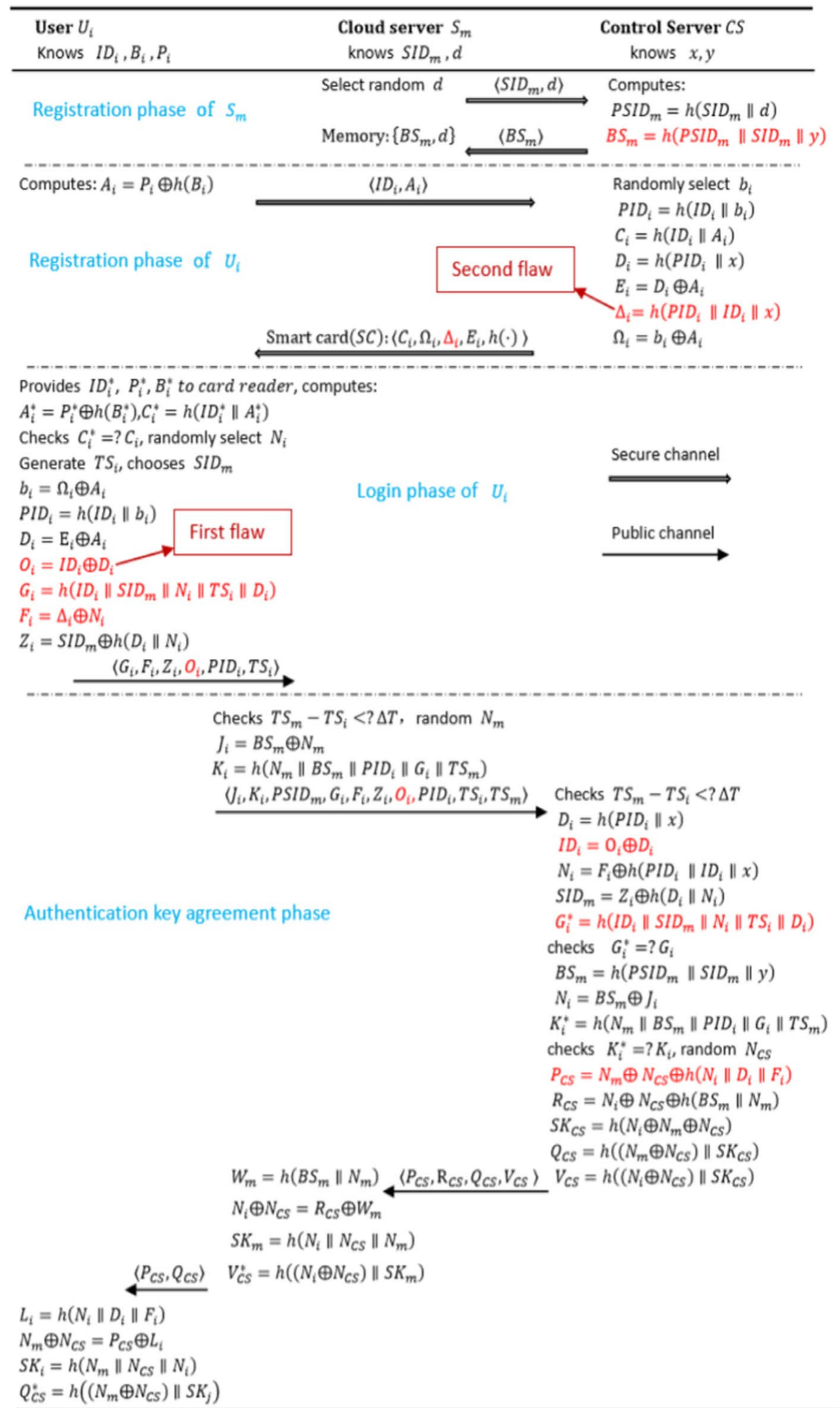



**Fig. 2** Implementation of Kang et al.'s protocol. Implementation of the registration, login, and authentication key agreement phases in Kang et al.'s protocol

Step 1: When receiving the login message from $U_i$, $S_m$ first checks the time interval condition $TS_m - TS_i < \Delta T$, where $TS_m$ is $S_m$'s current timestamp and $\Delta T$ is expected time interval during message transmission. If $TS_m - TS_i \geq \Delta T$, $S_m$ terminates the connection; otherwise, $S_m$ takes a random number $N_m$ to calculate

$$J_i = B_m \oplus N_m$$
$$K_i = h(N_m \parallel BS_m \parallel PID_i \parallel G_i \parallel TS_m)$$

Next, $S_m$ sends $\langle J_i, K_i, PSID_m, G_i, F_i, Z_i, O_i, PID_i, TS_i, TS_m \rangle$ to the control server $CS$ via an public channel.

Step 2: After getting the message, $CS$ checks time interval condition $TS_{CS} - TS_m < \Delta T$, where $TS_{CS}$ is $CS$'s current timestamp. If $TS_{CS} - TS_m < \Delta T$ , $CS$ computes

$$D_i = h(PID_i \parallel x)$$
$$ID_i = O_i \oplus D_i$$
$$N_i = F_i \oplus h(PID_i \parallel ID_i \parallel x)$$
$$SID_m = Z_i \oplus h(D_i \parallel N_i)$$
$$G_i^* = h(ID_i \parallel SID_m \parallel N_i \parallel TS_i \parallel D_i)$$

Then, $CS$ checks $G_i^*$ is equal to $G_i$ or not. If $G_i^* = G_i$, $CS$ thinks that the user $U_i$ is legal; otherwise, it terminates the session. After that, $CS$ calculates

$$BS_m = h(PSID_m \parallel SID_m \parallel y)$$
$$N_i = BS_m \oplus J_i$$
$$K_i^* = h(N_m \parallel BS_m \parallel PID_i \parallel G_i \parallel TS_m)$$

for authenticating the cloud server $S_m$. If $K_i^* \neq K_i$, $CS$ thinks the cloud server $S_m$ is illegal and terminates the session; otherwise, $CS$ randomly selects a number $N_{CS}$ and computes

$$P_{CS} = N_m \oplus N_{CS} \oplus h(N_i \parallel D_i \parallel F_i)$$
$$R_{CS} = N_i \oplus N_{CS} \oplus h(BS_m \parallel N_m)$$
$$K_{CS} = h(N_i \parallel N_m \parallel N_{CS})$$
$$Q_{CS} = h((N_m \oplus N_{CS}) \parallel SK_{CS})$$
$$V_{CS} = h((N_i \oplus N_{CS}) \parallel SK_{CS})$$

where $K_{CS}$ is the secret session key between $U_i$ and $S_m$. Finally, $CS$ sends $\langle P_{CS}, Q_{CS}, \mathrm{R_{CS}}, V_{CS} \rangle$ to $S_m$ through public communication.

Step 3: When obtaining the message from $CS$, $S_m$ calculates

$$W_m = h(BS_m \parallel N_m)$$
$$N_i \oplus N_{CS} = R_{CS} \oplus W_m$$
$$SK_m = h(N_i \parallel N_{CS} \parallel N_m)$$
$$V_{CS}^* = h((N_i \oplus N_{CS}) \parallel SK_m).$$

Next, $S_m$ checks whether $V_{CS}^*$ is equal to $V_{CS}$. If $V_{CS}^* = V_{CS}$, $S_m$ sends $\langle P_{CS}, Q_{CS} \rangle$ to the user $U_i$.

Step 4: On receiving the reply message from $S_m$, $U_i$ computes

$$
\begin{aligned}
&L_i = h(N_i \parallel D_i \parallel F_i)\\
&N_m \oplus N_{CS} = P_{CS} \oplus L_i\\
&SK_i = h(N_m \parallel N_{CS} \parallel N_i)\\
&Q^*_{CS} = h\big((N_m \oplus N_{CS}) \parallel SK_j\big).
\end{aligned}
$$

Then, the $U_i$ checks the condition whether $Q^*_{CS}$ is equal $Q_{CS}$ or not. If the condition is true, $U_i$ confirms $CS$ and $S_m$ are authentic.

### 3.2 Cryptanalysis of Kang et al.'s protocol

In this section, we make cryptanalysis of the protocol proposed by Kang et al. [10] in details. For analysis, there are some valid assumptions that can be found in [17–20].

#### *3.2.1 Off-line password guessing attack*

The authors in [10] stated that their protocol is protected against off-line password guessing attacks. However, we discover that a malicious attacker can obtain the master secret key of $CS$ after launching the above attack. The details are described as below:

Step 1: An attacker namely Eve first registers in the control server $CS$ with identity $ID_{Eve}$ like a normal user. Next, he logins in and sends the message $\langle G_{Eve}, F_{Eve}, Z_{Eve}, O_{Eve}, PID_{Eve}, TS_{Eve}\rangle$ to $S_m$. Because the message is transmitted publicly, he can easily obtain the values $O_{Eve}$ and $PID_{Eve}$. For example, using the wireshark tool to capture the packets locally.

Step 2: According to the description in the login phase, Eve computes $D_{Eve} = O_{Eve} \oplus ID_{Eve}$ , where has been shown the "First flaw" in the Fig. 2.

Step 3: Since $D_i = h(PID_i \parallel x)$, the off-line password guessing attack can be implemented by Algorithm 1.

**Algorithm 1:** Off-line Password Guessing Attack

**Input:** input parameters $D_{Eve}, PID_{Eve}, h(\bullet)$.

**Output:** output $x$ , which is the secret key only known to $CS$.

1 Eve generates a random number and takes it as key $x_{tmp}$;

2 Eve Computes $D_{tmp} = h\,(PID_{Eve} \parallel x_{tmp})$;

3 **if** $D_{tmp} == D_{Eve}$ **then**

4     Return $(x_tmp\ )$;

5 **else**

6     Go to 1 until correct key is obtained;

7 **end**

Although the algorithm may take a long time to execute, Eve will be willing to keep trying because the control server $CS$ uses the key $x$ to authenticate all the user $U_i$, which is crucial parameter to the whole system. Thus, the protocol proposed by Kang et al. is vulnerable to the above attack.

#### 3.2.2 Design redundant in the user registration phase

In order to avoid the impersonation attack in Amin et al.'s [5] protocol, the authors compute $BS_m = h(PSID_m \parallel SID_m \parallel y)$, which indicates the identity $SID_m$ and pseudoidentity $PSID_m$ of $S_m$ are bundled up with the secret key $y$ of $CS$ by hash function. As proved in the section of security analysis in [10], this technique can be effective against the cloud server impersonation attack. Similarly, the authors claim that the operation $\Delta_i = h(PID_i \parallel ID_i \parallel x)$ is aslo used to avoid that the user cheats $CS$ with a false identity. Unfortunately, we further research discovered that this design is redundant in the user registration phase.

As described in [8], the authentication scheme using smart card is mainly to resolve the problem, which the remote servers must store a verification table containing user identities and passwords. In the login phase of Kang et al.'s [10] protocol, only legal $U_i$ with the real identity $ID_i$, password $P_i$ and biometric $B_i$ can access the card reader. Moreover, the operation $PID_i = h(ID_i \parallel b_i)$ makes clear that pseudoidentity $PID_i$ is also bound to the real identity $ID_i$ by hash function during the subsequent login phase, and the value $b_i$ is protected in the smart card. So, if $U_i$ can login into the card reader, the control server $CS$ can authenticate $U_i$. That's why the smart card is used in this authentication protocol. Therefore, the operation $\Delta_i = h(PID_i \parallel ID_i \parallel x)$ is designed redundant in Amin et al.'s protocol. The detailed description will be presented in Sect. 4.2.

#### 3.2.3 Inconvenient for password change

Generally, it is essential to update password for the legal $U_i$. However, for the sake of brevity, the password change phase is not introduced in [10]. Furthermore, we further discover that even if this phase is designed according to the Kang et al.'s protocol [10], $U_i$ has to re-register to the control server $CS$ via a secure channel. $CS$ should deliver a new smartcard for the $U_i$ or requires the $U_i$ to mail the original smart card for replacement. Our following description will demonstrate that an existing $U_i$ could not change password with his/her smart card locally. Assumed that, $U_i$ can renew password with smart card during the login phase. Then the following these steps will be performed:

Step 1: After punching the smart card, $U_i$ provides $ID_i^*$, $P_i^*$ and $B_i^*$ to the card reader($CR$).

Step 2: $CR$ computes $A_i^* = P_i^* \oplus h(B_i^*)$ and $C_i^* = h(ID_i^* \parallel A_i^*)$. Then, it checks whether the condition $C_i^*$ equals $C_i$. If $C_i^* = C_i$, the terminal prompts $U_i$ for a new password.

Step 3: $U_i$ enters a new password $P_i^{new}$ to $CR$.

Step 4: When $U_i$ logins to the card reader normally, $CR$ executes the following operations according to the login phase of Kang et al's protocol:

$$A_i^{new} = P_i^{new} \oplus h(B_i)$$
$$C_i^{new} = h\left(ID_i \parallel A_i^{new}\right)$$
$$b_i^{new} = \Omega_i \oplus A_i^{new}$$
$$PID_i^{new} = h\left(ID_i \parallel b_i^{new}\right)$$
$$D_i^{new} = E_i \oplus A_i^{new}$$
$$O_i^{new} = ID_i \oplus D_i^{new}$$
$$b_i^{new} = \Omega_i \oplus A_i^{new}$$
$$G_i^{new} = h\left(ID_i \parallel SID_m \parallel N_i \parallel TS_i \parallel D_i^{new}\right)$$
$$F_i = \Delta_i \oplus N_i$$
$$Z_i = SID_m \oplus h(D_i \parallel N_i)$$

Obviously, since $b_i^{new} \neq b_i$, where $b_i$ is produced by $CS$; so $PID_i^{new} \neq PID_i$, where $PID_i = h(ID_i \parallel b_i)$. What's more, since $\Delta_i = h(PID_i \parallel ID_i \parallel x)$, the value $\Delta_i$ is also changed. If $U_i$ does not register again for substituting the recorded values $\langle C_i, \Omega_i, \Delta_i, E_i, h(\cdot) \rangle$ in the smart card, $CS$ could not authenticate $U_i$ in the subsequent communication phase. Therefore, it is inconvenient for password change in Kang et al.'s improved protocol.

## 4 Our protocol

This section introduces an enhanced authentication and key agreement protocol for the IoT-enabled devices in distributed cloud computing environment, as Fig. 1 is showing in smart city. The current scenario involves 3 main entities: the server control $CS$, the cloud server $S_m$ and each IoT-enabled device, which belong to the user $U_i$. There are 5 phases in our enhanced protocol: (1) Registration phase, (2) login phase, (3) authentication and key agreement phase, (4) password change phase, (5) Identity update phase. The detailed implementation of the first three phases is showed in Fig. 3.

### 4.1 Registration phase

Firstly, the control server $CS$ randomly produces two high-entropy numbers $x$ and $y$, which $x$ is used as the secret key only known to $CS$ for authenticate all $U_i$ and $y$ is used as another secret key only known to $CS$ for authenticate all $S_m$, respectively [21–23]. Then, any cloud server and user can register with $CS$. In addition, the secure channel referred to in this phase can be the Internet Key Exchange Protocol version 2(IKEv2) [13] or Secure Socket Layer Protocol (SSL) [24].

#### *4.1.1 Cloud server registration phase*

During the cloud server registration, $S_m$ sends the message$\langle SID_m, d \rangle$ to $CS$, where $SID_m$ is its identity and $d$ is a random number. On receiving the message, $CS$ calculates $PSID_m = h(SID_m \parallel d)$, $BS_m = h\left(PSID_m \parallel SID_m \parallel y\right)$ and sends $\langle BS_m \rangle$ back to $S_m$ via a secure channel. Finally, $S_m$ stores secret parameter $\langle BS_m, d \rangle$ into the memory.

#### *4.1.2 User registration phase*

When a user $U_i$ wishes to register with $CS$, $U_i$ selects desired identity $ID_i$ and password $P_i$ to enter his/her IoT-enabled device such as a card reader [25, 26]. Then, the device collects $U_i$'s biometric $B_i$ and generates a random number $b$ to compute $PID_i = h(ID_i \parallel b)$,

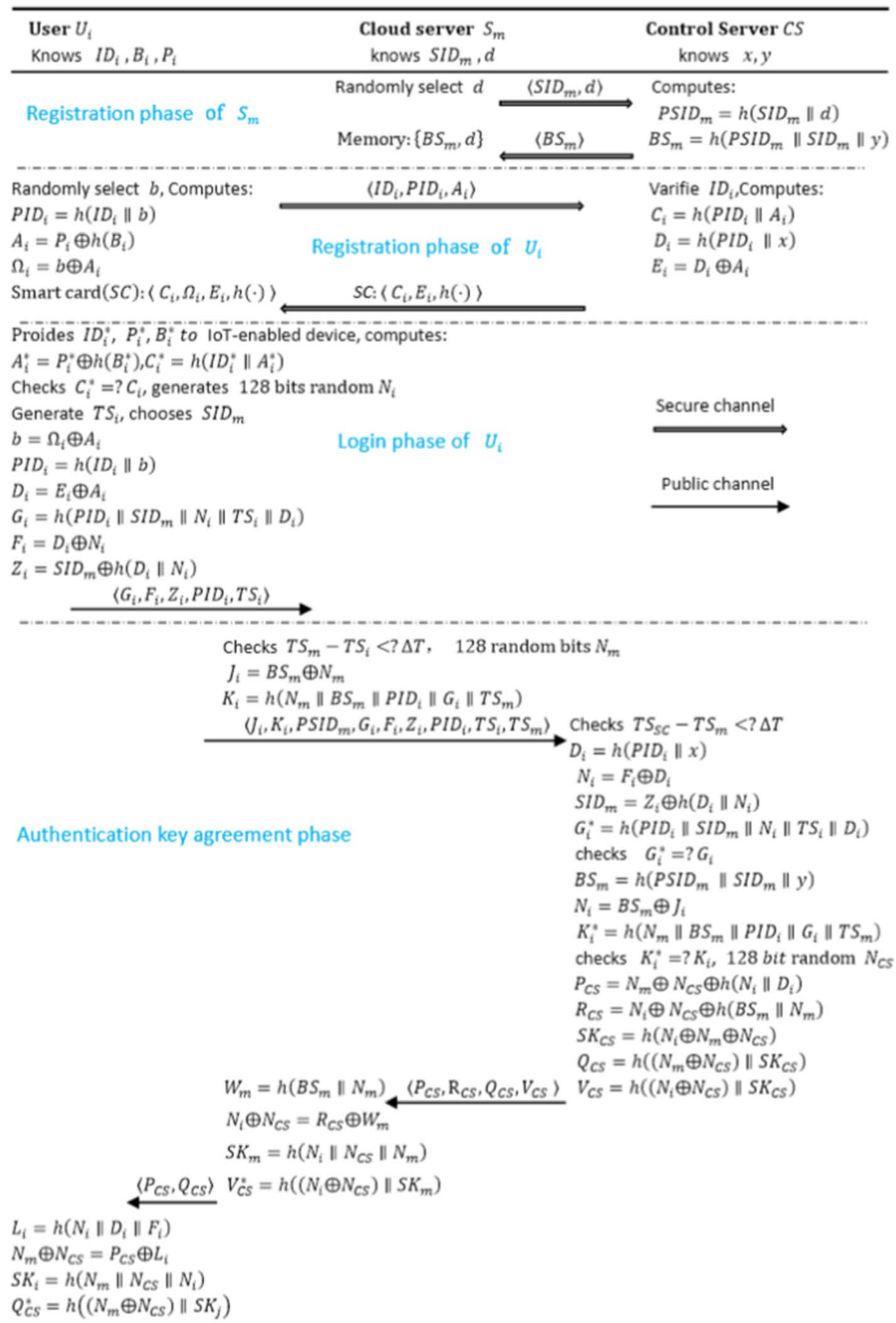



**Fig. 3** Implementation of our protocol. Implementation of the registration, login, and authentication key agreement phases in the proposed protocol

$A_i = P_i \oplus h(B_i)$ and $\Omega_i = b \oplus A_i$. Next, it sends $\langle ID_i, PID_i, A_i\rangle$ to $CS$ in a secure channel. After receiving the message, $CS$ verifies the authenticity of the user's identity $ID_i$. If $ID_i$ is illegal, $CS$ rejects $U_i$'s registration. Otherwise, $CS$ calculates $C_i = h(PID_i \parallel A_i)$, $D_i = h(PID_i \parallel x)$ and $E_i = D_i \oplus A_i$. Then, $CS$ writes the data $\langle C_i, \mathrm{E_i}, h(\cdot)\rangle$ to a smart card and delivers it to $U_i$ through private communication. When obtains the smart card, $U_i$ inserts it to IoT-enabled device and inputs $ID_i$ and $P_i$ to the device again. Then, the device writes $\Omega_i$ to the smart card. Finally, the smart card records the informations $\langle C_i, \Omega_i, \mathrm{E_i}, h(\cdot) \rangle$.

### 4.2 Login phase

If $U_i$ wants to get information from the private cloud server $S_m$, $U_i$ inserts the smart cart into the IoT-enabled device and provides $ID_i^*$,$P_i^*$ and $B_i^*$. The device computes $A_i^* = P_i^* \oplus h\left(B_i^*\right)$ and $C_i^* = h\left(ID_i^* \parallel A_i^*\right)$. Then, it verifies if $C_i^*$ is equal $C_i$. If $C_i^* = C_i$, the device authenticates the real $U_i$; otherwise, it rejects this login of $U_i$. Next, the device generates an at least 128 bits random number $N_i$ and executes the follow operations:

$$b = \Omega_i \oplus A_i$$
$$PID_i = h(ID_i \parallel b)$$
$$D_i = E_i \oplus A_i$$
$$G_i = h(PID_i \parallel SID_m \parallel N_i \parallel TS_i \parallel D_i)$$
$$F_i = D_i \oplus N_i$$
$$Z_i = SID_m \oplus h(D_i \parallel N_i)$$

where $TS_i$ is the current timestamp of the device, $SID_m$ is the private server $S_m$'s identity. After that, the device transmits $\langle G_i, F_i, Z_i, PID_i, TS_i \rangle$ to $S_m$ via a public channel.

### 4.3 Authentication and key agreement phase

In this phase, the mutual authentication and key agreement among three parties is mainly achieved through four-way handshake. In the first handshake, after receiving $U_i$'s login message, $S_m$ calculates its own verification condition to append with the login message and sends them to $CS$. In the second handshake, on receiving the message from $S_m$, $CS$ verifies the legitimacy of $U_i$ and $S_m$. If they are legit, $S_m$ produces itself authentication conditions for $U_i$ and $S_m$ respectively, and sends the conditions to $S_m$. In the third handshake, $S_m$ selects verification conditions related to itself to verify $CS$ and sends the remaining message to $U_i$. In the fourth handshake, $U_i$ verifies the legitimacy of $CS$. If any party fails to pass the authentication, the session will be ended in this phase. As a result, the entire authentication path $(U_i \rightarrow S_m \rightarrow SC \rightarrow S_m \rightarrow U_i)$ is established. In the meantime, a shared secret key $SK$ is negotiated to encrypt the subsequent communication traffic between $U_i$ and $S_m$. The detailed description is as follows:

Step 1: On receiving the login message, $S_m$ first checks the condition whether $TS_m - TS_i < \Delta T$ holds or not, If $TS_m - TS_i < \Delta T$, $S_m$ terminates the connection; otherwise, $S_m$ produce a 128 bits random number $N_m$ and calculates

$$J_i = B_m \oplus N_m$$
$$K_i = h(N_m \parallel BS_m \parallel PID_i \parallel G_i \parallel TS_m)$$

Then, $S_m$ sends $\langle J_i, K_i, PSID_m, G_i, F_i, Z_i, O_i, PID_i, TS_i, TS_m \rangle$ to the control server $CS$ publicly.

Step 2: After getting the message, $CS$ also checks whether $TS_{CS} - TS_m < \Delta T$ or not. If $TS_{CS} - TS_m < \Delta T$, $CS$ computes

$$D_i = h(PID_i \parallel x)$$
$$ID_i = O_i \oplus D_i$$
$$N_i = F_i \oplus h(PID_i \parallel ID_i \parallel x)$$
$$SID_m = Z_i \oplus h(D_i \parallel N_i)$$
$$G_i^* = h(ID_i \parallel SID_m \parallel N_i \parallel TS_i \parallel D_i)$$

Then, $CS$ checks the condition whether $G_i^*$ is equal $G_i$. If $G_i^* = G_i$, $CS$ authenticates the $U_i$ is legal; otherwise, $CS$ terminates the session. After that, $CS$ calculates

$$BS_m = h\left(PSID_m \parallel SID_m \parallel y\right)$$
$$K_i^* = h(N_m \parallel BS_m \parallel PID_i \parallel G_i \parallel TS_m)$$

Next, $CS$ checks if $K_i^*$ is equal $K_i$. If $K_i^* \neq K_i$, $CS$ thinks $S_m$ is illegal and terminates the session; otherwise, $CS$ randomly selects a 128 bits number $N_{CS}$ and computes

$$P_{CS} = N_m \oplus N_{CS} \oplus h(N_i \parallel D_i \parallel F_i)$$
$$R_{CS} = N_i \oplus N_{CS} \oplus h(BS_m \parallel N_m)$$
$$SK_{CS} = h(N_i \parallel N_m \parallel N_{CS})$$
$$Q_{CS} = h((N_m \oplus N_{CS}) \parallel SK_{CS})$$
$$V_{CS} = h((N_i \oplus N_{CS}) \parallel SK_{CS})$$

where, $SK_{CS}$ is the secret session key which can encrypt the following communicate message between $U_i$ and $S_m$. Finally, $CS$ sends $\langle P_{CS}, Q_{CS}, R_{CS}, V_{CS} \rangle$ to $S_m$ through public channel.

Step 3: When obtaining the messge from $CS$, the $S_m$ calculates

$$W_m = h(BS_m \parallel N_m)$$
$$N_i \oplus N_{CS} = R_{CS} \oplus W_m$$
$$V_{CS}^* = h((N_i \oplus N_{CS}) \parallel SK_m)$$

Next, $S_m$ checks whether $V_{CS}^* = V_{CS}$ or not. If $V_{CS}^* = V_{CS}$, $S_m$ authenticates $CS$ and sends $\langle P_{CS}, Q_{CS} \rangle$ to $U_i$.

Step 4: On receiving the reply message from $S_m$, $U_i$ computes

$$L_i = h(N_i \parallel D_i \parallel F_i)$$
$$N_m \oplus N_{CS} = P_{CS} \oplus L_i$$
$$SK_i = h(N_m \parallel N_{CS} \parallel N_i)$$
$$Q_{CS}^* = h\left((N_m \oplus N_{CS}) \parallel SK_j\right)$$

Then, $U_i$ checks whether $Q_{CS}^*$ is equal $Q_{CS}$. If $Q_{CS}^* = Q_{CS}$, $U_i$ confirms that $CS$ and $S_m$ are authentic. At last, the 3 participants of $U_i$ , $S_m$ and $CS$ negotiate a shared secret key

$$SK = h(N_m \parallel N_{CS} \parallel N_i)$$

### 4.4 Password change phase

This phase is invoked whenever $U_i$ wants to update his/her password without communicating with the control server $CS$. After inserting the smart card into the IoT-enabled device, $U_i$ provides $ID_i^*$, $P_i^*$ and $B_i^*$. Then, the device executes $A_i^* = P_i^* \oplus h\left(B_i^*\right)$

and $C_i^* = h\left(ID_i^* \parallel A_i^*\right)$. Then, it verifies if $C_i^*$ is equal $C_i$ or not. If $C_i^* = C_i$,the device prompts $U_i$ for a new password $P_i^{new}$ and generates a random number $b_i^{new}$; otherwise, it rejects $U_i$'s password change. Then, it computes the following operations

$$
\begin{aligned}
&A_i^{new} = P_i^{new} \oplus h(B_i)\\
&C_i^{new} = h\left(ID_i \parallel A_i^{new}\right)\\
&\Omega_i^{new} = b_i^{new} \oplus A_i^{new}\\
&b = \Omega_i \oplus A_i^*\\
&D_i = E_i \oplus A_i^*\\
&E_i^{new} = D_i \oplus A_i^{new}
\end{aligned}
$$

Finally, the device replaces recorded values $\langle C_i, \Omega_i, \mathrm{E_i} \rangle$ with $\left\langle C_i^{new}, \Omega_i^{new}, E_i^{new} \right\rangle$ in the smart card respectively. So, it is very convenient and fast for $U_i$ to update password using smart card locally in our protocol.

### 4.5 Identity update phase

It is practical that a legal $U_i$ updates his identity $ID_i$, such as the identity has expired. However, because the control server $CS$ needs to verify the authenticity of the user's $ID_i$, $U_i$ should re-register to $CS$ through the secure channel in this phase.

## 5 Results and discussion

In this section, we defines the capabilities of the attacker and makes a discussion on security analysis of our protocol. Based on adversarial model, we use the security protocol analysis tools of Automated Validation of Infinite-State Systems (AVISPA) and Scyther to prove the protocol can defend the various existing attacks. Then, we detailedly analyze the authentication paths among the three participators to ensure security protection from the most common vulnerabilities of impersonation attacks. Finally, the performance comparisons of our protocol with others are described briefly.

### 5.1 Adversarial model

In this section, we give the threat attack model, which the main reference is Dolev-Yao adversary threat model [27–29]. The detailed descriptions of Dolev–Yao adversary threat model are as follows:

(1) Adversary can eavesdrop and intercept all messages passing through the network;
(2) Adversary can store and send the intercepted or self-constructed messages;
(3) Adversary can participate in the operation of the protocol as a legal subject.
(4) The power analysis or side-channel attacks can help the attacker to extract the secret information stored in user's smart card.

### 5.2 Simulation of our protocol using security protocol analysis tools

This section presents simulation of our protocol using security protocol analysis tools of AVISPA and Scyther, both of which are complete and standard formal automatic analysis tools. The detailed instructions of AVISPA can refer to [30–33] and Scyther to [34–36].

#### *5.2.1 Simulation code description*

The first step in the use of simulation tools is to describe the target protocol in a formal language. This section introduces the AVISPA tool formal language HLPSL(High Level Protocol Specification Language) and the Scyther tool formal language SPDL(Security Protocol Description Language) to formally simulate our agreement.

*(1) The HLPSL simulation code of our protocol* The HLPSL simulation code of our protocol involves 5 roles: "role user" simulates real user $U_i$ ; "role server" simulates the cloud server $S_m$; 'role control server" simulates the server control $CS$; "role session" represent the role of the four interactive handshakes; "role environment" represent high-level corner with intruder; "role goal" represents the purpose of simulation. Below we only briefly introduce the part HLPSL description of user roles, environmental roles and security goals, as showing in Fig. 4.

In Fig. 4a, the user role process describes the parameters, initial states and transition that using at the beginning. The "transition" represents the acceptance of information and the sending of response information. "Channel (dy)" means that the attack mode is the Dolev–Yao attack model [37], in which the attacker can control of the network of the protocol. For example, an attacker can intercept, steal, modify, and replay the information transmitted on the channel in the protocol and even pretends to be a legal role in the protocol to perform operations to initiate an attack.

The Fig. 4b presents the role environment and the security goals. The high-level role process includes global constants and a mixed role process of one or more sessions. Among them, the intruder may pretend to be a legitimate user and run certain role processes. There are also some sentences that describe the knowledge known to the intruder in initial state, generally including the name of the agent, all the keys shared by other agents, and all known functions. For the HLPSL modeling of security goals, we only give the confidentiality goal of HLPSL supporting one of the two goals of confidentiality and authentication. For confidentiality, the target instance indicates which values are kept secret among the declared roles. If it cannot be achieved, it means that the intruder has obtained a confidential value and can successfully attack the protocol. For authentication, the main purpose is to verify identity masquerading attacks. Although Amin et al. [5] claimed that their protocol can reach the three authentication security goals

```
//The protocol description in HLPSL code
role user (U,S,CS: agent,
            Smartcard: symmetric_key,
            SND, RCV: channel (dy))
played_by U def=
local State : nat,
Na,Nb,Nc : text
init State := 0
transition
0. State = 0
 /\ RCV(start) =|>
State':= 2
/\ Na' := new()
/\ SND({U.Na'}_Smartcard)
/\ secret(Na',na,{U,CS})
2. State = 2
/\RCV({CS.Na.Nb'.Nc'}_Smartcard) =|>
State' := 7
end role
```

(a) User role

```

role environment() def=
const user, concrolserver, server : agent,
      smartcard,bs : symmetric_key,
      na,nb,nc : protocol_id

intruder_knowledge = {user,server,concrolserver}
composition
session(user,server,concrolserver,smartcard,bs)
/\ session(i,server,concrolserver,smartcard,bs)
/\ session(user,i,concrolserver,smartcard,bs)
/\ session(user,server,i,smartcard,bs)
end role

goal
secrecy_of na,nb,nc
end goal
```

(b) Environment role and security goal

**Fig. 4** Part HLPSL simulation code of our protocol. Figure includes two pictures. The **b** presents the HLPSL description of user roles. The role environment and the security goals in HLPSL code are showing in **b**

(*the authentication_on alice_server_ni, the authentication_on server_aserver_ncs, the authentication_on aserver_alice_nm*) [5], Kang et al. [10] pointed out the server cannot guarantee the cloud server chosen by the user, which is vulnerable to counterfeit attack. We will specifically demonstrate how our protocol resist this common attack in Sect. 4.2.

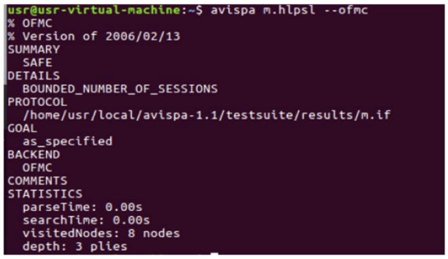
*(2) The SPDL simulation code of our protocol* It is similar to HLPSL that the SPDL simulation code of our protocol includes 3 roles: "role U" simulates real user $U_i$ ; "role S" simulates the cloude server $S_m$; "role CS" simulates the server control *CS*. Here, we take the control server *CS* role as an example to introduce the SPDL code, which is presented in Fig. 5. After defining the variables required for session protocol, the full implementation of our protocol is represented by the collection of events that occur in *CS*. The "send" and "recv" events indicate that *CS* sends a message and receives one respectively. One of the advantages of the Scyther tool is that it flexibly describe target attributes, whether it is the confidentiality of a variable or the authentication of a certain subject to another subject. The Scyther tool can analyze and verify the security attributes that users are interested in. The description of the target attribute is completed through the "claim" event, which can be used to describe the authentication of roles and the confidentiality of variables.

#### *5.2.2 Simulation results*

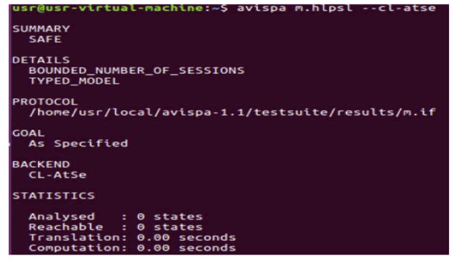
This section presents the simulation results of our protocol using two formal analysis tools. We personally build the AVISPA (Version of 2006/02/13) and Scyther(v1.1.3) in a virtual machine of an ubuntu operating system. Figure 6 presents the results of all the four back-end analysis tools provided by AVISPA to simulate the proposed protocols for all entities. The test results of OFMC, CL-AtSe, and SATMC modules show that our protocol is safe (SUMMARY SAFE), which means it can achieve the expected security goals; the TA4SP verification model represents INCONCLUSIVE, as the current TA4SP module does not support one-way hash function and the result of No ATTACK TRACE can be provided with the current version. When using the Scyther tool to simulate the protocol, we also use the Dolev-Yao attack model and the minimum number of execution rounds in the analysis parameters is set to 3. The simulation results of the Scyther tool is present in Fig. 7. Figure 7a shows the attack path of the Scyther tool's formal analysis under the Dolev-Yao model for our protocol. The reachability analysis report of our protocol messages is presented in Fig. 7b. The test results show that our proposed protocol

```
role CS
{
    var hello,helloU : Nonce;
    fresh BS : Nonce;
    fresh SC : Nonce;
    fresh Ncs : Nonce;
    var Ni,Ns : Nonce;
    recv_1(S,CS,{S,hello}k(S,CS));
    send_2(CS,S,{CS,BS}k(S,CS));
    recv_3(U,CS,{U,helloU}k(U,CS));
    send_4(CS,U,{CS,SC}k(U,CS));
    recv_6(S,CS,{Ni}SC,{Ns}BS);
    send_7(CS,S,{h(Ni,Ns,Ncs)}BS,{h(Ni,Ns,Ncs)}SC);
    claim(CS,Secret,h(Ni,Ns,Ncs));
}
```

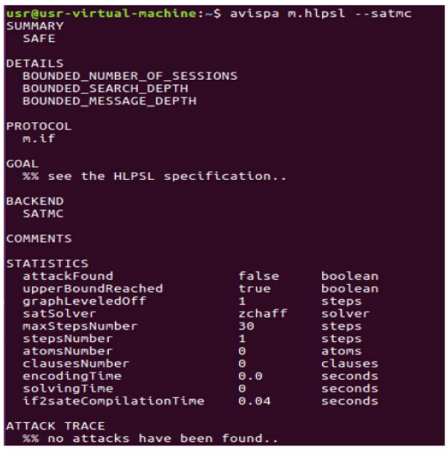
**Fig. 5** Control server CS role in SPDL. The control server *CS* role in the SPDL code using Scyther tool




(a) OFMC result



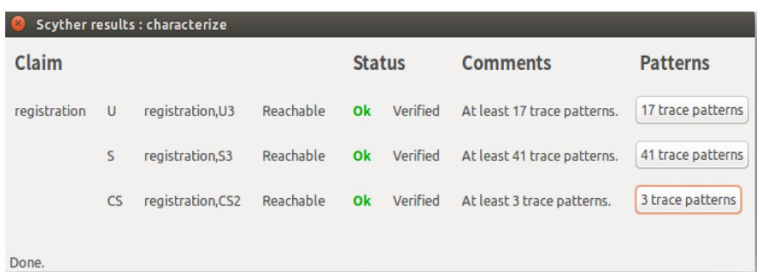

(b) CL-AtSe result




(c) SATMC result


(d) TA4SP result

**Fig. 6** Simulation results of the AVISPA tool under the four backends analysis. The results of all the four back-end analysis tools provided by AVISPA to simulate the proposed protocols for all entities. The test results of OFMC, CL-AtSe, and SATMC modules respectively




(a) Attack path under the Dolev-Yao model


(b) Reachability analysis report of our protocol

**Fig. 7** Simulation results of the Scyther tool. Figure includes two pictures. The **a** shows the attack path of the Scyther tool's formal analysis under the Dolev–Yao model for our protocol. The reachability analysis report of our protocol messages is presented (**b**) does not have any threat of attack under this model. Therefore, we can assert that our protocol can resist the various common attacks, such as insider attack, replay attack, session key discloser attack and so on.

### 5.3 Security analysis

In the following, we mainly use cryptography knowledge to analyze in detail the authentication paths among $U_i$, $S_m$, and $CS$ in our proposed, so as to protect against the most common attacks of impersonation attack [38–41].

*(1) Mutual authentication between $S_m$ and $CS$* In the cloud server registration phase, $S_m$ negotiates with $CS$ to produce a value $BS_m = h\left(PSID_m \parallel SID_m \parallel y\right)$, which can be regarded as the symmetric secret key for $S_m$ and $CS$, since the value $BS_m$ only can be calculate by $S_m$ and $CS$. Therefore, $S_m$ and $CS$ can achieve mutual authentication through the symmetric secret key $BS_m$ in the authentication phase, such as Kerberos protocol authentication. Moreover, since the identity $SID_m$ and pseudoidentity $PSID_m$ of $S_m$ all bind up with the secret number $y$ of the control server $CS$, $CS$ will authenticate both identities of $S_m$. Thus, our protocol can realize mutual authentication between $S_m$ and $CS$ in the authentication phase. Based on [5], we mark it with the following symbols:

$$\text{In the authentication phase: } S_m(SID_m) \Leftrightarrow S_m(PSID_m) \overset{BS_m}{\Leftrightarrow} CS\left(y\right) \tag{1}$$

*(2) Mutual authentication between $U_i$ and $CS$* As discussed in Chapter 2.2.2, in order to avoid recording the $U_i$'s identity and password information on the control server$CS$, $CS$ distributes a smart card to $U_i$ during the registration phase. The smart card records the values $\left\langle C_i, E_i, h(\cdot)\right\rangle$ in our protocol.

Firstly, as the only $U_i$ that knows $ID_i$,$B_i$ and $P_i$ can computes $C_i = h(ID_i \parallel A_i)$, and $A_i = P_i \oplus h(B_i)$ for logging into the IoT-enabled device, the value $C_i$ recording in the smart card is mainly used to verify $U_i$. So, we mark it with the following symbols:

$$\text{In the user logined phase:} U_i(ID_i) \overset{smart\ card(\ C_i)}{\Leftrightarrow} \text{IoT-enabled device} \tag{2}$$

The above symbol means that: with the help of value $C_i$ recording in the smart card, IoT-enabled devices can authenticate $U_i$. On the other hand, the user trusts the IoT-enabled device obviously.

Secondly, when $U_i$ logins into the device, the device will compute $b = \Omega_i \oplus A_i$ ,$PID_i = h(ID_i \parallel b)$ and$D_i = E_i \oplus A_i$. The value $E_i$ recording in the smart card can be regarded as an intermediate data in the process of authentication between the IoT-enabled device and $CS$. On the one hand, only the IoT-enabled device can compute $D_i = E_i \oplus A_i$ with the data $E_i$, if $U_i$ logined into the device with $B_i$ and $P_i$. On the other hand, only $CS$ that knows $x$ and $PID_i$ can compute $D_i = h(PID_i \parallel x)$, then computes $A_i = D_i \oplus E_i$ with the data $E_i$. Thus, IoT-enabled device and $CS$ can realize mutual authentication in the help of the smart card in the user login phase. So, we mark it with the following symbols:

$$\text{During the user login phase: IoT-enabled device} \overset{smart\ card(\ E_i)}{\Leftrightarrow} CS(x) \tag{3}$$

Thirdly, as $U_i$ logined into the IoT-enabled device, the device can compute $D_i$ with the value $E_i$. Then, the value $D_i$ can be the symmetric secret key for the IoT-enabled device and $CS$ in the authentication, since only the IoT-enabled device and $CS$ can calculate the value $D_i$. Therefore, the IoT-enabled device and $CS$ can achieve mutual authentication through the symmetric secret key $D_i$ in the authentication phase. So, we mark it with the following symbols:

$$\text{In the authentication phase: IoT-enabled device} \overset{D_i}{\Leftrightarrow} CS(x) \tag{4}$$

Based on the symbol (2), symbol (3) and symbol (4), we can deduce with the following symbol:

$$\text{In the authentication phase: } U_i(ID_i) \overset{D_i}{\Leftrightarrow} CS(x) \tag{5}$$

The above symbol means that: with the help of the smart card, $U_i$ with the identity $ID_i$ can authenticate each other with $CS$ in the authentication phase.

In addtion, after receiving $U_i$ registration message, $CS$ should verify the authenticity of $U_i$ 's identity $ID_i$ . When the identity $ID_i$ is confirmed to be legal, $CS$ will perform subsequent operations and delivers a smart card to $U_i$. Then, while $U_i$ logined into the IoT-enabled device, the device computes $PID_i = h(ID_i \parallel b_i)$ , which makes clear that pseudoidentity $PID_i$ is bound with the real identity $ID_i$ by hash function, and the value $b_i$ is protected by $\Omega_i$ recording in the smart card. So, the $U_i$'s identity $ID_i$ is indirectly controlled by $U_i$'s pseudoidentity $PID_i$, which is bound with the secret number $x$ of the control server $CS$ with operation $D_i = h(PID_i \parallel x)$. Thus, we mark it with the following symbol:

$$\text{In the authentication phase:} U_i(ID_i) \Leftrightarrow U_i(PID_i) \overset{D_i}{\Leftrightarrow} CS(x) \tag{6}$$

*(3) Mutual authentication between $U_i$ and $S_m$* Just like the above part (2) analysis, we can mark with the following symbols in this part:

$$\text{In the authentication phase: } U_i(PID_i) \overset{N_i,SID_m}{\Leftrightarrow} CS(x) \tag{7}$$

Since the values $N_i$ and $SID_m$ are encrypted and transmitted by the symmetric secret key $D_i$,where $F_i = D_i \oplus N_i$ and $Z_i = SID_m \oplus h(D_i \parallel N_i)$.

$$\text{In the authentication phase: } S_m(PSID_m) \overset{N_m}{\Leftrightarrow} CS(y) \tag{8}$$

Since the value $N_m$ is encrypted and transmitted by the symmetric secret key $BS_m$, where $J_i = BS_m \oplus N_m$.

$$\text{In the authentication phase: } U_i(PID_i) \overset{N_m \oplus N_{CS}}{\Leftrightarrow} CS(x) \tag{9}$$

Since the value $N_m \oplus N_{CS}$ is encrypted and transmitted by the secret value $N_i$ and $D_i$, where $P_{CS} = N_m \oplus N_{CS} \oplus h(N_i \parallel D_i)$ .

$$\text{In the authentication phase: } S_m(PSID_m) \overset{N_i \oplus N_{CS}}{\Leftrightarrow} CS(y) \tag{10}$$

**Table 2** Security functionality comparison of our protocol with the related protocols

| Security functionality | Ours | Kang et al. [10] | Amin et al. [5] |
|---|---|---|---|
| User's anonymity | YES | YES | YES |
| User auditability | YES | NO | NO |
| Simple and secure password change | YES | NO | YES |
| Resist off-line password guessing attack | YES | NO | YES |
| Resist impersonation attack | YES | YES | NO |
| Protection of the biometric | YES | YES | NO |
| Resist insider attack | YES | YES | YES |
| Resist replay attack | YES | YES | YES |
| Resist session key discloser attack | YES | YES | YES |

**Table 3** Operations comparison among our scheme with other related schemes

| | Ours | Kang et al. [10] | Amin et al. [5] |
|---|---|---|---|
| Cloud server registration phase | 2H | 2H | 2H |
| User registration phase | 4H + 3X | 5H + 3X | 5H + 3X |
| Login phase | 5H + 5X | 5H + 6X | 6H + 5X |
| Authertication and key agreement phase | 17H + 20X | 18H + 21X | 17H + 20X |
| Total count | 28H + 28X | 30H + 30X | 30H + 28X |

H, hash operation and it's numbers; X, xor operation and it's numbers

Since the value $N_i \oplus N_{CS}$ is encrypted and transmitted by the secret value $BS_m$ and $N_m$, where $R_{CS} = N_i \oplus N_{CS} \oplus h(BS_m \parallel N_m)$.

Therefore,we we can deduce with the following symbol:

$$\text{In the authentication phase: } U_i(PID_i) \overset{SK_i}{\Leftrightarrow} CS(x, y) \overset{SK_m}{\Leftrightarrow} S_m(PSID_m) \tag{11}$$

As the symbol (11) shows, our protocol realize mutual authentication between $U_i$ and $S_m$ through the mediator of $CS$. What's more, the 3 parties share the same session key $SK = h(N_m \parallel N_{CS} \parallel N_i)$. As a result, we can assert that our protocol can effectively resist impersonation attacks.

### 5.4 Performance comparisons

In the following, we concretely compare our protocol with the other two protocols [5, 10] in terms of resistance to security functionality and computational performance. In the Table 2, we list the 9 general security requirements of a robust authentication protocol for IoT-enabled devices and cloud servers. The results in Table 2 show the superiorities of our protocol are User auditability, simple and secure password change, resist off-line password guessing attack, resist impersonation attack and protection of the biometric.

Moreover, the Table 3 shows the number of times the hash function and XOR operation have cost in each phase of our protocol with other related protocol. From the total count in the last line, we can see that our protocol uses the hash function and XOR the least number of times. Thus, it is more suitable for the environment in which the applications are resource-constrained and data-intensive, such as IoT-enabled devices in the smart city.

## 6 Concluding remarks

In this paper, we deeply researched the authentication protocols for IoT-enabled devices in distributed cloud computing environment. We discover that Kang et al.'s protocol has 3 security drawbacks, such as vulnerable to off-line password guessing attack, designed redundant in the user registration phase and inconvenient for password change. Then, we introduced a lightweight pseudonym identity based authentication and key agreement protocol using smart card. To illustrate the security of our protocol, the security protocol analysis tools of AVISPA and Scyther are used to prove the proposed protocol can defend the various existing attacks, such as repaly attack, weak password guessing attack, man-in-the-middle attack, session key discloser attack and so on. We further analyze the authentication paths among participants in our proposed with cryptography knowledge, so as to avoid the most common attacks of impersonation attack. Moreover, we concretely compare our protocol with the other two protocols in terms of resistance to security requirements and computational performance. Both results show that our protocl is superior to the other two related protocols. As a result, the enhanced protocol will be applicable in distributed cloud computing architecture for smart city.

#### Abbreviations

IoT: Internet of things; SaaS: Software as a service; PaaS: Platform as a service; IaaS: Infrastructure as a service; CS: The server control; CR: Card reader; AVISPA: Automated validation of infinite-state systems; HLPSL: High level protocol specification language; SPDL: Security protocol description language; OFMC: On-the fly model-checke; AtSe: Attack searcher; SATMC: SAT-based model-checke; A4SP: Tree automatabased protocol analyzer.

#### Acknowledgements

This work was supported the National key research and development projects (No. 2019QY0501) and the National Natural Science Foundation of China (No. 2904020211).

#### Authors' contributions

In this paper, HH conceived and designed the study. The paper was written by HH and ZW. SL discussed the formal analysis and QW worked as the advisors to discuss. All authors read and revised the manuscript.

#### Availability of data and materials

The corresponding author shall keep the analysis and full simulation code set. If necessary, the data set can be requested from the corresponding author according to reasonable requirements.

## Declarations

#### Competing interests

The authors declare that they have no competing interests.

#### Author details

[1]State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450001, Henan, China.
[2]Henan Key Laboratory of Network Cryptography Technology, Zhengzhou 450001, Henan, China.

Received: 20 August 2020 Accepted: 24 June 2021
Published online: 10 July 2021

#### References

1. A.B. Zaslavsky, C. Perera, D. Georgakopoulos, in *Sensing as a Service and Big Data in International Conference on Advances in Cloud Computing, ACC-2012, Bangalore, India*, (2012), p. 219
2. G. Ateniese, R. Burns, R. Curtmola, et al., in *Provable Data Possession at Untrusted Stores in Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS'07), Virginia, VA, USA*, (2007), p. 598–609
3. S.R. Chandra, Yf. WANG, Cloud things construction—the integration of internet of things and cloud computing. Future Gener. Comput. Syst. **56**(C), 684–700 (2016)
4. M. Díaz, C. Martín, B. Rubio, State-of-the-art, challenges, and open issues in the integration of Internet of things and cloud computing. J. Netw. Comput. Appl. **67**, 99–117 (2016)
5. R. Amin, N. Kumar, G.P. Biswas, R. Iqbal, V. Chang, A light weight authentication protocol for IoT-enabled devices in distributed cloud computing environment. Future Gener. Comput. Syst. **78**, 1005–1019 (2018)

6. S.K. Sood, A.K. Sarje, K. Singh, A secure dynamic identity based authentication protocol for multi-server architecture. J. Netw. Comput. Appl. **34**(2), 609–618 (2011)
7. X. Li, Y. Xiong, J. Ma, W. Wang, An efficient and security dynamic identity based authentication protocol for multi server architecture using smart cards. J. Netw. Comput. Appl. **35**(2), 763–769 (2012)
8. K. Xue, P. Hong, C. Ma, A lightweight dynamic pseudonym identity based authentication and key agreement protocol without verification tables for multi-server architecture. J. Comput. Syst. Sci. **80**(1), 195–206 (2014)
9. M.C. Chuang, M.C. Chen, An anonymous multi-server authenticated key agreement scheme based on trust computing using smartcards and biometrics. Expert Syst. Appl. **41**(4), 1411–1418 (2014)
10. B. Kang, Y. Han, K. Qian et al., Analysis and improvement on an authentication protocol for IoT-enabled devices in distributed cloud computing environment. Math. Probl. Eng. **2020**(2), 1–6 (2020)
11. L. Zhou, X. Li, K. Yeh, C. Su, W. Chiu, Lightweight IoT based authentication scheme in cloud computing circumstance. Future Gener. Comput. Syst. **91**, 1244–1251 (2019)
12. O. Ruan, N. Kumar, D. He, J.-H. Lee, Efficient provably secure password-based explicit authenticated key agreement. Pervasive Mob. Comput. **24**, 50–60 (2015)
13. C. Kaufman, Internet Key Exchange (IKEv2) Protocol. RFC 4306, (2005)
14. M.S. Hwang, L.H. Li, A new remote user authentication scheme using smart cards. IEEE Trans. Ind. Electron. **46**(1), 28–30 (2000)
15. A. Vilmos, C. Medaglia, A. Moroni, Vision and challenges for realising the internet of things[J]. hot working technology, 2010
16. A. Whitmore, A. Agarwal, L. Da Xu, The internet of things—a survey of topics and trends. Inf. Syst. Front. **17**(2), 261–274 (2015). https://doi.org/10.1007/s10796-014-9489-2
17. C.C. Chang, T.C. Wu, Remote password authentication with smartcards. IEEProc. Comput. Digit. Tech. **38**(3), 165–168 (1999)
18. R. Amin, G.P. Biswas, A secure light weight scheme for user authentication and key agreement in multi-gateway based wireless sensor networks. Ad Hoc Netw. (2015). https://doi.org/10.1016/j.adhoc
19. P. Kocher, J. Jaffe, B. Jun, Differential power analysis, in Proceedings of Advances in Cryptology, (1999), p. 388–397
20. T.S. Messerges, E.A. Dabbish, R.H. Sloan, Examining smart-card security under the threat of power analysis attacks. IEEE Trans. Comput. **51**(5), 541–552 (2002)
21. D.B. He, N. Kumar, J. Chen et al., Robust anonymous authentication protocol for healthcare applications using wireless medical sensor networks. Multimed. Syst. **21**(1), 49–60 (2015). (**14**)
22. W. Diffie, M.E. Hellman, New directions in cryptography. IEEE Trans. Inf. Theory **22**(6), 644–654 (1976)
23. K.Y. Yoo, E.J. Yoon, G.R. Alavalapati, Comment on efficient and secure dynamic ID-based remote user authentication scheme for distributed systems using smart cards. IET Inf. Secur. **11**(4), 220–221 (2016)
24. C. Heinrich, Transport layer security (TLS). Hit.bme.hu **31**(4), 2009 (2005)
25. J.S. Leu, W.B. Hsieh, Efficient and secure dynamic ID-based remote user authentication scheme for distributed systems using smart cards. IET Inf. Secur. (2014). https://doi.org/10.1049/iet-ifs.2012.0206
26. D. He, N. Kumar, N. Chilamkurti, A secure temporal-credential-based mutual authentication and key agreement scheme with pseudo identity for wireless sensor networks. Inf. Sci. **321**, 263–277 (2015)
27. D. Dolev, A. Yao, On the security of public key protocols. IEEE Trans. Inf. Theory **29**(2), 198–208 (1983)
28. D. He, Y. Zhang, D. Wang, K.-K. Raymond Choo, Secure and efficient two-party signing protocol for the identity-based signature scheme in the IEEE P1363 standard for public key cryptography. IEEE Trans. Dependable Secure Comput. **17**(5), 1124–1132 (2020)
29. Y. Zhang, D. He, X. Huang, D. Wang, K.-K.R. Choo, J. Wang, White-box implementation of the identity-based signature scheme in the IEEE P1363 standard for public key cryptography. IEICE Trans. Inf. Syst. **E103–D**(2), 188–195 (2020)
30. AVISPA Team. (2014). AVISPA Tool. http://www.avispa-project.org. Accessed: Aug 2020
31. A. Armando et al., The AVISPA tool for the automated validation of Internet security protocols and applications, in *17th International Conference on Computer Aided Verification, (CAV05), Lecture Notes in Computer Science*, (vol. 3576, Springer-Verlag, 2005), p. 281–285
32. L. Viganò, Automated security protocol analysis with the AVISPA tool. Electron. Notes Theor. Comput. Sci. **155**, 61–86 (2006)
33. Scyther Team. (2014). Scyther Tool. https://people.cispa.io/cas.cremers/scyther/index.html. Accessed: Aug 2020
34. C.J.F. Cremers, Scyther: semantics and verification of security protocols/door Casimier Joseph Franciscus Cremers (2006)
35. C.J.F. Cremers, The scyther tool: verification, falsification, and analysis of security protocols, in *International Conference on Computer Aided Verification* (Springer, Berlin, Heidelberg, 2008)
36. C.J.F. Cremers, P. Lafourcade, P. Nadeau, Comparing state spaces in automatic security protocol analysis, in *Formal to Practical Security—Papers Issued from the 2005–2008 French-Japanese Collaboration* (2009)
37. D. He, S. Zeadally, N. Kumar, J.H. Lee, Anonymous authentication for wireless body area networks with provable security. IEEE Syst. J. **PP**(99), 1–12 (2016). https://doi.org/10.1109/JSYST.2016.2544805
38. L. Wu, Y. Zhang, K.K.R. Choo et al., Efficient and secure identity-based encryption scheme with equality test in cloud computing. Future Gener. Comput. Syst. **73**(Aug.), 22–31 (2017)
39. S.M. Bellovin, M. Merritt, Encrypted key exchange: password-based protocols secure against dictionary attacks, in *Proceedings of IEEE Symposium on Security & Privacy*, (1992), p.72–84
40. A. Groce, J. Katz, A new framework for efficient password-based authenticated key exchange, in *17th ACM Conference on Computer and Communications Security (CCS)*, (2010), p. 516–525
41. Q. Feng, D. He, Z. Liu, D. Wang, R. Choo, Multi-party signing protocol for the identity-based signature scheme in IEEE P1363 standard. IET Inf. Secur. **14**(4), 443–451 (2020)