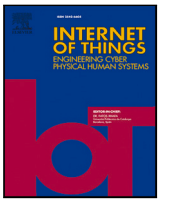
Internet of Things 21 (2023) 100669

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.elsevier.com/locate/iot



Research article

A modified lightweight authenticated key agreement protocol for Internet of Drones

Dharminder Chaudhary [a], Tanmay Soni [a], Kondeti Lakshmi Vasudev [b,*], Kashif Saleem [c]

[a] Department of Computer Science and Engineering, Amrita School of Computing, Amrita Vishwa Vidyapeetham, Chennai, India
[b] Department of Mechanical Engineering, Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Chennai, India
[c] Center of Excellence in Information Assurance, King Saud University, Riyadh, Saudi Arabia

ARTICLE INFO

Keywords:
Internet of Drones
Authentication
Key agreement
Collision resistant hash
Cryptography

ABSTRACT

Drones can be embedded in the system Internet of Drones can make a military observation of a region, a transport system embedded with sensors. Drones played a highly important role in different fields and brought great convenience to production and lifestyle. But we know all the data collected by sensors embedded in drones suffer from security challenges and privacy issues. There are a lot of authenticated key agreement protocols for the security of transmitted data through drones. We have analyzed a recent protocol, "A lightweight authentication and key agreement scheme for Internet of Drones" by Zhang et al. in the year 2020. We have found that this protocol is not secure against stolen smart card attacks, and the control server stores extra data. We have tried to address security issues of password guessing, anonymity, user/server impersonation, insider attack, and stolen smart card attacks, and man in the middle attack. We have proposed a modified lightweight authenticated key agreement protocol for Internet of Drones. We have proved its security in the random oracle model. We have done a performance analysis of the proposed protocol with relevant protocols that ensures its efficiency and security as well.

1. Introduction

The application of Internet of drones in various fields is gaining prominence because of their employability in military surveillance, aerial photography, medicines transport, logistics, disaster relief transport, drone policing, spraying of pesticides etc. [1,2]. A typical surveillance application using drones is depicted in Fig. 1. The embedded sensors inside the drone can collect and analyze the physical characteristics like humidity, pressure, and temperature, along with the videos and audio recordings collected by the embedded camera and microphone. This information can be transmitted to the controller through wireless communication like Bluetooth, wifi, etc. In this manner, the controller can obtain real-time information by controlling these drones at a distance. Application of these drones for the above-mentioned purposes gained popularity because of the mobility these vehicles can provide [3,4]. The design of each drone depends on the mission it has to fulfill. For example, if the drone has to be deployed for disaster relief, then it is required to be equipped with infrared detectors and cameras, similarly, if its application is aerial photography, then cameras and microphones are required, and for logistics transportation, positioning systems are required. Hence, the design process of these vehicles depends on the mission specifications. Even though each drone has its own objectives to fulfill a mission, when it comes to its usage, the design objectives are endurance, low drag, range, and less power consumption for flight.

* Corresponding author.
E-mail addresses: lakshmivasudev.k@gmail.com, kl_vasudev@ch.amrita.edu (K.L. Vasudev).

https://doi.org/10.1016/j.iot.2022.100669
Received 7 October 2022; Received in revised form 18 November 2022; Accepted 11 December 2022
Available online 22 December 2022

This gives restrictions on the weight, communication system and energy source [5,6]. An alternate method is to implement multiple drones such that data can be collected in a distributed manner while the model can prolong battery life for the sensor device thus reducing the exploitation costs of the operation [7]. But, this method also has challenges related to the security of the data collected. In some cases, the data collected may have highly sensitive information, and the fragile communication network between drones may lead to easy interception into the transmitted data and can be captured. In these dynamic and continuously changing and updating eras, data collected from IoD systems, is very valuable for big data analytics in understanding and predicting the trends of individual behaviors. In this scenario, it is very difficult to say the existing security techniques for data protection are sophisticated and cannot be broken easily by the cyber criminals. This is because, first, the existing methods follow a mechanism based on computational security. Second, drones have limited computing power hence, they cannot perform complex encryption and decryption of datasets. Third, collected data by private IoD are stored in public cloud servers, this result in giving direct access to the collected data to the other tenants of the public cloud where it is stored. Chen and Wang [1] proposed a pseudonym scheme-based network coding that enhances the privacy against the above mentioned threats.

These pseudonym schemes can be of two types. First, techniques that disconnect the user's location data with her/his true identity.

The initial attempts in this direction were put by Lamport [8]. He proposed a one-way hash function in the whole remote authentication scheme. Many updated versions of Lamport [8] work have been developed. The first attempt in the authenticated key agreement was made by Turkanovic et al. [9]. They implemented authenticated key exchange between nodes and users without using gate way node. Resource limited nodes are befitted as bitwise XOR operator and hash function. However, this scheme is incapable to resist node impersonation and man in the middle attacks, along user traceability and with nodes anonymity. These gaps were identified and addressed by Farash et al. [10]. They ignored the drawbacks like specific temporary information, user impersonation, and offline password guessing attacks. Amin et al. [11] proposed a robust AKA scheme that overcomes the drawbacks of Farashet al.'s work. The scheme has relied on smart card. This scheme suffers if the smart card is lost, and offline password guessing this was addressed by Jiang et al. [12]. Challa et al. [13] proposed a new signature based authenticated key exchange that uses elliptic curve cryptography. In 2020, Zhang et al. [14] proposed a lightweight authenticated key agreement protocol for Internet of Drones. But, we have found that the protocol is not secure against stolen smart card attack. In the last year 2021, Tanveer et al. [15] proposed a new robust authenticated key exchange for the Internet of Drones environment. This protocol verifies the user is legal or not and then it helps to establish a session key between the user and a specified drone for secure communication. But, this protocol is lightweight as compared to Zhang et al. [14]. In 2021, Hussain et al. [16] proposed lightweight user access to drone for smart city environment. But, this protocol [16] is vulnerable to drone capture attacks and it does not support forward secrecy discussed in Liu et al. [17].

1.1. Motivation and contribution

While designing an authenticated key agreement protocol, few factors play a major role because of asset constrained drone gadgets. Firstly, we need to design a secure authentication protocol for open channel communication. Secondly, with restricted assets, drones cannot execute complicated operations on expansive datasets, so at the same time we need efficiency. This motivates us to design a secure lightweight authenticated key agreement protocol for Internet of Drones. We have designed a low resource and proficient authenticated key agreement for the Internet of Drone, which consists of only a one-side hash function and bitwise XOR operations. This is much more efficient in terms of both computation and communication overheads.

1.2. Organization

Section 1 contains introduction and literature survey, and Section 2 provides some basic mathematical notations, terminologies, and basic definitions required for discussion and analysis of the proposed protocol given in Section 4. A detailed formal and informal security analysis has been given Section 5. The Section 6 contains a comparative study of proposed scheme with relevant ones. This section illustrates both computational and communication aspects of the proposed protocol. The last Section 7 concludes the remarks.

2. Preliminaries

In this section, we have illustrated some useful notations, symbols, and terminologies used in the proposed protocol (see Table 1). The symbol u_i or U_i denotes the ith user, and v_j or V_j denotes jth drone in the proposed protocol. The user is specified by d_i denotes identity of ith user, along with pw_i denotes the password of ith user, respectively. The symbol id_j denotes identity of jth drone, pid_i denotes masked secret identity of ith user, pid_j denotes masked secret identity of jth drone, respectively. The symbol msk denotes the master secret of the control server, and st_1 denotes the current time stamp used by the user.

Table 1
Notations used.

Notation	Description
u_i or U_i	ith user
v_j or V_j	jth drone
id_i	Identity of ith user
pw_i	Password of ith user
id_j	Identity of jth drone
pid_i	Masked identity of ith user
pid_j	Masked identity of jth drone
msk	Master secret of CS
st_1	Current time stamp
$h(.)$	Collision resistant hashing
$\oplus$	Bitwise XOR operation
.	Usual multiplication

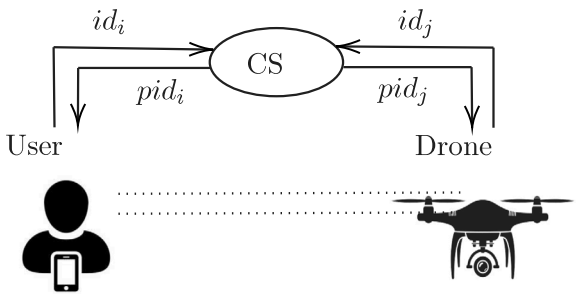


Fig. 1. A description of the proposed design with dotted line open channel and dark line secure channel.

2.1. Architecture

Fig. 1 shows the network model of our proposed plan or scheme. This model consists of three main components: Control Server(CS), end Users(U_i) and Drones(V_j).

(1) **Control Server**: Its job is to register the users and the drones before they are allowed to communicate with each other. To establish the connection between them the control server generates the secret key and issues them to users and drones based on their identity.
(2) **Users**: Users or End users using any portable electronic device gets the secret key from the control server after registering. The verification of the user takes place before the communication between them so that the data can be shared securely.
(3) **Drones**: Similar to the users, they also get their secret keys from the control server after registering. Once the user is verified, the connection is established and a secure channel is formed for their communication.

2.2. Defense requirements

We have seen the features of our proposed AKA scheme for Internet of Drone physical layout. But certain security requirements need to be taken care of before implementing it.

(1) **Mutual Authentication**: This is the process in which both the users and drones authenticate themselves before sending the messages through the network channel. It also authenticates their identity, which is transmitted along with the messages.
(2) **anonymity**: Our proposed plan or scheme should ensure that the user's identity is not publicly shared with anyone. Only the legal communicator to whom the user has registered will only have their details. If the attacker tries to get the information, then only the encrypted information will be seen.
(3) **Un-traceability**: Our scheme should assure the users and drones by providing un-traceability as security. If any adversary tries to gather the location of the drones or users by intercepting the network channel, the un-traceability feature prevents such actions from them.
(4) **Session Key Agreement**: Once the scheme is implemented effectively, the session key will be generated for further communication between users and drones. For other genuine users, who are not a part of this session, the adversary will not get any information even if they have the session key for the same.
(5) **Resistance against Various Attacks**: To prevent the loss of information or sharing of users' data, the scheme should be resistant to attacks like impersonation attacks, man-in-the-middle attacks, drone capture attacks, server spoofing attacks, modification attacks, replay attacks and known session key attacks.

3. A security analysis of Zhang et al.'s protocol [14]

This section describes Zhang et al. protocol [14] in brief. This is very helpful to analyze weaknesses in the protocol. At the last subsection of this section, we have shown how to perform attack on Zhang et al. protocol.

3.1. Setup phase

A control server is responsible for generating its master secret and other public parameters justified in the following steps:

(1) The control server randomly takes a 160 bits master secret, and then it chooses a 160 bits masking number κ and the public parameter n.
(2) In this step, the control server takes a secure collision resistant hashing $h : \{0, 1\}^* \rightarrow \{1, 1\}^{160}$, an identity id_{cs} and it computes pseudo identity $pid_{cs} = h(id_{cs} \| \kappa)$.
(3) Finally, the control server stores κ, and it publishes parameters $(n, pid_{cs}, h(.))$

From this, an adversary $\mathcal{A}$ gets the public parameters $(n, pid_{cs}, h(.))$.

3.2. Authentication and key agreement phase

After registration, a user u_i becomes the registered user and v_j is the registered drone. After this authentication phase, a user and drone can communicate securely with session key which is generated at the end of the authentication phase.

(1) First a user enters the identity id_i and password pw_i, then the mobile device uses this information, and it calculates $pid_i = pid_i^m \oplus h(id_i \| pw_i)$, and $\alpha_i = \alpha_i^m \oplus h(id_i \| pw_i)$, and the messages m_1, m_2, m_3, m_4, by using current timestamp st_1 and a 60 bits random number r. It computes the messages as $m_1 = h(pid_{cs} \| st_1) \oplus pd_i$, $m_2 = h(pid_i \| pid_{cs} \| \alpha_i) \oplus r_1$, $m_3 = h(pid_i \| pid_{cs} \| \alpha_i \| r_1) \oplus pid_j$, and $m_4 = h(pid_i \| pid_j \| pid_{cs} \| \alpha_i \| r_1)$. Finally, it sends $\langle m_1, m_2, m_3, m_4, st_1 \rangle$ to the server.
(2) The server receives authentication request $< m_1, m_2, m_3, m_4, st_1$ from the user and then it checks the user and validates time by checking whether $st_1 \leq \triangle t$, if the condition does not satisfy, then server stops the process and rejects the authentication request. If it satisfies the condition, then it furthers calculates pid_i' and gets α_i' and then it computes $pid_i' = m_1 \oplus h(pid_{cs} \| st_1)$, $r_1' = m_2 \oplus h(pid_i' \| pid_{cs} \| \alpha_i')$, $pid_j' = m_3 \oplus h(pid_i' \| pid_{cs} \| \alpha_i' \| r_1')$, and $m_4' = h(pid_i' \| pid_j' \| pid_{cs} \| \alpha_i' \| r_1')$. Now, the server checks whether $m_4' = m_4$ or not. If it is not equal, then it signals an error by rejecting the authentication request. If it is true, then it furthers computes $m_5 = h(pid_j' \| \alpha_j') \oplus r_1'$, $m_6 = h(pid_j' \| pid_{cs} \| \alpha_j' \| r_1') \oplus pid_i'$, and $m_7 = h(pid_i' \| pid_j' \| pid_{cs} \| \alpha_j' \| r_1')$. Finally, it sends $\langle m_5, m_6, m_7 \rangle$ to the v_j.
(3) The v_j receives messages $\langle m_5, m_6, m_7 \rangle$ from the server, and it calculates r_1'', pid_i'', m_7' and then it checks whether $m_7' = m_7$. If they are equal then it authenticates the server and randomly selects a number which is 160 bits where $r_2 \in z_n^*$ and then it computes $r_1'' = m_5 \oplus h(pid_j \| \alpha_j)$, $pid_i'' = m_6 \oplus h(pid_j \| pid_{cs} \| \alpha_j \| r_1'')$, $m_7' = h(pid_i'' \| pid_j \| pid_{cs} \| \alpha_j \| r_1'')$, $m_8 = h(pid_j \| pid_i'' \| r_1'') \oplus r_2$, and $m_9 = h(r_1'' \| r_2)$. Finally, it computes $sk_{ji} = h(pid_i'' \| pid_j \| pid_{cs} \| m_9)$, $m_{10} = h(pid_i'' \| pid_j \| pid_{cs} \| r_1'' \| r_2 \| m_9)$, and it sends m_8, m_{10} and sends it to the user u_i.
(4) The user u_i receives $\langle m_8, m_{10} \rangle$ from the v_j, and then u_i computes m_{10}' and then compares that with m_{10}. If $m_{10}' = m_{10}$ then user authenticates v_j and then generates a session key (sk_{ij}). If it is not equal then it will through an error by rejecting the authentication request sent by v_j. The computations done by the user u_i are $r_2' = m_8 \oplus h(pid_j \| pid_i \| r_1)$, $m_9' = h(r_1 \| r_2')$, $m_{10}' = h(pid_i \| pid_j \| pid_{cs} \| r_1 \| r_2')$, and $sk_{ij} = h(pid_i \| pid_j \| pid_{cs} \| m_9')$.

3.3. Attack on Zhang et al. authentication and key agreement protocol for Internet of Drones [14]

The Zhang et al. authentication and key agreement protocol for Internet of Drones suffers from stolen smart card attack, privileged insider attack, and usual login without verification of identity and password of the user.

(1) **Stolen Smart Card Attack:** The adversary $\mathcal{A}$ knows the public parameters $(n, pid_{cs}, h(.))$, and it gets the stolen smart card. Moreover, $\mathcal{A}$ has access to the public channel, and it gets the message $\langle m_1, m_2, m_3, m_4, st_1 \rangle$ from the public channel in the step(1) of authentication phase. Therefore, $\mathcal{A}$ can compute $m_1 \oplus h(pid_{cs} \| st_1) = pid_i$, and $pid_i^m \oplus pid_i = h(id_i \| pw_i)$ because the values $\alpha_i^m, pid_i^m, pid_j$ are stored in the smart card. The adversary $\mathcal{A}$ uses α_i^m, and $h(id_i \| pw_i)$ to compute $\alpha_i^m \oplus h(id_i \| pw_i) = \alpha_i$, then it can play the role of legal user. As a legal user, $\mathcal{A}$ chooses r_1' in place of r_1, and it computes the messages as $m_1 = h(pid_{cs} \| st_1) \oplus pd_i$, $m_2 = h(pid_i \| pid_{cs} \| \alpha_i) \oplus r_1'$, $m_3 = h(pid_i \| pid_{cs} \| \alpha_i \| r_1') \oplus pid_j$, and $m_4 = h(pid_i \| pid_j \| pid_{cs} \| \alpha_i \| r_1')$. Finally, it sends $\langle m_1, m_2, m_3, m_4, st_1 \rangle$ to the server. In the last step, $\mathcal{A}$ receives $\langle m_8, m_{10} \rangle$ from the v_j, and then it computes $r_2' = m_8 \oplus h(pid_j \| pid_i \| r_1')$, $m_9' = h(r_1' \| r_2')$, $m_{10}' = h(pid_i \| pid_j \| pid_{cs} \| r_1' \| r_2')$, and establishes a valid session key $sk_{ij} = h(pid_i \| pid_j \| pid_{cs} \| m_9')$ with the drone.

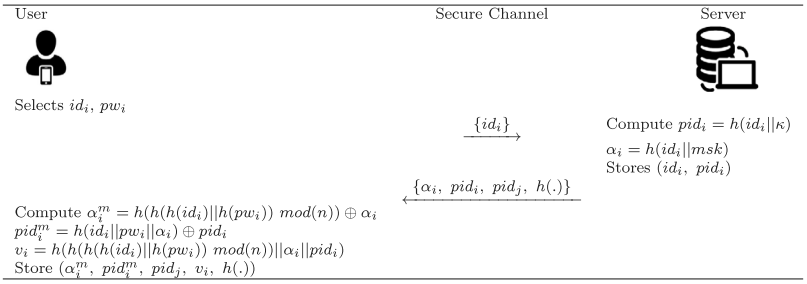


Fig. 2. Illustration of user registration phase.

(2) **Privileged Insider Attack:** A malicious privileged insider $\mathcal{A}$ has an access to the secure database of the server except the private key of the server. The database contains the information id_i, α_i, pid_i related to the user U_i. The insider $\mathcal{A}$ knows the public parameters $(n, pid_{cs}, h(.))$, and it gets $\langle m_1, m_2, m_3, m_4, st_1\rangle$ from the public channel sent from the user U_i. Now, it computes $m_2 \oplus h(pid_i \parallel pid_{cs} \parallel \alpha_i) = r_1$, and it tries to impersonate the user. As a legal user, $\mathcal{A}$ chooses r_1' in place of r_1, and it computes the messages as $m_1 = h(pid_{cs} \parallel st_1) \oplus pd_i$, $m_2 = h(pid_i \parallel pid_{cs} \parallel \alpha_i) \oplus r_1'$, $m_3 = h(pid_i \parallel pid_{cs} \parallel \alpha_i \parallel r_1') \oplus pid_j$, and $m_4 = h(pid_i \parallel pid_j \parallel pid_{cs} \parallel \alpha_i \parallel r_1')$. Finally, it sends $\langle m_1, m_2, m_3, m_4, st_1\rangle$ to the server. In the last step, $\mathcal{A}$ receives $\langle m_8, m_{10}\rangle$ from the v_j, and then it computes $r_2' = m_8 \oplus h(pid_j \parallel pid_i \parallel r_1')$, $m_9' = h(r_1' \parallel r_2')$, $m_{10}' = h(pid_i \parallel pid_j \parallel pid_{cs} \parallel r_1' \parallel r_2')$, and establishes a valid session key $sk_{ij} = h(pid_i \parallel pid_j \parallel pid_{cs} \parallel m_9')$ with the drone.

(3) **Login without Verification:** First a user enters the identity id_i and password pw_i, then the mobile device uses this information, and it computes $pid_i = pid_i^m \oplus h(id_i \parallel pw_i)$, and $\alpha_i = \alpha_i^m \oplus h(id_i \parallel pw_i)$, and sends $\langle m_1, m_2, m_3, m_4, st_1\rangle$ to the server. There is no verification for the user credential i.e. user's identity and password.

(4) **User Impersonation:** A malicious user $\mathcal{A}$ gets the messages m_1, m_2, m_3, m_4, st_1, and pid_{cs} is public parameter. He computes $m_1 \oplus h(pid_{cs} \parallel st_1) = pid_i$, and $pid_i \oplus pid_i^m = h(id_i \parallel pw_i)$ with the help of the smart card. Moreover, he can compute $\alpha_i = \alpha_i^m \oplus h(id_i \parallel pw_i)$ using the credential stored in the smart card. In this way, an adversary $\mathcal{A}$ can play a role of legal user.

4. Proposed modified authenticated key agreement protocol for Internet of Drones

This section describes the idea into four phases: (1) setup, (2) user registration, (3) drone registration, (4) authentication and key agreement, respectively.

4.1. Setup phase

A control server is responsible for generating its master secret and other public parameters justified in the following steps:

(1) The control server (CS) randomly takes a 160 bits master secret msk, and then it further chooses a 160 bits masking number κ and an integer $n = 160$ bits (system parameter).

(2) In this step, the control server takes a secure collision resistant hashing $h : \{0, 1\}^* \rightarrow \{0, 1\}^{160}$, an identity id_{cs} and it computes pseudo identity $pid_{cs} = h(id_{cs} \parallel \kappa)$.

(3) Finally, the control server stores $msk,\ \kappa$, and it publishes parameters $(n, pid_{cs}, h(.))$

4.2. User registration phase

In this phase, the user u_i registers through a secure channel. The essential steps to proceed with registration are shown in Fig. 2.

(1) The user selects id_i, pw_i, and he sends id_i to the control server. The server computes $pid_i = h(id_i \parallel \kappa)$, $\alpha_i = h(id_i \parallel msk)$, where κ is a random number generated during setup phase, and msk is the master secret of the server. The server stores $(id_i,\ pid_i)$ in its database, and it sends $\{\alpha_i,\ pid_i,\ pid_j,\ h(.)\}$ to the user. This step does not store α_i that makes it more secure than Zhang et al.'s scheme.

(2) In this step, the user have adopted the "fuzzy verifier" technique by Wang et al. [18] and computed the parameter $\alpha_i^m = h(h(h(id_i) \parallel h(pw_i))\ mod(n)) \oplus \alpha_i$, $pid_i^m = h(id_i \parallel pw_i \parallel \alpha_i) \oplus pid_i$, and a verification factor $v_i = h(h(h(h(id_i) \parallel h(pw_i))\ mod(n)) \parallel \alpha_i \parallel pid_i)$, and he stores $\{\alpha_i^m,\ pid_i^m,\ pid_j,\ v_i,\ h(.)\}$.

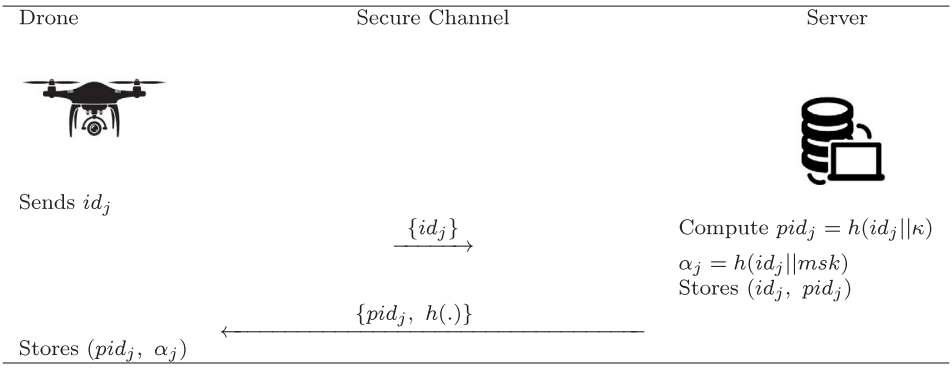


Fig. 3. Illustration of drone registration phase.

4.3. Drone registration phase

In this phase, the drone v_j registers through a secure channel. The essential steps to proceed with registration are shown in Fig. 3. We have assumed drone is a temper proof device.

(1) The drone sends id_j to the control server, where the identity of the drone may be its serial number/fixed string.
(2) The server computes $pid_j = h(id_j \parallel \kappa)$, $\alpha_j = h(id_j \parallel msk)$, and it stores $(id_j,\ pid_j)$, and it sends $pid_j,\ h(.)$ to the drone. Finally, the drone stores $(pid_j,\ \alpha_j)$ inside it.

4.4. Authentication and key agreement phase

After registration, a user u_i becomes the registered user and v_j is the registered drone. After this authentication phase, a user and drone can communicate securely with session key, which is generated at the end of the authentication phase (see Fig. 4).

(1) First, a user enters the identity id_i and password pw_i, then the mobile device uses this information, and it computes $\alpha_i = \alpha_i^m \oplus h(h(h(id_i) \parallel h(pw_i))\ mod(n))$, $pid_i = pid_i^m \oplus h(id_i \parallel pw_i \parallel \alpha_i)$, then it verifies the user by checking $v_i = h(h(h(h(id_i) \parallel h(pw_i))\ mod(n)) \parallel \alpha_i \parallel pid_i)$. After verification, it computes the messages m_1, m_2, m_3, m_4, by using current timestamp st_1 and a 160 bits random number r. It computes $m_1 = h(pid_{cs} \parallel st_1) \oplus pid_i$, $m_2 = h(pid_i \parallel pid_{cs} \parallel \alpha_i) \oplus r_1$, $m_3 = h(pid_i \parallel pid_{cs} \parallel \alpha_i \parallel r_1) \oplus pid_j$, and $m_4 = h(pid_i \parallel pid_j \parallel pid_{cs} \parallel \alpha_i \parallel r_1)$. Finally, it sends $\langle m_1, m_2, m_3, m_4, st_1 \rangle$ to the server.
(2) The control server receives authentication request $< m_1, m_2, m_3, m_4, st_1$ from the user, then it checks the user and validates time by checking time stamp whether $st_1 \leq \Delta\ t$, if the condition is not satisfied, then server stops the process and rejects the authentication request. If the condition is satisfied, then it furthers computes $pid_i' = m_1 \oplus h(pid_{cs} \parallel st_1)$, and it searches corresponding identity in its database. If the masked identity pid_i exists in its database, then it fetches the corresponding identity, and it computes $\alpha_i' = h(id_i \parallel msk)$ gets α_i' and it computes $r_1' = m_2 \oplus h(pid_i' \parallel pid_{cs} \parallel \alpha_i')$, $pid_j' = m_3 \oplus h(pid_i' \parallel pid_{cs} \parallel \alpha_i' \parallel r_1')$, and $m_4' = h(pid_i' \parallel pid_j' \parallel pid_{cs} \parallel \alpha_i' \parallel r_1')$. Now, the control server checks whether $m_4' = m_4$ or not. If it is not equal, then it signals an error by rejecting the authentication request. If it is true, then it searches real identity of a drone and masked identity pid_j' in its database. Furthers, it computes $\alpha_j' = h(id_j' \parallel msk)$, $m_5 = h(pid_j' \parallel \alpha_j') \oplus r_1'$, $m_6 = h(pid_j' \parallel pid_{cs} \parallel \alpha_j' \parallel r_1') \oplus pid_i'$, and $m_7 = h(pid_i' \parallel pid_j' \parallel pid_{cs} \parallel \alpha_j' \parallel r_1')$. Finally, it sends $\langle m_5, m_6, m_7 \rangle$ to the v_j.
(3) The v_j receives messages $\langle m_5, m_6, m_7 \rangle$ from the server, and it calculates r_1'', pid_i'', m_7' and then it checks whether $m_7' = m_7$. If they are equal then it authenticates the server and randomly selects a number which is 160 bits where r_2 and then it computes $r_1'' = m_5 \oplus h(pid_j \parallel \alpha_j)$, $pid_i'' = m_6 \oplus h(pid_j \parallel pid_{cs} \parallel \alpha_j \parallel r_1'')$, $m_7' = h(pid_i'' \parallel pid_j \parallel pid_{cs} \parallel \alpha_j \parallel r_1'')$, $m_8 = h(pid_j \parallel pid_i'' \parallel r_1'') \oplus r_2$, and $m_9 = h(r_1'' \parallel r_2)$. Finally, it computes $sk_{ji} = h(pid_i'' \parallel pid_j \parallel pid_{cs} \parallel m_9)$, $m_{10} = h(pid_i'' \parallel pid_j \parallel pid_{cs} \parallel r_1'' \parallel r_2 \parallel m_9)$, and it sends m_8, m_{10} and sends it to the user u_i.
(4) The user u_i receives $\langle m_8, m_{10} \rangle$ from the v_j, and then u_i computes m_{10}' and then compares that with m_{10}. If $m_{10}' = m_{10}$ then user authenticates v_j and then generates a session key (sk_{ij}). If it is not equal, then it will through an error by rejecting the authentication request sent by v_j. The computations done by the user u_i are $r_2' = m_8 \oplus h(pid_j \parallel pid_i \parallel r_1)$, $m_9' = h(r_1 \parallel r_2')$, $m_{10}' = h(pid_i \parallel pid_j \parallel pid_{cs} \parallel r_1 \parallel r_2')$, and $sk_{ij} = h(pid_i \parallel pid_j \parallel pid_{cs} \parallel m_9')$.

5. Security analysis

In this part, there will be a detailed analysis of our proposed plan or scheme. Firstly, we will show how secure our scheme is when we will implement it using the random oracle model. After this, we will be discussing about how our scheme can fulfill the security requirements as mentioned below.

Fig. 4. A brief overview of authentication phase.

5.1. Formal security analysis

5.1.1. Password guessing

First, a user enters the identity id_i and password pw_i, then the mobile device uses this information, and it computes $\alpha_i = \alpha_i^m \oplus h(h(h(id_i) \parallel h(pw_i))\ mod(n))$, $pid_i = pid_i^m \oplus h(id_i \parallel pw_i \parallel \alpha_i)$, then it verifies the user by checking $v_i = h(h(h(h(id_i) \parallel h(pw_i))\ mod(n)) \parallel \alpha_i \parallel pid_i)$. Therefore, this is hard to find password from $h(h(h(id_i) \parallel h(pw_i))\ mod(n))$ because of collision resistant hashing. Another stored value is $v_i = h(h(h(h(id_i) \parallel h(pw_i))\ mod(n)) \parallel \alpha_i \parallel pid_i)$ which is again hashed with pw_i, and pid_i only known to the user.

5.1.2. Stolen device attack

The user selects id_i, pw_i, and he sends id_i to the control server. The server computes $pid_i = h(id_i \parallel \kappa)$, $\alpha_i = h(id_i \parallel msk)$, where κ is a random number generated during setup phase, and msk is the master secret of the server. The server stores $(id_i,\ pid_i)$ in its database, and it sends $\{\alpha_i,\ pid_i,\ pid_j,\ h(.)\}$ to the user. The user computes $\alpha_i^m = h(h(h(id_i) \parallel h(pw_i))\ mod(n)) \oplus \alpha_i$, $pid_i^m = h(id_i \parallel pw_i \parallel \alpha_i) \oplus pid_i$, and a verification factor $v_i = h(h(h(h(id_i) \parallel h(pw_i))\ mod(n)) \parallel \alpha_i \parallel pid_i)$, and he stores $\{\alpha_i^m,\ pid_i^m,\ pid_j,\ v_i,\ h(.)\}$ in the device. But α_i, pid_i, and pw_i only known to the user implies safety against stolen device attacks.

5.1.3. Anonymity

First, a user enters the identity id_i and password pw_i, then the mobile device uses this information, and it computes $\alpha_i = \alpha_i^m \oplus h(h(h(id_i) \parallel h(pw_i))\ mod(n))$, $pid_i = pid_i^m \oplus h(id_i \parallel pw_i \parallel \alpha_i)$, then it verifies the user by checking $v_i = h(h(h(h(id_i) \parallel$

$h(pw_i)) \ mod(n)) \parallel \alpha_i \parallel pid_i)$. Furthers, it computes the messages m_1, m_2, m_3, m_4, by using current timestamp st_1 and a 60 bits random number r. It computes $m_1 = h(pid_{cs} \parallel st_1) \oplus pid_i$, $m_2 = h(pid_i \parallel pid_{cs} \parallel \alpha_i) \oplus r_1$, $m_3 = h(pid_i \parallel pid_{cs} \parallel \alpha_i \parallel r_1) \oplus pid_j$, and $m_4 = h(pid_i \parallel pid_j \parallel pid_{cs} \parallel \alpha_i \parallel r_1)$. Finally, it sends $\langle m_1, m_2, m_3, m_4, st_1 \rangle$ to the server. Therefore, this is hard to find identity from $m_1 = h(pid_{cs} \parallel st_1) \oplus pid_i$, $m_2 = h(pid_i \parallel pid_{cs} \parallel \alpha_i) \oplus r_1$, $m_3 = h(pid_i \parallel pid_{cs} \parallel \alpha_i \parallel r_1) \oplus pid_j$, and $m_4 = h(pid_i \parallel pid_j \parallel pid_{cs} \parallel \alpha_i \parallel r_1)$.

5.1.4. Server impersonation

The control server receives authentication request $< m_1, m_2, m_3, m_4, st_1$ from the user, then it checks the user and validates time by checking time stamp whether $st_1 \leq \triangle t$, if the condition is not satisfied, then server stops the process and rejects the authentication request. If the condition is satisfied, then it furthers calculates pid_i' and gets α_i' and then it computes $pid_i' = m_1 \oplus h(pid_{cs} \parallel st_1)$, $\alpha_i = h(id_i \parallel msk)$, $r_1' = m_2 \oplus h(pid_i' \parallel pid_{cs} \parallel \alpha_i')$, $pid_j' = m_3 \oplus h(pid_i' \parallel pid_{cs} \parallel \alpha_i' \parallel r_1')$, and $m_4' = h(pid_i' \parallel pid_j' \parallel pid_{cs} \parallel \alpha_i' \parallel r_1')$. Now, the server checks whether $m_4' = m_4$ or not. If it is not equal, then it signals an error by rejecting the authentication request. If it is true, then it furthers computes $m_5 = h(pid_j' \parallel \alpha_j') \oplus r_1'$, $m_6 = h(pid_j' \parallel pid_{cs} \parallel \alpha_j' \parallel r_1') \oplus pid_i'$, and $m_7 = h(pid_i' \parallel pid_j' \parallel pid_{cs} \parallel \alpha_j' \parallel r_1')$. Finally, it sends $\langle m_5, m_6, m_7 \rangle$ to the υ_j. Therefore, $\mathcal{A}$ cannot impersonate the server without knowing random number κ, and the master secret.

5.1.5. User impersonation

First, a user enters the identity id_i and password pw_i, then the mobile device uses this information, and it computes $\alpha_i = \alpha_i^m \oplus h(h(h(id_i) \parallel h(pw_i)) \ mod(n))$, $pid_i = pid_i^m \oplus h(id_i \parallel pw_i \parallel \alpha_i)$, then it verifies the user by checking $\upsilon_i = h(h(h(h(id_i) \parallel h(pw_i)) \ mod(n)) \parallel \alpha_i \parallel pid_i)$. Furthers, it computes the messages m_1, m_2, m_3, m_4, by using current timestamp st_1 and a 60 bits random number r. It computes $m_1 = h(pid_{cs} \parallel st_1) \oplus pid_i$, $m_2 = h(pid_i \parallel pid_{cs} \parallel \alpha_i) \oplus r_1$, $m_3 = h(pid_i \parallel pid_{cs} \parallel \alpha_i \parallel r_1) \oplus pid_j$, and $m_4 = h(pid_i \parallel pid_j \parallel pid_{cs} \parallel \alpha_i \parallel r_1)$. Finally, it sends $\langle m_1, m_2, m_3, m_4, st_1 \rangle$ to the server. Therefore, it is not possible to impersonate a user without knowing pid_i, pw_i, id_i, and α_i.

5.1.6. Session key agreement

First, a user enters the identity id_i and password pw_i, then the mobile device uses this information, and it computes $\alpha_i = \alpha_i^m \oplus h(h(h(id_i) \parallel h(pw_i)) \ mod(n))$, $pid_i = pid_i^m \oplus h(id_i \parallel pw_i \parallel \alpha_i)$, then it verifies the user by checking $\upsilon_i = h(h(h(h(id_i) \parallel h(pw_i)) \ mod(n)) \parallel \alpha_i \parallel pid_i)$. Furthers, it computes the messages m_1, m_2, m_3, m_4, by using current timestamp st_1 and a 60 bits random number r. It computes $m_1 = h(pid_{cs} \parallel st_1) \oplus pid_i$, $m_2 = h(pid_i \parallel pid_{cs} \parallel \alpha_i) \oplus r_1$, $m_3 = h(pid_i \parallel pid_{cs} \parallel \alpha_i \parallel r_1) \oplus pid_j$, and $m_4 = h(pid_i \parallel pid_j \parallel pid_{cs} \parallel \alpha_i \parallel r_1)$. Finally, it sends $\langle m_1, m_2, m_3, m_4, st_1 \rangle$ to the server. The server calculates pid_i' and gets α_i' and then it computes $pid_i' = m_1 \oplus h(pid_{cs} \parallel st_1)$, $r_1' = m_2 \oplus h(pid_i' \parallel pid_{cs} \parallel \alpha_i')$, $pid_j' = m_3 \oplus h(pid_i' \parallel pid_{cs} \parallel \alpha_i' \parallel r_1')$, and $m_4' = h(pid_i' \parallel pid_j' \parallel pid_{cs} \parallel \alpha_i' \parallel r_1')$. Now, the server checks whether $m_4' = m_4$ or not. If it is not equal, then it signals an error by rejecting the authentication request. If it is true, then it furthers computes $m_5 = h(pid_j' \parallel \alpha_j') \oplus r_1'$, $m_6 = h(pid_j' \parallel pid_{cs} \parallel \alpha_j' \parallel r_1') \oplus pid_i'$, and $m_7 = h(pid_i' \parallel pid_j' \parallel pid_{cs} \parallel \alpha_j' \parallel r_1')$. Finally, it sends $\langle m_5, m_6, m_7 \rangle$ to the υ_j. The υ_j receives messages $\langle m_5, m_6, m_7 \rangle$ from the server, and it calculates r_1'', pid_i'', m_7' and then it checks whether $m_7' = m_7$. If they are equal then it authenticates the server and randomly selects a number which is 160 bits where $r_2 \in z_n^*$ and then it computes $r_1'' = m_5 \oplus h(pid_j \parallel \alpha_j)$, $pid_i'' = m_6 \oplus h(pid_j \parallel pid_{cs} \parallel \alpha_j \parallel r_1'')$, $m_7' = h(pid_i'' \parallel pid_j \parallel pid_{cs} \parallel \alpha_j \parallel r_1'')$, $m_8 = h(pid_j \parallel pid_i'' \parallel r_1'') \oplus r_2$, and $m_9 = h(r_1'' \parallel r_2)$. Finally, it computes $sk_{ji} = h(pid_i'' \parallel pid_j \parallel pid_{cs} \parallel m_9)$, $m_{10} = h(pid_i'' \parallel pid_j \parallel pid_{cs} \parallel r_1'' \parallel r_2 \parallel m_9)$, and it sends m_8, m_{10} and sends it to the user u_i. The user u_i receives $\langle m_8, m_{10} \rangle$ from the υ_j, and then u_i computes m_{10}' and then compares that with m_{10}. If $m_{10}' = m_{10}$ then user authenticates υ_j and then generates a session key (sk_{ij}). If it is not equal, then it will through an error by rejecting the authentication request sent by υ_j. The computations done by the user u_i are $r_2' = m_8 \oplus h(pid_j \parallel pid_i \parallel r_1)$, $m_9' = h(r_1 \parallel r_2')$, $m_{10}' = h(pid_i \parallel pid_j \parallel pid_{cs} \parallel r_1 \parallel r_2')$, and $sk_{ij} = h(pid_i \parallel pid_j \parallel pid_{cs} \parallel m_9')$. Therefore, a user and a specified drone establish a secure session key.

5.1.7. Man in the middle attack

First, a user enters the identity id_i and password pw_i, then the mobile device uses this information, and it computes $\alpha_i = \alpha_i^m \oplus h(h(h(id_i) \parallel h(pw_i)) \ mod(n))$, $pid_i = pid_i^m \oplus h(id_i \parallel pw_i \parallel \alpha_i)$, then it verifies the user by checking $\upsilon_i = h(h(h(h(id_i) \parallel h(pw_i)) \ mod(n)) \parallel \alpha_i \parallel pid_i)$. Furthers, it computes the messages m_1, m_2, m_3, m_4, by using current timestamp st_1 and a 60 bits random number r. It computes $m_1 = h(pid_{cs} \parallel st_1) \oplus pid_i$, $m_2 = h(pid_i \parallel pid_{cs} \parallel \alpha_i) \oplus r_1$, $m_3 = h(pid_i \parallel pid_{cs} \parallel \alpha_i \parallel r_1) \oplus pid_j$, and $m_4 = h(pid_i \parallel pid_j \parallel pid_{cs} \parallel \alpha_i \parallel r_1)$. Finally, it sends $\langle m_1, m_2, m_3, m_4, st_1 \rangle$ to the server. The server calculates pid_i' and gets α_i' and then it computes $pid_i' = m_1 \oplus h(pid_{cs} \parallel st_1)$, $r_1' = m_2 \oplus h(pid_i' \parallel pid_{cs} \parallel \alpha_i')$, $pid_j' = m_3 \oplus h(pid_i' \parallel pid_{cs} \parallel \alpha_i' \parallel r_1')$, and $m_4' = h(pid_i' \parallel pid_j' \parallel pid_{cs} \parallel \alpha_i' \parallel r_1')$. Now, the server checks whether $m_4' = m_4$ or not. If it is not equal, then it signals an error by rejecting the authentication request. If it is true, then it furthers computes $m_5 = h(pid_j' \parallel \alpha_j') \oplus r_1'$, $m_6 = h(pid_j' \parallel pid_{cs} \parallel \alpha_j' \parallel r_1') \oplus pid_i'$, and $m_7 = h(pid_i' \parallel pid_j' \parallel pid_{cs} \parallel \alpha_j' \parallel r_1')$. Finally, it sends $\langle m_5, m_6, m_7 \rangle$ to the υ_j. The υ_j receives messages $\langle m_5, m_6, m_7 \rangle$ from the server, and it calculates r_1'', pid_i'', m_7' and then it checks whether $m_7' = m_7$. If they are equal then it authenticates the server and randomly selects a number which is 160 bits where $r_2 \in z_n^*$ and then it computes $r_1'' = m_5 \oplus h(pid_j \parallel \alpha_j)$, $pid_i'' = m_6 \oplus h(pid_j \parallel pid_{cs} \parallel \alpha_j \parallel r_1'')$, $m_7' = h(pid_i'' \parallel pid_j \parallel pid_{cs} \parallel \alpha_j \parallel r_1'')$, $m_8 = h(pid_j \parallel pid_i'' \parallel r_1'') \oplus r_2$, and $m_9 = h(r_1'' \parallel r_2)$. Finally, it computes $sk_{ji} = h(pid_i'' \parallel pid_j \parallel pid_{cs} \parallel m_9)$, $m_{10} = h(pid_i'' \parallel pid_j \parallel pid_{cs} \parallel r_1'' \parallel r_2 \parallel m_9)$, and it sends m_8, m_{10} and sends it to the user u_i. The user u_i receives $\langle m_8, m_{10} \rangle$ from the υ_j, and then u_i computes m_{10}' and then compares that with m_{10}. If $m_{10}' = m_{10}$ then user authenticates υ_j and then generates a session key (sk_{ij}). If it is not equal, then it will through an error by rejecting the authentication request sent by υ_j. The computations done by the user u_i are $r_2' = m_8 \oplus h(pid_j \parallel pid_i \parallel r_1)$, $m_9' = h(r_1 \parallel r_2')$, $m_{10}' = h(pid_i \parallel pid_j \parallel pid_{cs} \parallel r_1 \parallel r_2')$. Therefore, this is not possible to execute man in the middle attack without knowing server master secret, user credentials id_i, pid_i, pw_i, and α_i.

5.2. Security model

To explain the security of our plan we will take a reference from Zhang et al. [14]. Based on his work, we intend to show the security model in which an adversary $\mathcal{A}$ and a challenger C play a game with each other. The adversary will imitate a Turing machine. This machine works on probability polynomial time and produces non-deterministic results. While on the other hand challenger imitates all the oracles, which generates a random response. $\prod_{\Lambda}^{t}$ represents the occurrence of tth players $\Lambda \in (U_i, CS, V_j)$. The adversary $\mathcal{A}$ is permitted by the oracle to ask multiple queries from them. As a result, the machine will give the response accordingly.

(1) $h(t)$: This term represents the hash oracle. It manages the hash list L_h. When the adversary $\mathcal{A}$ tries to run a hash query along with some message x then the challenger C is responsible for verifying the message if it present in the hash list L_h or not. If it is present, then the result will be returned by the challenger in the form of hashed message $h(t)$. If not present, then a number X is chosen by the challenger from $X \in Z_n^*$ and it will return it. After returning, the same number (x, X) is stored in the hash list L_h by the challenger C.
(2) $Extract(ID_i)$: The method of destroying the integrity of an authorized drone is explained in this query. This is possible when the adversary $\mathcal{A}$ somehow got the secret key. For achieving the secret key the adversary may use this method on the identity of the drone ID_i and as a result, the challenger C returns the secret key.
(3) $Send(\prod_{\Lambda}^{t}, M)$: In this query, an active attack can be performed by the attacker $\mathcal{A}$. When the adversary tries to send any message to the instance of players $\prod_{\Lambda}^{t}$, it gets the response from the instance with the message. For the new instances, this process can be performed by sending the $Send(\prod_{\Lambda}^{t}, Start)$to the oracle machine.
(4) $Reveal(\prod_{\Lambda}^{t})$: This query tells us what will happen if we try to use the session key incorrectly. If the adversary $\mathcal{A}$ somehow gets the desired instance, then the challenger C will return the session key of that particular instance $\prod_{\Lambda}^{t}$. Or else it will return L.
(5) $Execute(Ui, Vj)$: In this query, the adversary $\mathcal{A}$gets all the secret information by intercepting the messages which are sent through a public channel.
(6) $Test(\prod_{\Lambda}^{t})$: This query is used by the adversary $\mathcal{A}$ to identify the difference between the original session key and the random secret key Here, the adversary $\mathcal{A}$ has one constraint for this query, which can be executed only once. A random bit $b \in 0, 1$ is chosen by the challenger C and if b = 1, the original session key is returned to the adversary $\mathcal{A}$. If b = 0 then a random secret key is returned. While for all other cases, the challenger C returns $\perp$ to the adversary $\mathcal{A}$.

The adversary $\mathcal{A}$ can perform these queries after executing the test query in the following sequence: $Extract, Send, Reveal, Execute$. At this stage, the restriction is there because the adversary $\mathcal{A}$ cannot perform the $Reveal$ query for the oracle and its pattern. They already executed when the $Test$ query was under the running stage. Finally, the adversary $\mathcal{A}$ can break the AKA and win the game if $b' = b$. For calculating the advantage of the adversary we can use :

$$adv_{\Sigma}^{AKA}(A) = |2Pr[b' - b] - 1|$$

Definition 5.1 (*Secure-AKA*)**.** If there is no probability polynomial adversary $\mathcal{A}$ can win this game with the major advantage $adv_{\Sigma}^{AKA}(A)$, then we can say that our scheme is Secure-AKA.

If the adversary $\mathcal{A}$ can make an illegal copy of the login, response or communication message then the mutual authentication of our scheme will be compromised. Let E_{U-CS} represent adversary $\mathcal{A}$ is imitating U_i and creates the login message which is validated by CS. Let E_{U-V} represent adversary $\mathcal{A}$ is imitating V_i and creates the response message which is validated by U_i. The advantage for this is defined as follows:

$$adv_{\Sigma}^{MA}(A) = Pr[E_{U-CS}] + Pr[E_{U-V}]$$

Definition 5.2 (*Secure-MA*)**.** If there is no probability polynomial adversary $\mathcal{A}$ can win this game with the major advantage $adv_{\Sigma}^{MA}(A)$, then we can say that our scheme is Secure-MA.

5.3. Provable security

We will prove that in non-negligible probability, no adversary exists who can imitate the legal login and the response message. What we try to say is that our proposed scheme has secure AKA and MA and is safe to use.

Lemma 5.3. *Assume that there is a probability polynomial adversary $\mathcal{A}$ who can find the legal login or response message with non-negligible probability. Then there exists a challenger who can guess the 160-bit random number with a non-negligible probability.*

Proof. Suppose challenger C chooses the 160-bit random number and sends the parameters to the adversary. After this challenger will generate the hash list, which will be empty so that the inputs and outputs of the hash oracles would be recorded. Then the challenger chooses two challenge drone identities at the initial stage. After the hash oracles, other oracles are queried and their answers are as follows:

(1) h(x): Challenger checks if x exists in the hash list or not. If it is present then the challenger returns X to the adversary. If it is not present then the challenger chooses a number X, then adds (x, X) to the hash list and returns X to the adversary.
(2) Extract: If i is not equal to I, then the challenger finds a tuple in the hash list and returns alpha to the adversary. If it is equal, then the challenger rejects the query and ends the game.
(3) Send: To simulate the active attack in four different formats this query is launched by the adversary. Send Challenger checks if i is not equal to me. Then challenger finds the hash list for users' secret key alpha if they are equal. The secret key is then used by the challenger to choose the random number, and the current time stamp computes M. If they are not the same, then the challenger chooses R and sets M.
(4) Send: Once the message is received, the challenger verifies if j and J are the same or not. If same, then challenger rejects it and chooses two numbers R and sets M. If not same, then challenger finds the hash list for secret key alpha and continues the process. Send Challenger checks if j is not equal to J. If yes, then the challenger finds the hash list of the drones' secret key. Now challenger chooses a random number with the help of this secret key and computes M. If they are different then the challenger chooses three numbers, sets and returns M to the user.
(5) Reveal: Challenger returns the correct session key if the instance is approved. If not approved then the challenger returns inverse.

Let us assume $\mathcal{A}$ can compute a legal login request that means the message reply (m_1, m_2, m_3, m_4) is sent to $(\pi^t_{U_i}$, start) query with initialization $i = I$ and (m_8, m_{10}), and send query $(\pi^k_{V_j}, m_5, m_6, m_7)$ starting with $j = J$ are submitted to CS and U_i for verification. To compute the advantage of C the events are (1) e_1 denotes simulation without abortion, (2) e_2 denotes a legal submission of login request (m_1, m_2, m_3, m_4) with the help of Send$(\pi^t_{U_i}$, start) queries or valid response (m_8, m_{10}) with the help of send$(\pi^k_{V_j}, m_5, m_6, m_7)$ queries, at the same time $Extract(id_I)$ and $Extract(id_J)$ those are never queried, (3) e_3 denotes $U_i = U_I$ or $V_j = V_J$, and (4) e_4 denotes that C chooses the correct entry from the list L_h. Let us consider furthers q_s, q_{L_s}, and q_{L_h} represents sends queries, L_s-queries, and L_h-queries submitted by $\mathcal{A}$. This implies $Pr[e_1] \geq \frac{1}{q_s}$, then obviously $Pr[e_2 \parallel e_1] \geq \epsilon$, $Pr[e_3 \parallel (e_2 \wedge e_1)] \geq \frac{1}{q_{L_s}}$, $Pr[e_4 \parallel (e_3 \wedge e_2 \wedge e_1)] \geq \frac{1}{q_{L_s}}\frac{1}{q_{L_s}} + \frac{a}{q_{L_h}}\frac{b}{q_{L_h}-a}$ in which, we denote "a" is a legal number in $Sends(\pi^t_{U_i}, start)$-queries and "$b$" is corresponding legal entry in $Send(\pi^t_{U_i}, (m_8, m_{10}))$-queries. Therefore, C makes a successful guess of 160 bits random number with probability $Pr[e_1 \wedge e_2 \wedge e_3 \wedge e_4] = Pr[e_4 \parallel e_3 \wedge e_2 \wedge e_1]Pr[e_3 \parallel e_2 \wedge e_1]Pr[e_2 \parallel e_1]Pr[e_1] = \frac{1}{q_s}\frac{1}{q_{L_s}}(\frac{1}{q_{L_s}}\frac{1}{q_{L_s}-1} + \frac{a}{q_{L_h}}\frac{b}{q_{L_h}-a})\epsilon$. □

Theorem 5.4. *The proposed scheme has Secure-MA under guessing 160 bits random number is hard. From the lemma, no adversary can generate the legal login or response message because guessing a 160-bit random number is very difficult. So we can say that our proposed scheme is secure.*

Theorem 5.5. *The proposed scheme has Secure Authenticated Key Agreement under the assumption that guessing 160-bits random number is hard.*

Proof. After executing the test query assume that the probability polynomial adversary gets the correct $b' = b$ with a non-negligible probability. Then the challenger can guess the 160-bit random number. The advantage of the challenger is defined with the help of certain events (1) E_{sk} denotes adversary receives the correct session key after the query of test-query, (2) E_u denotes successfully executed a test query to an instance by $\mathcal{A}$, (3) E_v denotes successfully executes a test query to an instance by $\mathcal{A}$, and (4) E_{u-cs-v} denotes the authentication between user and drone, and the user and control server can be exploited by the adversary. We have already observed that probability $\mathcal{A}$ makes a correct guess of "a", and "b" without any useful information is $\frac{1}{2}$, therefore, we have $Pr[e_{sk}] \geq \frac{\epsilon}{2}$ that implies $Pr[e_{sk}] = Pr[e_{sk} \wedge e_V] + Pr[e_{sk} \wedge e_V \wedge e_{U-CS-V}] + Pr[e_{sk} \wedge e_V \wedge e_{U-CS-V}] \leq Pr[e_{sk} \wedge e_U] + Pr[e_{U-CS-V}] + Pr[e_{sk} \wedge e_V \wedge e_{U-CS-V}]$. In this observation, we have $Pr[e_{sk} \wedge e_U] + Pr[e_{sk} \wedge e_V \wedge e_{U-CS-V}] \geq Pr[e_{sk}] - Pr[e_{U-CS-V}] \geq \frac{\epsilon}{2} - Pr[e_{U-CS-V}]$, with probability $Pr[e_V \wedge e_{U-CS-V}] = Pr[e_V]$, that implies $Pr[e_{sk} \wedge e_V] \geq \frac{\epsilon}{4} - \frac{Pr[e_{U-CS-V}]}{2}$. The event $e_{sk} \wedge e_{V_i}$ shows $\mathcal{A}$ impersonates user U_i and establishes a legal session. With the help of the Lemma 5.3, $Pr[e_{U-CS-V}]$ is a negligible probability, so that $\frac{\epsilon}{4} - \frac{Pr[e_{U-CS-V}]}{2}$ is non negligible. □

6. Performance analysis

Here, we have evaluated the performance of the proposed algorithm model in the area of computation costs and communication costs. We also compare the execution time of various operations and results of our proposed model with Wazid et al.'s algorithm and Singh et al.'s algorithm. We have followed notations t_f : time for executing an extraction, t_h : time for executing hash, t_{exp} : time for executing exponentiation, and t_{mul} : time for executing multiplication. These notations are used to signify the execution time in this paper. All the above operations are executed between a mobile(drone) and a computer. Mobile is used in the simulation instead of drone in the simulation as they both have similar technology. Here in the place of drone a Samsung Galaxy S5 with the specifications of Quad-core 2.45G processor, 2G bytes memory, Android 4.4.2 operating system is used and a desktop computer with the specifications of I5-4460S 2.90 GHz processor, 4G bytes memory and windows 8 operating system are used for the server simulation (see Table 2). We have considered a cyclic group G of order n is used to attain enough security for 1024 bits RSA.

In **Wazid et al.'s protocol**, on the user side sixteen hash and fuzzy extraction need to be executed, while on the drone side seven of them need to be executed and the server need to execute eight hash functions. Therefore, user's execution time $= t_f + 16t_h =$

Table 2
An overview of cost in milliseconds.

Symbols	User-side computation cost	Drone-side computation cost	Server-side computation cost
t_f	13.045	13.045	5.427
t_h	0.056	0.056	0.007
t_{exp}	2.249	2.249	0.339
t_{mul}	0.008	0.008	0.001

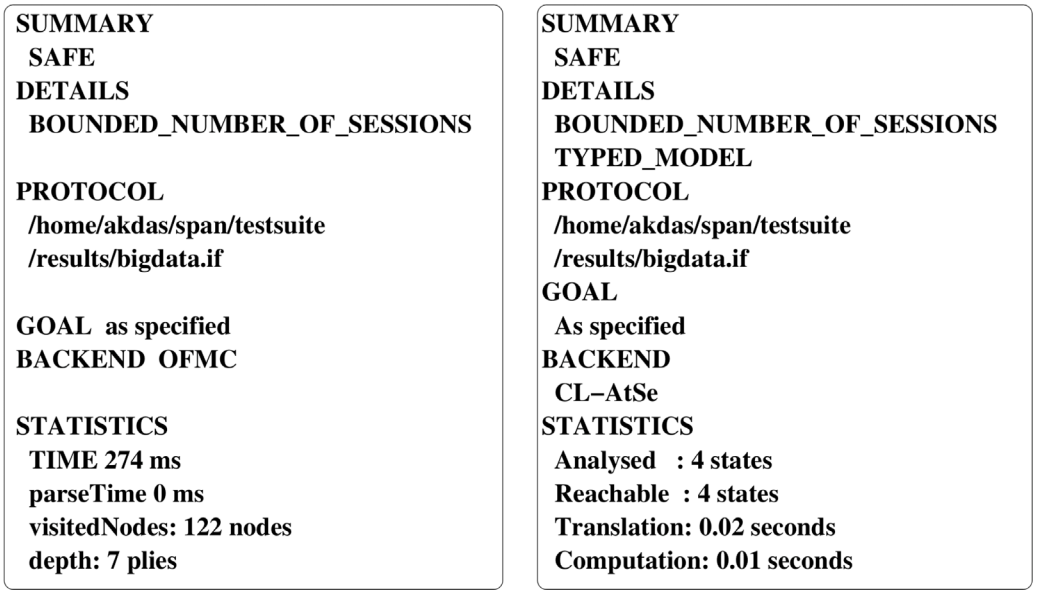


Fig. 5. Proposed protocol implementation in Automated Validation of Internet Security Protocols and Applications tool.

14.301 ms, drone side execution time = $7t_h = 0.392$ ms, server side execution time = $8t_h = 0.056$ ms. In **Singh et al.'s protocol**, the server is not involved in authentication, only the drone and user are involved in execution process. Where user need to compute two exponentiation and five modular. Drone need to compute two exponentiation and seven modular. Therefore, user's computational time = $2t_{exp} + 5t_{mul} = 4.538$ ms, drone's computational time = $2t_{exp} + 7t_{mul} = 4.554$ ms. In the **Zhang et al.'s protocol**, it uses only hash function in the authentication process. Where, 10 hash functions need to be calculated on user side, seven hash functions by drone and server. Therefore, user's execution time = $10t_h = 0.56$ ms, drone side execution time = $7t_h = 0.392$ ms, and server side execution time = $7t_h = 0.049$ ms. In the **proposed protocol**, we have also used only hash function in the authentication process, where, 12 hash functions need to be calculated on user side, seven hash functions by drone and server. Therefore, user's execution time = $12t_h = 0.672$ ms, drone side execution time = $7t_h = 0.392$ ms, and server side execution time = $7t_h = 0.049$ ms. Now, the computational cost our proposed protocol is compared with the other three schemes i.e with Wazid et al.'s scheme, Singh et al.'s scheme, Zhang et al.'s scheme have been shown in Table 3 and Fig. 6, where, $|G|$ is determined by 1024 bits, $|Z_n|$ is the 160 bits, and $|id|$ is the time stamp of length 32 bit and also the identity of the user.

In Wazid et al.'s protocol [19] $(m_1, \ldots, m_7, m_{10}, m_{11}, m_{12}, t_1, t_2, t_3)$ are the transmitted messages, where t_i is the time stamp of length 32 bits and $m_i \in Z_n$. Therefore, it takes $10|G| + 3|D| = 1696$ bits during communication, Singh et al.'s protocol [20] transfers messages $(x_i, y_i, Time_i, id_i)$ on user side and $(x_j, y_j, Time_j, id_j)$ on drone side communication, where, $(Time_i, id_i)$ are the 32 bit time stamp and user identity. Therefore, the total communication cost is $4|G| + 4|id_i| = 4256$ bits. In Zhang et al.'s scheme [14], the user sends $(m_1, \ldots, m_4, st_1)$ to the server and the server sends (m_5, m_6, m_7) to the drone and then drone computes all those and send (m_8, m_{10}) to the user. Here, st_1 is a 32 bit time stamp and all the remaining messages are of 160 bit hash value. Therefore, the total communication cost is $9|Z_n| + |id| = 1472$ bits. In the proposed protocol, the user sends $(m_1, \ldots, m_4, st_1)$ to the server and the server sends (m_5, m_6, m_7) to the drone and then drone computes all those and send (m_8, m_{10}) to the user. Here, st_1 is a 32 bit time stamp and all the remaining messages are of 160 bit hash value. Therefore, the total communication cost is $9|Z_n| + |id| = 1472$ bits. By the analysis and comparison of all the mentioned protocols, we can conclude that our proposed protocol has the least computational cost and time compared to Wazid et al.'s scheme and Singh et al.'s scheme, and it ensures security as compared to Zhang et al.'s scheme. Proposed protocol also tested in Automated Validation of Internet Security Protocols and Applications tool (see Fig. 5). AVISPA contains four backends, namely, "On-the-fly Model-Checker (OFMC)", "Constraint Logic based Attack Searcher (CL-AtSe)", "SAT-based Model-Checker (SATMC)", and "Tree Automata based on Automatic Approximations for the Analysis of Security Protocols (TA4SP)". A security protocol is first implemented using the "High-Level Protocol Specification Language (HLPSL)" of AVISPA, which

Table 3
Comparison between computation and communication cost.

Schemes	User-side computation cost	Drone-side computation cost	Server-side computation cost	Total cost
[*A*] Wazid et al.	$1t_f + 16t_h(14.301)$	$7t_h(.392)$	$8t_h(0.056)$	14.794
[*B*] Singh et al.	$2t_{exp} + 5t_{mul}(4.538)$	$2t_{exp} + 7t_{mul}(4.554)$	–	9.092
[*C*] Zhang et al.	$10t_h(0.56)$	$7t_h(0.392)$	$7t_h(0.049)$	1.001
[*D*] Proposed scheme	$12t_h(0.672)$	$7t_h(0.392)$	$7t_h(0.049)$	1.113

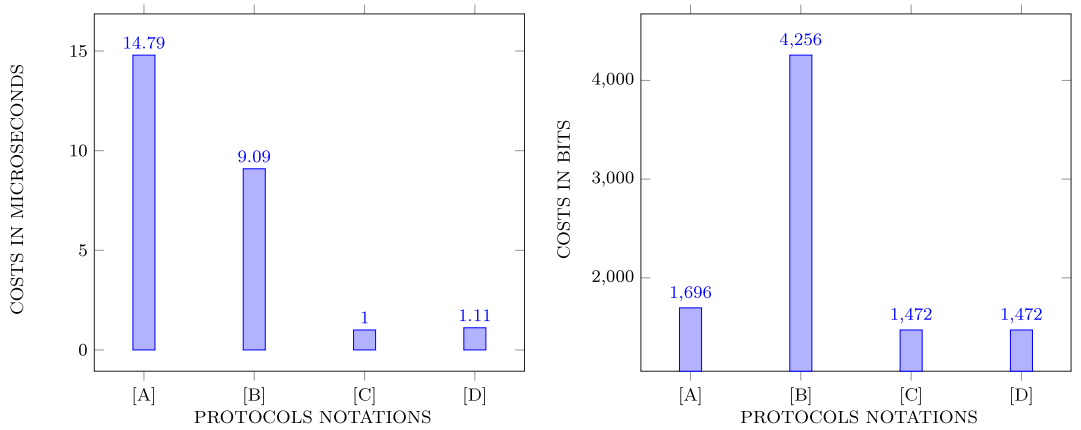


Fig. 6. Illustration of computations and communications costs of protocols.

is converted to "Intermediate Format (IF)" with the help of HLPSL2IF translator. The IF acts an input to one of the four available backends of AVISPA to output the simulation results, which indicate whether the test protocol is "safe", "unsafe" or "conclusive". A comprehensive treatment on AVISPA tool and its HLPSL implementation aspects can be found in [17].

We have utilized the "SPAN (Security Protocol ANimator for AVISPA)" [19] for simulating the proposed protocol. The Fig. 5 shows the output of the OFMC and CL-AtSe implementation results. Under OFMC backend, a total of 122 nodes were visited at a depth of 7 plies which took the time of 274 milliseconds. On the other side, under the CL-AtSe backend 4 states were analyzed where all these states were also reachable, and the required translation and computation time were 0.02 s and 0.01 s, respectively. The simulation results in this figure clearly demonstrate the safe execution of the proposed protocol with no report on "replay attack" and "man-in-the-middle attack".

7. Concluding remarks

There exist a lot of authenticated key agreement protocols for the security of transmitted data through drones. We have analyzed a recent authentication and key agreement protocol by Zhang et al. in year 2020. We have found that this protocol is not secure against stolen smart card attack, and the control server stores extra data. Therefore, We have tried to address security issues of password guessing, anonymity, user/server impersonation, insider attack, and stolen smart card attack, and man in the middle attack. We have proposed a modified lightweight authenticated key agreement protocol for Internet of Drones. The proposed protocol is proved secure in the random oracle model. A performance analysis of proposed protocol with relevant protocols ensures its efficiency, and security as well. This protocol useful for internet of drones, e-healthcare, and internet of unmanned vehicles.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

We are very thankful to the editor and anonymous reviewers for their valuable suggestions. They have played a highly important role to improve the quality of this article. We are thankful to the "Researcher Supporting Project number (RSPD2023R697), King Saud University, Riyadh, Saudi Arabia" for supporting the work.

References

[1] Yu-Jia Chen, Li-Chun Wang, Privacy protection for Internet of drones: A network coding approach, IEEE Internet Things J. 6 (2) (2018) 1719–1730.
[2] Shubhani Aggarwal, Neeraj Kumar, Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges, Comput. Commun. 149 (2020) 270–299.
[3] Rico Valentino, Woo-Sung Jung, Young-Bae Ko, A design and simulation of the opportunistic computation offloading with learning-based prediction for unmanned aerial vehicle (uav) clustering networks, Sensors 18 (11) (2018) 3751.
[4] Milan Erdelj, Borey Uk, David Konam, Enrico Natalizio, From the eye of the storm: An IoT ecosystem made of sensors, smartphones and UAVs, Sensors 18 (11) (2018) 3814.
[5] Bas Vergouw, Huub Nagel, Geert Bondt, Bart Custers, Drone technology: Types, payloads, applications, frequency spectrum issues and future developments, in: The Future of Drone Use, Springer, 2016, pp. 21–45.
[6] Sandeep Saharan, Seema Bawa, Neeraj Kumar, Dynamic pricing techniques for Intelligent Transportation System in smart cities: A systematic review, Comput. Commun. 150 (2020) 603–625.
[7] Rajanpreet Kaur Chahal, Neeraj Kumar, Shalini Batra, Trust management in social Internet of Things: A taxonomy, open issues, and challenges, Comput. Commun. 150 (2020) 13–46.
[8] Leslie Lamport, Password authentication with insecure communication, Commun. ACM 24 (11) (1981) 770–772.
[9] Muhamed Turkanović, Boštjan Brumen, Marko Hölbl, A novel user authentication and key agreement scheme for heterogeneous ad hoc wireless sensor networks, based on the Internet of Things notion, Ad Hoc Netw. 20 (2014) 96–112.
[10] Mohammad Sabzinejad Farash, Muhamed Turkanović, Saru Kumari, Marko Hölbl, An efficient user authentication and key agreement scheme for heterogeneous wireless sensor network tailored for the Internet of Things environment, Ad Hoc Netw. 36 (2016) 152–176.
[11] Ruhul Amin, S.K. Hafizul Islam, G.P. Biswas, Muhammad Khurram Khan, Lu Leng, Neeraj Kumar, Design of an anonymity-preserving three-factor authenticated key exchange protocol for wireless sensor networks, Comput. Netw. 101 (2016) 42–62.
[12] Qi Jiang, Sherali Zeadally, Jianfeng Ma, Debiao He, Lightweight three-factor authentication and key agreement protocol for internet-integrated wireless sensor networks, Ieee Access 5 (2017) 3376–3392.
[13] Sravani Challa, Mohammad Wazid, Ashok Kumar Das, Neeraj Kumar, Alavalapati Goutham Reddy, Eun-Jun Yoon, Kee-Young Yoo, Secure signature-based authenticated key establishment scheme for future IoT applications, Ieee Access 5 (2017) 3028–3043.
[14] Yunru Zhang, Debiao He, Li Li, Biwen Chen, A lightweight authentication and key agreement scheme for Internet of Drones, Comput. Commun. 154 (2020) 455–464.
[15] Muhammad Tanveer, Neeraj Kumar, Mohammad Mehedi Hassan, et al., RAMP-IoD: A robust authenticated key management protocol for the Internet of Drones, IEEE Internet Things J. 9 (2) (2021) 1339–1353.
[16] Sajid Hussain, Khalid Mahmood, Muhammad Khurram Khan, Chien-Ming Chen, Bander A. Alzahrani, Shehzad Ashraf Chaudhry, Designing secure and lightweight user access to drone for smart city surveillance, Comput. Stand. Interfaces 80 (2022) 103566.
[17] Shuangshuang Liu, Chien-Ming Chen, Comments on "A secure and lightweight drones-access protocol for smart city surveillance", IEEE Trans. Intell. Transp. Syst. (2022).
[18] Ding Wang, Ping Wang, Understanding security failures of two-factor authentication schemes for real-time applications in hierarchical wireless sensor networks, Ad Hoc Netw. 20 (2014) 1–15.
[19] Mohammad Wazid, Ashok Kumar Das, Neeraj Kumar, Athanasios V. Vasilakos, Joel J.P.C. Rodrigues, Design and analysis of secure lightweight remote user authentication and key agreement scheme in Internet of drones deployment, IEEE Internet Things J. 6 (2) (2018) 3572–3584.
[20] Jaya Singh, Ashish Gimekar, Subramanian Venkatesan, An efficient lightweight authentication scheme for human-centered industrial Internet of Things, Int. J. Commun. Syst. (2019) e4189.