A Secure Anonymous D2D Mutual Authentication and Key Agreement Protocol for IoT Environments

Rahman Hajian
Islamic Azad University South Tehran Branch
Abbas Haghighat
Islamic Azad University of Safashahr
S.Hossein Erfani (h_erfani@azad.ac.ir)
Islamic Azad University https://orcid.org/0000-0002-7893-4191

Research Article

Keywords: Internet of Things (IoT), Device to Device (D2D) authentication, Key agreement, Key Compromise Impersonation (KCI) attack.

Posted Date: June 16th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-223151/v1

A Secure Anonymous D2D Mutual Authentication and Key Agreement Protocol for IoT Environments

[1]R. Hajian, [2]A. Haghighat, *[3]S. H. Erfani

[1] Department of Information Technology Engineering, South Tehran Branch, Islamic Azad University, Tehran, Iran
[2] Department of Electrical and Computer Engineering, Safashahr Branch, Islamic Azad University, Safashahr, Iran
[3]Department of Computer Engineering, South Tehran Branch, Islamic Azad University, Tehran, Iran

*Corresponding: h_erfani@azad.ac.ir

Abstract

Internet of Things (IoT) is a developing technology in our time that is prone to security problems as it uses wireless and shared networks. A challenging scenario in IoT environments is Device-to-Device (D2D) communication that an authentication server as a trusted third-party, does not involve in the authentication and key agreement process. It is only involved in the process of allocating long-term secret keys and their update. A lot of authentication protocols have been suggested for such situations. This article demonstrated that three state-of-the-art related protocols failed to remain anonymous, insecure against key compromise impersonation (KCI) attack, and clogging attack. To counter the pitfalls of them, a new D2D mutual authentication and key agreement protocol is designed here. The proposed protocol is anonymous, untraceable, and highly secure. Moreover, there is no need for a secure channel to generate a pair of private and public keys in the registration phase.) Formal security proof and security analysis using BAN logic, Real-Or-Random (ROR) model, and Scyther tool showed that our proposed protocol satisfied security requirements. Furthermore, communication cost, computation cost, and energy consumption comparisons denoted our schema has better performance, compared to other protocols.

Keywords: Internet of Things (IoT), Device to Device (D2D) authentication, Key agreement, Key Compromise Impersonation (KCI) attack.

1. introduction

The promising prospect of 5G cellular networks and future-generation 6G networks and their wide applications in different areas make it hard to design them. Millions of people and devices are linked to each other with the Internet of Everything (IoE), often through wireless networks [1]. Therefore, they face many security risks. Different architectures have been proposed for IoT and IoE networks which consider communication infrastructures, communication range, computational power, and transmission capacity of IoT/IoE devices. Some include smart transportation systems, smart healthcare systems, internet of vehicles (IoV), electric vehicular network, distributed heterogeneous environment without surveillance, industrial applications, smart satellite communications, and UAV control [2].

Device-to-device (D2D) and machine-to-machine (M2M) communications have attracted great scholarly attention in this regard. Due to the lack of an authentication server as a trusted third party, this communication faces many challenges [3, 4]. In mutual key-agreement protocols, a third-party authentication server is used in registration and initialization phases to load credentials, long keys, and other secret parameters. Two IoT devices can identify each other and share a secure session key without a trusted authority (TA). However, data transmitted in the public channels can be eavesdropped on by an adversary who performs some attacks such as violation of confidentiality, location forgery attack, user/devise impersonation, node compromise attack, and geographical position revealing. It can also breach anonymity and untraceability [5].

Alzahrani et al. [6] proposed an ECC-based key-agreement protocol for D2D communications in IoV networks that does not need a TA in the authentication and key-agreement phase. They analyzed and revealed security pitfalls of protocols by Islam and Biswas [7] and Mandal et al. [8]. In this article, we demonstrate that the protocol proposed by Alzahrani et al. [6] suffers from some problems in design and is insecure against insider attacks, key compromise attack, and fails to provide anonymity. On the other hand, the proposed protocol by

Li et al. [9] for D2D-based communication is elaborated here and is shown to be vulnerable against key compromise attack and replay attack. Furthermore, it does not support anonymity and untraceability. Also, Chaudhry et al.' protocol [10] cannot counter the clogging attack. Thus, a novel protocol for D2D communications without the presence of a trusted third party to improve protocols by Alzahrani et al. [6], Chaudhry et al. [10], and Li et al. [9] is proposed here. The proposed protocol is highly efficient and secure. Also, it does not need a private channel in the registration phase for generating private-public keys. It also provides key update capability.

1.1. Motivation

Secure and effective interaction in Device-to-Device (D2D) and Machine Type Communication (MTC), without involving a trusted authority, is essential for 5G, 6G, and its application. Exclusion of a trusted authority from authentication and key-agreement process may decrease delay and overhead, but may also bring about security challenges and impaired resistance against known attacks. This article offers a mutual authentication protocol between two IoT devices that are anonymous and untraceable while being effective and secure, making it a proper model for resource-restrained IoT devices. Here, IoT devices communicate with the trusted server only when they register or update the password in the network. Moreover, the private key is calculated by the IoT device and a single point failure problem is resolved.

1.2. Contribution

We analyzed protocols by Alzarani et al. [6], Chaudhry et al. [10], and Li et al. [9] in terms of security issues. Then, we proposed a mutual key-agreement protocol that includes four phases: (i) initialization, (ii) registration and long secret key generation, (iii) authentication and key agreement, (iv) public and private keys updating. The contributions of this article are listed below:

I) A two-party authenticated key agreement protocol is proposed to solve the key escrow problem without the involvement of an online Trusted Third-Party (TTP) authentication server.
II) Mutual authentication phase is done without a TA, and registration and the key updating phase are done in an insecure channel.
III) Three State-of-the-art related works and their security flaws are investigated.
IV) Strict formal security verification is achieved using the Scyther tool and BAN logic in the proposed protocol.
V) Resistance to different kinds of attacks is elaborated with the help of informal security verification.
VI) Evaluation of network performance, including communication/computation overheads and energy consumption, demonstrates that our proposed protocol is more effective compare to other protocols.

1.3. Organization

The rest of the article is organized as follows: Section 2 provides a review of the literature. Section 3 discusses the shortcomings of protocols proposed by Alzahrani et al., Chaudhry et al., and Li et al. Our proposed protocol is presented in section 4. The security and efficiency of our protocol are evaluated in section 5. Finally, the conclusion and future work are given in section 6.

2. Literature review

Amin et al. [11] proposed a mutual key-agreement protocol between users and the cloud server for distributed and mobile cloud environments. They used bilinear pairing in the authentication phase which is not suitable for D2D communications because of excessive overhead. A lightweight authentication protocol based on hash and XOR operations for wearable devices was introduced by Das et al. [12]. But the protocol is not secure against desynchronization and offline guessing attacks. In [13], a protocol was designed for IoT environments that supports anonymity but allows forged nodes to penetrate into the network. Wu et al. [14] offered a mutual key agreement protocol for smart power grids to be applied on smart meter and smart service provider, but it is insecure against known session-specific temporary information attack. D2D communications are significant

used in vehicular ad-hoc networks, especially vehicle-to-vehicle (V2V) communication. Li et al. [9] presented an identity-based V2V key agreement scheme. But it is not anonymous and provides no subtle mechanism to identify the vehicles. A mutual design was offered for body sensor networks by Shuai et al. [15]. A similar V2V protocol for IoT-based industrial environments was proposed by Lara et al. [16]. Islam [17] introduced a two-factor lattice-based encryption protocol for post-quantum cryptography to solve discrete logarithm problems and factoring problems in quantum environments. In order to reduce evaluation overhead and authenticate public keys, and to solve the key escrow problem, Islam and Biswas [7] designed an ECC-based key agreement protocol along with the self-certified public key for two-party communications. Mandal et al. [8] improved the security issues of Islam and Biswas' protocol. Nevertheless, Alzahrani et al. [6] showed that both protocols by Islam-Biswas [7] and Mandal et al. [8] suffered from key compromise impersonation attacks and do not ensure the anonymity of IoT devices. Chaudhry et al. [10] improved Das et al.'s [18] protocol (LACKA-IoT) and proposed iLACKA-IoT to access IoT-based cloud systems for D2D communication. They showed that LACKA-IoT was insecure to man-in-the-middle attack and device impersonation attack. Table 1 has provided explanations for previous methods and features.

Table 1. Summary of the related works

protocol	year	Features and good points	Weak points
Alzahrani et al. [6]	2020	ECC-based, D2D communication, review previous work, self-certificate, security analysis using BAN logic and random oracle model (ROM)	Lack of anonymity Vulnerable to key compromise impersonation (KCI) attack
Islam and Biswas [7]	2015	ECC-based, D2D communication, review previous work, self-certified public keys, security analysis using AVISPA tool and ROM	Lack of anonymity and mutual authentication, vulnerable to KCI, replay, and clogging attacks
Mandal et al. [8]	2018	ECC-based, user-to-user interaction, review pervious work, self-certificate, security analysis using AVISPA tool and BAN logic	Lack of anonymity Vulnerable to KCI attack
Li et al. [9]	2019	ECC-based, vehicular ad hoc networks (VANET), vehicle-to-vehicle (V2V), security analysis using random oracle model	Lack of anonymity and mutual authentication, vulnerable to replay, clogging, and KCI attacks
Chaudhry et al. [10]	2020	ECC-based, IoT based sensor cloud systems, D2D, review pervious work, ROR model	Lack of anonymity and forward secrecy, vulnerable to clogging attack
Amin et al. [11]	2016	bilinear pairing, user-to-cloud communication, review pervious work, password renewal phase, security analysis using AVISPA tool	High computation and storage cost, poor scalability, vulnerable to offline guessing attack and insider attack
Das et al. [12]	2017	lightweight cryptography, wearable devices, user-to-wearable device interaction, dynamic wearable device addition, security analysis using real-or-random (ROR) model and AVISPA tool	Lack of forward secrecy Vulnerable to desynchronization attack, and offline guessing attack,
Simplicio Jr et al. [13]	2016	ECC-based, review pervious work, combination of strengthened-Menezes-Qu-Vanstone (SMQV) with implicit certificates	Lack of anonymity and formal security analysis, vulnerable to KCI, impersonation, and replay attacks
Wu et al. [14]	2019	ECC-based, smart grid architecture, ECC, Random Oracle, ProVerif tool, Simulation, smart meter-to-service provider	Vulnerable to KCI attack, replay attack, and insider attack
Shuai et al. [15]	2020	ECC-based, wireless body area networks, client-to-application provider interaction, random oracle model	Vulnerable to KCI attack
Lara et al. [16]	2020	Lightweight operation, M2M communications (sensor-gateway), industrial internet of things (IIoT), security analysis using AVISPA and BAN logic	Vulnerable to desynchronization attack and secret leakage attack
Das et al.'s [18]	2019	ECC-based, smart device (D2D), device access method, ROR model, AVISPA tool, simulation	Lack of anonymity, vulnerable to impersonation attack and man-in-the-middle (MITM) attack

In this article, we analyze the protocols by Alzahrani et al. [6], Li et al. [9], and Chaudhry et al. [10] to demonstrate their security deficiencies. Our proposed protocol features an acceptable computational and communication overhead which, considering the real-time nature of D2D systems and resource scarcity, greatly improves performance. Table 2 summarizes symbols and corresponding definitions used in this article.

Table 2. Symbols and their definitions

Symbol	Description	Symbol	Description
TA	Trusted Authority	K_{pub}, K_{pri}	Public and Private key pair of TA
D_i	i^{th} IoT Device	d_i, Q_i	Public and Private key pair of D_i
ID_i	Identity of D_i	x, y	Random nonce chosen by D_i and D_j
TID_i	Temporal identity computed by D_i	t_X	Current timestamps (X=1, 2)
p	A large prime number	$\mathcal{A}$	Adversary
F_p	A prime finite field	SK_{ij}	Session Key between D_i and D_j
Z_p^*	A set of prime numbers $\{1, \dots, p-1\}$ $(p > 3)$	ΔT	Maximum allowable transmission delay
G	Base point over E/F_p	$\oplus$, $\|$	Bitwise XOR, Concatenation operation
$E(a,b)$ $/F_p$	An elliptic curve over prime field F_p defined by non-singular elliptic curve equation: $y^2 = x^3 + ax + b\ (mod\ p)$ a, b are constants. $x, y, a, b \in F_p$ and $4a^2 + 27b^2 (mod\ p) \neq 0$.	$h(\cdot)$	One-way and collision-resistant and cryptographic hash function defined by: $h\{0,1\}^* \rightarrow h\{0,1\}^l$ with arbitrary length inputs and fixed length outputs.

3. Detailed review

In this section, protocols proposed by Alzahrani et al. [6], Chaudhry et al [10], and Li et al [9] are discussed in detail in terms of security against known attacks.

3.1. Threat model

In our model, IoT devices communicate through a public channel. The following security issues are concerned with the adversary:

I) The adversary $\mathcal{A}$ cannot distinguish xG and xyG.
II) The adversary is unlikely to retrieve x from $h(x)$.
III) The adversary can eavesdrop, modify, remove, and duplicate messages transmitted in the public channel.
IV) The adversary can act as an insider to obtain secret parameters of other members in order to implement attacks.
V) The adversary can compromise all network entities and obtain all temporary as well as permanent credentials.

3.2. Review of Alzahrani et al.'s protocol

Alzahrani et al. [6] proposed an ECC-based key agreement protocol for IoT-based environments without TA intervention. The protocol included three phases: Initial system setup, registration, and authentication, which are described below.

3.1.1. Initial system setup phase. The server (a trusted third authority) calculates private key K_{pri} and public key $K_{pub} = K_{pri}G$ as $K_{pri} \in Z_p^*$. It then publishes system parameters $\{E/F_p, h(.), G, K_{pub}\}$ in the public space; where G is the base point, $h(.)$ is one-way hash function, and E/F_p is the Elliptical curve E on finite field F of order q.

3.1.2. Registration phase. In this phase, device D_i picks a unique identity ID_i and random number $r_i \in Z_p^*$ to be registered on a network. Then, it computes $R_i = h(ID_i \parallel r_i)G$ and transmits the message $< ID_i, R_i >$ to TA. Upon receiving the message, the trusted authority selects a random nonce $r_i^{TA} \in Z_p^*$ and calculates $R_i^{TA} = h(ID_i \parallel r_i^{TA})K_{pub} + R_i$, $f_i = [h(ID_i \parallel r_i^{TA}) + h(ID_i \parallel R_i^{TA})]K_{pri}$, and $Q_i = R_i^{TA} + H(ID_i \parallel R_i^{TA})K_{pub}$. After that, it sends $< ID_i, R_i^{TA}, f_i >$ via a secure channel to D_i, and publishes Q_i as the public key of D_i. Finally, D_i receives the second message and calculates private key d_i as $d_i = [f_i + h(ID_i \parallel r_i)]$. If $Q_i =? d_iG = R_i^{TA} + h(ID_i \parallel R_i)G$ is met, it stores pair of (d_i, Q_i) in its memory.

3.2.3. Authentication and key agreement phase. This phase is established between D_i and D_j and shown in Fig. 1.

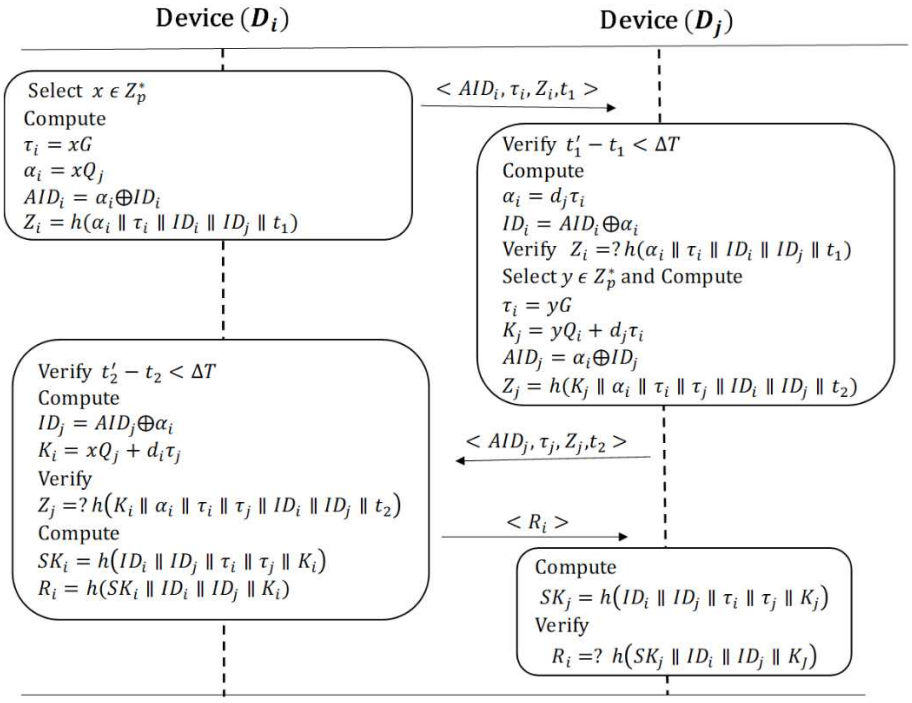

Fig. 1. Authentication process in Alzahrani et al.'s protocol

Step 1. D_i chooses a random number $x \in Z_p^*$ and calculates values of $\tau_i = xG$ and $\alpha_i = xQ_j$ where Q_j is the public key of D_j. Then, it calculates pseudo-ID $AID_i = \alpha_i \oplus ID_i$ and calculates Z_i as $Z_i = h(\alpha_i \parallel \tau_i \parallel ID_i \parallel ID_j \parallel t_1)$ where t_1 is the current timestamp of the message. Finally, the message $< AID_i, \tau_i, Z_i, t_1 >$ is sent to D_j.

Step 2. D_j receives the message $< AID_i, \tau_i, Z_i, t_1 >$ at time t_1'. If $t_1' - t_1 < \Delta T$, it verifies the freshness of the message. Then, it calculates $\alpha_i = d_j\tau_i$ and retrieves $ID_i = AID_i \oplus \alpha_i$. After that, it checks whether or not $Z_i =? h(\alpha_i \parallel \tau_i \parallel ID_i \parallel ID_j \parallel t_1)$ and verifies D_i. Otherwise, it immediately aborts the session. D_j then, picks the random nonce $y \in Z_p^*$ and timestamp t_2 and calculates the values of $\tau_i = yG$, $K_j = yQ_i + d_j\tau_i$, $AID_j = \alpha_i \oplus ID_j$, and $Z_j = h(k_j \parallel \alpha_i \parallel \tau_i \parallel \tau_j \parallel ID_i \parallel ID_j \parallel t_2)$. Finally, it transmits the message $< AID_j, \tau_j, Z_j, t_2 >$ to D_i.

Step 3. D_i checks freshness of the received message and retrieves $ID_j = AID_j \oplus \alpha_i$. Then, it calculates $K_i = xQ_j + d_i\tau_j$ and checks if $Z_j =? h(k_i \parallel \alpha_i \parallel \tau_i \parallel \tau_j \parallel ID_i \parallel ID_j \parallel t_2)$. If the condition holds, D_j is verified. D_i then calculates the session key SK_i as $SK_i = h(ID_i \parallel ID_j \parallel \tau_i \parallel \tau_j \parallel K_i)$ and generates message $< R_i >$ as $R_i = h(SK_i \parallel ID_i \parallel ID_j \parallel K_i)$ to transmit to D_j.

Step 4. Upon the reception of the message, D_j calculates session key as $SK_j = h(ID_i \parallel ID_j \parallel \tau_i \parallel \tau_j \parallel K_j)$ and checks whether or not $R_i =? h(SK_j \parallel ID_i \parallel ID_j \parallel K_j)$. If not, it immediately terminates the session; otherwise, $SK = SK_i = SK_j$ is verified.

3.2. Weakness of Alzahrani et al.'s protocol

In this section, we will discuss the weaknesses in Alzahrani et al.'s protocol.

3.2.1. Problem 1. In the authentication phase, AID_i is calculated as $AID_i = \alpha_i \oplus ID_i$, where $\alpha_i = xQ_j$, Q_j is the public key of D_j, x is a point on elliptical curve, and ID_i is the unique identity of D_j. In practice, XOR operation may not be applied on them. Thus, $AID_i = h(\alpha_i) \oplus ID_i$ is suggested instead.

3.2.2. Problem 2. D_i has the value of ID_j while generating $Z_i = h(\alpha_i \parallel \tau_i \parallel ID_i \parallel ID_j \parallel t_1)$. This decreases protocol scalability. Moreover, saving all IDs increases the risk of node compromise attack because once the adversary seizes a node's ID, he can fetch other IDs and violate protocol anonymity. On the other hand, because of node mobility, it is not efficient to stores all IDs in the memory.

3.2.3. Insecurity against privileged insider attack. An insider can get the ID of a device and reach the ID of all other connected devices, as explained below.

Step 1. The insider agent picks the random nonce $x^* \in Z_p^*$ and calculates values of $\tau_i^* = x^*G$, $\alpha^* = x^*Q_j$, and $AID_i^* = \alpha_i^* + ID_i^*$. Then, it generates $Z_i^* = h(\alpha_i^* \parallel \tau_i^* \parallel ID_i^* \parallel ID_j \parallel t_1)$ and transmits the message $< AID_i^*, Z_i^*, \tau_i^*, t_1 >$ to D_j.

Step 2. D_j checks for verifies the freshness of the message and calculates $AID_j = \alpha_i^* \oplus ID_j$, and sends it to D_i^*.

Step 3. Upon the reception of AID_j, D_i^* obtains $ID_j = \alpha_i^* \oplus AID_j$. D_i^* eavesdrops all transmissions from D_j to other devices and obtains $AID_j^{new} = \alpha_j^{new} \oplus ID_j$. Applying XOR operation, it achieves α_j^{new}. Hence, protocol secrecy is violated.

Step 4. Once the insider obtains α_j^{new}, he applies XOR operation on $AID_j^{new} \oplus \alpha_j^{new}$ to obtain ID_j^{new} of the device. Thus, Alzahrani et al.'s protocol loses anonymity and untraceability. To solve this problem, a temporary ID is suggested to be used for communications.

3.2.4. Key Compromise Impersonation (KCI) attack. Once a key is compromised by the adversary, the replay attack could run as follows.

Step 1. Similar to the insider attack, the adversary $\mathcal{A}$ obtains the key d_i and IDs of nodes. $\mathcal{A}$ then selects $x^A \in Z_p^*$ and calculates $\tau_i^A = x^A G$, $\alpha_i^A = x^A Q_i$, $AID_i^A = \alpha_i^A \oplus ID_i$, $Z_i^A = h(\alpha_i^A \parallel \tau_i^A \parallel ID_i \parallel ID_j \parallel t_1)$, and transmits the message $< AID_i^A, \tau_i^A, Z_i^A, t_i >$ to D_j.

Step 2. D_j receives the first message $AID_i^A \oplus \alpha_i^A$ and obtains ID_i. It checks if $Z_i^A =? h(\alpha_i^A \parallel Z_i^A \parallel ID_i \parallel ID_j \parallel t_1)$ and authenticates ID_i. Thus, the adversary $\mathcal{A}$ can easily forge a device. Hence, their protocol is vulnerable against key compromise impersonation attack.

3.3. Review of Li et al.'s protocol

The proposed protocol by Li et al. includes three phases as described below.

3.3.1 Initial setting phase. TA creates a finite cyclic additive group $\mathbb{G}$ with generator G and chooses a big prime number p. Then, it selects the private key $K_{pri} \in Z_p^*$ and calculates the public key $K_{pub} = K_{pri}G$, and chooses hash functions $h_1(.), h_2(.), h_3(.)$ and publishes parameters $< \mathbb{G}, p, G, K_{pub}, h_1(.), h_2(.), h_3(.) >$.

3.3.2. Registration phase. The vehicle V_i selects an ID_i and sends it to TA through a secure channel. TA chooses a random number $r_i \epsilon Z_p^*$ and calculates $R_i = r_i G$. It then calculates $h_i = h_1(ID_i \parallel R_i)$ and $d_i = r_i + h_i K_{Pri} (mod\ p)$ and transmits the pair (d_i, R_i) to V_i through a secure channel. V_i checks whether or not $d_i G = ? R_i + h_1(ID_i \parallel R_i) K_{pub}$. If the condition holds, it accepts $Q_i = d_i G$ as its public key.

3.3.3 Key agreement phase. This phase occurs between the vehicles V_i and V_j without an intervention of a third-party server. This phase is demonstrated in Fig. 2.

Step 1. V_i selects a random number $x \epsilon Z_p^*$ and calculates $\tau_i = xG$, and sends $\{ID_i, R_i, \tau_i\}$ to V_j .

Step 2. Upon reception of the message, V_j selects a random number $y \epsilon Z_p^*$ and $\tau_i = yG$ and sends the message $< ID_j, R_j, \tau_j >$ to V_i. At the same time, it obtains the public key of V_i as $Q_i = R_i + h_1(ID_i \parallel R_i) K_{pub}$ and calculates $K_j = h_2(d_j Q_i) y \tau_i$ and the session key $SK_j = h_3(ID_i \parallel ID_j \parallel K_j \parallel yQ_i \parallel d_j \tau_i)$.

Step 3. V_i receives the message $< ID_j, R_j, \tau_j >$ and obtains public key $Q_j = R_j + h_1(ID_j \parallel R_j) K_{pub}$, and calculates $K_i = h_2(d_i Q_j) x \tau_j$ and obtains the session key $SK_i = h_3\ (ID_i \parallel ID_j \parallel K_i \parallel xQ_j \parallel d_i \tau_j)$.

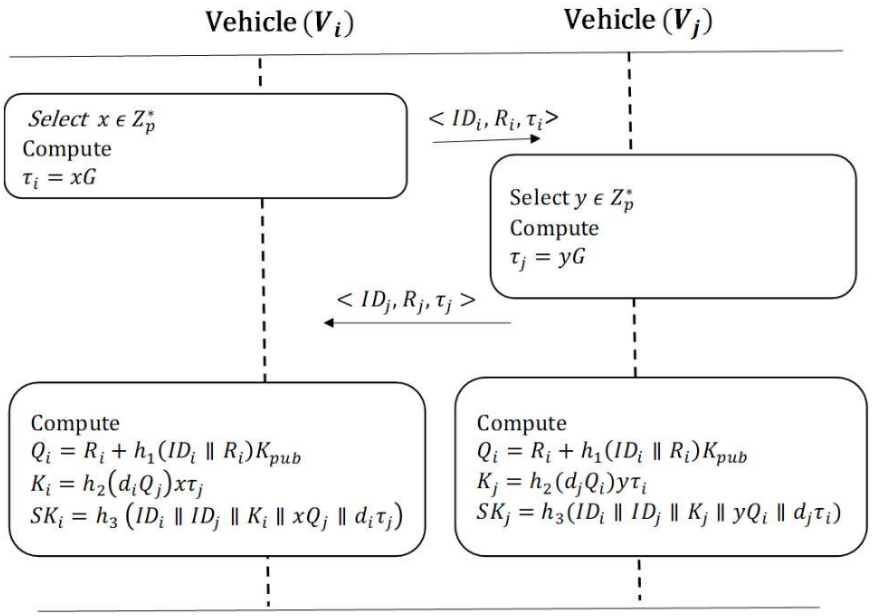

Fig. 2. Authentication process in Li et al.'s protocol

3.4. Weakness of Li et al.'s protocol

The following summarizes security issues of Li et al.'s protocol.

3.4.1. Lack of mutual authentication. V_i and V_j focus on session key agreement and fail to authenticate each other. Thus, the adversary $\mathcal{A}$ can easily insert forged messages into the network.

3.4.2. Clogging attack. It is a subclass of DoS attacks wherein the adversary clogs the receiver and wastes its communication and computation sources in an attempt to paralyze the receiver [8]. In Li et al.'s protocol, the adversary runs the clogging attack as follows:

Step 1. Adversary $\mathcal{A}$ captures the first message $< ID_i, R_i, \tau_i >$ in key agreement phase. $\mathcal{A}$ then selects a random nonce $x^A \epsilon Z_p^*$ and calculates $\tau^A = x^A G$, and transmits the message $< ID_i, R_i, \tau^A >$ to V_j.

Step 2. Upon receiving the message, V_j selects a random nonce y and calculates $\tau_j = yG$, public key $Q_i = R_i + h_1(ID_i \parallel R_i)K_{pub}$, and session key $SK_j^A = h_3(ID_i \parallel ID_j \parallel K_j \parallel yQ_i \parallel d_j\tau^A)$. Then, it transmits the message $< ID_j, R_j, \tau_j >$ to $\mathcal{A}$.

Step 3. Adversary $\mathcal{A}$ transmits the message $< ID_j, R_j, \tau^A >$ to V_i with only calculating $\tau^A = xG$.

Step 4. V_i receives the message and calculates $Q_j = R_j + h_1(ID_j \parallel R_j)K_{pub}$, $K_i = h_2(d_iQ_j)x\tau^A$, and session key $SK_i^A = h_3(ID_i \parallel ID_j \parallel K_i \parallel xQ_j \parallel d_i\tau^A)$.

This attack desynchronizes the agreed session key between the agents, i.e., $SK_j^A \neq SK_i^A$. Adversary $\mathcal{A}$ performs a multiplication operation of ECC and makes the two agents V_i and V_j run 11 scalar multiplication, 2 point addition, and 6 hash function. This causes a huge loss of time and costs that is only recognized by desynchronized session key after transmission of encrypted data.

3.4.3. Lack of anonymity and untraceability. In Li et al.'s protocol, vehicle IDs are transmitted clearly and an adversary can easily track and store messages to use them for its own purposes. Hence, their protocol loses anonymity and untraceability.

3.4.4. Replay attack. Transmitted messages include x and y nonces but have no timestamp. Because of traffic constraints of vehicles and IoT devices, they cannot store all the nonces. Thus, an adversary can capture the first $< ID_i, R_i, \tau_i >$ or the second $< ID_j, R_j, \tau_j >$ message and transmit it later to be verified by the receiver. It is, therefore, vulnerable to replay attack.

3.4.5. Key compromise impersonation. If private keys d_i and d_j are revealed, the adversary can easily forge a vehicle as follows:

Step 1. The adversary $\mathcal{A}$ obtains d_i, ID_i and R_i and uses them to obtain public key Q_i as $Q_i = R_i + h_1(ID_i \parallel R_i)K_{pub}$. Then, $\mathcal{A}$ selects a random nonce $x^A \epsilon Z_p^*$ and calculates $\tau^A = x^AG$, and transmits the message $< ID_i, R_i, \tau^A >$ to V_j.

Step 2. Upon receiving the message, V_j retrieves the public key Q_i because there are no changes in ID_i and R_i. Then, session keys SK_i and SK_j will be equal as follows: $SK_i^A = h_3(ID_i \parallel ID_j \parallel h_2(d_jQ_i)x^A\tau_j \parallel x^AQ_j \parallel d_i\tau_j) = h_3(ID_i \parallel ID_j \parallel h_2(d_iQ_j)y\tau^A \parallel yQ_i \parallel d_j\tau^A) = SK_j^A$. Therefore, once d_i is revelaed, the adversary can forge a node.

3.5. Analysis of Chaudhry et al.'s protocol

In Chaudhry et al.'s protocol, a TA selects ID_i and private key $d_i \epsilon Z_p^*$ and uses random nonce $l_i \epsilon Z_p^*$ to calculate the certificate C_i of private key Q_i, as $Q_i = d_iG, A_i = (d_i + l_i)G, C_i = K_{pri}(d_i + l_i)h(ID_i \parallel A_i) + d_i$ where K_{pri} is private key of TA. These are used in D2D authentication and key agreement between two IoT devices. An authentication and key agreement phase in this protocol is shown in Figure 3.

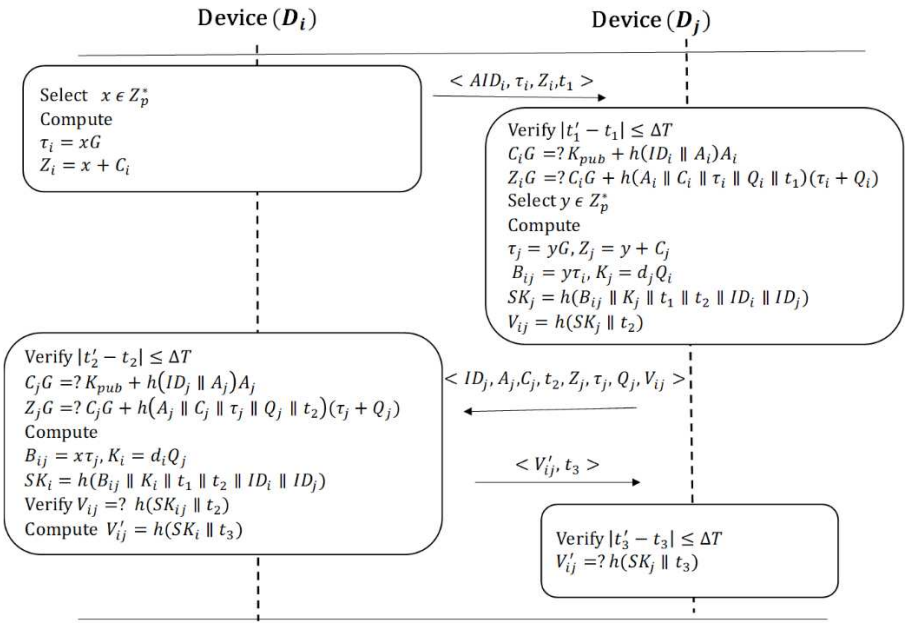

Fig. 3. Authentication process in Chaudhry et al.'s protocol

3.5.1. Lack of anonymity and untraceability. Since ID_i and ID_j for D_i and D_j devices are clearly transferred in the channel, the adversary can easily eavesdrop messages. Therefore, anonymity and untraceability are breached in this protocol.

3.5.2. Clogging Attack. Chaudhry et al.'s protocol is insecure against clogging attack as shown below.

Step 1. The adversary $\mathcal{A}$ captures the message $< ID_i, A_i, t_1, Z_i, \tau_i, Q_i >$ that consists the current timestamp t_1, $\tau_i = xG$, and $Z_i = x + C_i$, is transmitted by D_i in which $x \epsilon Z_p^*$. Then, $\mathcal{A}$ generates random nonce x^A and calculates $\tau_i^A = x^AG$, $\tau_i^{A''} = \tau_i^A + \tau_i$, and $Z_i^A = Z_i + x^A = (x + C_i + x^A)$. After that, the adversary transmits the fake message $< ID_i, A_i, t_1, Z_i^A, \tau_i^{A''}, Q_i >$ to D_j.

Step 2. Upon reception of the message, D_j checks its freshness $Z_i^AG \,?= \tau_i^{A''} + K_{pub} + h(ID_i \| A_i)A_i + Q_i$ to ensure:

$$Z_i^AG = (x + C_i + x^A)G = x^AG + xG + C_iG = x^AG + xG + K_{pri}G + h(ID_i \| A_i)(d_i + l_i)G + Q_i = \tau_i^A + \tau_i + K_{pub} + h(ID_i \| A_i)(d_i + l_i)G + Q_i = \tau_i^{A''} + K_{pub} + h(ID_i \| A_i)A_i + Q_i \quad (1)$$

The condition is met and the adversary can deceive D_j and breach message integration. This waste D_j resources because it will perform 4 ECC multiplication operations and 2 hash functions to transmit the message $< ID_j, A_j, t_2, Z_j, \tau_j, Q_j, V_{ij} >$ to D_i. Then, D_i will do some more calculations and realize that the session key is different, implying that clogging attack has been successful.

3.5.3. Lack of perfect forward secrecy. Once the adversary compromises d_i, C_i of D_i, s/he obtains agreed session keys in authentication phase. This is because C_i is fixed and retrievable in mutual authentication between D_i and D_j. The following elaborates on this attack:

Step 1. $\mathcal{A}$ eavesdrops and saves the messages $< ID_i, A_i, t_1, Z_i, \tau_i, Q_i >$ and $< ID_j, A_j, t_2, Z_j, \tau_j, Q_j, V_{ij} >$ from earlier sessions.

Step 2. $\mathcal{A}$ captures D_i to obtain long-term credentials d_i and $C_i = K_{pri}(d_i + l_i)h(ID_i \parallel A_i) + d_i$. These two parameters are fixed. The adversary needs to obtain $B_{ij} = y\tau_i = x\tau_j$ and $K = d_iQ_j = d_jQ_i$ to retrieve session keys from earlier sessions as $SK_{ij}^{old} = h(B_{ij}^{old} \parallel K \parallel t_1 \parallel t_2 \parallel ID_i \parallel ID_j)$.

Step 3. Adversary $\mathcal{A}$ uses Z_i^{old} to obtain x^{old}, and then calculates $K = d_iQ_j$, $x^{old} = Z_i^{old} - C_i$, and ${B_{ij}}^{old} = x^{old}\tau_j^{old}$. Thus, the session key from the earlier session is recalled. In the same way, the keys of the future sessions can be obtained. Hence, Chaudhry et al.'s protocol fails to support perfect forward/backward secrecy.

4. System description and proposed protocol

4.1 Network model

6G technology is not limited to cellular mobile networks and is aimed at expanding digital communications. However, improve effectiveness in the real-time communication and dense IoT networks requires reducing computational and communication overhead. However, adversaries and insiders can penetrate into the network due to the use of insecure channels and temporary connections for high network mobility. Thus, in addition to promoting performance, security, and privacy issues need to be considered.

The proposed model includes three major entities: IoT devices, Trusted authority (TA), and an adversary. IoT devices are 6G communication devices that communicate with their peers or far servers with no human involvement. In fact, they provide Machine Type Communication (MTC).

TA provides offline information for IoT devices. Considering scalability and widespread use of 6G networks, it is more optimal for devices to register online in their preferred network. These IoT devices include sensors embedded on smart vehicles, sensors in smart houses, smart health systems, UAVs, smart agriculture, and other single-hop or hierarchical networks. Fig 4. demonstrates the network model.

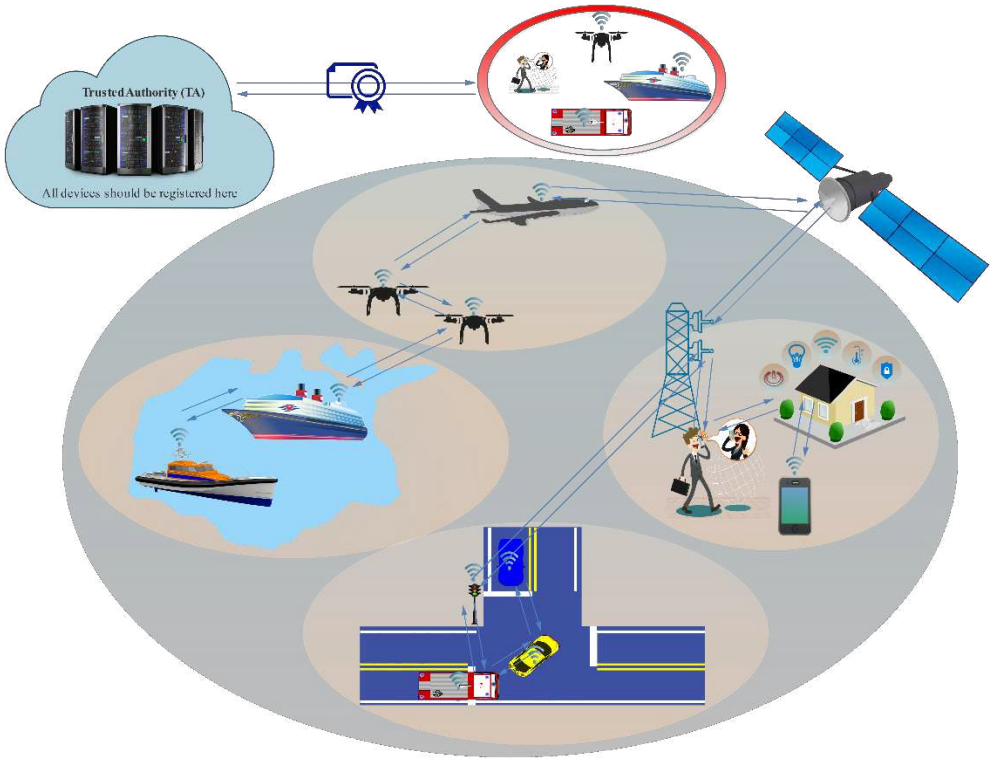

Fig. 4. Network model

4.2. Communication model

An IoT device may communicate with another device or a TA through channels including Bluetooth, Wi-Fi, ZigBee, cellular spectrum, optical fiber, etc. We analyze the most challenging communication scenario between two devices and TA-free authentication.

4.3. Proposed protocol

In this section, a mutual key agreement and authentication protocol for IoT devices without the intervention of a Trusted Authority (TA) is proposed. It includes four phases: initial system configuration, registration and key generation, authentication and key agreement, public and private keys updating. TA intervenes in initialization, the long-secret key generating, and updating phases to allocate device-specific public keys. To avoid key escrow problem, in case the TA's key is disclosed, the private keys of devices are not exposed. This is because each device separately calculates its private key. The proposed protocol uses ECC encryption, one-way hash function, and XOR operation. ECC-based encryption is advantageous for its high security compared to symmetric encryption and short key length compared to asymmetric encryption such as RSA. This increases computational performance and security features. All the phases in the protocol, except the first phase, are done in a public channel that is accessible to the invader. A summary of the first and second phases of the proposed protocol is demonstrated in Figure 5.

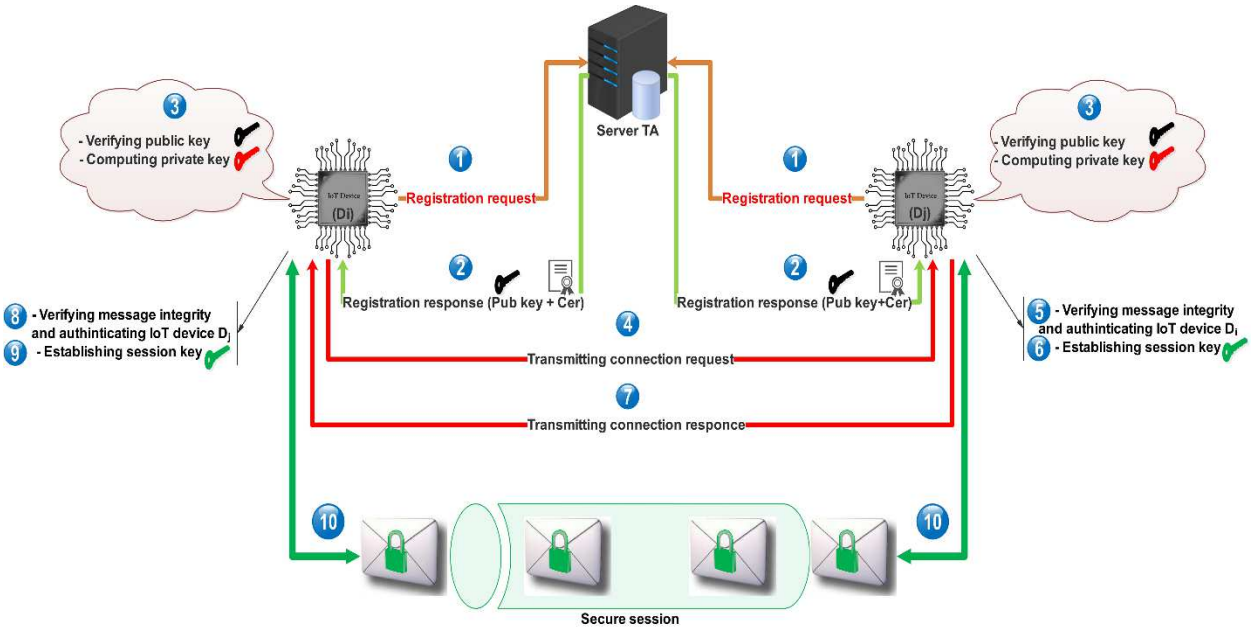

Fig. 5. An overview of the second and third phases of the proposed protocol

4.2.1. Initial system configuration

In this phase, TA selects Elliptic curve E on finite field F with big prime order p and collision resistant one-way hash function $h(.)$. TA, then selects the private key $K_{pri} \epsilon Z_p^*$ for itself and calculates the corresponding public key $K_{pub} = K_{pri}G$. It also selects random nonce $s \epsilon Z_p^*$ and assigns unique IDs to each IoT devices and stores $HID_x = h(ID_x \parallel s \parallel K_{pri}), sG$ in them. Finally, TA publishes public system parameters including $\{E/F_p, h(.), G, K_{pub}\}$.)

4.2.2. IoT device registration and key generation

This phase occurs between IoT device D_i and trusted server TA in a public channel as follows:

Step RKG1. D_i selects the random nonce $r_i \epsilon Z_p^*$ and unique ID_i.

It then calculates $R_i = r_i G$, masked identity $MID_i = ID_i \oplus h(r_i\, sG)$, and message authentication code $MAC_i = h(ID_i \parallel HID_i \parallel t_1)$. After that, it transmits message $< MID_i, R_i, MAC_i, t_1 >$ in timestamp t_1, to TA through a public channel.

Step RKG2. TA receives the message $< MID_i, R_i, MAC_i, t_1 >$, and retrieves $ID_i = MID_i \oplus h(sR_i G)$. It then checks the freshness of the message and computes $HID_i = h(ID_i \parallel s \parallel K_{pri})$ and verifies message integrity by checking $MAC_i \stackrel{?}{=} h(ID_i \parallel HID_i \parallel t_1)$. If the condition is met, it selects a random nonce $e_i \in Z_p^*$ and obtains values of $P_i = e_i K_{pub} + R_i$, $f_i = (e_i + P_i) K_{pri}$ and public key $Q_i = P_i + P_i K_{pub}$. Also, TA calculates $M_i = K_{pri} Q_i$ to obtain values of $N_i = M_i + h(ID_i)G$, $F_i = f_i \oplus h(ID_i \parallel R_i)$ and $MAC_{TA} = h(HID_i \parallel N_i \parallel F_i)$. Finally, the second message $< F_i, N_i, MAC_{TA} >$ is sent to D_i through a public channel. TA calculates $\xi_i = e_i \oplus h(R_i \parallel K_{pri})$ and stores ξ_i and R_i in its memory. It publishes P_i and public key Q_i in the public space.

Step RKG3. Upon receiving the message $< F_i, N_i, MAC_{TA} >$,D_i retrieves $f_i = F_i \oplus h(ID_i \parallel R_i)$ and calculates its private key d_i as $d_i = f_i + r_i$, then verifies $Q_i \stackrel{?}{=} d_i G$ to ensure accuracy of its calculations. After that, it retrieves $M_i = N_i - h(ID_i)G$ and Checks the validity of $MAC_{TA} \stackrel{?}{=} h(HID_i \parallel N_i \parallel F_i)$. I=If the equality test is satisfied, d_i and Q_i are considered to be legal pairs of private and public keys. The public key Q_i and $d_i G$ are equal, as below.

$$d_i G = (f_i + r_i)G = (\big[(e_i + P_i)K_{pri}\big]G + R_i = e_i K_{pub} + R_i + P_i\, K_{pub} = P_i + P_i K_{pub} = Q_i \qquad (2)$$

Thus, TA calculates and sends Q_i to the IoT device D_i without having access to private key d_i. This phase is briefly demonstrated in Fig 6.

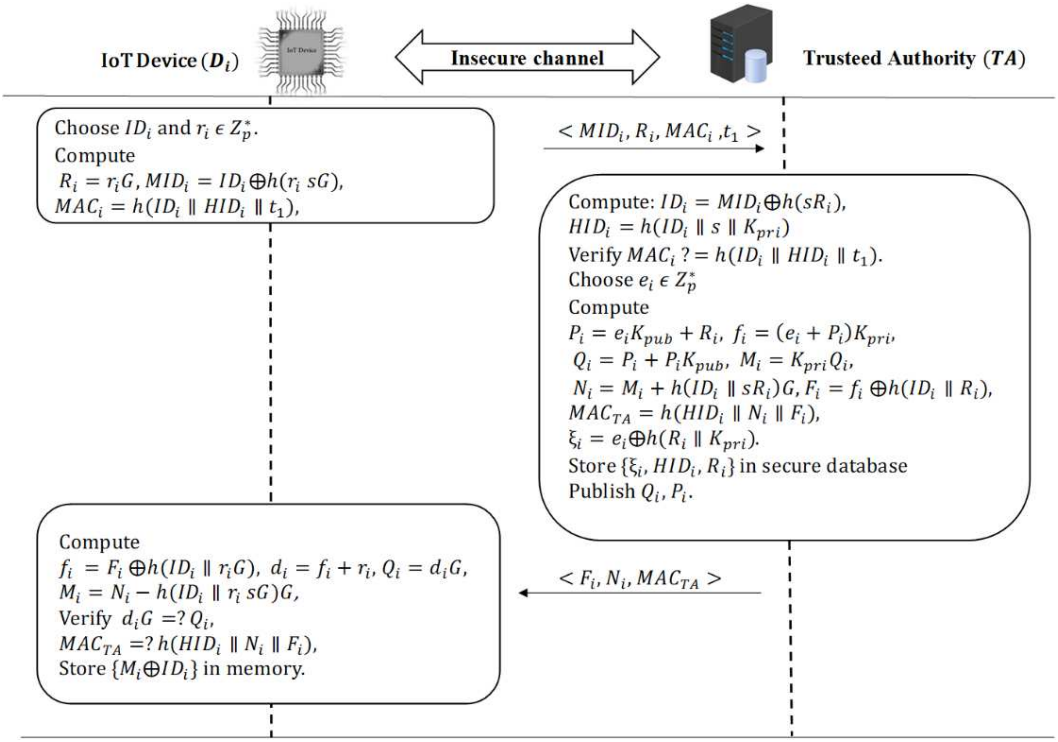

Fig. 6. IoT Device registration and key generation phase

4.2.3. Authentication and key agreement phase

Two IoT devices D_i and D_j need to authenticate each other and agree on a shared session key to establish a secure connection. Then, they encrypt their information by the session key SK_{ij} and publish that in the public channel. Authentication and key agreement are done in the public channel where adversaries may be present. This phase is described below in three steps.

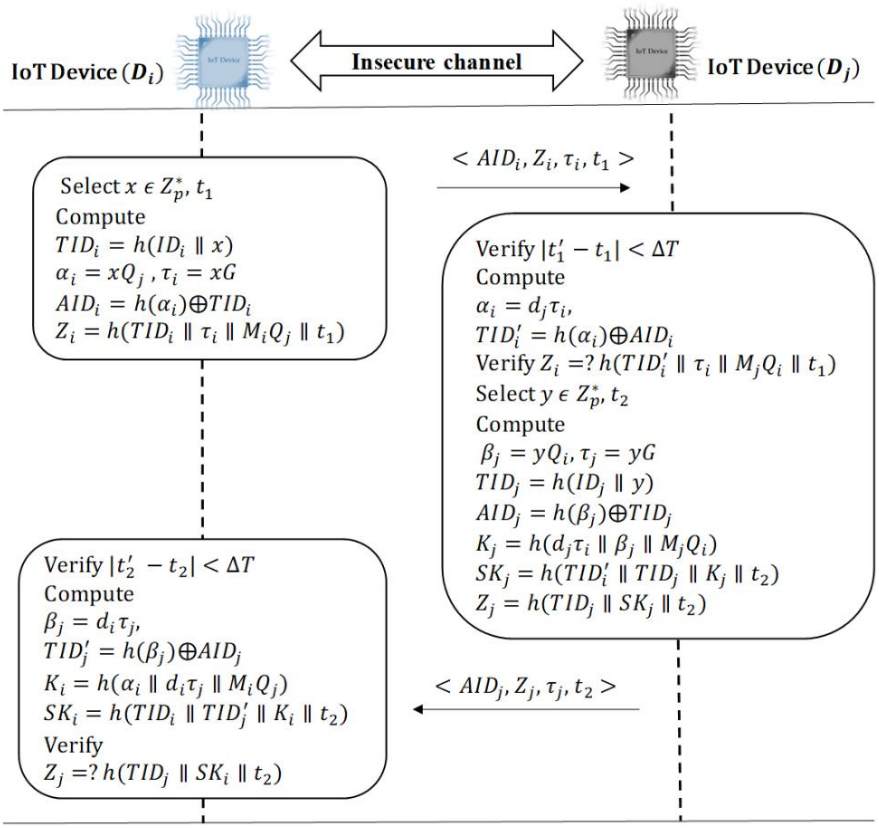

Fig. 7. Authentication and key agreement phase

Step AKA1: $D_i \rightarrow D_j$: $< AID_i, Z_i, \tau_i, t_1 >$

The IoT device D_i selects a temporary identity TID_i, a random nonce $x \in Z_p^*$, and current timestamp t_1. It then calculates $\alpha_i = xQ_j$, $\tau_i = xG$, $AID_i = h(\alpha_i) \oplus TID_i$, and $Z_i = h(TID_i \| \tau_i \| M_iQ_j \| t_1)$. Finally, it transmits the message $< AID_i, Z_i, \tau_i, t_1 >$to the IoT device D_j.

Step AKA2: $D_j \rightarrow D_i$: $< AID_j, Z_j, \tau_j, t_2 >$

Upon receiving the message $< AID_i, Z_i, \tau_i, t_1 >$, D_j checks its freshness by verifying the condition $|t_1' - t_1| < \Delta T$, where t_1' is the time of message delivery. If the condition is met, D_j calculates $\alpha_i = d_j\tau_i$ and $TID_i' = h(\alpha_i) \oplus AID_i$, where d_j is the private key of D_j. It then checks $Z_i =? h(TID_i' \| \tau_i \| M_jQ_i \| t_1)$ to verify the equality authentication test. If the condition is not met, it immediately aborts the session; otherwise, verifies the message and TID_i as a legal entity. D_j selects TID_j and random nonce $y \in Z_p^*$, and calculates $\beta_j = yQ_i$, $\tau_j = yG$, and $AID_j = h(\beta_j) \oplus TID_j$. After that, it calculates $K_j = h(d_j\tau_i \| \beta_j \| M_jQ_i)$ using public key of D_i, the value of M_j in its memory, secret nonce y, and τ_i of the first message. Next, it calculates temporary session key $SK_j = h(TID_i' \| TID_j \| K_j \| t_2)$ where t_2 is the current timestamp of message generated by D_j. It also calculates $Z_j = h(TID_j \| SK_j \| t_2)$ to preserve message integrity, and transmits the message $< AID_j, Z_j, \tau_j, t_2 >$ to D_i through a public channel.

Step AKA3: $D_i \leftarrow < AID_j, Z_j, \tau_j, t_2 >$

D_i receives the message $< AID_j, Z_j, \tau_j, t_2 >$ and verifies its freshness by checking $|t_2' - t_2| < \Delta T$. It then retrieves $\beta_j = d_i\tau_j$ and TID_j' as $TID_j' = h(\beta_j) \oplus AID_j$. Next, D_i calculates $K_i = h(\alpha_i \| d_i\tau_j \| M_iQ_j)$ using public key Q_j, stored value M_i, nonce x, and τ_j. Then, it calculates $SK_i = h(TID_i \| TID_j' \| K_i \| t_2)$ and $Z_j^* = h(TID_j \|$

$SK_i \parallel t_2$). D_i checks whether or not $Z_j =? Z_j^*$. If the condition holds, it authenticates D_j, and verifies $SK_i = SK_j$ as a secure session key; otherwise, it immediately aborts the session. Since $M_i = K_{pri}Q_i$ and $M_j = K_{pri}Q_j$ are delivered to D_i and D_j in registration phase, K_i and K_j are equal as $K_j = h(d_j\tau_i \parallel \beta_j \parallel M_jQ_i) = h(d_jxG \parallel \beta_j \parallel K_{pri}Q_jQ_i) = h(xQ_j \parallel yQ_i \parallel K_{pri}Q_iQ_j) = h(\alpha_i \parallel d_iyG \parallel M_iQ_j) = h(\alpha_i \parallel d_i\tau_j \parallel M_iQ_j) = K_i$. Eventually, $SK_i = SK_j$. This phase is briefly demonstrated in Fig 7.

4.2.4. Public and Private Keys updating phase

An IoT device may have to update private and public keys because of key expiration or disclosure. In our proposed protocol, the pair of private and public keys are updated through a public channel between the IoT device and TA, as follows.

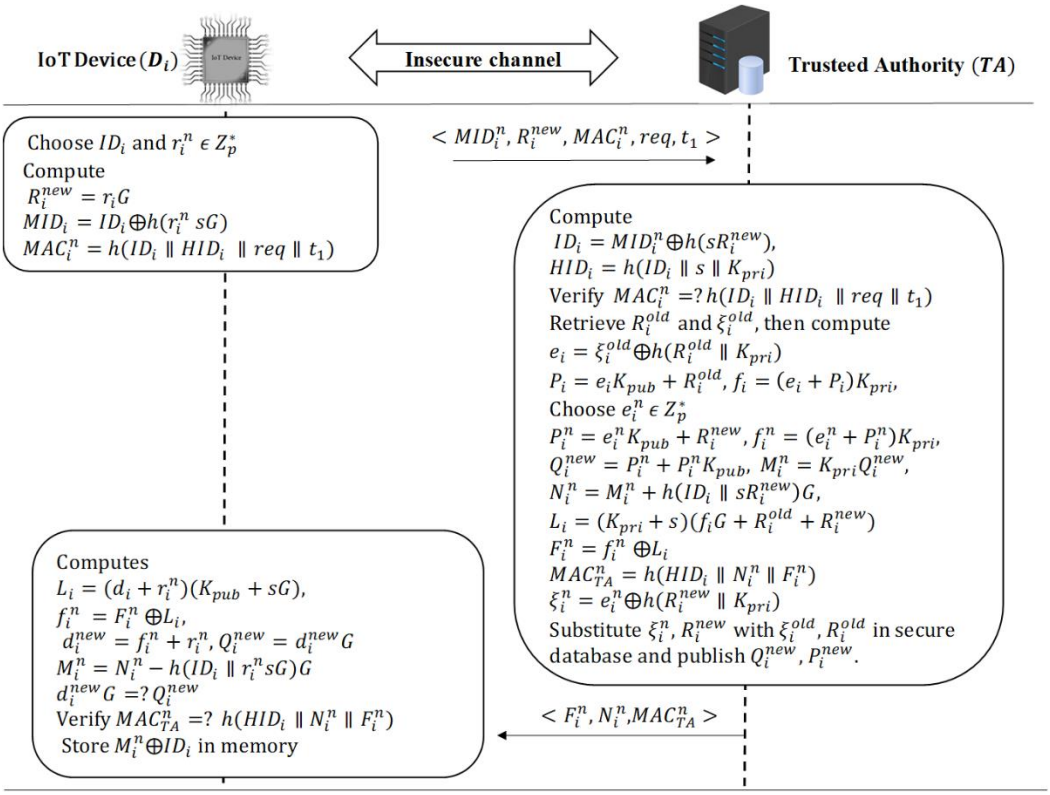

Fig .8. Updating pair of private and public keys phase

Step KU1: $D_i \rightarrow TA$: $< MID_i^n, R_i^{new}, MAC_i^n, req, t_1 >$

D_i selects the random nonce $r_i^n \epsilon Z_p^*$. It then calculates $R_i^{new} = r_i^nG$ and masked identity $MID_i^n = ID_i \oplus h(r_i^n sG)$, $MAC_i^n = h(ID_i \parallel HID_i \parallel req \parallel t_1)$ and transmits message $< MID_i^n, R_i^{new}, MAC_i^n, req, t_1 >$ to TA through a public channel. t_1 indicates the current timestamp of generating message.

Step KU2: $TA \rightarrow D_i$: $< F_i^n, N_i^n, MAC_{TA}^n >$

TA receives the message $< MID_i^n, R_i^{new}, MAC_i^n, req, t_1 >$, and retrieves $ID_i = MID_i \oplus h(sR_i^{new}G)$. It calculates $HID_i = h(ID_i \parallel s \parallel K_{pri})$ and check the equality test $MAC_i^n =? h(ID_i \parallel HID_i \parallel req \parallel t_1)$. If condition is not met, it terminate the session; otherwise, it recalls the corresponding values of ξ_i^{old} and R_i^{old} and computes $e_i = \xi_i^{old} \oplus h(R_i^{old} \parallel K_{pri})$, $P_i = e_iK_{pub} + R_i^{old}$, and $f_i = (e_i + P_i)K_{pri}$. After that, it selects a random nonce $e_i^n \epsilon Z_p^*$ and obtains values of $P_i^n = e_i^nK_{pub} + R_i^{new}$, $f_i^n = (e_i^n + P_i^n)K_{pri}$ and new public key $Q_i^{new} = P_i^n + P_i^nK_{pub}$. Also,

TA calculates $M_i^n = K_{pri}Q_i^{new}$ to obtain values of $N_i^n = M_i^n + h(ID_i \parallel sR_i^{new})G$, $F_i^n = f_i^n \oplus L_i$, and $L_i = (K_{pri} + s)(f_iG + R_i^{old} + R_i^{new})$. Finally, it calculates $MAC_{TA}^n = h(HID_i \parallel N_i^n \parallel F_i^n)$ and transmits the second message $< F_i^n, N_i^n, MAC_{TA}^n >$ to D_i through a public channel. TA also calculates $\xi_i^n = e_i^n \oplus h(R_i^{new} \parallel K_{pri})$ and replaces ξ_i^{old}, R_i^{old} with ξ_i^n, R_i^{new}. It publishes new public key Q_i^{new} and P_i^n in the public space.

Step KU3: $D_i \leftarrow < F_i^n, N_i^n, MAC_{TA}^n >$

Upon receiving the message $< F_i^n, N_i^n, MAC_{TA}^n >$, D_i calculates $L_i = (d_i + r_i^n)\left(K_{pub} + sG\right)$. Then, it retrieves $f_i^n = F_i^n \oplus L_i$ and calculates new public key d_i^{new} as $d_i^{new} = f_i^n + r_i^n$ and verifies new public key Q_i^{new} as $Q_i^{new} =? d_i^{new}G$. Then, it retrieves $M_i^n = N_i^n - h(ID_i)G$ and stores $M_i^n = N_i^n - h(ID_i)G$ in its memory. Two secrets $L_i = (d_i + r_i^n)(K_{pub} + sG)$ and $(K_{pri} + s)(f_iG + R_i^{old} + R_i^{new})$ are equal as $L_i = (d_i + r_i^n)\left(K_{pub} + sG\right) = d_iK_{pub} + d_isG + r_i^n\, K_{pub} + r_i^n sG = K_{pri}\left(f_i + r_i^{old}\right)G + K_{pri}\, r_i^n G + s\left(f_i + r_i^{old}\right)G + sr_i^n G = (K_{pri} + s)(f_iG + R_i^{old} + R_i^{new})$. Figure 8. demonstrate this phase.

5. Security analysis and efficiency

In this section, the proposed protocol is analyzed formally and informally to demonstrate the mutual authentication between two participants and shares a secure and temporary session key that is safe against well-known attacks.

5.1. Security proof

Real-or-Random (ROR) model [10, 18, 19] is used to prove semantic security of the proposed protocol and to obtain session key security (SK-security), as shown in theorem 1.

5.1.1. Preliminaries

Only two partnerships (see definition 1) exist for session key establishing in the proposed protocol. TA actively engages in registration and public key assignment. Public parameters are $\{E/F_p, h(.), G, K_{pub}\}$. Values of $\{M_i, ID_i, d_i, Q_i\}$ are stored in IoT device memory. An entity may have several instances known as oracles. TA server and the IoT device are two entities in the network. IoT devices include instances of Dev^i and Dev^j. Oracles have three modes. (i) $accept$: the oracle receives a correct message and responds to it. It denotes a situation in which the last message by the oracle is accepted before the session expires. (ii) $reject$: the oracle receives a wrong message. (iii) $null$: the oracle returns no responses. PID^i and PID^j imply partnership identity of instances of Dev^i and Dev^j. SID^{ij} is the current session ID. The following definitions help clarify our analyses.

Definition 1. (Participant). Includes all entities of IoT devices with their specific private and public keys in the proposed model.

Definition 2. (Partnering). Suppose Dev^i and Dev^j identified as PID^i and PID^j generate session $\mathcal{S}$ with the identity SID^{ij}. If Dev^i and Dev^j have similar session IDs in accepted state that are directly linked to each other and complete a successful session, they will establish a partnership, one being the initiator and the other responder.

Definition 3. (Freshness). Session $\mathcal{S}$ is fresh if the adversary issues queries of $SKReveal(Dev)$, $EphReveal(Dev)$, $Corrupt(Dev)$, and $Compromise(Dev)$ while it or its matching session is at risk. These queries need to be requested before SK^{ij} expires. This is to distinguish fresh and random session keys.

Definition 4. (Difference lemma). Suppose $Succ_n^{\mathcal{A}} | \; n = 1, 2, 3$ denote the event in probability density function and $Succ_1^{\mathcal{A}} \wedge \neg\, Succ_3^{\mathcal{A}} \Leftrightarrow Succ_2^{\mathcal{A}} \wedge \neg\, Succ_3^{\mathcal{A}}$. Then we have $\left|\Pr[Succ_1^{\mathcal{A}}] - \Pr[Succ_2^{\mathcal{A}}]\right| \leq \Pr[Succ_3^{\mathcal{A}}]$.

The adversary can request the following queries to breach semantic security of the proposed protocol.

$\boldsymbol{Send}(\boldsymbol{Dev^i}, \boldsymbol{Dev^j}, <\boldsymbol{m_1}>)$. Once this query is implemented, $\mathcal{A}$ transmits the message $m_1 =< AID_i, Z_i, \tau_i, t_1 >$ instead of Dev^i to Dev^j. Dev^j checks for validity of the query and, as specified above (see sec 4.2.3), calculates the session key $SK^j = h(TID_i' \parallel TID_j \parallel K_j \parallel t_2)$ and returns the message $< AID_j, Z_j, \tau_j, t_2 >$.

$\boldsymbol{Send}(\boldsymbol{Dev^j}, \boldsymbol{Dev^i}, <\boldsymbol{m_2}>)$. $\mathcal{A}$ sends the message $m_2 =< AID_j, Z_j, \tau_j, t_2 >$ to Dev^i to forge the oracle Dev^j. upon receiving the message, Dev^ichecks for values of the query and, as specified above, calculates the session key $SK^i = h(TID_i \parallel TID_j' \parallel K_i \parallel t_2)$ and completes the session. If the conditions are not met or the session expires, the query is rejected.

$\boldsymbol{Execute}(\boldsymbol{Dev})$. For this query, $\mathcal{A}$ obtains $< m_1 >$ and $< m_2 >$ in authentication and key agreement phase. It is like a passive eavesdropping attack.

$\boldsymbol{Hash}(\boldsymbol{Var})$. The oracle Dev generates a list L_H to store hash records. When the adversary issues $Hash(Var)$, Dev searches in the list L_H and returns the corresponding v in tuple (Var,v). Otherwise, it generates a random value v' and adds (Var,v') to its list, returning the v'.

$\boldsymbol{SKReveal}(\boldsymbol{Dev})$. It simulates session key leakage by the adversary. If SK^{ij} is generated, Dev returns it in response to the query. Otherwise, it returns $null$.

$\boldsymbol{EphReveal}(\boldsymbol{Dev^i})$. $\mathcal{A}$ uses this query to obtain ephemeral secret parameters α and x of Dev^i to perform ephemeral secret leakage attack.

$\boldsymbol{Corrupt}(\boldsymbol{Dev^i})$. By running this query, the adversary obtains long-static secret parameters d_i and M_i of Dev^i.

$\boldsymbol{Compromise}(\boldsymbol{Dev^i})$. By querying this, all static and dynamic secret parameters of Dev^i (x, α_i, M_i and d_i) are delivered to the adversary. The objective is to implement a successful insider attack to capture the whole network.

$\boldsymbol{TestID}(\boldsymbol{Dev})$. Once this is requested, Dev flips the coin $c = \{0, 1\}$. If $c = 1$, it returns the original ID to $\mathcal{A}$. Otherwise, it generates a string of random bits of similar length and returns in response.

$\boldsymbol{TestSK}(\boldsymbol{Dev})$. The adversary needs to have successfully implemented $SKReveal(Dev)$ to run this query in an attempt to obtain the session key SK^{ij} and disrupt semantic security. Upon receiving the query, Dev returns $null$ if the key has not been generated. Otherwise, it flips a neutral coin. If adversary's guess (c') and flipped coin (c) are equal, it yields the session key to the adversary. Otherwise, it generates a random value of similar length and returns it in response.

$\boldsymbol{Expire}(\boldsymbol{Dev})$. It removes the session key generated by Dev.

Definition 5. (Semantic security). To simulate semantic security of the proposed protocol, we designed a series of consecutive and undistinguishable games between the adversary and oracle Dev. The adversary issues different queries to Dev to successfully lunch an attack. The adversary tries to guess the flipped coin c through a $TestSK$ query to increase his chance of winning. This query is asked when Dev is in $accept$ state and the session $\mathcal{S}$ is fresh and not expired. Dev returns SK when session key is generated and $c = c'$.

The advantage of the adversary in breaching our protocol in the semantic security model is $Adv_{\mathcal{P}}^{\mathcal{A}}(t) = \left|\Pr\left[Succ_{\mathcal{P}}^{\mathcal{A}}(t)\right] - 1\right|$, where $Adv_{\mathcal{P}}^{\mathcal{A}}(t) \leq \varepsilon$ and $\varepsilon > 0$ is a trivial value.

5.1.2. Theorem

Suppose $\mathcal{A}$ a probabilistic polynomial adversary at time t that seeks to breach semantic security of the proposed protocol. If $\mathcal{A}$ can issue maximum q_s $Send$, q_h $Hash$, and q_e $Execute$ queries, its advantage for winning the proposed protocol in consecutive games $G_n|n = 1, \dots, 8$ will be less than the following value because of the one-way hash function $h(.)$ and difficulty of ECCDHP. The output string of hash oracle is λ. And $\mathfrak{T}$ is the number of uncompromised instances of Dev in the network. The adversary's likelihood to breach the semantic security of the proposed protocol is calculated as follows:

$$Adv_{\mathcal{P}}^{\mathcal{A}}(t) \leq \frac{5q_s + 4q_h^2 + (q_s + q_e)^2}{2\lambda} + q_h^2\, Adv_{ECCDHP}(t)$$

Eventually, the adversary's likelihood to compromise the whole network is calculated as follows:

$$Adv_{Net}^{\mathcal{A}}(t) = \prod_{i=1}^{\mathfrak{T}} Adv_{\mathcal{P}(i)}^{\mathcal{A}}(t)$$

5.1.3. Proof

To prove the robustness of the proposed protocol, the following games $G_n | n = 1, \dots, 8$ are simulated between the adversary and Dev. $Succ_n^{\mathcal{A}}(t)$ is the chance of winning n^{th} game by the adversary and $\Pr[Succ_n^{\mathcal{A}}(t)]$ is the probability of winning at time t in game G_n. Dev returns a response based on $accept$ or $reject$ state of the query. The games are simulated based on ROR as follows:

Game G_1. It simulates a real attack to the protocol. The adversary $\mathcal{A}$ needs to correctly guess the flipped coin. Thus, the probability of winning for $\mathcal{A}$ in real protocol with random oracles is

$$Adv_1^{\mathcal{A}}(t) = 2\left|\Pr[Succ_1^{\mathcal{A}}(t)] - 1\right| \quad (3)$$

Game G_2. The adversary simulates $Send$ and $Execute$ queries in a real attack to obtain $<m_1>$ and $<m_2>$ messages, or tries to forge Dev. Thus, the probability of winning for the adversary is

$$Adv_1^{\mathcal{A}}(t) = Adv_2^{\mathcal{A}}(t) \quad (4)$$

Game G_3. It simulates a situation in which the adversary wins the game because of hash oracle collision. When $\mathcal{A}$ issues the query, Dev returns the appropriate response from L_T and L_H lists that store transcripts and hash records, respectively. The game is over when random numbers or hash oracles collide. Based on birthday paradox, the probability of collision for hash oracles and random numbers is $\frac{q_h^2}{2\lambda}$ and $\frac{(q_s+q_e)^2}{2\lambda}$, respectively. This game is undistinguishable from the previous one, and thus

$$\left|\Pr[Succ_3^{\mathcal{A}}(t)] - \Pr[Succ_2^{\mathcal{A}}(t)]\right| \leq \frac{q_h^2 + (q_s+q_e)^2}{2\lambda} \quad (5)$$

Game G_4. It simulates the probability of adversary's winning without sending a $Hash$ query. For this, the adversary needs to forge $Send(Dev^i, Dev^j, <m_1>)$ and $Send(Dev^j, Dev^i, <m_2>)$. Based on the difference lemma in consecutive games, the difference between G_3 and G_4 is insignificant. Thus,

$$\left|\Pr[Succ_4^{\mathcal{A}}(t)] - \Pr[Succ_3^{\mathcal{A}}(t)]\right| \leq \frac{q_s}{2\lambda} \quad (6)$$

Game G_5. This game aims at breaking perfect forward secrecy using $Corrupt(Dev)$. $\mathcal{A}$ tries to obtain Dev's temporal secret parameters. The probability of adversary's winning the game is undistinguishable from the previous game as

$$\left|\Pr[Succ_5^{\mathcal{A}}(t)] - \Pr[Succ_4^{\mathcal{A}}(t)]\right| \leq q_h Adv_{ECCDHP}^{\mathcal{A}}(t) + \frac{q_h^2}{2\lambda} \quad (7)$$

Game G_6. This game is over when the adversary obtains the original ID of oracle Dev by issuing a $TestID(Dev)$ query. $\mathcal{A}$ needs to calculate $TID_i = h(ID_i \parallel x)$, $\alpha_i = xQ_j$, and $AID_i = h(\alpha_i) \oplus TID_i$. Assuming the nonce x, the difference between this game and the previous one is

$$\left|\Pr[Succ_6^{\mathcal{A}}(t)] - \Pr[Succ_5^{\mathcal{A}}(t)]\right| \leq \frac{q_h^2}{2\lambda} + \frac{q_s}{\lambda} \quad (8)$$

Game G_7. It simulates a situation in which the adversary issues a $TestSK(Dev)$ query. This requires a successful implementation of $SKReveal(Dev)$. The oracle flips the coin c. If $c = 1$, it returns the session key. Using $Hash$

query at most probability $\frac{q_h^2}{2\lambda}$ and $TestSK$ query at most probability $\frac{q_s}{\lambda}$, the difference between this game and the previous one is

$$\left|\Pr[Succ_7^{\mathcal{A}}(t)] - \Pr[Succ_6^{\mathcal{A}}(t)]\right| \leq \frac{q_h^2}{2\lambda} + \frac{q_s}{\lambda} \quad (9)$$

The adversary's advantage to break security of the proposed protocol by guessing the coin inside $TestSK$ query is $Adv_{\mathcal{P}}^{\mathcal{A}}(t) = 2\Pr[c = c'] - 1$ where c' is the adversary's guess. Since the adversary is unable to distinguish real and random session keys, without issuing a $Hash$ query with accurate entries we have $\Pr[Succ_7^{\mathcal{A}}(t)] = 1/2$. Therefore, the theorem is proves based on formulas 3-9.

$$Adv_{\mathcal{P}}^{\mathcal{A}}(t) \leq 2 \times \sum_{G=1}^{6}\left|Adv_{G+1}^{\mathcal{A}}(t) - Adv_{G}^{\mathcal{A}}(t)\right| \leq \frac{q_h^2+(q_s+q_e)^2}{2\lambda} + \frac{q_s}{2\lambda} + q_h\left(Adv_{ECCDHP}^{\mathcal{A}}(t)\right) + \frac{q_h^2}{2\lambda} + \frac{q_h^2}{2\lambda} + \frac{q_s}{\lambda} + \frac{q_h^2}{2\lambda} + \frac{q_s}{\lambda} \leq \frac{5q_s+4q_h^2+(q_s+q_e)^2}{2\lambda} + q_h^2\, Adv_{ECCDHP}(t) \quad (10)$$

Game G_8. The adversary implements $Compromise$ (Dev) to obtain all static and dynamic secret parameters of Dev. The aim is to perform a privileged-insider attack and compromise the whole network. In fact, by capturing one device, $\mathcal{A}$ seeks to capture the whole network. Since oracle instances are independent from each other, the probability of seizing one instance is independent from others. Thus, the adversary needs to compromise all oracle instances to capture the whole network. Therefore, his chance is trivial as follows:

$$Adv_{Net}^{\mathcal{A}}(t) = \prod_{i=1}^{\mathfrak{T}} Adv_{\mathcal{P}(i)}^{\mathcal{A}}(t) \quad (11)$$

5.1. Formal security analysis by BAN logic

Table 3 shows signs and symbols of BAN logic [20], and the following rules are used in the BAN logic to prove mutual authentication between two IoT devices D_i and D_j.

Table 3. Definition of BAN logic notations.

Symbol	Description	Symbol	Description
P,Q	Entities in the authentication phase	$P \lhd X$	P has received X
$\#(X)$	The formula X is fresh	$P\mid\sim X$	P once sent X
$P\mid\equiv X$	The entity P believes X	$< X >_Y$	X is XORed by Y
$P \Longrightarrow X$	P has jurisdiction on X	$(X)_Y$	X is hashed by Y
$P \overset{K}{\leftrightarrow} Q$	Shared secret K between P and Q	$\{X,Y\}_K$	formulas X and Y encrypted by K

- (R1) Message-Meaning: $\frac{P \mid\equiv P \overset{K}{\leftrightarrow} Q, P \lhd \{X\}_K}{P \mid\equiv (Q\mid\sim X)}$
- (R2) Nonce-Verification: $\frac{P \mid\equiv \#(X), P \mid\equiv (Q\mid\sim X)}{P \mid\equiv (Q\mid\equiv X)}$
- (R3) Jurisdiction: $\frac{P \mid\equiv (Q \Rightarrow X), P \mid\equiv (Q \mid\equiv X)}{P \mid\equiv X}$
- (R4) Freshness: $\frac{P \mid\equiv \#(X)}{P \mid\equiv \#(X,Y)}$
- (R5) Belief: $\frac{P \mid\equiv Q\mid\equiv (X,Y)}{P \mid\equiv Q\mid\equiv (X)}$, $\frac{P \mid\equiv (X), P\mid\equiv (Y)}{P \mid\equiv (X,Y)}$

The proposed protocol is based on following assumptions.

- $A1:\ D_i\mid\equiv \#(x), D_i\mid\equiv \#(t_i), D_i\mid\equiv \#(t_j)$
- $A2:\ D_j\mid\equiv \#(y), D_j\mid\equiv \#(t_j), D_j\mid\equiv \#(t_i)$
- $A3:\ D_i\mid\equiv D_i \overset{M_i}{\leftrightarrow} TA$
- $A5:\ D_i\mid\equiv D_i \overset{SK_{ij}}{\longleftrightarrow} D_j$
- $A6:\ D_j\mid\equiv D_i \overset{SK_{ji}}{\longleftrightarrow} D_j$
- $A7:\ D_i\mid\equiv D_j \Longrightarrow D_i \overset{SK_{ij}}{\longleftrightarrow} D_j, D_i\mid\equiv D_j \Longrightarrow M_j$

- $A4:\ D_j| \equiv D_j \overset{M_j}{\leftrightarrow} TA$
- $A8:\ D_j| \equiv D_i \Longrightarrow D_i \overset{SK_{ij}}{\longleftrightarrow} D_j, D_j| \equiv D_i \Longrightarrow M_i$

The process of mutual authentication aims to obtain the following goals

- $G1:\ D_j| \equiv D_i| \equiv U_i \overset{SK_{ji}}{\longleftrightarrow} D_j$
- $G3:\ D_i| \equiv D_j| \equiv D_i \overset{SK_{ij}}{\longleftrightarrow} D_j$
- $G2:\ D_j| \equiv D_i \overset{SK_{ij}}{\longleftrightarrow} D_j$
- $G4:\ D_i| \equiv D_i \overset{SK_{ji}}{\longleftrightarrow} D_j$

Messages are transmitted between D_i and D_j in a public channel. The idealized message has the following specifications

$$M1: D_j \lhd \left\{< TID_i >_{\alpha_i}, xG, (TID_i, \tau_i, t_i)_{M_iQ_j}, t_i\right\},\ M2: D_i \lhd \{< TID_j >_{\beta_j}, yG, \left(TID_j, \left(TID_i, TID_j, t_j\right)_K\right)_{(M_jQ_i)}, t_j\}$$

Based on $A4$, message $M1$, applying the message-meaning rule $R1$, and the fact that D_j receives the first message, we have

$$S1: D_j \mid \equiv D_i \mid \sim (TID_i, \alpha_i, \tau_i (TID_i, \tau_i, t_i)_{M_iQ_j}, t_i)$$

Using $A2$, $S1$, the freshness rule $R4$, and the nonce verification rule $R2$, we have

$$S2: D_j \mid \equiv D_i \mid \equiv (TID_i, \alpha_i, \tau_i (TID_i, \tau_i, t_i)_{M_iQ_j}, t_i)$$

Based on $A6$, $S2$, and message $M2$, we have

$$S3: D_j \mid \equiv D_i \mid \equiv (TID_i, TID_j, M_iQ_j, t_i)$$

We achieve the first goal by applying the belief rule $R5$ on $S3$.

$$\boldsymbol{G_1}: \boldsymbol{D_j}| \equiv \boldsymbol{D_i}| \equiv (\boldsymbol{D_i} \overset{\boldsymbol{SK_{ji}}}{\longleftrightarrow} \boldsymbol{D_j})$$

The second goal is achieved based on $A8$, the jurisdiction rule $R3$ and $S3$.

$$\boldsymbol{G_2}: \boldsymbol{D_j}| \equiv \boldsymbol{D_i} \overset{\boldsymbol{SK_{ij}}}{\longleftrightarrow} \boldsymbol{D_j}$$

Based on receiving the message $M2$ at D_j, $A3$ and $M2$, we have

$$S4:\ D_i \mid \equiv D_j \mid \sim (TID_j, \beta_j, \tau_j, \left(D_i \overset{SK_{ji}}{\longleftrightarrow} D_j\right)_{(M_jQ_i)}, t_j)$$

Using $A1$, $S4$, the message freshness, and the nonce verification, we achieve $S5$.

$$S5:\ D_i \mid \equiv D_j \mid \equiv (TID_j, \beta_j, \tau_j, \left(D_i \overset{SK_{ji}}{\longleftrightarrow} D_j\right)_{(M_jQ_i)}, t_j)$$

According to $A5$, ad $S5$, we achieve the third goal.

$$\boldsymbol{G_3}: \boldsymbol{D_i}| \equiv \boldsymbol{D_j}| \equiv \boldsymbol{D_i} \overset{\boldsymbol{SK_{ij}}}{\longleftrightarrow} \boldsymbol{D_j}$$

Eventually, based on $A7$, $R3$, and by assuming the correctness of the third goal, we achieve the fourth goal.

$$\boldsymbol{G_4}: \boldsymbol{D_i}| \equiv \boldsymbol{D_i} \overset{\boldsymbol{SK_{ji}}}{\longleftrightarrow} \boldsymbol{D_j}$$

5.2. Scyther tool

Scyther tool is widely used for analyzing security protocols [21, 22]. In this section, we show that the proposed protocol meets security claims like "*Alive*", "*Nisynch*", "*Niagree*", "*weakagree*", and "*secret*" for aliveness, non-injective synchronization, non-injective agreement, minimum agreement, and confidentiality, respectively. Moreover, confidentiality of session key is preserved and IoT device IDs remain secret. Scyther code and evaluation results are given below in Fig 9 and Fig 10.

```
usertype TimeStamp;
const G;
secret IDi, IDj, di, dj, Kpri,x,y;
hashfunction h1;
secret XOR: Function;
secret ScalarMulti: Function;
macro Kpub = ScalarMulti(Kpri, G); macro Qi=ScalarMulti(di, G); macro Qj=ScalarMulti(dj, G); macro TIDi=h1(IDi,x);
macro TIDj=h1(IDj,y);
protocol IoVD2D (IoTDevi, IoTDevj) {
role IoTDevi        {
fresh t1: TimeStamp;
macro TIDi=h1(IDi,x); macro Etai=ScalarMulti (x,G); macro Alphai= ScalarMulti (x, Qj); macro AIDi= XOR(h1(Alphai),
TIDi);  macro MiQj= ScalarMulti(Kpri,Qi,Qj); macro Zi =h1(TIDi, Etai, MiQj, t1);
send_1(IoTDevi, IoTDevj ,(AIDi, Zi, Etai ,t1));
var  AIDj, Zj, Etaj, t2;
recv_2( IoTDevj ,IoTDevi ,(AIDj,Zj,Etaj,t2));
macro Betajhat= ScalarMulti(di,Etaj); macro TIDjhat= XOR(h1(Betajhat),AIDj); macro
Ki=h1(Alphai,ScalarMulti(di,Etaj),MiQj); macro SKi=h1(TIDi, TIDjhat, Ki, t2);
macro MjQi=ScalarMulti(Kpri,Qj,Qi); macro Zjhat= h1(TIDjhat, SKi, MjQi, t2);
match (Zjhat, Zj);
claim (IoTDevi,Secret,IDi); claim (IoTDevi,Secret,IDj); claim (IoTDevi, Alive); claim (IoTDevi, Nisynch);
claim (IoTDevi, Niagree); claim (IoTDevi, Weakagree); claim (IoTDevi, Commit, IoTDevj , x,y); };
role  IoTDevj     {
var t1; recv_1(IoTDevi, IoTDevj,(AIDi,Zi, Etai,t1));
macro Alphaihat= ScalarMulti(dj,Etai); macro TIDihat= XOR(h1(Alphaihat),AIDi); macro
MjQi=ScalarMulti(Kpri,Qj,Qi); match (MjQi,MiQj); macro Zihat=h1(TIDihat, Etai,MjQi, t1);
match (Zihat,Zi);
macro TIDj1=h1(IDj,y); macro Betaj=ScalarMulti (y,Qi); macro AIDj= XOR(h1(Betaj), TIDj1);
macro Etaj=ScalarMulti (y,G); macro Kj=h1(ScalarMulti(dj, Betaj, Etai),MjQi);
fresh t2: TimeStamp;
macro SKj=h1(TIDihat, TIDj1, Kj, t2); macro Zj=h1(TIDj1, SKj, MjQi, t2); send_2 (IoTDevj ,IoTDevi ,(AIDj,Zj,Etaj,t2));
claim (IoTDevj,Secret,IDi); claim (IoTDevj,Secret,IDj); claim (IoTDevj, Alive); claim (IoTDevj, Nisynch);
claim (IoTDevj, Niagree); claim (IoTDevj, Weakagree); claim (IoTDevj, Commit IoTDevi, x,y); };
}:
```

Fig .9. Scyther code of the proposed protocol

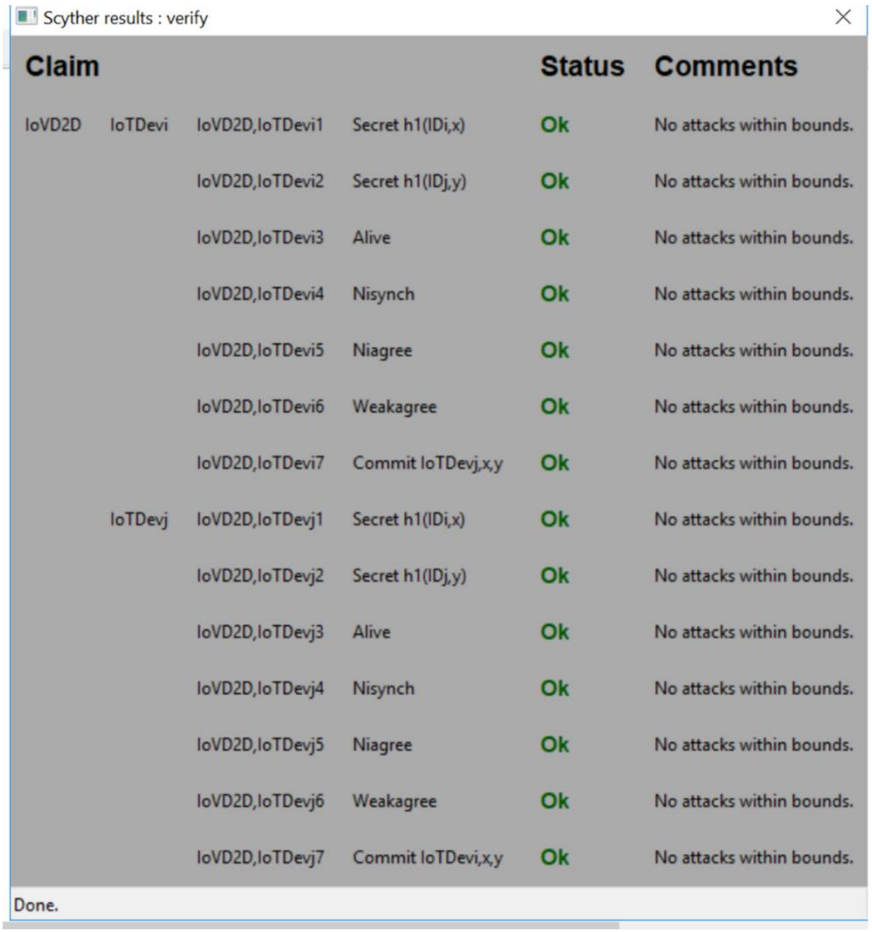

Fig .10. Security analysis of the proposed protocol using Scyther tool

As seen in Fig 10, ID_i, ID_j, $TID_i = h(ID_i \parallel x)$, and $TID_i = h(ID_i \parallel y)$ are secure against intruder attacks and their secrecy is preserved. Thus, the proposed protocol is highly anonymous and untraceable.

5.3. Informal security analysis

It is proved in this section that the proposed protocol provides security measures for D2D key agreement protocol. Table 4 compares the proposed protocol with other existing protocols in features and resisting known attacks.

5.3.1. Replay Attack. The protocol uses timestamps t_1 and t_2, random-temporary nonces x and y, that are used as hashed values $Z_i = h(TID_i \parallel \tau_i \parallel M_i Q_j \parallel t_1)$. Any changes in them violates message integrity and the condition $Z_i =? Z_i' = h(TID_i' \parallel \tau_i \parallel M_j Q_i \parallel t_1)$ will no longer hold. Hence it is secure against replay attack because all the messages are sent in a valid ΔT time period.

5.3.2. Man-In-The-Middle (MITM) Attack. It is assumed that adversary $\mathcal{A}$ has access to ID_i and ID_j of IoT devices. He seeks to seize and manipulate messages $< AID_i, Z_i, \tau_i, t_1 >$ and $< AID_j, Z_j, \tau_j, t_2 >$ to IoT devices and forge an authorized device. Since the adversary does not have long secret key d_i and d_j, he cannot retrieve $\alpha_i = d_j \tau_i$ and $\beta_j = d_i \tau_j$, and is unable to perform the man-in-the-middle attack.

5.3.3. Privilege-Insider Attack. An unauthorized insider IoT device cannot seize device IDs or session keys in the network pseudo-identity $AID_i = h(\alpha_i) \oplus TID_i$ are sent to the channel where $\alpha_i = xQ_j$ and only D_j has access

to TID_i because α_i is retrieved as $\alpha_i = d_j\tau_i$. This prevents all unauthorized insider access to other session keys and device IDs. While Alzahrani et al.'s [6] protocol cannot withstand the privileged insider attack.

5.3.4. Device capture attack. If an adversary $\mathcal{A}$ captures an IoT device and extract its sensitive information by power analysis techniques to achieve M_i and ID_i, he cannot get other devices' information such as secret stored value M_j, private key d_j, and original identity ID_j. In addition, if private key d_i is not stored in D_i, the adversary cannot compromise that node. Thus, the proposed protocol is secure against device compromise attack, while [6, 7, 8, 9] are insecure against that.

5.3.5. Known Specific Temporary Information Attack. If an adversary $\mathcal{A}$ obtains secret parameters x and y, he cannot get session key $SK_{ij} = h(TID_i \parallel TID_j \parallel K \parallel t_2)$ because hidden value K is $K = h(\alpha_i \parallel d_i\tau_j \parallel M_iQ_j) = h(d_j\tau_i \parallel \beta_j \parallel M_jQ_i)$ wherein M_i and M_j are stored in each device and is inaccessible. Therefore, our protocol is secure against temporary secret leakage attack.

5.3.6. Key Compromise Impersonation (KCI) Attack. Once an adversary $\mathcal{A}$ gains access to private keys d_i or d_j, he is unable to get session key $SK_i = SK_j$ because hidden parameters M_i/M_j are stores in devices as $M_i = K_{pri}Q_i$ which is extracted only when the Elliptic Curve Diffie-Hellman (ECDHP) Problem is solved. Thus, the proposed protocol is secure against KCI attack.

5.3.7. Anonymity and Untraceability. Messages are transmitted between two devices with pseudo-temporary identity $AID_i = h(\alpha_i)\oplus TID_i$ and $AID_j = h(\beta_j)\oplus TID_j$ where TID_i and TID_j are temporary IDs. If an adversary $\mathcal{A}$ obtains private keys d_i and d_j, he can only gain access to the temporary, not permanent, IDs. Moreover, earlier messages cannot be retrieved because temporary IDs change in each authentication session. Thus, the proposed protocol is anonymous and untraceable. In other words, it provides dual anonymous communication.

5.3.8. Perfect Forward/Backward Secrecy. If an adversary $\mathcal{A}$ obtains the session key in a session, he will not have access to other session keys in other session because session keys are computed independently, including random nonces (x and y) and temporary TID_i and TID_j that change in each authentication session. Therefore, the proposed protocol preserves perfect forward backward secrecy.

5.3.9. Impersonation Attack. To successfully perform this attack and forge IoT devices, the adversary needs to duplicate messages $Z_i = h(TID_i \parallel \tau_i \parallel M_iQ_j \parallel t_1)$ and $Z_j = h(TID_j \parallel SK_j \parallel t_2)$ to be certified by the shared protocol. An adversary can never generate a valid message to forge an authorized device in the network because it does not have access to private key d_i and original ID_i. Our proposed protocol is able to withstand impersonation attack.

5.3.10. Known-Key Attack. When an IoT device shares a secure session key with another device in the network, an adversary $\mathcal{A}$ who seizes the key cannot obtain the keys of other devices because each key has its own specific parameters x, y, K_i, K_j, and temporary TID. To overcome this challenge, the adversary must solve Elliptic Curve Discrete Logarithm (ECDL) problem, ECDHP problem, and one-way hash function, which is practically infeasible.

5.3.11. Key Escrow Problem. If private key of TA is disclosed, other keys remain secret because only TA is involved in key generation process and the private key $d_i = f_i + r_i = (e_i + P_i)K_{pri} + r_i$ consists of random nonce $r_i \in Z_p^*$ that is unknown to TA. Therefore, the proposed protocol solves key escrow problem and single point of failure.

5.3.12. Self-Certification Mechanism. Each IoT device D_i stores $M_i = K_{pri}Q_i$ in its memory in registration phase, and does not need an online trusted third-party TA in authentication and key agreement phase. Therefore, our protocol solves key escrow problem, single-point of failure, and has a great performance in certificate generation and verification process.

5.3.13. Clogging Attack. The adversary cannot perform clogging attack and waste IoT resources because both agents mutually authenticate each other before sharing a session key. however, Li et al.'s and Chaudhry et al.'s protocols are insecure against this attack.

5.3.14. Other features. Many protocols neglect updating long secret credentials. Nevertheless, our protocol offers a secure authentication and key agreement session between IoT devices while updating paired private and public keys of each device in the presence of a TA in a pubic channel. This increases its applicability. Moreover, registration phase of this protocol is done in an insecure channel that increases its scalability.

Table 4. Security Comparisons

Comparative point		protocols				
		Chaudhry et al. [10]	Das et al. [18]	Alzahrani et al. [6]	Li et al. [9]	Our protocol
Features	Mutual authentication	✓	✓	✓	×	✓
	Perfect forward secrecy	×	✓	✓	✓	✓
	IoT device anonymity and untraceability	×	×	×	×	✓
	Long secret keys updating	×	×	×	×	✓
	Using public channel in registration and updating keys phase	×	×	×	×	✓
	Security proof	ROM model	ROR model	ROM model, BAN logic	ROM model	ROR model, BAN logic
	Security analysis	-	AVISPA tool	-	-	Scyther tool
Resistance to	Strong replay attack	✓	✓	✓	×	✓
	Clogging attack	×	✓	✓	×	✓
	Desynchronization attack	×	✓	✓	×	✓
	Man-in-the-Middle attack	×	×	✓	✓	✓
	IoT device impersonation attack	✓	×	✓	✓	✓
	IoT deice Key compromise Impersonation attack	✓	✓	×	×	✓
	IoT device capture attack	✓	✓	✓	✓	✓
	Privilege-insider attack	✓	✓	×	✓	✓
	Known specific temporary information attack	✓	✓	✓	✓	✓
	Key escrow problem	✓	✓	✓	✓	✓
	Known key attack	✓	✓	✓	✓	✓

✓: Yes, ×: No

Based on Table 4 and other analyses, our protocol has a perfect security profile while other protocols [6, 9, 10, 18] suffer from flaws. Thus, our proposed protocol is better than others for authentication of D2D communications.

5.4. Analysis of efficiency

Our protocol is compared with other existing protocols [6, 9, 10, 18] in terms of communication and computational overhead. We assume that timestamp, ID, nonce, hash value and elliptical curve point are 32, 128, 128, 160, and 320 bits, respectively. The time for hash operation (T_h), symmetric encryption/decryption

($T^{s}_{enc/dec}$), point addition (T_{pa}), and point scalar multiplication on elliptic curve (T_{pm}) is 0.00032, 0.0056, 0.0044, and 0.0171 sec [23, 18]. Computation and communication costs of our protocol are 0.14128 sec and 1344 bits, respectively. Table 5 compares our protocol with existing ones in terms of communication and computation overheads. It shows that the proposed protocol has lower computational overhead than other related protocols. However, it resists KCI attack, replay attack, insider attack, clogging attack, and provides anonymity (see Table 4). In [9], only one session key is shared and no authentication mechanism is applied. Security is an essential factor in IoT networks and overhead pays its due for that. Our protocol is compared with others in terms of computational overhead and communication overhead in Fig 11.

Table 5. Comparison of Communication and Computation Overheads

Protocol	**Communication Cost**	**Computation Cost**
Alzahrani et al. [6]	$4\|ECC\| + 3\|Hash\| + 2\|Timestamp\|$	$8\,T_{pm} + 2\,T_{pa} + 8\,T_h$
Li et al. [9]	$2\|ID\| + 4\|ECC\|$	$12\,T_{pm} + 2\,T_{pa} + 6\,T_h$
Chaudhry et al. [10]	$2\|ID\| + 6\|ECC\| + 2\|Hash\| + 3\|Timestamp\|$	$10\,T_{pm} + 6\,T_{pa} + 8\,T_h$
Das et al. [18]	$2\|ID\| + 8\|ECC\| + 2\|Hash\| + 3\|Timestamp\|$	$18\,T_{pm} + 6\,T_{pa} + 12\,T_h$
Our protocol	$2\|ECC\| + 4\|Hash\| + 2\|Timestamp\|$	$8\,T_{pm} + 14\,T_h$

$|x|$: Bit length of variable x, T_x: execution time for the operation x.

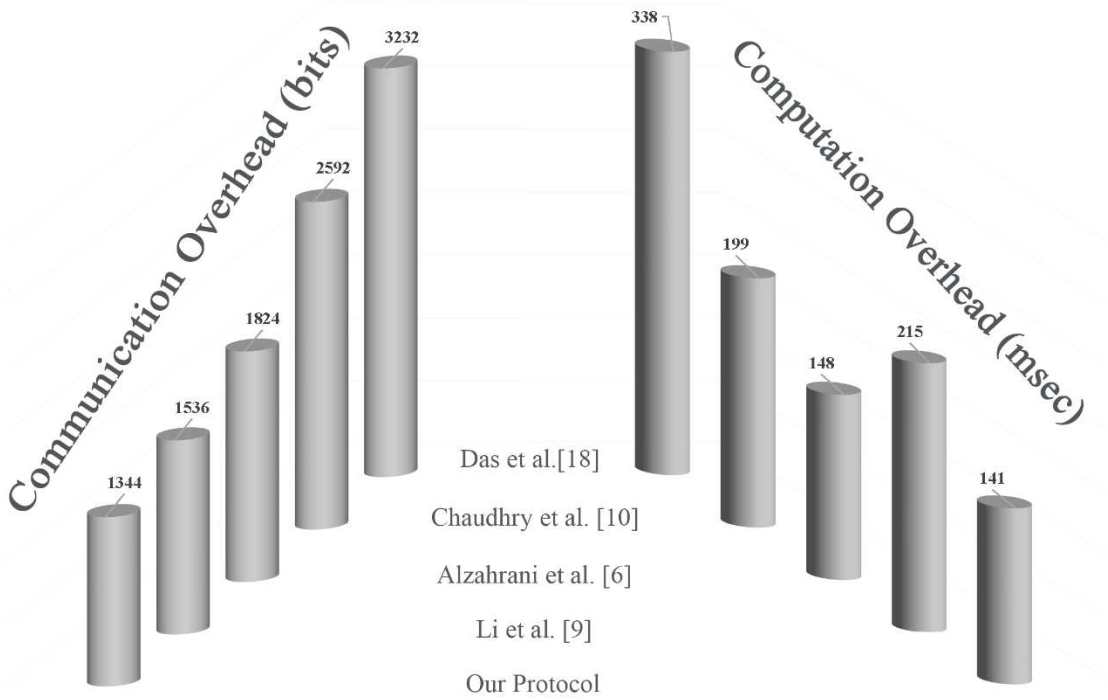

Fig .11. Communication and communication overhead comparison.

The performance of the proposed protocol, compared to existing protocols, in terms of energy consumption for 500 authentication and key agreement sessions for IoT devices in a 100×100 m^2 environment is simulated in Matlab R2017a. Results are shown in Fig. 12. Increased number of sessions leads to increase energy

consumption in [10] and [18]. But the proposed protocol offers the most optimal performance because energy management in ubiquity 6G networks is important along with security issues. Our protocol improves performance to 20-65%.

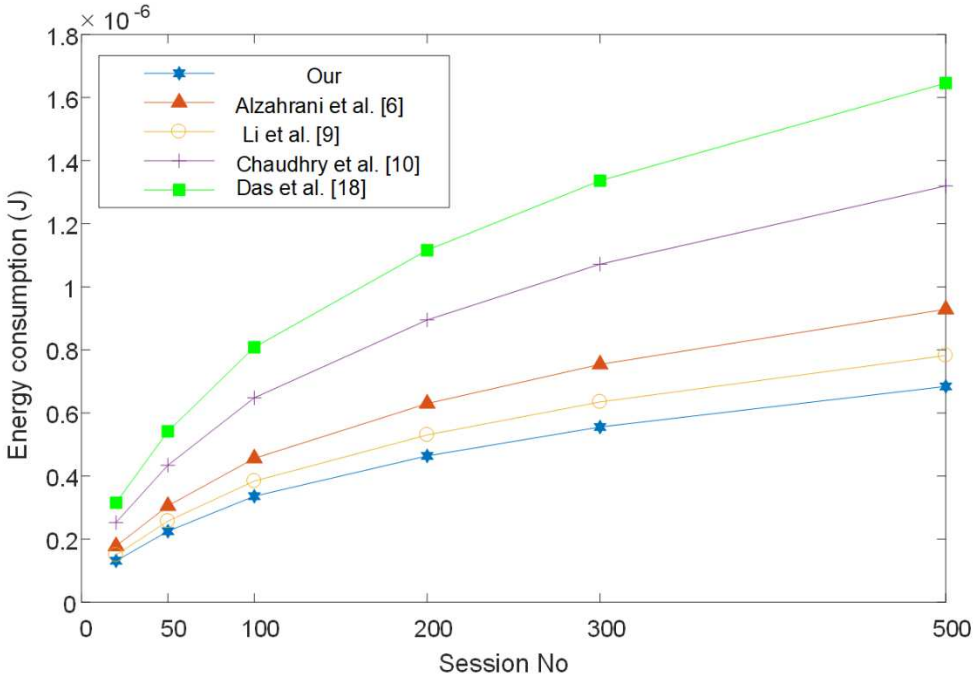

Fig. 12. Energy consumption comparison.

6. Conclusion

Widespread use of IoT technologies and lack of comprehensive standard protocols for IoT environments call for new authentication and key agreement protocols. The present study examined the flaws of mutual authentication and key agreement protocols for D2D communications in IoT networks. It also proposed a new protocol include four phases: system initialization, IoT device registration, authentication and key agreement, key pair updating. Our protocol could overcome security issues and resist well-known attacks such as KCI attack, replay attack, insider attack, and provide strong anonymity. Also, it used public channels in IoT device registration and public/private key updating phases. BAN logic, ROR model, and Scyther tool were used for security analysis. The protocol performance was also compared with other protocols in terms of computational, communication overhead, and energy consumption and showed to have a better performance than other compared protocols. Finally, a blockchain-based group authentication protocol in D2D communications while provides anonymity is proposed for future work.

Compliance with ethical standards

Conflict of interest. The authors declare that they have no conflict of interest.

Ethical approval. This article does not contain any studies with human participants or animals performed by any of the authors.

Informed consent. Informed consent was obtained from all individual participants included in the study.

Author contributions

All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by Rahman Hajian, Abbas Haghighat, and Seyed Hossein Erfani. All authors read and approved the final manuscript.

References

[1] Saad, W., Bennis, M., & Chen, M. (2019), A vision of 6G wireless systems: Applications, trends, technologies, and open research problems, IEEE network, 1-9, https://doi.org/.10.1109/MNET.001.1900287.

[2] Wang, M., Yan, Z., & Niemi, V. (2017), UAKA-D2D: Universal authentication and key agreement protocol in D2D communications. Mobile Networks and Applications, 22(3), 510-525, https://doi.org/10.1007/s11036-017-0870-5.

[3] Seok, B., Sicato, J. C. S., Erzhena, T., Xuan, C., Pan, Y., & Park, J. H. (2020), Secure D2D Communication for 5G IoT Network Based on Lightweight Cryptography, Applied Sciences, 10(1), 217, https://doi.org/10.3390/app10010217.

[4] Hajian, R., & Erfani, S. H. (2020). CHESDA: continuous hybrid and energy-efficient secure data aggregation for WSN. The Journal of Supercomputing, 1-31, https://doi.org/10.1007/s11227-020-03455-z.

[5] Banerjee, S., Odelu, V., Das, A. K., Srinivas, J., Kumar, N., Chattopadhyay, S., & Choo, K. K. R. (2019), A Provably Secure and Lightweight Anonymous User Authenticated Session Key Exchange Scheme for Internet of Things Deployment, *IEEE Internet of Things Journal*, *6*(5), 8739-8752, https://doi.org/10.1109/JIOT.2019.2923373.

[6] Alzahrani, B. A., Chaudhry, S. A., Barnawi, A., Al-Barakati, A., & Shon, T. (2020), An Anonymous Device to Device Authentication Protocol Using ECC and Self Certified Public Keys Usable in Internet of Things Based Autonomous Devices, *Electronics*, *9*(3), 520, https://doi.org/10.3390/electronics9030520.

[7] Islam, S. H., & Biswas, G. P. (2015), Design of two-party authenticated key agreement protocol based on ECC and self-certified public keys, *Wireless Personal Communications*, *82*(4), 2727-2750, https://doi.org/10.1007/s11277-015-2375-5.

[8] Mandal, S., Mohanty, S., & Majhi, B., (2018), Cryptanalysis and enhancement of an anonymous self-certified key exchange protocol, *Wireless Personal Communications*, *99*(2), 863-891, https://doi.org/10.1007/s11277-017-5156-5.

[9] Li, Q., Hsu, C. F., Raymond Choo, K. K., & He, D. (2019), A Provably Secure and Lightweight Identity-Based Two-Party Authenticated Key Agreement Protocol for Vehicular Ad Hoc Networks, *Security and Communication Networks*, *7871067,* https://doi.org/10.1155/2019/7871067.

[10] Chaudhry, S. A., Yahya, K., Al-Turjman, F., & Yang, M. H. (2020). A secure and reliable device access control scheme for IoT based sensor cloud systems. IEEE Access, 8, 139244-139254, https://doi.org/10.1109/ACCESS.2020.3012121.

[11] Amin, R., Islam, S. H., Biswas, G. P., Giri, D., Khan, M. K., & Kumar, N., (2016), A more secure and privacy-aware anonymous user authentication scheme for distributed mobile cloud computing environments, *Security and Communication Networks*, *9*(17), 4650-4666,https://doi.org/10.1002/sec.1655.

[12] Das, A. K., Wazid, M., Kumar, N., Khan, M. K., Choo, K. K. R., & Park, Y, (2017), Design of secure and lightweight authentication protocol for wearable devices environment, *IEEE journal of biomedical and health informatics*, *22*(4), 1310-1322, https://doi.org/10.1109/JBHI.2017.2753464.

[13] Simplicio Jr, M. A., Silva, M. V., Alves, R. C., & Shibata, T. K, (2017), Lightweight and escrow-less authenticated key agreement for the internet of things. *Computer Communications*, *98*, 43-51, https://doi.org/10.1016/j.comcom.2016.05.002.

[14] Wu, F., Xu, L., Li, X., Kumari, S., Karuppiah, M., & Obaidat, M. S., (2018), A lightweight and provably secure key agreement system for a smart grid with elliptic curve cryptography, *IEEE Systems Journal*, *13*(3), 2830-2838, https://doi.org/10.1109/JSYST.2018.2876226.

[15] Shuai, M., Liu, B., Yu, N., Xiong, L., & Wang, C. (2020), Efficient and privacy-preserving authentication scheme for wireless body area networks, *Journal of Information Security and Applications*, *52*, 102499, https://doi.org/10.1016/j.jisa.2020.102499.

[16] Lara, E., Aguilar, L., Sanchez, M. A., & García, J. A. (2020). Lightweight Authentication Protocol for M2M Communications of Resource-Constrained Devices in Industrial Internet of Things, *Sensors*, *20*(2), 501, https://doi.org/10.3390/s20020501.

[17] Islam, S. H. (2020), Provably secure two-party authenticated key agreement protocol for post-quantum environments. *Journal of Information Security and Applications*, *52*, 102468, https://doi.org/10.1016/j.jisa.2020.102468.

[18] Das, A. K., Wazid, M., Yannam, A. R., Rodrigues, J. J., & Park, Y. (2019). Provably secure ECC-based device access control and key agreement protocol for IoT environment. *IEEE Access*, *7*, 55382-55397, https://doi.org/10.1109/ACCESS.2019.2912998.

[19] R. Vinoth, L. J. Deborah, P. Vijayakumar and N. Kumar, (2020), "Secure Multi-factor Authenticated Key Agreement Scheme for Industrial IoT," in *IEEE Internet of Things Journal*, https://doi: 10.1109/JIOT.2020.3024703.

[20] Burrows, Michael, Martin Abadi, and Roger Michael Needham. "A logic of authentication." Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences, 426(1871), 233-271 (1989), https://doi.org/10.1098/rspa.1989.0125.

[21] C. Cremers, Scyther tool, (n.d.). http://www.cs.ox.ac.uk/people/cas.cremers/scyther/ (accessed June 13, 2018).

[22] Nikooghadam, M., & Amintoosi, H. (2020). A secure and robust elliptic curve cryptography-based mutual authentication scheme for session initiation protocol. *Security and Privacy*, *3*(1), e92, https://doi.org/10.1002/spy2.92.

[23] Srinivas, J., Das, A. K., Kumar, N., & Rodrigues, J. (2018). Cloud centric authentication for wearable healthcare monitoring system. *IEEE Transactions on Dependable and Secure Computing*, https://doi.org/10.1109/TDSC.2018.2828306.