# A Lightweight Authentication and Key Exchange Protocol With Anonymity for IoT

Daojing He, *Member, IEEE*, Yanchang Cai, Shanshan Zhu, Ziming Zhao, Sammy Chan, *Senior Member, IEEE*, and Mohsen Guizani, *Fellow, IEEE*

***Abstract*— The number of IoT devices is growing rapidly, and the interaction between devices and servers is also more frequent. However, IoT devices are often at the edge of the network, which leads their communications with the server to be completely exposed, making it more vulnerable to attacks. Moreover, IoT devices have limited energy and computational resources. Therefore, we propose in this paper a lightweight authentication and key exchange protocol with anonymity for IoT devices. The proposed scheme supports mutual authentication between IoT devices and the server. We verify the security of the protocol through formal and informal analyses. Finally, we compare security and performance with other protocols, which shows that our protocol has the advantages of being lightweight and secure.**

***Index Terms*— Internet of Things (IoT), authentication protocol, IoT device, lightweight.**

## I. Introduction

INTERNET of Things (IoT) is a concept of the interconnection of all things, which provides the function of collecting and generating data for services and users. Due to the open network environment, the communication among these devices and the cloud server (CS) is extremely vulnerable to a wide variety of attacks including replay attacks, impersonation attacks, etc. In addition, IoT devices are often limited by low battery power, low computational resources, and low storage. At the same time, since IoT devices provide the data foundation for the operation of IoT systems, secure communications between devices and CS becomes more important. Therefore, it is particularly important to have a low-cost and a highly secure authentication protocol.

The protocol proposed by Amin et al. [1] is mainly aimed to resist smart card stolen attacks and offline password guessing attacks, and the protocol is used on medical devices, which provides convenience for the use of more secure biometric passwords, but cannot be applied to other scenarios without biological information. Sureshkumar et al. [2] proposed a protocol for device cluster authentication, which has the feature of privacy protection, but because it involves multiple uses of the elliptic curve algorithm, it requires higher computational cost.

### A. Challenges and Motivation

Authentication protocols for IoT devices face many challenges. Anonymity is an effective feature in preventing an adversary from obtaining device information. At present, the protocols that satisfy anonymity are usually based on asymmetric encryption methods, and the computational cost is relatively expensive. Although some protocols use pseudonyms, they increase storage and management cost. To reduce the computational cost incurred at devices, some protocols use a third-party assisted method, but this method is not suitable for two-party authentication of IoT devices. Those protocols that use the pre-computation method could reduce the computational cost during authentication, but total computational cost is not reduced. At the same time, the authentication protocol needs to defend against attacks, such as replay attacks, anti-server simulation attacks, etc. In addition, some protocols use biometric information to achieve authentication, which has limitations in common scenarios.

IoT devices need to meet communication security requirements while being limited by resource shortages. Therefore, we need to design a protocol for IoT devices that can satisfy anonymity, low cost and security.

### B. Research Contributions

Motivated by the above challenges, we propose a lightweight authentication and key exchange protocol with

Manuscript received 16 July 2022; revised 13 January 2023; accepted 1 March 2023. Date of publication 24 March 2023; date of current version 13 November 2023. This work was supported in part by the National Key Research and Development Program of China under Grant 2021YFB2700900; in part by the Shenzhen Key Technical Project under Grant 2022N009; in part by the National Natural Science Foundation of China under Grant U1936120; in part by the Fok Ying Tung Education Foundation of China under Grant 171058; in part by the Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies under Grant 2022B1212010005; and in part by the University Grants Committee of the Hong Kong Special Administrative Region, China, under Project CityU 11201421. The associate editor coordinating the review of this article and approving it for publication was J. Li. *(Corresponding author: Daojing He.)*

Daojing He is with the Software Engineering Institute, Nanchang University, Nanchang 330047, China, and also with the School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen 518055, China (e-mail: hedaojinghit@163.com).

Yanchang Cai and Ziming Zhao are with the Software Engineering Institute, Nanchang University, Nanchang 330047, China (e-mail: 13864008846@163.com).

Shanshan Zhu is with the Harbin Institute of Technology, Shenzhen 518055, China (e-mail: zhushanshan85@163.com).

Sammy Chan is with the Department of Electrical Engineering, City University of Hong Kong, Hong Kong (e-mail: eeschan@cityu.edu.hk).

Mohsen Guizani is with the Machine Learning Department, Mohamed bin Zayed University of Artificial Intelligence (MBZUAI), Abu Dhabi, United Arab Emirates (e-mail: mguizani@ieee.org).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TWC.2023.3257028.

Digital Object Identifier 10.1109/TWC.2023.3257028

anonymity for IoT that is low-cost, and resistant to various attacks. The main contributions of this paper are as follows.

- We propose a lightweight authentication and key exchange protocol with anonymity for IoT that can resist multiple attacks. This protocol only includes a pair of elliptic curve encryption and decryption, simple symmetric encryption and hash operations, and only a small number of message exchange.
- We use the Real-Or-Random (ROR) model to demonstrate the key indistinguishability, and show the formal verification of the proposed protocol using the popularly used AVISPA tool. By informal analysis, we prove that our protocol is secure facing various well-known attacks.
- We evaluate the total cost in terms of computational and communication costs between IoT devices and CS, and compare it with other authentication protocols. Comparisons show that our proposed protocol has significant performance advantages.

### C. Paper Organization

The rest of this article is organized as follows. Section II discusses the related work. In Section III, we describe the system architecture and present our proposed protocol. In Section IV, we present a comprehensive security analysis of the proposed protocol. The comparison with other protocols is provided in Section V and conclusions are made in Section VI.

## II. Related Work

In recent years, the wide application of IoT devices has brought great convenience to our life. However, IoT devices are often plagued by security and high latency. In addition to cloud computing, the corresponding concept of fog computing has also been proposed, which is regarded as the future of the IoT industry because it can provide high-performance, low-latency services, but its cost also increases with the enhancement of corresponding services. In this section, we review some relevant IoT device authentication schemes.

In 2015, Kalra et al. [3] proposed a mutual authentication protocol based on HTTP cookies and elliptic curve cryptography to enhance communication security between IoT devices and CS. In the same year, Xie et al. proposed a uniqueness-and-anonymity-preserving user authentication scheme for a health care system [4]. This scheme is later pointed out by Xu et al. that it could not resist the desynchronization attack [5]. An improved scheme is also proposed in [5] that could resist a number of attacks such as insider attacks and offline attacks. In 2018, Chuang et al. [6] proposed a lightweight continuous authentication protocol for devices to authenticate data which are exchanged over a pre-defined period. The concept of valid authentication time period is proposed and the dynamic characteristics of token technology and IoT devices are adopted to meet the continuity and robustness requirements. In 2018, Qiu et al. [7] analyzed various protocols [8], [9] [10], pointed out some issues, and made some improvements. In 2019, a multi-device authentication and key distribution protocol was proposed by Sureshkumar et al. [2], which is characterized by providing more effective privacy protection and communication security authentication, and is suitable for multi-device security authentication in a small range. In 2019, Tai et al. [11] pointed out the shortcomings of the scheme proposed by Li et al. [12], and made further discussions. In 2020, Nikooghadam et al. [13] proposed a lightweight authentication protocol between IoT devices and CS. The Scyther tool and BAN logic are used to verify the security of this protocol. Another lightweight authentication protocol was proposed by Nakkar et al. [14] to reduce latency for critical applications and provide forward secrecy. The comparison with other protocols proved the main security properties of this protocol. Also, Melki et al. [15] proposed a lightweight and secure multi-factor authentication protocol for IoT devices based on the unique PUF value and random channel characteristics within IoT devices. In 2021, in order to address the high complexity and vulnerability issues, Shahidinejad et al. [16] proposed a lightweight authentication protocol for IoT devices. The protocol involves a trust center and the cloud service provider. The protocol has certain advantages in anti-attack, communication cost and computation cost. In 2022, Qiu et al. [17] proposed a provably secure authentication and key distribution protocol based on extended chaotic map using “fuzzy validator” and “honey talk” technology, which combines biometrics and chaotic map, and all 13 safety evaluation criteria were proved to be met through safety analysis and simulation results.

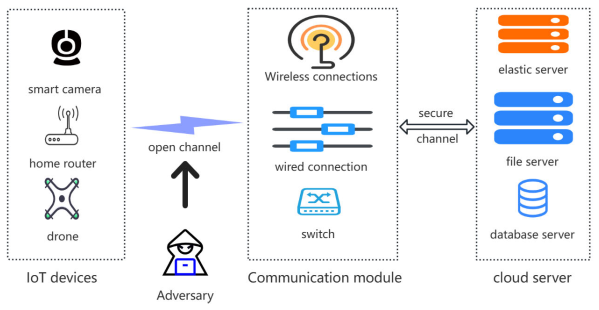


Fig. 1. System architecture of IoT device authentication.

## III. The Proposed Protocol

We present our proposed authentication and key exchange protocol in this section. The protocol consists of two phases, one is the registration phase and the other is the authentication communication phase. In the following, we first present the system architecture, and the threat model, then we present the details of the proposed authentication protocol.

### A. System Architecture

There are two types of entities in this protocol: IoT devices and CS, which are connected by a communication module as depicted in Fig. 1.

*1) IoT Devices:* An IoT device is first registered with the CS. After registration, the IoT device and CS authenticate each other. After authentication, a shared session key *sk* is established, and the IoT devices can use *sk* to access the services of the CS.

TABLE I
SYMBOLS AND THEIR DESCRIPTIONS

| Symbol | Description |
|---|---|
| $ED_i$ | $i\_th$ device |
| $CS$ | Cloud server |
| $ID_{cs}$ | ID of $CS$ |
| $Id_i$ | Identity information of $ED_i$ |
| $Pw_i$ | Password of $ED_i$ |
| $T_1, T_2, T_3, T_4$ | Timestamps |
| $r_{cs}, e_i, q_i$ | High entropy random numbers |
| $\Delta T$ | Maximum transmission delay |
| $A$ | Adversary |
| $h(.)$ | Hash function |
| $E_t$ | Expiration time of the device |
| $X_{cs}$ | Server's private key |
| $P_{cs}$ | Server's public key |
| ECC | Elliptic Curve Cryptography algorithm |
| $sk$ | Session key |
| $\oplus$ | Bitwise XOR operation |
| $\|\|$ | Data concatenation |
| $ENC_{sk}/DEC_{sk}$ | Symmetric encryption/decryption |

*2) Cloud Server:* The cloud server is responsible for the registration of IoT devices, providing services for authenticated devices or obtaining information uploaded by trusted devices. Before authentication, an IoT device needs to confirm that it has been registered with the CS.

*3) Communication Module:* IoT devices and CS communicate through communication modules, which can be basic network facilities or wireless transmission devices. In this protocol, it only has the function of information transmission, and is not an entity requiring identity verification.

## *B. Threat Model*

The Dolev-Yao adversarial model [18] is widely used to verify the security of protocols, so we use the same threat model here. Attacker $\mathcal{A}$ has the following capabilities.

- Adversary $\mathcal{A}$ can intercept, modify, delete and retransmit all information in the open channel.
- Adversary $\mathcal{A}$ has the ability to steal the device and obtain the information stored in the device.
- Adversary $\mathcal{A}$ has the ability to impersonate device and CS.
- For encrypted information, adversary $\mathcal{A}$ needs to obtain the key to decrypt it.
- Servers are trusted entities, $\mathcal{A}$ is unable to compromise it.

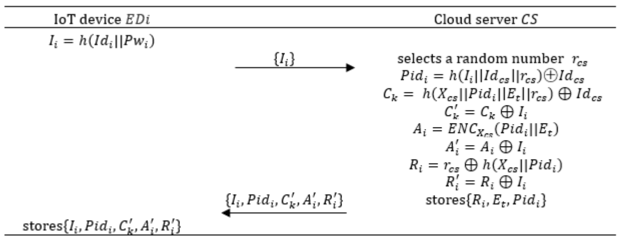
## *C. Protocol Description*

There are two phases in our protocol: the device registration phase and the mutual authentication phase. Table I presents the symbols used in the proposed protocol.

*1) Registration Phase:* As shown in Fig. 2, to register with the timestamp, an IoT device generates a registration request message with its own identity information $Id_i$ and password $Pw_i$, and sends this request to the CS. The registration phases are as follows:

*Step 1:* The IoT device $ED_i$ uses its own $Id_i$ and $Pw_i$ to calculate:

$$I_i = h(Id_i||Pw_i) \tag{1}$$



IoT device $EDi$
Cloud server $CS$
$I_i = h(Id_i||Pw_i)$
$\{I_i\}$
selects a random number $r_{cs}$
$Pid_i = h(I_i||Id_{cs}||r_{cs}) \oplus Id_{cs}$
$C_k = h(X_{cs}||Pid_i||E_t||r_{cs}) \oplus Id_{cs}$
$C'_k = C_k \oplus I_i$
$A_i = ENC_{X_{cs}}(Pid_i||E_t)$
$A'_i = A_i \oplus I_i$
$R_i = r_{cs} \oplus h(X_{cs}||Pid_i)$
$R'_i = R_i \oplus I_i$
$\{I_i, Pid_i, C'_k, A'_i, R'_i\}$
stores$\{R_i, E_t, Pid_i\}$
stores$\{I_i, Pid_i, C'_k, A'_i, R'_i\}$

Fig. 2. Device registration phase.

Then it is sent to the CS. Here, since original $Id_i$ and $Pw_i$ are not sent, but only the value $I_i$ after the hash operation is sent, this can effectively prevent internal attacks.

*Step 2:* Having received the request message $\{I_i\}$ from the device, the CS selects a random number $r_{cs}$. To calculate the device identity related information $Pid_i$ and $R_i$, the CS performs the following operations.

$$Pid_i = h(I_i \,||\, Id_{cs} \,||\, r_{cs}) \oplus Id_{cs} \tag{2}$$
$$C_k = h(X_{cs} \,||\, Pid_i \,||\, E_t \,||\, r_{cs}) \oplus Id_{cs} \tag{3}$$
$$C'_k = C_k \oplus I_i \tag{4}$$
$$A_i = ENC_{X_{cs}}(Pid_i \,||\, E_t) \tag{5}$$
$$A'_i = A_i \oplus I_i \tag{6}$$
$$R_i = r_{cs} \oplus h(X_{cs} \,||\, Pid_i) \tag{7}$$
$$R'_i = R_i \oplus I_i \tag{8}$$

The CS stores $\{R_i, E_t, Pid_i\}$ in a secure database. Then it sends $\{Pid_i, A'_i, R'_i, C'_k\}$ to the IoT device, this information is used in the first step of the authentication phase to generate the transmitted information.

*Step 3:* The IoT device securely stores the information $\{I_i, Pid_i, C'_k, A'_i, R'_i\}$ in its own database after receiving it.

*2) Authentication Phase:* In this phase, the CS and the IoT device complete the authentication through the following steps. Fig. 3 shows the authentication process of the proposed scheme and its flow chart is shown in Fig. 4.

*Step 1:* IoT device $ED_i$ inputs $Id'_i$ and $Pw'_i$, and computes:

$$I^*_i = h(Id'_i \,||\, Pw'_i) \tag{9}$$
$$C_k = C'_k \oplus I_i \tag{10}$$

Then $ED_i$ chooses a random number $e_i$ and generates a current timestamp $T_1$. To compute the encrypted information $N_i$ and $E_i$, $ED_i$ performs the following operations.

$$A_i = A'_i \oplus I_i \tag{11}$$
$$R_i = R'_i \oplus I_i \tag{12}$$
$$N_i = ECC_{P_{cs}}(Pid_i \,||\, R_i) \tag{13}$$
$$E_i = ENC_{C_k}(e_i, T_1, A_i) \tag{14}$$

Here, the formula $N_i = ECC_{P_{cs}}(Pid_i \,||\, R_i)$ is used to encrypt the device identity information $P_{cs}, R_i$. The random number $e_i$ used to generate the session key is encrypted by the formula $E_i = ENC_{C_k}(e_i, T_1, A_i)$. Then, $ED_i$ sends the request information $\{N_i, T_1, E_i\}$ to the CS.

*Step 2:* After the CS receives the request message from the device, it first checks $T_2 - T_1 < \Delta T$, where $T_2$ is the current timestamp of the CS. If the condition is satisfied, the

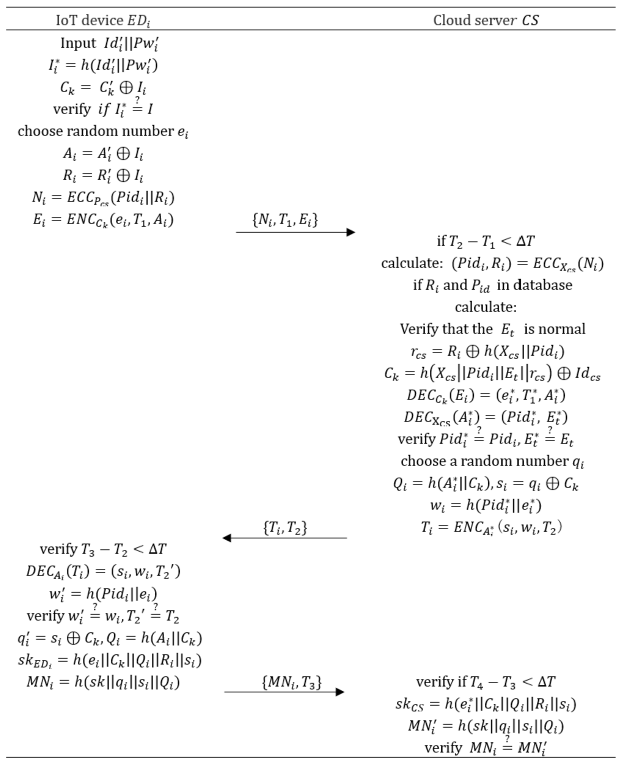

IoT device $ED_i$
Cloud server $CS$
Input $Id_i'||Pw_i'$
$I_i^* = h(Id_i'||Pw_i')$
$C_k = C_k' \oplus I_i$
verify $if\ I_i^* \overset{?}{=} I$
choose random number $e_i$
$A_i = A_i' \oplus I_i$
$R_i = R_i' \oplus I_i$
$N_i = ECC_{P_{cs}}(Pid_i||R_i)$
$E_i = ENC_{C_k}(e_i, T_1, A_i)$
$\{N_i, T_1, E_i\}$
if $T_2 - T_1 < \Delta T$
calculate: $(Pid_i, R_i) = ECC_{X_{cs}}(N_i)$
if $R_i$ and $P_{id}$ in database
calculate:
Verify that the $E_t$ is normal
$r_{cs} = R_i \oplus h(X_{cs}||Pid_i)$
$C_k = h(X_{cs}||Pid_i||E_t||r_{cs}) \oplus Id_{cs}$
$DEC_{C_k}(E_i) = (e_i^*, T_1^*, A_i^*)$
$DEC_{X_{cs}}(A_i^*) = (Pid_i^*, E_t^*)$
verify $Pid_i^* \overset{?}{=} Pid_i, E_t^* \overset{?}{=} E_t$
choose a random number $q_i$
$Q_i = h(A_i^*||C_k), s_i = q_i \oplus C_k$
$w_i = h(Pid_i^*||e_i^*)$
$T_i = ENC_{A_i^*}(s_i, w_i, T_2)$
$\{T_i, T_2\}$
verify $T_3 - T_2 < \Delta T$
$DEC_{A_i}(T_i) = (s_i, w_i, T_2')$
$w_i' = h(Pid_i||e_i)$
verify $w_i' \overset{?}{=} w_i, T_2' \overset{?}{=} T_2$
$q_i' = s_i \oplus C_k, Q_i = h(A_i||C_k)$
$sk_{ED_i} = h(e_i||C_k||Q_i||R_i||s_i)$
$MN_i = h(sk||q_i||s_i||Q_i)$
$\{MN_i, T_3\}$
verify if $T_4 - T_3 < \Delta T$
$sk_{CS} = h(e_i^*||C_k||Q_i||R_i||s_i)$
$MN_i' = h(sk||q_i||s_i||Q_i)$
verify $MN_i \overset{?}{=} MN_i'$

Fig. 3. Authentication phases for IoT devices.

CS computes $(Pid_i \mid\mid R_i) = ECC_{X_{CS}}(N_i)$ gets $Pid_i$ and $R_i$; otherwise, the session is terminated.

Then, the CS searches its database for $Pid_i$ and $R_i$, if it exists, the CS obtains the corresponding information $E_t$. If $E_t$ expires, the session is abandoned; if not, the CS performs the following calculations to obtain the decryption key $C_k$.

$$r_{cs} = R_i \oplus h(X_{cs} \mid\mid Pid_i) \tag{15}$$

$$C_k = h\left(X_{cs} \mid\mid Pid_i \mid\mid E_t \mid\mid r_{cs}\right) \oplus Id_{cs} \tag{16}$$

The CS can calculate $DEC_{C_k}(E_i) = (e_i^*, T_1^*, A_i^*)$ to decrypt $E_i$ to get $e_i^*, T_1^*, A_i^*$, $DEC_{X_{cs}}(A_i^*) = (Pid_i^*, E_t^*)$ to decrypt $A_i^*$ to get $Pid_i^*, E_t^*$. The CS checks $Pid_i^* \overset{?}{=} Pid_i,\ E_t^* \overset{?}{=} E_t$. If so, the IoT device is accepted. It will also check $T_1^* \overset{?}{=} T_1$ to confirm that the timestamp has not been modified during the information exchange. Next, the current timestamp $T_2$ and the random number $q_i$ generated by the CS are used as follows.

$$Q_i = h(A_i \mid\mid C_k) \tag{17}$$

$$s_i = q_i \oplus C_k \tag{18}$$

$$w_i = h(Pid_i \mid\mid e_i^*) \tag{19}$$

$$T_i v = ENC_{A_i^*}(s_i, w_i, T_2) \tag{20}$$

The formula $T_i = ENC_{A_i^*}(s_i, w_i, T_2)$ encrypts $s_i$, $w_i$ and $T_2$. $s_i$ and $w_i$ in $T_i$ are used to generate the session key and the identity of the device verification server, respectively. Then, CS sends $T_i$ and $T_2$ to the IoT device.

*Step 3:* After receiving the message from the CS, the IoT device immediately compares the current timestamp $T_3$ with $T_2$ to check if the message is fresh ( $T_3 - T_2 < \Delta T$). Then, it uses $A_i$ to decrypt $T_i$, $DEC_{A_i}(T_i) = (s_i, w_i, T_2')$ to get $s_i, w_i, T_2'$. Then, it calculates:

$$w_i' = h(Pid_i \mid\mid e_i) \tag{21}$$

Device $ED_i$ confirms the identity of the CS by verifying $w_i' \overset{?}{=} w_i$. Verifying $T_2' \overset{?}{=} T_2$ ensures that the timestamp contained in the message has not been modified. Then, the device performs the following operations to calculate the session key *sk* and the authentication information MN of *sk*.

$$q_i\prime = s_i \oplus C_k \tag{22}$$

$$Q_i = h(A_i \mid\mid C_k) \tag{23}$$

$$sk_{ED_i} = h(e_i \mid\mid C_k \mid\mid Q_i \mid\mid R_i \mid\mid s_i) \tag{24}$$

$$MN_i = h(sk \mid\mid q_i \mid\mid s_i \mid\mid Q_i) \tag{25}$$

*Step 4:* After the CS receives the message, it first generates the current timestamp $T_4$ and checks the freshness of the message by $T_4 - T_3 < \Delta T$. If the freshness is satisfied, it calculates:

$$sk_{CS} = h(e_i^* \mid\mid C_k \mid\mid Q_i \mid\mid R_i \mid\mid s_i) \tag{26}$$

$$MN_i{}' = h(sk \mid\mid q_i \mid\mid s_i \mid\mid Q_i) \tag{27}$$

Next, the CS verifying $MN_i \overset{?}{=} MN_i{}'$ to ensure the *sk* is consistent with the device. By now, the IoT device and the CS authentication is completed.

## IV. Security Analysis

In this section, we perform the formal and informal verifications to prove the security of our protocol.

### A. Formal Analysis

In this section, we simulate our protocol in the Automated Verification of Applications (AVISPA) tool, which evaluates the security of the scheme against an adversary. Here, we describe the role of the CS and the IoT device $ED_i$, as well as the environment, session and target.

*1) Protocol Description:* The language HLPSL (High Level Protocol Specification Language) of the tool AVISPA [19] is a role-based language. Below, we define different roles and their operations according to the behaviours of different entities, and describe what information is passed between them.

We implement two basic roles of $ED_i$ and the CS through the HLPSL language. The $ED_i$ is described in the HLPSL language as shown in Fig. 5, and the CS is described in Fig. 6.

The $ED_i$ is the initiator of the protocol and receives the start signal. This signal maintains the operation of the protocol, and the state changes from 0 to 1. Then $I_i = h(Id_i \mid\mid Pw_i)$ is calculated, and $I_i$ is sent to the CS through a secure channel TX(). Declaring the channel type (dy) indicates that the channel is used for the Dolev-Yao [18] threat model. Then, the $ED_i$ receives the message $Pid_i, A_i', R_i', C_k'$ returned by the CS through the secure channel RX(). During the login phase, the $ED_i$ generates a random number through the new() method and timestamp, and then sends the message $\{N_i, T_1, E_i\}$ to the CS through the open channel. After receiving the message, the CS generates a random number through the new() method, and returns the message $\{T_i, T_2\}$ through the open channel.

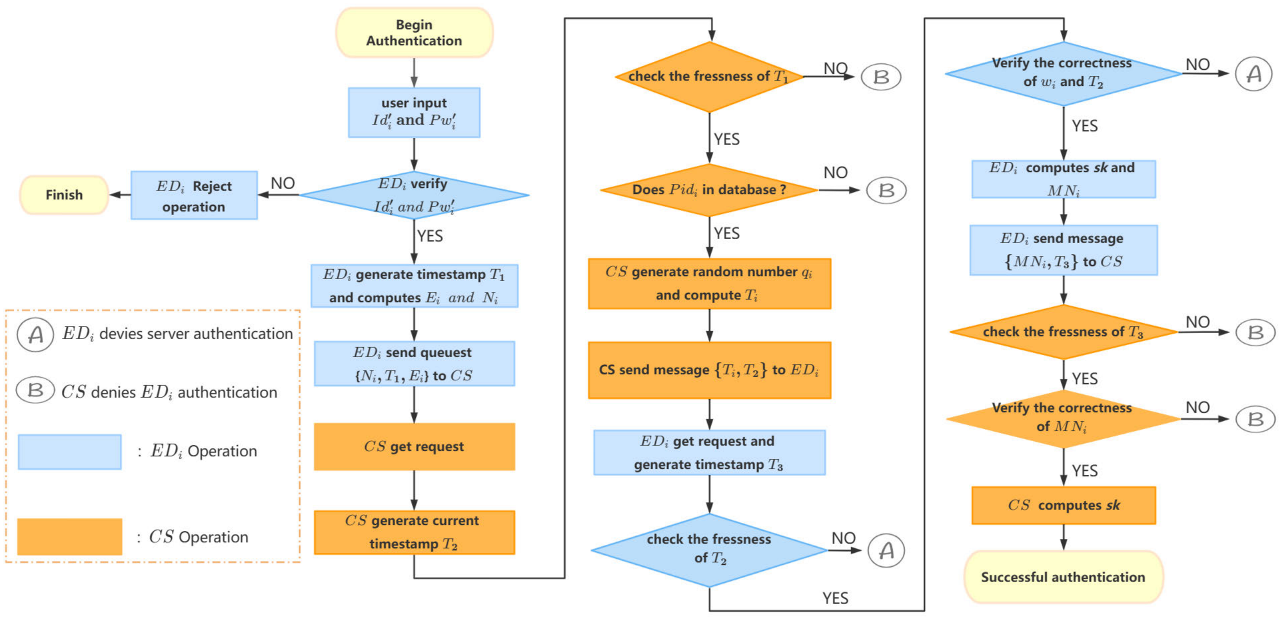

Begin Authentication
user input $Id_i'$ and $Pw_i'$
$ED_i$ verify $Id_i'$ and $Pw_i'$
NO
YES
$ED_i$ Reject operation
Finish
$ED_i$ generate timestamp $T_1$ and computes $E_i$ and $N_i$
$ED_i$ send queuest $\{N_i, T_1, E_i\}$ to $CS$
$CS$ get request
$CS$ generate current timestamp $T_2$
check the fressness of $T_1$
Does $Pid_i$ in database ?
$CS$ generate random number $q_i$ and compute $T_i$
CS send message $\{T_i, T_2\}$ to $ED_i$
$ED_i$ get request and generate timestamp $T_3$
check the fressness of $T_2$
Verify the correctness of $w_i$ and $T_2$
$ED_i$ computes sk and $MN_i$
$ED_i$ send message $\{MN_i, T_3\}$ to $CS$
check the fressness of $T_3$
Verify the correctness of $MN_i$
$CS$ computes sk
Successful authentication
A $ED_i$ devies server authentication
B $CS$ denies $ED_i$ authentication
: $ED_i$ Operation
: $CS$ Operation

Fig. 4. The work flow of the proposed authentication protocol.

Finally, the $ED_i$ replies the message $\{MN_i, T_3\}$ to the CS through the open channel.

As shown in Fig. 7, at the end of the protocol, we specify the session roles, goals, privacy goals, and environment of the program. There are two confidentiality goals and two authentication goals in our implementation, respectively. The first one is verify $(Edi, CS, alice\_bob\_r1, Eri')$ and request $(CS, Edi, alice\_bob\_r1, Eri)$. $ED_i$ implements server-side authentication by generating a new random number $N1$. The second is verify $(CS, Edi, bob_alice\_r2, Qri')$ and request $(Edi, CS, bob_alice\_r2, Qri)$.

*2) Simulation Results:* OFMC and CL_AtSe are two back-end analysis techniques of AVISPA. While the OFMC can identify errors in the protocol and complete limited sessions. Besides these functions, CL_AtSe executes the protocol using the detection and redundancy elimination methods.

The analysis results of the OFMC backend and CL_AtSe backend models are shown in Fig. 8 and Fig. 9, respectively. In this mode, the attacker has complete control over the external network, so that it can impersonate any communicating entity and communicate with other entities. As long as the attacker knows the required *sk*, it can intercept, analyze and modify the message. The backend needs to consider all possible security threats when analyzing the protocol. That is because, according to this model, the attacker can listen to any message in a communication session. Simulation results has shown that our protocol is secure.

### B. Security Proof in ROR Model

*1) Security Model:* Abdalla et al.'s Real-Or-Random (ROR) model [20] is used to prove the security of our protocol. In the ROR model, some definitions are as follow:

*Definition 1: A collision-resistant one-way hash function $h : \{0,1\}^* \to \{0,1\}^w$ is taken as a deterministic algorithm in nature, which generates a fixed-length w bits binary string output $h(u) \in \{0,1\}^l$, known as message digest or hash value, on the input as a binary string $u \in \{0,1\}^*$ of arbitrary length. An adversary A's advantage in finding collision with the execution time t is denoted and defined by $Adv_{\mathcal{A}}^{HASH}(t) = Pr\left[(u_1,u_2) \leftarrow_R \mathcal{A} : U_1 \neq u_2 \quad and \quad h(u_1) = h(u_2)\right]$, where $Pr\left[X\right]$ is an event X's probability, and $(u_1,u_2) \leftarrow_R \mathcal{A}$: indicates that the pair$(u_1,u_2)$ is randomly picked by $\mathcal{A}$. An$(\psi, t)$-adversary $\mathcal{A}$ attacking the collision resistance of $h(\cdot)$ means that the execution time of $\mathcal{A}$ will be at most t and that $Adv_{\mathcal{A}}^{HASH}(t) \leq \psi$.*

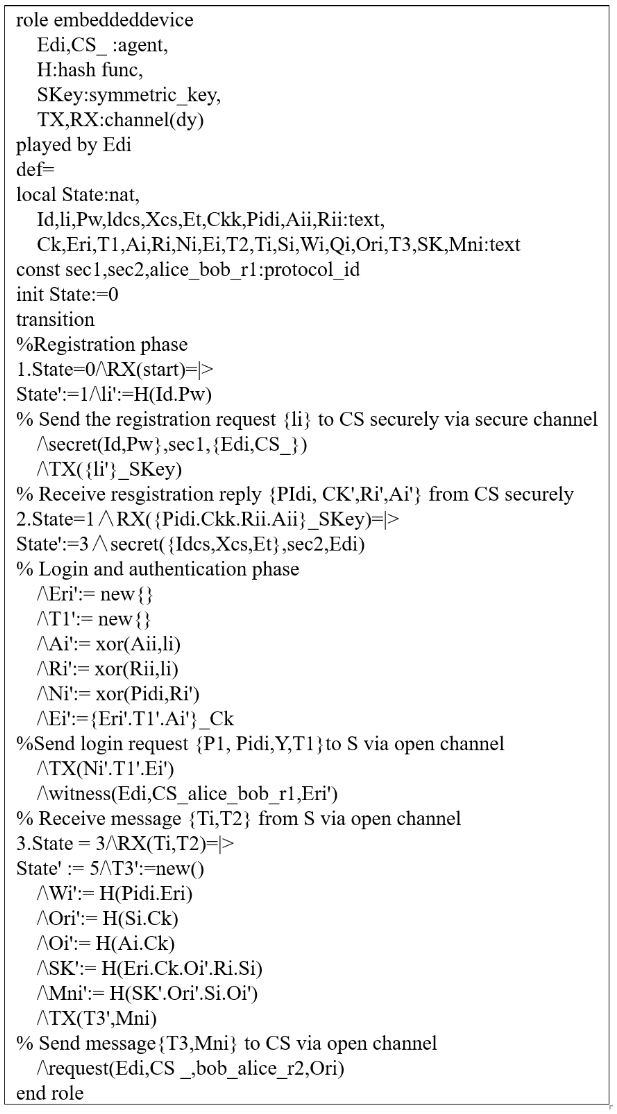
*Participants:* Let $\Pi_{IoT}^d$ and $\Pi_{CS}^v$ denote the instance d and v of $ED_i$ and the CS.

*Partnering:* Two instances are partnered if they hold the same non-null session identifications (sid). The $\Pi_{IoT}^d$ and $\Pi_{CS}^v$ partner each other when there is a non-empty session ID(SID) or other unique text between them.

*Freshness:* If adversary $\mathcal{A}$ does not obtain the *sk* between $\Pi_{IoT}^d$ and $\Pi_{CS}^v$ through the following query, then $\Pi_{IoT}^d$ and $\Pi_{CS}^v$ is fresh.

*Adversary:* Adversary $\mathcal{A}$ participates in the communication process through the following ORACLE query. It can read, modify and replay all the information in the process of communication.

- *Execute*($\Pi_{IoT}^d$,$\Pi_{CS}^v$): This query is a passive attack in which $\mathcal{A}$ eavesdrops the real messages between $\Pi_{IoT}^d$ and $\Pi_{CS}^v$.
- *Send*($\Pi_{IoT}^d$,$\Pi_{CS}^v$, m): Through this query, $\mathcal{A}$ can initiate active attacks, in which $\mathcal{A}$ may create, intercept, modify, and resend a message to the target entity $\Pi_{IoT}^d$ or $\Pi_{CS}^v$. The output is the information returned by $\Pi_{IoT}^d$ or $\Pi_{CS}^v$ after receiving information m.
- *CorruptIoT*($\Pi_{IoT}^d$): This query simulates the loss of an $ED_i$ by returning information stored in $\Pi_{IoT}^d$.
- *Test*($\Pi_{IoT}^d$,$\Pi_{CS}^v$): Flip an unbiased coin at the beginning of the query and get the result b $\epsilon\{0,1\}$. The value of b is kept secret from $\mathcal{A}$. If *sk* is defined and fresh for instances $\Pi_{IoT}^d$, $\Pi_{CS}^v$, return the *sk* for instances $\Pi_{IoT}^d$,



```
role embeddeddevice
   Edi,CS_ :agent,
   H:hash func,
   SKey:symmetric_key,
   TX,RX:channel(dy)
played by Edi
def=
local State:nat,
   Id,li,Pw,ldcs,Xcs,Et,Ckk,Pidi,Aii,Rii:text,
   Ck,Eri,T1,Ai,Ri,Ni,Ei,T2,Ti,Si,Wi,Qi,Ori,T3,SK,Mni:text
const sec1,sec2,alice_bob_r1:protocol_id
init State:=0
transition
%Registration phase
1.State=0/\RX(start)=|>
State':=1/\li':=H(Id.Pw)
% Send the registration request {li} to CS securely via secure channel
   /\secret(Id,Pw},sec1,{Edi,CS_})
   /\TX({li'}_SKey)
% Receive resgistration reply {PIdi, CK',Ri',Ai'} from CS securely
2.State=1/\RX({Pidi.Ckk.Rii.Aii}_SKey)=|>
State':=3/\secret({Idcs,Xcs,Et},sec2,Edi)
% Login and authentication phase
   /\Eri':= new{}
   /\T1':= new{}
   /\Ai':= xor(Aii,li)
   /\Ri':= xor(Rii,li)
   /\Ni':= xor(Pidi,Ri')
   /\Ei':={Eri'.T1'.Ai'}_Ck
%Send login request {P1, Pidi,Y,T1}to S via open channel
   /\TX(Ni'.T1'.Ei')
   /\witness(Edi,CS_alice_bob_r1,Eri')
% Receive message {Ti,T2} from S via open channel
3.State = 3/\RX(Ti,T2)=|>
State' := 5/\T3':=new()
   /\Wi':= H(Pidi.Eri)
   /\Ori':= H(Si.Ck)
   /\Oi':= H(Ai.Ck)
   /\SK':= H(Eri.Ck.Oi'.Ri.Si)
   /\Mni':= H(SK'.Ori'.Si.Oi')
   /\TX(T3',Mni)
% Send message{T3,Mni} to CS via open channel
   /\request(Edi,CS _,bob_alice_r2,Ori)
end role
```

Fig. 5. HLPSL language description of IoT devices.

$\Pi_{CS}^{v}$ or a random *sk* if b = 0; otherwise, return invalid symbol $\perp$.

*Semantic Security of the Session Key:* If our protocol is indistinguishable under the ROR model, it means that the generation of *sk* is random enough to avoid information leakage. Adversary $\mathcal{A}$ will launch as many test queries as possible to obtain the encrypted message c of message m. $\mathcal{A}$ executes the Test query and guesses that the value of $b'$ is either 0 or 1. If $b = b'$, $\mathcal{A}$ wins (denoted as $SUCC$). Let $\mathcal{P}$ denote our protocol, $Adv_{\mathcal{P}}^{ake}$ denote the advantage function of $\mathcal{A}$ violating semantic security in the ROR model. We have:

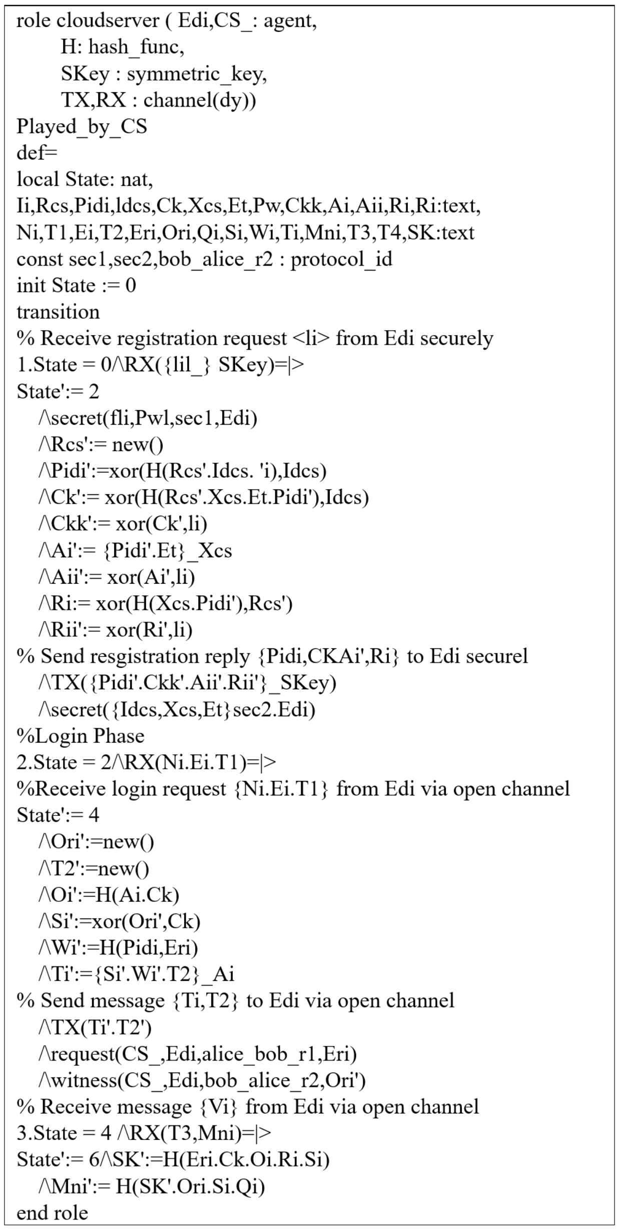
$$Adv_{\mathcal{A}}^{ake} = \mid 2B7Pr[SUCC] - 1 \mid .$$

If $Adv_{\mathcal{P}}^{ake}$ is negligible, we call the protocol $\mathcal{P}$ is considered safe in the ROR model.

*Random Oracle:* All entities and $\mathcal{A}$ have access to a collusion-resistant one-way cryptographic hash function $h(\cdot)$, modeled as a random oracle.

*Theorem 1:* In a random oracle model, $\mathcal{A}$ is an adversary running in polynomial time t against our protocol. Let $q_h$ denote the number of $Hash$ queries, $q_{send}$ denote the number of $Send$ queries, $\mid Hash \mid$ denote the range of space of the hash function, $\mid D \mid$ denote the size of the uniformly distributed password dictionary $D$. Then:

$$Adv_{\mathcal{P}}^{ake} \leq \frac{q_h^2}{|Hash|} + \frac{2q_{send}}{|D|}.$$



```
role cloudserver ( Edi,CS_: agent,
      H: hash_func,
      SKey : symmetric_key,
      TX,RX : channel(dy))
Played_by_CS
def=
local State: nat,
Ii,Rcs,Pidi,ldcs,Ck,Xcs,Et,Pw,Ckk,Ai,Aii,Ri,Ri:text,
Ni,T1,Ei,T2,Eri,Ori,Qi,Si,Wi,Ti,Mni,T3,T4,SK:text
const sec1,sec2,bob_alice_r2 : protocol_id
init State := 0
transition
% Receive registration request <li> from Edi securely
1.State = 0/\RX({lil_} SKey)=|>
State':= 2
   /\secret(fli,Pwl,sec1,Edi)
   /\Rcs':= new()
   /\Pidi':=xor(H(Rcs'.Idcs. 'i),Idcs)
   /\Ck':= xor(H(Rcs'.Xcs.Et.Pidi'),Idcs)
   /\Ckk':= xor(Ck',li)
   /\Ai':= {Pidi'.Et}_Xcs
   /\Aii':= xor(Ai',li)
   /\Ri:= xor(H(Xcs.Pidi'),Rcs')
   /\Rii':= xor(Ri',li)
% Send resgistration reply {Pidi,CKAi',Ri} to Edi securel
   /\TX({Pidi'.Ckk'.Aii'.Rii'}_SKey)
   /\secret({Idcs,Xcs,Et}sec2.Edi)
%Login Phase
2.State = 2/\RX(Ni.Ei.T1)=|>
%Receive login request {Ni.Ei.T1} from Edi via open channel
State':= 4
   /\Ori':=new()
   /\T2':=new()
   /\Oi':=H(Ai.Ck)
   /\Si':=xor(Ori',Ck)
   /\Wi':=H(Pidi,Eri)
   /\Ti':={Si'.Wi'.T2}_Ai
% Send message {Ti,T2} to Edi via open channel
   /\TX(Ti'.T2')
   /\request(CS_,Edi,alice_bob_r1,Eri)
   /\witness(CS_,Edi,bob_alice_r2,Ori')
% Receive message {Vi} from Edi via open channel
3.State = 4 /\RX(T3,Mni)=|>
State':= 6/\SK':=H(Eri.Ck.Oi.Ri.Si)
   /\Mni':= H(SK'.Ori.Si.Qi)
end role
```

Fig. 6. HLPSL language description of the cloud server.

*Proof:* We follow the proof approach in [21]. $Succ_i$ denote the event in the game $G_i$ where $\mathcal{A}$ guesses b′=b. The game details are as follows:

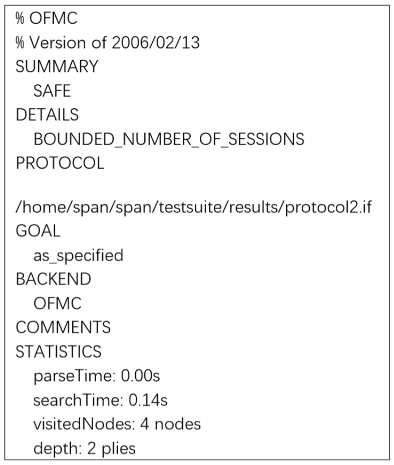
- *Game* $G_0$: In this Game, $\mathcal{A}$ performs a real attack on protocol $\mathcal{P}$, and b is randomly generated before the game starts. We have:

$$Adv_{\mathcal{P}}^{ake} = \left| 2 \cdot Pr\left[SUCC_0\right] - 1 \right| . \qquad (28)$$

```
role session ( Edi,CS_: agent,
      H : hash_func,
      SKey : symmetric_key)
   def=
      local
         Tx1, Tx2 : channel (dy),
         Rx1, Rx2 : channel (dy)
      composition
         embeddeddevice(Edi,CS_,H,SKey,Tx1,Rx1)
            ∧cloudserver(Edi,CS_,H,SKey,Tx2,Rx2)
end role

role environment()
def=
const edi,cloudserver : agent,
      hash_: hash_func,
      skey : symmetric_key,
      pidi,ni,ei,t1,ti,t2,mni,t3:text,
      sec1,sec2,alice_bob_rl,bob_alice_r2 : protocol_id
   intruder_knowledge={edi,cloudserver,hash_skey,
                        pidi,ni,ei,t1,ti,t2,mni,t3}
   composition
            session(edi,cloudserver,hash_,skey)
            ∧session(edi,i,hash_,skey)
            ∧session(i,cloudserver,hash_,skey)
end role
goal
secrecy_of sec1
secrecy_of sec2
authentication_on alice_bob_rl
authentication_on bob_alice_r2
end goal
environment()
```

Fig. 7. HLPSL language descriptions of sessions, targets and environments.

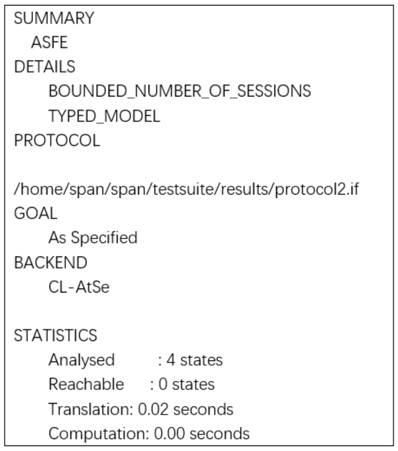


```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL

/home/span/span/testsuite/results/protocol2.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.14s
  visitedNodes: 4 nodes
  depth: 2 plies
```

Fig. 8. OFMC backend model output.

- $Game\ G_1$: Under this Game, $\mathcal{A}$ eavesdrops on the information between the entities $\Pi^d_{IoT}$ and $\Pi^v_{CS}$ by executing $Execute(\Pi^d_{IoT}$,$\Pi^v_{CS})$ queries. Finally, $\mathcal{A}$ makes $Test()$ oracle and decides whether $Test()$ output is the correct *sk* or a random value. The *sk* consists of random numbers $e_i$, $q_i$, $C_k$, $Q_i$, $R_i$, where $C_k = C'_k \oplus I_i$, $Q_i = h(A_i \mid\mid C_k)$, $R_i = R'_i \oplus I_i$. But the value of $l_i$ depends on $Id'_i$ and



```
SUMMARY
  ASFE
DETAILS
    BOUNDED_NUMBER_OF_SESSIONS
    TYPED_MODEL
PROTOCOL

/home/span/span/testsuite/results/protocol2.if
GOAL
    As Specified
BACKEND
    CL-AtSe

STATISTICS
    Analysed       : 4 states
    Reachable      : 0 states
    Translation: 0.02 seconds
    Computation: 0.00 seconds
```

Fig. 9. CL_AtSe backend model output.

$Pw'_i$. At this time, $\mathcal{A}$ cannot obtain $C_k$, $Q_i$, $R_i$ from the eavesdrop. Therefore, $\mathcal{A}$ cannot calculate the real *sk* in this experiment. Therefore, the experiment cannot be successful until $\mathcal{A}$ obtains $ED_i$ and knows its *sk*. So $Game_0$ and $Game_1$ are the same, we have:

$$Pr[Succ_0] = Pr[Succ_1]. \tag{29}$$

- $Game\ G_2$: Based on $Game_1$, attacker $\mathcal{A}$ can initiate Send and H($\cdot$)oracle queries. $Game_2$ simulates $\mathcal{A}$ spoofing attack where adversary $\mathcal{A}$ can create, edit and send a request to entities $\Pi^d_{IoT}$ and $\Pi^v_{CS}$. In our proposed protocol, the editable information of $\mathcal{A}$ is $N_i, T_1, E_i$, $T_i, T_2$, $MN_i, T_3$. However, each piece of information is timestamp dependent, so $\mathcal{A}$ executes the $Send()$ query without causing any collisions. According to the birthday paradox, we have:

$$Pr[Succ_1] \ - \ Pr[Succ_2] \leq \frac{q_h^2}{|Hash|}. \tag{30}$$

- $Game\ G_3$: In this game, $\mathcal{A}$ can obtain information stored by the $ED_i$. $\mathcal{A}$ obtains information of $ED_i$ through the enforcement of CorruptIoT($\Pi^d_{IoT}$). However, the CS also does not store $ED_i$ passwords, and the *sk* depends on the random numbers $e_i$ and $q_i$. However, these two random numbers are generated only by $ED_i$ and the CS in the authentication process, so $\mathcal{A}$ cannot calculate $sk$. Thus, we have:

$$Pr[Succ_2] \ - \ Pr[Succ_3] \leq \frac{q_{send}}{|D|}. \tag{31}$$

In the final Game, the attacker used all of the oracle try to break the security of protocol $\mathcal{P}$, and wants winning the game only by guessing the value of b. So,

$$Pr[Succ_3] = \frac{1}{2}. \tag{32}$$

By deriving formulas (28), (29), (30), (31) and (32), we obtain:

$$Adv^{ake}_{\mathcal{P}} \leq \frac{q_h^2}{|Hash|} + \frac{2q_{send}}{|D|}.$$

When the range space of hash function $Hash$ and password dictionary $D$ are very large numbers, the value of $Adv_{\mathcal{P}}^{ake}$ is a sufficiently small number. Hence, our protocol is secure under the ROR model.

### C. Correctness Analysis

In this section, we will prove the correctness of our protocol, which is the consistency of the key $sk_{ED_i} = sk_{CS}$. On the device, $C_k, Q_i, R_i$ are generated by the information $\{Id_i', Pw_i'\}$ input by the user and the data $\{C_k', A_i'\}$ stored locally during the registration phase. $s_i$ is obtained by calculating $DEC_{A_i}(T_i) = (s_i, w_i, T_2')$, so the value of $s_i$ is the same on the device and the CS. $A_i$ is obtained by calculating $A_i = A_i' \oplus I_i = A_i' \oplus h(Id_i' \mid\mid Pw_i')$. On the server side, the CS uses the private key to decrypt the information $N_i$ to obtain the corresponding data $Pid_i, R_i$ and search the local database to confirm the validity of the device identity. At this point, $Pid_{ED_i} = Pid_{CS}$, $R_{i_{ED_i}} = R_{i_{CS}}$. The information $C_k$ on the CS is obtained by calculating $C_k = h(X_{CS} \mid\mid Pid_i \mid\mid E_t \mid\mid R_{cs}) \oplus Id_{CS}$. Where $X_{CS}, R_{cs}, Id_{CS}$ is the information of the CS itself, so $C_{k_{ED_i}} = C_{k_{CS}}$. The CS calculates $Q_i = h(A_i' \mid\mid C_k)$ to obtain $Q_i$, calculates $DEC_{C_k}(E_i) = (e_i^*, T_1^*, A_i^*)$ to obtain $\{e_i^*, T_1^*, A_i^*\}$. At this time, we have $A_i^* = A_i$ and $e_i* = e_i$, so $Q_{i_{ED_i}} = Q_{i_{CS}}$. Finally, $sk_{ED_i} = h(e_i \mid\mid C_{k_{ED_i}} \mid\mid Q_{i_{ED_i}} \mid\mid R_{i_{ED_i}} \mid\mid s_i) = h(e_i^* \mid\mid C_{k_{CS}} \mid\mid Q_{i_{CS}} \mid\mid R_{i_{CS}} \mid\mid s_i) = sk_{CS}$.

### D. Informal Analysis

We believe that the protocol can provide the following security features and resist the following attacks.

*1) Perfect Forward Secrecy:* The protocol provides perfect forward secrecy. The attacker cannot recalculate the $sk$, even if he knows the long-term password $Pw_i$ of the IoT device or the long-term *sk X* of the CS. In the second step of our protocol, $R_i$ is stored in a private and secure database, $sk$ is calculated by $sk = h(e_i \mid\mid C_k \mid\mid Q_i \mid\mid R_i \mid\mid s_i)$. $C_k$ is calculated from the private $C_k'$ in the secure database. In other words, $sk$ relies on security parameters that the attacker cannot access, and therefore, the attacker cannot recompute $sk$. Therefore, our proposed protocol satisfies the complete forward secrecy.

*2) Device Anonymity:* With this security feature, the adversary cannot obtain the identity of the IoT device $Id_i$. In our protocol, only $I_i = h(Id_i \mid\mid Pw_i)$ and $Pid_i = h(I_i \mid\mid Id_{cs} \mid\mid r_{cs}) \oplus Id_{cs}$ are used in both phases of the protocol (where $r_{cs}$ is a random number generated by the $CS$), so even if the attacker can get hold of the exchanged message via eavesdropping, he cannot get the real identity of the device, so our protocol can provide device anonymity.

*3) Mutual Authentication:* Our protocol supports mutual authentication as follow. The CS verifies the device twice. The first time is in the second step of the authentication phase. The CS compares the $Pid_i^*$ in $N_i$ sent by the device with the local one. If it is the same, it means that the device information is real. The second time is in the fourth step, the CS also performs a similar operation on the information $MN_i'$.

For authenticating the CS by the IoT device, after the IoT device receives the information from the CS, it first takes the current time $T_3$, and calculates the time difference between $T_3$ and $T_2$ to check if the information is fresh or not. The device decrypts $T_i$, $DEC(T_i)_{A_i^*} = (s_i^*, w_i^*, T_2^*)$ to get $s_i^*$, $w_i^*$, $T_2^*$, and checks $w_i' \overset{?}{=} w_i$ to verify the identity of the CS. Note that verifying $T_2^* \overset{?}{=} T_2$ ensures that the timestamp contained in the message has not been modified.

*4) Replay Attack:* A replay attack means that the attacker resends the packet $N_i, T_1, E_i$ to the CS. In our protocol, the time is verified by both the CS and the IoT device. In the authentication process, each communication will calculate the time difference between the time in the information and the current time, such as $T_2 - T_1 < \Delta T$ in step 2 and $T_3 - T_2 < \Delta T$ in step 3 and $T_4 - T_3 < \Delta T$ in step 4. If an attacker uses old packets to launch a replay attack, both the CS and the IoT device will drop the session due to insufficient timestamp freshness.

*5) Device Impersonation Attack:* To launch such an attack, the attacker could send his own parameters $N_i'$ and $E_i'$ instead of $N_i$ and $E_i$ to the CS through an open channel. Next, we explain why an adversary cannot emulate an IoT device using these two parameters:

In the authentication phase, if the attacker sends his own parameter $E_i'$, then when the CS receives it, it cannot use $C_k$ to decrypt $E_i'$ to get the correct information. That is, the correct $A_i^*$ cannot be obtained, and the subsequent $Pid_i^*$ cannot be successfully authenticated, and the session will be discarded.

As described in the second step of the authentication phase, the server obtains $Pid_i$ by calculating $Pid_i = N_i \oplus R_i$, decrypts $A_i^*$ to obtain parameters $Pid_i^*$ and $E_t^*$, and verifies $Pid_i^* \overset{?}{=} Pid_i$, $E_t^* \overset{?}{=} E_t$, the next steps will not be performed until these conditions are met. As above, $E_i$ is impossible to forge, so the parameters $Pid_i$ and $R_i$ of $N_i$ are also impossible to forge. Therefore, if the adversary sends its own parameters $Pid_i'$ and $R_i'$ instead of $Pid_i$ and $R_i$ to CS. $Pid_i'$ will fail the verification and the request of the adversary will not be accepted. So the adversary could not impersonate the IoT device using $Pid_i'$.

*6) Server Impersonation Attack:* The attacker sends his fake parameter $T_i'$ through the public channel instead of the real parameter $T_i$ to implement the attack of the simulated CS. According to the description of step 3 of the authentication phase, when the IoT device receives the information $T_i$ from the CS, it will use $A_i^*$ to decrypt $DEC(T_i)_{A_i^*} = (s_i^*, w_i^*, T_2^*)$ to get the parameters $s_i^*$, $w_i^*$ and $T_2^*$. Then it computes $w_i' = h(Pid_i \mid\mid e_i)$ and compares it with $w_i$ to verify the identity of the CS. The session is terminated if the verification result is negative. Therefore, the adversary cannot simulate the CS through its own $T_i'$.

*7) Stolen Device Attack:* In this protocol, the CS does not store any parameters, including the password or *sk* of the IoT device, so the attacker cannot calculate the *sk* even if it is stolen. When the IoT device is attacked, since the *sk* depends on the random number $e_i$ and the random number $q_i$ in $s_i = q_i \oplus C_k$, and they are only generated during the authentication process with the IoT device and the CS, the attacker cannot calculate the *sk*.

TABLE II
COMPARISON WITH OTHER PROTOCOLS

| Security Requirements | [3] | [22] | [23] | [24] | [25] | Proposed |
|---|---|---|---|---|---|---|
| Perfect forward secrecy | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Device anonymity | × | ✓ | × | × | × | ✓ |
| Mutual authentication | × | ✓ | × | ✓ | ✓ | ✓ |
| Replay attack | ✓ | × | × | ✓ | ✓ | ✓ |
| Device impersonation Attack | ✓ | ✓ | × | ✓ | ✓ | ✓ |
| Server impersonation attack | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Stolen device attack | × | ✓ | ✓ | × | × | ✓ |
| DoS attack | ✓ | × | × | ✓ | ✓ | ✓ |
| Known key secrecy | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Formal security analysis | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Session key security | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

TABLE III
COMPUTATION COST ANALYSIS

| protocol | computation cost | | | |
|---|---|---|---|---|
| | device | server | total cost | estimated time |
| [2] | $5T_p + 5T_h$ | $5T_p + 6T_h$ | $10T_p + 11T_h$ | 13.032 ms |
| [3] | $3T_p + 4T_h$ | $4T_p + 5T_h$ | $7T_p + 9T_h$ | 9.242 ms |
| [22] | $4T_p + 3T_h$ | $4T_p + 4T_h$ | $7T_p + 8T_h$ | 9.150 ms |
| [23] | $11T_p + 10T_h$ | $3T_p + 8T_h$ | $14T_p + 18T_h$ | 18.484 ms |
| Proposed | $5T_h + 1T_{en} + 1T_d + 1T_p$ | $6T_h + 1T_{en} + 2T_d + 1T_p$ | $11T_h + 2T_{en} + 3T_d + 2T_p$ | 5.939 ms |

*8) DoS Attack:* In this scheme, we use timestamps to detect the freshness of information. And in the authentication process, every communication has a timestamp involved. During this phase, if the timestamps do not match, the session will be dropped, so an attacker cannot use any DoS attack. In addition, we still use random numbers to resist replay attacks, so our proposed protocol is resistant to DoS attacks.

*9) Known Key Secrecy:* Known key secrecy is when the old *sk* is obtained by an attacker and used to calculate the current *sk*. In this protocol, each session uses two random numbers $e_i$ and $q_i$, which are generated from the CS and the IoT device, respectively. The *sk* is obtained by calculating $sk = h(e_i \,||\, C_k \,||\, Q_i \,||\, R_i \,||\, s_i)$, so the old *sk* is not related to the current *sk*. Therefore, even if the attacker has the old *sk*, he cannot deduce the current *sk*.

## *E. Security Features Comparison*

Here, we compare the security features of our proposed protocol with with other relevant protocols [3], [22], [23], [24] and [25] in Table II. In summary, while achieving anonymity and mutual authentication, our proposed protocol is secure against a wide variety of attacks.

## V. PERFORMANCE ANALYSIS

In this section, we evaluate the computational cost and the communication cost of our proposed protocol.

## *A. Computational Cost*

It is well-known that the computation cost for XOR and concatenation operation are negligible in comparison with other operations such as hash function, symmetric key encryption/decryption operation, scalar point multiplication operation, etc. Therefore, we will not consider XOR and concatenation operations as computation cost in the comparison table.

During the authentication phase, the IoT device needs four hash operations, one-time symmetric encryption and elliptic curve encryption to get the message $\{N_i, T_1, E_i\}$ in step 1. The CS needs four hash operations, two symmetric decryption operations and one elliptic curve decryption operation in step 2 to obtain message $\{T_i, T_2\}$. The IoT device executes one decryption operation and four hash operations in step 3. Finally, CS executes the hash operation twice. To sum up, the device executes five hashes, one encryption, and one decryption; the CS executes six hashes, one encryption, and two decryptions. Table III shows the efficiency comparison of other protocols and ours.

The configuration of our experiments is RK3568 development board, 2GB ROM and KaihongOS Standard version 1.2.2.010. From the experimental results, we get $T_h$, $T_p$, $T_{en/d}$ to be 0.092, 1.202 and 0.5046, respectively, in milliseconds(ms).

- $T_h-$ the time it takes to execute a hash function
- $T_{en/d}-$ the time it takes to perform a symmetric encryption/decryption
- $T_p-$ the time it takes to perform one elliptic curve scalar multiplication operation

As can be seen from Table III and Fig. 10, the computational cost of the protocols in [2], [3], [22], and [23] are 13.032 ms, 9.242 ms, 9.150 ms, and 18.484 ms, respectively, while our proposed protocol's computational cost is 5.939 ms. We also performed the execution time cost of our protocol. Here, the CS is resource-rich, so in the experiments, we only count the time for the device to perform the first step, the third step and the total cost in the authentication process. The average values of the first step, the third step, and the total number of trials are 1.8141 ms, 0.7988 ms, and 2.6129 ms, respectively. So our

TABLE IV
COMMUNICATION COST OF EACH PROTOCOL

| protocol | communication cost unit (bits) | | | | | |
|---|---|---|---|---|---|---|
| | number of messages | message 1 | message 2 | message 3 | message 4 | total cost |
| [2] | 3 | $ID_i, P_1, P_2$ | $P_3, P_4, T_i$ | $V_i$ | – | 1760 |
| [3] | 3 | $ID_i, P_1, P_2$ | $P_3, P_4, T_i$ | $V_i$ | – | 1760 |
| [22] | 3 | $PID_i, P_1, P_2$ | $P_3, P_4, T_i$ | $V_i$ | – | 1760 |
| [23] | 4 | $A_7, A_9, A_{10}, T_1$ | $A_{11}, A_{12}, A_{15}, A_{16}, T_2$ | $A_8, A_{19}, A_{20}, T_3$ | $A_{17}, A_{21}$ | 4416 |
| Proposed | 3 | $N_i, T_1, E_i$ | $T_i, T_2$ | $MN_i, T_3$ | – | 662 |

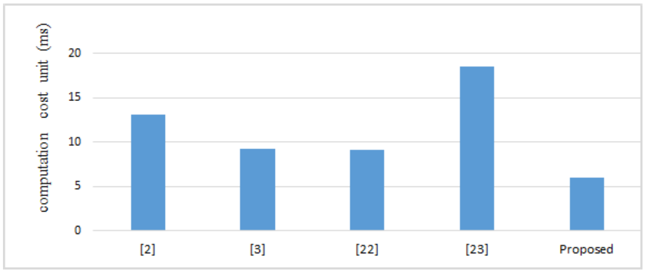


Fig. 10. The comparison of computational cost.

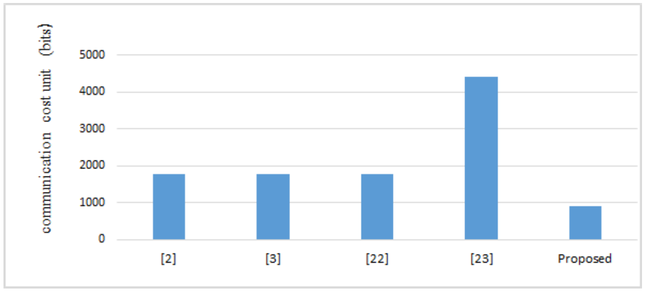


Fig. 11. The comparison of communication cost.

protocol has an absolute advantage for resource-constrained IoT device.

## B. Communication Cost

In the following, we compare the communication cost of our protocol with other protocols. We stipulate the length of the hash function, random number, timestamp, symmetric encryption/decryption, and group element in ECC as 160 bits, 32 bits, 32 bits, 128 bits and 320 bits. The server's key length is 1024 bits. The communication cost of our protocol is calculated as follows: the message $\{N_i, T_1, E_i\}$ needs (320+32+128) = 480 bits, the message $\{T_i, T_3\}$ needs (128+32) = 160 bits, and the message $\{MN_i, T_3\}$ needs (160+32) = 192 bits. In total, the cost of our protocol is 892 bits, similarly, the communication cost of the protocols of [2], [3], [22], and [23] are 1760, 1760, 1760, and 4416 bits, respectively. Table IV and Fig. 11 show that the communication cost is lower than other protocols.

## VI. CONCLUSION

There are many challenges in the authentication of IoT devices such as anonymity, lightweightness, and resistance to common attacks. Existing protocols are superior in some aspects, but they do not satisfy all the above-mentioned characteristics. This paper has proposed a lightweight authentication and key exchange protocol with anonymity for IoT devices with all the above features. We have also compared our protocol with other protocols. The results show that our proposed scheme is more lightweight and secure. In future work, we plan to enhance our protocol so that it can complete real-time authentication between servers and devices, and devices and devices according to different requirements.

## REFERENCES

[1] R. Amin, S. H. Islam, G. P. Biswas, M. K. Khan, and M. S. Obaidat, "Design and analysis of an enhanced patient-server mutual authentication protocol for telecare medical information system," *J. Med. Syst.*, vol. 39, no. 11, pp. 1–20, Nov. 2015.
[2] V. Sureshkumar, R. Amin, V. R. Vijaykumar, and S. R. Sekar, "Robust secure communication protocol for smart healthcare system with FPGA implementation," *Future Gener. Comput. Syst.*, vol. 100, pp. 938–951, Nov. 2019.
[3] S. Kalra and S. K. Sood, "Secure authentication scheme for IoT and cloud servers," *Pervas. Mobile Computing*, vol. 24, pp. 210–223, Dec. 2015.
[4] Q. Xie, W. Liu, S. Wang, L. Han, B. Hu, and T. Wu, "Improvement of a uniqueness-and-anonymity-preserving user authentication scheme for connected health care," *J. Med. Syst.*, vol. 38, no. 9, pp. 1–10, Sep. 2014.
[5] L. Xu and F. Wu, "Cryptanalysis and improvement of a user authentication scheme preserving uniqueness and anonymity for connected health care," *J. Med. Syst.*, vol. 39, no. 2, pp. 1–9, Feb. 2015.
[6] Y.-H. Chuang, N.-W. Lo, C.-Y. Yang, and S.-W. Tang, "A lightweight continuous authentication protocol for the Internet of Things," *Sensors*, vol. 18, no. 4, p. 1104, Apr. 2018.
[7] Y. Qiu and J. Lu, "Critical analysis of new protocols on lightweight authentication," in *Proc. 24th Asia–Pacific Conf. Commun. (APCC)*, Nov. 2018, pp. 325–330.
[8] M. D. Yu, M. Hiller, J. Delvaux, R. Sowell, S. Devadas, and I. Verbauwhede, "A lockdown technique to prevent machine learning on PUFs for lightweight authentication," *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 2, no. 3, pp. 146–159, Apr. 2016.
[9] H. Luo, G. Wen, J. Su, and Z. Huang, "SLAP: Succinct and lightweight authentication protocol for low-cost RFID system," *Wireless Netw.*, vol. 24, no. 1, pp. 69–78, 2018.
[10] A.-I. Radu and F. D. Garcia, "LeiA: A lightweight authentication protocol for CAN," in *Proc. 21st Eur. Symp. Res. Comput. Secur.* Heraklion, Greece: Springer, Sep. 2016, pp. 283–300.
[11] W.-L. Tai, Y.-F. Chang, and P.-L. Hou, "A three-factor anonymous authentication scheme for wireless sensor networks in Internet of Things environments," *J. Netw. Comput. Appl.*, vol. 21, no. 6, pp. 1014–1020, 2019.
[12] X. Li, J. Niu, S. Kumari, F. Wu, A. K. Sangaiah, and K.-K. R. Choo, "A three-factor anonymous authentication scheme for wireless sensor networks in Internet of Things environments," *J. Netw. Comput. Appl.*, vol. 103, pp. 194–204, Feb. 2018.
[13] M. Nikooghadam and H. Amintoosi, "Secure communication in CloudIoT through design of a lightweight authentication and session key agreement scheme," *Int. J. Commun. Syst.*, vol. 36, no. 1, p. e4332, Jan. 2023.
[14] M. Nakkar, R. Altawy, and A. Youssef, "Lightweight broadcast authentication protocol for edge-based applications," *IEEE Internet Things J.*, vol. 7, no. 12, pp. 11766–11777, Dec. 2020.
[15] R. Melki, H. N. Noura, and A. Chehab, "Lightweight multi-factor mutual authentication protocol for IoT devices," *Int. J. Inf. Secur.*, vol. 19, no. 6, pp. 679–694, Dec. 2020.

[16] A. Shahidinejad, M. Ghobaei-Arani, A. Souri, M. Shojafar, and S. Kumari, “Light-edge: A lightweight authentication protocol for IoT devices in an edge-cloud environment,” *IEEE Consum. Electron. Mag.*, vol. 11, no. 2, pp. 57–63, Mar. 2022.

[17] S. Qiu, D. Wang, G. Xu, and S. Kumari, “Practical and provably secure three-factor authentication protocol based on extended chaotic-maps for mobile lightweight devices,” *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 2, pp. 1338–1351, Mar./Apr. 2022.

[18] D. Dolev and A. C. Yao, “On the security of public key protocols,” *IEEE Trans. Inf. Theory*, vol. IT-29, no. 2, pp. 198–208, Mar. 1983.

[19] T. A. Team. *HLPSL Tutorial—The AVISPA Project*. Accessed: May 11, 2022. [Online]. Available: https://manualzz.com/doc/7191890/hlpsl-tutorial

[20] M. Abdalla, P.-A. Fouque, and D. Pointcheval, “Password-based authenticated key exchange in the three-party setting,” in *Proc. 8th Int. Workshop Public Key Cryptogr.* Les Diablerets, Switzerland: Springer, 2005, pp. 65–84.

[21] C.-C. Chang and H.-D. Le, “A provably secure, efficient, and flexible authentication scheme for ad hoc wireless sensor networks,” *IEEE Trans. Wireless Commun.*, vol. 15, no. 1, pp. 357–366, Jan. 2016.

[22] S. Kumari, M. Karuppiah, A. K. Das, X. Li, F. Wu, and N. Kumar, “A secure authentication scheme based on elliptic curve cryptography for IoT and cloud servers,” *J. Supercomput.*, vol. 74, no. 12, pp. 6428–6453, Dec. 2018.

[23] K.-H. Wang, C.-M. Chen, W. Fang, and T.-Y. Wu, “A secure authentication scheme for Internet of Things,” *Pervasive Mobile Comput.*, vol. 42, pp. 15–26, Dec. 2017.

[24] A. K. Das, M. Wazid, A. R. Yannam, J. J. P. C. Rodrigues, and Y. Park, “Provably secure ECC-based device access control and key agreement protocol for IoT environment,” *IEEE Access*, vol. 7, pp. 55382–55397, 2019.

[25] S. Li, T. Zhang, B. Yu, and K. He, “A provably secure and practical PUF-based end-to-end mutual authentication and key exchange protocol for IoT,” *IEEE Sensors J.*, vol. 21, no. 4, pp. 5487–5501, Feb. 2021.

**Daojing He** (Member, IEEE) received the B.Eng. and M.Eng. degrees from the Harbin Institute of Technology, China, in 2007 and 2009, respectively, and the Ph.D. degree from Zhejiang University, China, in 2012, all in computer science. He is currently a Professor with the School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, China. His research interests include networks and systems security. He is on the editorial board of some international journals, such as IEEE NETWORK.

**Yanchang Cai** was born in 1997. He is currently pursuing the master’s degree with the Software Engineering Institute, Nanchang University, Nanchang, China.

**Shanshan Zhu** was born in 1985. She is currently pursuing the Ph.D. degree with the Harbin Institute of Technology, Shenzhen, China.

**Ziming Zhao** was born in 1995. He is currently pursuing the master’s degree with the Software Engineering Institute, Nanchang University, Nanchang, China.

**Sammy Chan** (Senior Member, IEEE) received the B.E. and M.Eng.Sc. degrees in electrical engineering from the University of Melbourne, Australia, in 1988 and 1990, respectively, and the Ph.D. degree in communication engineering from the Royal Melbourne Institute of Technology, Australia, in 1995. Since December 1994, he has been with the Department of Electrical Engineering, City University of Hong Kong, where he is currently an Associate Professor.

**Mohsen Guizani** (Fellow, IEEE) is currently a Professor of machine learning and the Associate Provost with the Mohamed bin Zayed University of Artificial Intelligence (MBZUAI), Abu Dhabi, United Arab Emirates. Previously, he worked at different institutions in the USA. His research interests include applied machine learning and artificial intelligence, the Internet of Things, smart city, and cybersecurity. He was listed as a Clarivate Analytics Highly Cited Researcher in Computer Science in 2019, 2020, and 2021.