Scientific Reports

https://doi.org/10.1038/s41598-025-33020-8

Article in Press

Two secure authentication protocols for mitigating vulnerabilities in IoD

Received: 13 July 2025

Accepted: 15 December 2025

Published online: 24 December 2025

Cite this article as: Safkhani M. & Ghorbani Fard M. Two secure authentication protocols for mitigating vulnerabilities in IoD. *Sci Rep* (2025). https://doi.org/10.1038/s41598-025-33020-8

Masoumeh Safkhani & Mahmoud Ghorbani Fard

Two Secure Authentication Protocols for Mitigating Vulnerabilities in IoD

Masoumeh Safkhani[1,2,*] **and Mahmoud Ghorbani Fard**[1]

[1]Department of Computer Engineering, Shahid Rajaee Teacher Training University, Tehran, Iran, Postal code: 16788-15811, Email: Safkhani@sru.ac.ir
[2]School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran.
[*]Corr-author

ABSTRACT

The Internet of Drones (IoD) is a network layer control system that manages the communication of Unmanned Aerial Vehicles (UAVs). Drones have emerged as a novel approach to addressing everyday human challenges and are now used in a variety of domains, such as personal activities (e.g., photography and videography), urban applications (e.g., traffic monitoring and structural inspection), commercial operations (e.g., power line and tower inspection), agriculture, and military operations. Given the rapid growth of UAVs and their expanding applications, interconnecting drones to form an IoD is a desirable trend for enhancing flight safety and quality. However, challenges related to security, privacy, and inter-drone communication remain significant obstacles.

Numerous authentication protocols have been developed to address these concerns. Recently, Zhang et al. proposed a PUF-based authentication scheme that uses unique identifiers and hash functions to secure authentication in the IoD environment. However, in this paper, we demonstrate that Zhang et al.'s scheme is vulnerable to several attacks, including secret value disclosure, integrity violation, key extraction, traceability, and anonymity violation. The presented attacks are shown to have a success probability of one.

We also introduce two enhanced protocols that, through both informal and formal security proofs using the Scyther tool, demonstrate that they do not suffer from the vulnerabilities found in the earlier protocol. The communication costs of the proposed protocols (a) and (b) have increased by 29% and 59%, respectively, compared to the previous protocol. The computational costs for the proposed protocols (a) and (b) have also increased by 26% and 56%, respectively, while the storage costs in both proposed protocols remain unchanged compared to the previous protocol. It is true that the costs in the proposed protocols have risen; however, the previous design was vulnerable to various attacks, whereas the proposed protocols have demonstrated better security and have successfully achieved all their security objectives.

1 Introduction

The rapid advancement of technology has accelerated the pace of work and problem-solving. Among these technologies, the integration of the Internet of Things (IoT) into human society is clearly evident, promising a more convenient and better life for humanity[1]. IoT has permeated various sectors of industry and daily human needs.

The significance of security within the Internet of Things (IoT) is immense, as these devices are becoming increasingly embedded in vital infrastructure and daily use cases, such as smart homes, healthcare applications, and industrial settings [2]. As the number of connected devices grows, the complexity and scope of security threats also expand. Smart homes, in particular, face notable security challenges due to their reliance on interconnected devices that facilitate automation and convenience[3]. Many of these devices have limited computational capabilities, leaving them susceptible to various attacks, including unauthorized access, secret data disclosure, and denial-of-service (DoS) attacks[4]. Additionally, the vast array of devices and their varying communication protocols broaden the potential points of vulnerability. Thus, ensuring the security of IoT systems is crucial for protecting personal data, preserving privacy, and ensuring the stability of critical infrastructures. With the rising frequency of cyberattacks targeting IoT devices, it is vital to continually evaluate security weaknesses and deploy effective countermeasures to reduce risks[5].

A particular focus of this paper is the use of data fusion and exchange in remotely piloted aircraft, commonly known as drones, via communication networks. The Internet of Drones (IoD), also referred to as the Internet of Unmanned Aerial Vehicles (UAVs)[6], is itself part of the IoT ecosystem and is an architecture designed to provide coordinated access to controlled airspace for UAVs. With the continuous miniaturization of sensors and processors, alongside ubiquitous wireless connectivity, the IoD has enabled new applications and public services for diverse human needs, ranging from on-demand package delivery, traffic and wildlife monitoring, and infrastructure inspection, to search and rescue, agriculture, cinematography, and specific military operations. All of these applications share a common requirement for navigation, airspace management, and secure information

sharing.

Data exchange in IoD environments occurs via insecure communication channels due to the nature of wireless connections and the unique characteristics of their operational environments. Consequently, the IoD is susceptible to all threats associated with wireless communications. Furthermore, the risks inherent to the IoT are present in the IoD as well. Thus, while IoD technology offers significant advantages, it also introduces numerous security threats and privacy concerns. All IoD components, from hardware (e.g., drone sensors) to data-related elements (e.g., drone-collected information), must be systematically examined and protected from a security perspective[7,8].

The practicality of integrating tamper-resistant hardware in low-cost IoT devices is often limited due to budget constraints, size limitations, and power consumption requirements. Instead, manufacturers can adopt software-based security measures like secure boot and code obfuscation, implement lightweight cryptographic algorithms, and establish robust device authentication methods. Additionally, network-level security practices such as segmentation and the use of (Virtual Private Network) VPNs can protect data in transit. Fallback mechanisms, such as graceful degradation of functionality and user notifications about security states, can further enhance resilience. By combining these strategies, manufacturers can improve the security of commodity hardware without the need for specialized tamper-resistant components.

Maintaining data security and ensuring the authenticity of communicating parties are among the most significant challenges in implementing the Internet of Drones. These challenges are further complicated by the limited computational resources and restricted power supplies of drones. As a result, authentication schemes designed for the IoD environment must be both lightweight and reliable. In addition to securing communication, these schemes must minimize resource consumption and perform authentication computations as efficiently as possible, given the operational environment and critical needs of drones. Implementing stronger security features in communication systems, especially in constrained environments like drones or IoT devices, involves navigating several trade-offs that can affect bandwidth, latency, and overall system performance. Key challenges include increased data overhead from encryption and authentication processes, frequent key exchanges that escalate communication costs, and added latency due to encryption/decryption times. Additionally, these security measures often demand more computational resources, which can be problematic in resource-limited settings.

To tackle communication cost trade-offs challenges, while ensuring strong security, several optimization strategies can be utilized. The use of lightweight security protocols, data aggregation at the source, and effective key management can lead to a significant reduction in communication costs. Furthermore, employing techniques such as data compression, selective encryption of sensitive data, and batch processing can help decrease overhead. Additionally, implementing adaptive security levels based on the sensitivity of the data and utilizing Trusted Execution Environments (TEEs) for secure computations can improve security without sacrificing communication efficiency. It is essential to find a balance among these elements for successful operation in constrained environments. Therefore, in this paper, two versions of the protocol, (a) and (b), are proposed, each with different communication and computational costs. For applications where user untraceability is not critical, protocol (a) can be used, as it has lower computational and communication costs compared to protocol (b). On the other hand, for applications that require a higher level of security, the proposed protocol (b) should be utilized.

1.1 Main Contributions

The contributions of this paper can be summarized as follows:

1. In this research, we conducted a comprehensive analysis of the protocol proposed by Zhang et al.[9] with the aim of identifying and highlighting its security vulnerabilities. Through a rigorous evaluation of the protocol, we have identified several significant security vulnerabilities. These include a violation of message integrity, indicating that the protocol lacks a suitable mechanism for verifying the integrity of certain messages. Additionally, the protocol fails to guarantee the secrecy of the session key. Furthermore, we found that the protocol is susceptible to traceability and anonymity violation attacks and does not adequately protect the confidentiality of user identities and passwords.

2. Building upon our findings, we have developed two novel PUF-based protocols (a) and (b) to mitigate the security weaknesses exposed in Zhang et al.'s[9] protocol. These protocols are specifically designed to meet the requirements of various deployment scenarios.

3. Furthermore, we have employed both formal and informal security analysis to demonstrate the security efficacy of our proposed protocols. The results of our analysis, when compared to recent protocols, are presented. Notably, the Scyther security protocol analyzer was utilized for formal verification.

4. In this study, we conducted a comparative analysis of the newly proposed protocols (a) and (b) against similar recent protocols. The results of our calculations indicated that the communication costs for the proposed protocols (a) and (b) increased by 0.29% and 59%, respectively, when compared to Zhang et al.'s protocol[9]. Additionally, the computational

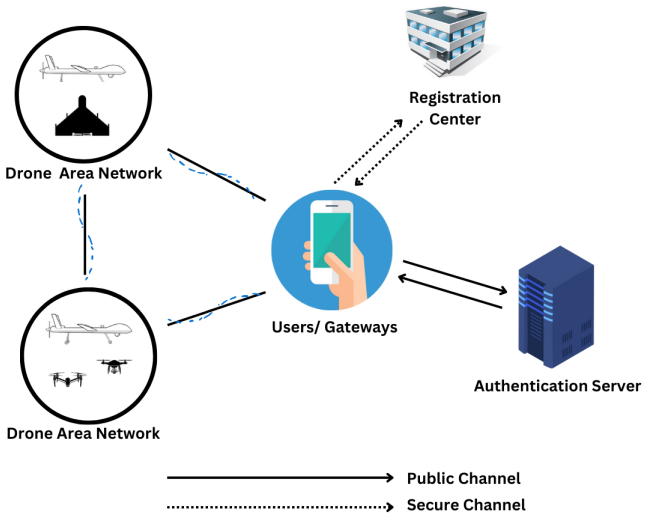

Figure 1. The IoD network model

costs for proposed protocols (a) and (b) were found to be 26 % and 56 %, respectively, compared to its predecessor while the storage costs remained unchanged across both new designs compared to their predecessor. Although it is acknowledged that the costs associated with the proposed protocols have risen, it is important to note that the previous design exhibited vulnerabilities to various attacks. In contrast, the proposed protocols have demonstrated enhanced security features and successfully met all specified security objectives, thereby providing a more robust solution despite the increase in costs. It is important to highlight that the proposed schemes, in comparison to other recent similar designs, offer a significant advantage in terms of computational, communication, and storage costs. Some existing schemes have higher costs than the ones presented in this paper.

1.2 Paper Organization

The rest of the paper is organized as follows: In Section 2, we examine the existing literature in this area. Section 3 provides a concise overview of Zhang et al.'s proposed scheme called PRLAP-IoD[9]. Section 4 delves into the security vulnerabilities of Zhang et al.'s scheme[9] , analyzing potential attacks. Section 5 introduces enhancements to the original protocol. Informal and formal security analysis of proposed schemes are presented in Section 6. The performance evaluations of proposed schemes are assessed against their counterparts in Section 7. This paper ends in Section 8 with concluding remarks.

2 Related Work

In recent years, many researchers have entered the field of IoD with multiple authentication schemes for the Internet of Drones mechanism, which aimed to provide security against various known security attacks. In[10], Tanveer et al. proposed a scheme in 2024 that combined hash functions and symmetric encryption. However, their scheme was vulnerable to advanced attacks that compromised data security, such as forward secrecy. In[11], Irshad et al. proposed a scheme allowing registered users to access real-time data from the environment directly after gateway authentication. However, their scheme was vulnerable to the forward secrecy breach attack. The authors of[12] examined[10,11] schemes and addressed their security vulnerabilities and demonstrated that their proposed protocol achieved a 67.86% and 17.80% reduction in computational and communication costs, respectively, compared to related protocols. The authors in[12] performed a thorough analysis of the schemes[10] and[11] and identified and fixed their security vulnerabilities. Their proposed protocol showed significant improvements in computational and communication efficiency and achieved a reduction of 67.86% and 17.80%, respectively, compared to existing protocols.
In[13], Srinivas et al. proposed an authentication scheme in 2020 to enable secure real-time data access from Internet of Things (IoT) devices (e.g., vehicles, which can also be implemented in IoD environments) by a legitimate external party (user). Their proposed scheme used a three-factor authentication mechanism based on Elliptic Curve Cryptography (ECC). The authors claimed that their scheme offers improved security features while its computational cost equals other schemes. To substantiate

their claims, they comprehensively analyzed and compared various schemes, including those proposed by Turkanovic et al.[14] and Pourambij et al.[15]. Srinivas et al.[13] identified security vulnerabilities in[14,15], including weaknesses against insider attacks, user impersonation, and stolen smart card attacks.
Alzahrani et al. in[16] analyzed various schemes, including[17] by Srinivas et al.,[18] by Zhou et al., and[19] by Wazid et al. and discovered that each scheme suffered from security vulnerabilities, such as the inability of[17] to resist impersonation attacks,[18]'s vulnerability to IoT device spoofing attacks, and[19]'s susceptibility to drone/user/control center impersonation attacks. In[16], also a scheme based on symmetric key cryptography and elliptic curve cryptography is proposed. By discussing security features, the authors demonstrated that the proposed scheme in[16] offers higher reliability and resistance to attacks mentioned in[17–19].
In 2024, Tanveer et al. proposed a scheme[20] that uses Physical Unclonable Function (PUF) and the AEGIS authentication encryption scheme[21] to ensure secure and reliable communication between users and unmanned aerial vehicles (UAVs) in smart cities. The authors of[20] analyzed various schemes, including[13,16], identifying vulnerabilities such as impersonation attacks, Man-In-The-Middle (MITM) attacks, lack of user anonymity, and mutual authentication failures in[13], and identity disclosure and user ID guessing attacks in[16]. They claimed to have addressed these shortcomings through a rigorous informal analysis.
In[22], Zhang et al. proposed a scheme to reduce communication costs by using XOR operations in conjunction with a one-way hash function for authentication. They claimed to have achieved perfect security and resistance to known attacks. Gupta et al. also proposed a scheme[23] to reduce hardware resource consumption and enhance authentication efficiency by employing simple XOR operations and a secure one-way hash function. Security analysis tools demonstrated that the Gupta et al.'s proposed protocol provides security and maintains a balance between performance and communication overhead.
In [24], Choi et al. examiined Sharma et al.'s Authentication and Key Agreement (AKA) scheme [25] and showed that their scheme is susceptible against user impersonation, stolen verifier, and Ephemeral Secret Leakage (ESL) attacks. Then, Choi et al proposed a lightweight and secure AKA scheme for the IoD to rectify the vulnerabilities of Sharma et al.'s scheme. In[26], Modarres and Sarbishaei introduced an authentication approach for IoD that leverages PUFs and multiple verification factors. In their proposed scheme, a ground station coordinates the identity checks between the user and the drone. Their analysis—both formal and informal—indicates that their proposed authentication scheme resists numerous attack scenarios. In[27], Nyangaresi et al. introduced a novel authentication protocol combining PUF, biometrics, and Elliptic Curve Cryptography (ECC) for the Internet of Drones. They demonstrated that their proposed scheme provides strong security protections against various types of attacks.
In[9], Zhang et al. highlighted UAVs' limited computational resources and memory constraints in 2024. They argued that ensuring strong security while minimizing computational and communication overhead is a significant challenge. To address this, they proposed a lightweight authentication scheme that uses XOR operations, a secure one-way hash function, and PUF. The authors also showed in[9] that[22,23]'s schemes are vulnerable to physical attacks, allowing adversaries to extract information using side-channel attacks after physically capturing the UAV.
Table 1 presents a comprehensive overview of various IoD authentication schemes. It summarizes the key contributions of each work, the methodologies employed, and the vulnerabilities identified in their approaches.

As shown in the Table 1, the protocol proposed by Zhang et al. (2024)[9] is still vulnerable to secret values disclosure, session key disclosure, traceability and integrity contradiction attacks, as will be demonstrated in this paper. We have modified the protocol to make it resilient against these attacks as well. Analyzing the presented protocols contributes significantly to ensuring their security and advances the science of protocol design and analysis.

3 PRLAP-IoD authentication scheme

PRLAP-IoD , a scheme presented by Zhang et al. in[9], is designed to verify identity and increase security in data exchanges in the IoD environment. Figure 1 illustrates the network model and IoD environment as proposed by Zhang et al.[9].

3.1 PRLAP-IoD assumptions

PRLAP-IoD scheme uses a noise-resistant and reliable PUF function capable of achieving 100 percent accuracy in complex environments (voltage fluctuations and temperature variations), as reported in[28]. In PRLAP-IoD, secure one-way hash functions and XOR operations are also utilized to enhance security and resistance against high-level attacks. The PRLAP-IoD authentication scheme includes five phases: pre-deployment, registration, login, authentication, and password update, which will be discussed further.
The notations used in this paper are shown in Table 2.

3.2 Pre-deployment phase

Server Ser is entrusted with the responsibility of registering each user and drone at this point. Ser also produces the master secret key, symbolized by X_{Ser}, and sets up GID_i (the gateway node's unique identity) and $h(.)$ (the one-way cryptographic hash function). The details are as follows:

Table 1. Summary of authentication schemes, vulnerabilities and their improved scheme in IoD. In this table, the (–) symbol means no information reported in this regard.

Article	Method Used	Vulnerabilities	Improved Scheme
Turkanovic et al. (2014)[14]/ Porambage et al.(2014)[15] ,	Hash function, XOR operation/hash function, ECC	Insider attacks, user impersonation, stolen smart card attack	Srinivas et al. (2020)[13]
Sinivas et al. (2019)[17]/ Zhou et al.(2019)[18]/ ,Wazid et al. (2018)[19]	Hash function, Biometric fuzzy extractor/hash function, bilinear pairing, ECC/hash function, Fuzzy extractor	Impersonation attacks/spoofing attacks/impersonation attacks	Alzahrani et al. (2021)[16]
Gupta et al. (2019)[23]	XOR operation, Hash function	Physical attacks	Zhang et al. (2024)[9]
Srinivas et al. (2020)[13]	Three-factor authentication based on ECC	Insider attacks, user impersonation, stolen smart card attacks	Tanveer et al. (2024)[10]
Zhang et al. (2020)[22]	XOR operations, Hash function	Physical attacks; side-channel attacks	Zhang et al. (2024)[9]
Tanveer et al. (2023)[10]	Hash function, symmetric encryption	Forward secrecy contradiction attacks	Mahmood et al. (2024)[12]
Irshad et al. (2024)[11]	Gateway authentication for real-time data access	Forward secrecy contradiction attacks	Mahmood et al. (2024)[12]
Tanveer et al. (2024)[20]	PUF and AEGIS authentication encryption[21]	Impersonation attacks, MITM attacks, lack of user anonymity, mutual authentication failures	-
Modarres and Sarbishaei (2025)[26]	PUF, Hash Function	-	-
Nyangaresi et al.(2025)[27]	Hash Function, PUF, Biometrics, ECC	-	-
Sharma et al. (2025)[25]	PUF, Hash function	User impersonation, stolen verifier, and ESL attacks	Choei et al. (2025)[24]
Zhang et al. (2024)[9]	Hash function, PUF, XOR operation	Secret values disclosure, integrity violation, key extraction, traceability, anonymity violation(in this paper)	Our proposed protocols(a), (b)

1. Initially, *Ser* selects a 128-bit random number (X_{Ser}) as the master private key and subsequently stores X_{Ser} securely.
2. For each gateway, *Ser* generates a 128-bit number (GID_i) as the identity of the i^{th} gateway node (GWN_i) and stores it in the i^{th} gateway node's database.
3. Finally, *Ser* selects a secure one-way hash function ($h(.)$) and publishes the values ($h(.), GID_i$).

3.3 Registration phase

At this phase, the drone and user are registered. First, we describe the drone registration step. For the drone registration phase, the drone receives challenge-response pairs from *Ser* to serve as its long-term secret key. Figure 2 illustrates the PRLAP-IoD drone registration process in detail.

Drone Registration:

1. The drone randomly selects a number r_b and calculates a pseudonym $HDID_j$ as $HDID_j = h(HDID_j \| r_b)$. Subsequently, the drone sends its pseudonym to *Ser* with a registration request.
2. Upon receiving a message from the drone D_j, *Ser* computes a challenge as $C_j = h(HID_j \| X_{Ser})$ and transmits C_j, GID_i to D_j over a secure and reliable channel.
3. Upon receiving the values C_j, GID_i from *Ser*, the drone D_j computes the response using its PUF function as $R_j = PUF(C_j)$ and also calculates $B_1 = h(C_j \| R_j \| GID_i)$. Subsequently, the drone stores the values $\langle B_1, HDID_j, GID_i \rangle$ in its memory and transmits the response R_j to *Ser* over a secure channel.

Table 2. Notations

Symbol	Description
U_i	The i^{th} User
GWN	Gateway node/Smart device
Ser	Trusted Authority Server
ID_i	The i^{th} User's Identity
PW_i	The i^{th} User's Password
GID_i	Unique Identity of Gateway Node
DID_i	Unique Identity of Drone
HID_i	Pseudonym Identity of i^{th} User
$NHID_i$	New Pseudonym Identity of i^{th} User
HPW_i	Pseudonym Password of i^{th} User
$HDID_j$	Pseudonym Identity of j^{th} Drone
$NHDID_j$	New Pseudonym Identity of j^{th} Drone
X_{Ser}	Master Secret key of Ser
K_{GS}^i	The Long-term Shared Secret Key of U_i and Ser
(C_j, R_j)	Challenge and Response Pair
SK	Established Session Key
Δ	Maximum Transmission Delay Time
$\oplus$	Bit-wise OR operation
$h(.)$	One-way Cryptographic Hash Function
$\|\|$	Concatenation Operation
$PUF(.)$	Physical unclonable function
$\|ID\|$	The bit length of ID
$\|PW\|$	The bit length of PW
T_h	Hash Function Execution time
T_{mp}	ECC Multiplication Execution Time
T_{PUF}	PUF Execution Time
T_{fe}	Fuzzy Extractor Execution Time
T_{pa}	ECC Add Execution Time
T_s	Symmetric Encryption/Decryption Execution Time
T_{cmap}	Chaotic Map Execution Time

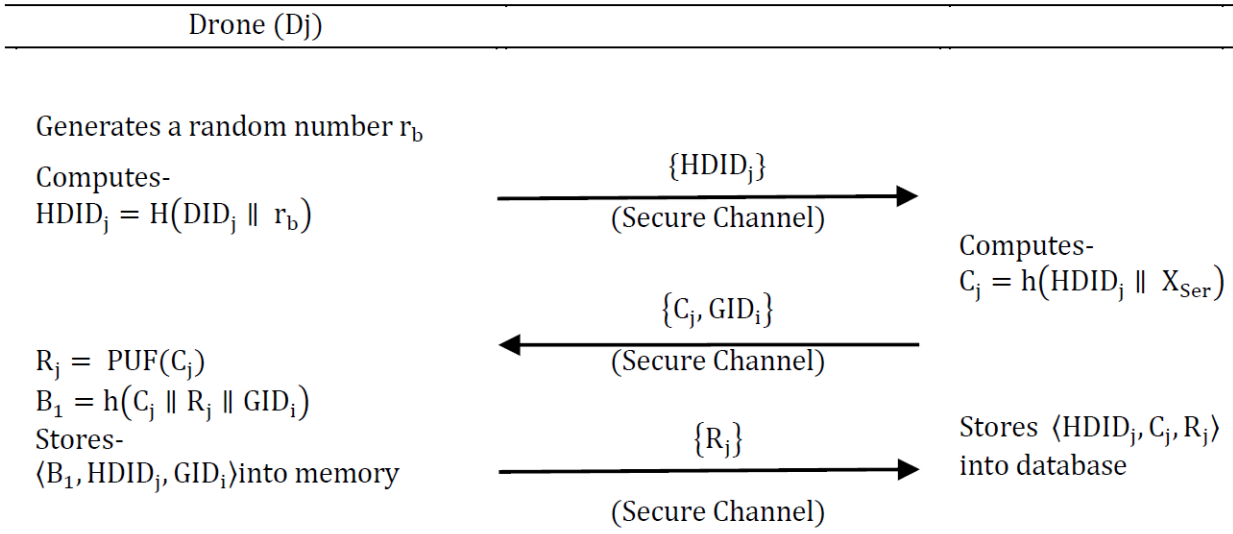

Figure 2. Drone registration phase of PRLAP-IoD[9]

4. Finally, *Ser* receives the value R_j and stores the values $\langle HDID_j, C_j, R_j \rangle$ in its database.

Initial registration in the system is mandatory to grant users access and control over their drones throughout missions. This process involves submitting a registration request to the server. Figure 3 provides a detailed overview of User registration phase.

User Registration:

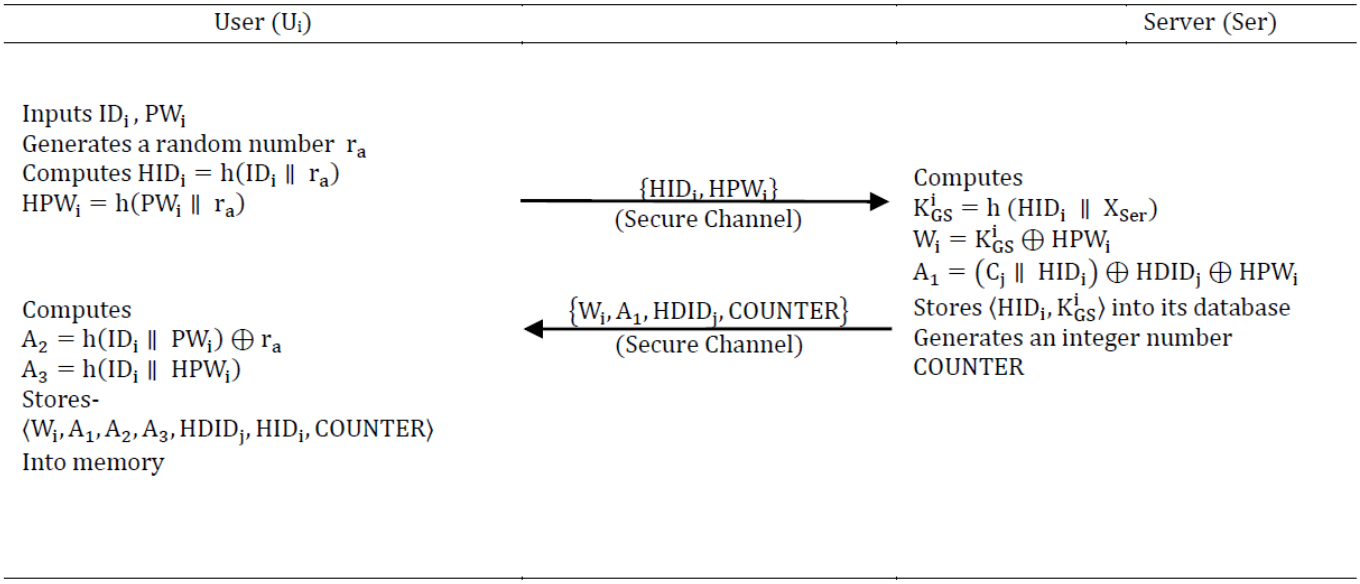

Figure 3. User registration phase of PRLAP-IoD[9]

1. A new user U_i must enter a unique identifier ID_i and password PW_i at the gateway node GWN_i. Subsequently, a random number r_a is generated and used to calculate a pseudonymous identity as $HID_i = h(ID_i \| r_a)$. A pseudo-password is also calculated as $HPW_i = h(PW_i \| r_a)$. These values $\{HID_i, HPW_i\}$ are then securely transmitted to the *Ser* via a secure channel.

2. As soon as the message $\{HID_i, HPW_i\}$ is received, *Ser* performs a set of processes for the user. *Ser* selects the $HDID_j$ that belongs to the registered UAV and calculates the user's long-term secret key denoted by K_{GS}^i as $K_{GS}^i = h(HID_i \| X_{Ser})$.

Then it obtains the $W_i = K_{GS}^i \oplus HPW_i$ and then calculates $A_1 = (C_j \| HID_i) \oplus HDID_j \oplus HPW_i$. After that, *Ser* generates the $COUNTER$ variable initialized to zero and uses it to record the number of failed user logins. Then, it sends the values $\{W_i, HDID_j, A_1, COUNTER\}$ to the user U_i through a secure channel and also stores the $\langle HID_i, K_{GS}^i \rangle$ in its database.

3. After collecting credits $\{W_i, HDID_j, A_1, COUNTER\}$ from *Ser*, the user calculates A_2 and A_3 as $A_2 = h(ID_i \| PW_i) \oplus r_a$ and $A_3 = h(ID_i \| HPW_i)$ respectively. Next, the user stores the $< W_i, A_1, A_2, A_3, HDID_j, HID_i, COUNTER >$ in his/her database safely.

3.4 Login and authentication phase

In this section, we will explain more about the login and authentication process of PRLAP-IoD. In the PRLAP-IoD protocol, both the user and the drone can generate a session key (SK) consisting of a unique response R_j and a random number K_1. The steps involved in this process will be explained below.

1. Initially, the GWN_i prompts the user U_i to input his unique user identifier and corresponding password. Upon the user's input of these confidential values, the GWN_i proceeds to compute $r_a^* = A_2 \oplus h(ID_i \| PW_i)$, $HPW_i^* = h(PW_i \| r_a^*)$, and $A_3^* = h(ID_i \| HPW_i^*)$. If $A_3^* = A_3$ is not established, GWN_i rejects the login request and the failed login attempt counter, denoted by $COUNTER$, is incremented. Otherwise, the user U_i calculates the current timestamp T_1 as well as the new challenge C_J^{new} for the drone D_j and calculates the message $A_1^* = A_1 \oplus HPW_i \oplus T_1$, then, the user sends $\langle A_1^*, C_j^{new}, T_1 \rangle$ for the drone D_j through the public channel.

2. Drone D_j, upon receiving the connection request and message $\langle A_1^*, C_j^{new}, T_1 \rangle$ from the user U_i, initially calculates its current timestamp and assigns it the label T_2. Subsequently, it verifies whether the inequality $|T_2 - T_1| < \Delta T$ holds, where ΔT represents the maximum allowable time difference, if it is not, the drone will reject and ignore the U_i request until it sends a new request. Otherwise, the drone retrieves $(C_j^* \| HID_i^*)$ as $A_1^* \oplus HDID_j \oplus T_1$, then computes $R_j^* = PUF(C_j^*)$, and $B_1^* = h(C_j^* \| R_j^* \| GID_i)$ and checks if $B_1^* = B_1$, then uses a Physically Unclonable Function PUF to generate a novel response as $R_j^{new} = PUF(C_J^{new})$. Subsequently, the drone computes new values $M_1 = R_j^{new} \oplus h(R_j^* \| T_2)$ and $M_2 = h(HID_i^* \| R_j^* \| T_2)$, then transmits $\langle M_1, M_2, T_2 \rangle$ to the gateway node via a public channel.

3. As soon as messages $\langle M_1, M_2, T_2 \rangle$ are received at the time T_3, the user U_i calculates the time validity, upon successful timestamp verification, the gateway node generates a random number R_1 and calculates the values $K_{GS}^{i^*} = W_i \oplus HPW_i^*$, $M_3 = R_1 \oplus K_{GS}^{i^*} \oplus M_2$ and $M_4 = h(HDID_j \| GID_i \| R_1 \| T_3)$ and sends it as $\langle M_3, M_4, HDID_j, T_2, T_3 \rangle$ to *Ser*; otherwise, the user rejects the connection.

4. After receiving the $\langle M_3, M_4, HDID_j, T_2, T_3 \rangle$ from the gateway at the time T_4, *Ser* checks the time stamp validity. During the D_j and U_i registration phase, *Ser* had stored $(HDID_j, C_j, R_j)$ and (HID_i, K_{GS}^i) respectively. Therefore, *Ser* can retrieve the challenge-response pair (R_j, C_j) from its database using the drone's pseudonym $(HDID_j)$. However, it can not retrieve (HID_i, K_{GS}^i), because in the messages sent from the gateway to the server there is no parameter that indicates which use/gateway it is connected to. We have fixed this issue in our proposed protocols. The rest of the protocol is explained assuming this problem does not exist. If $|T_4 - T_3| < \Delta T$ holds, *Ser* calculates the values $R_1^* = M_3 \oplus K_{GS}^i \oplus h(HID_i \| R_j \| T_2)$ and $M_4^* = h(HDID_j \| GID_i \| R_1^* \| T_3)$, after that *Ser* compares M_4 and M_4^* to confirm D_j and U_i. If M_4^* is not equal to M_4, the session will conclude. After successful authentication of D_j and U_i, *Ser* calculates $M_5 = h(R_j \| T_2) \oplus h(R_1^* \| T_4)$, $M_6 = h(K_{GS}^i \| R_1^* \| T_4)$ and $M_7 = h(R_j \| T_2 \| T_4)$, then sends $\langle M_5, M_6, M_7, T_4 \rangle$ to the gateway node.

5. When getting the messages $\langle M_5, M_6, M_7, T_4 \rangle$ from *Ser* at T_5, GWN_i first checks the validation of time by $|T_5 - T_4| < \Delta T$, if it holds, GWN_i calculates $M_6^* = h(K_{GS}^{i^*} \| R_1 \| T_4)$ and checks whether $M_6^* \stackrel{?}{=} M_6$ is or not. The session is terminated if it does not hold. Otherwise, GWN_i generates a new random number K_1 and calculates a new response $R_j^{new} = M_1 \oplus M_5 \oplus h(R_1 \| T_4)$, $M_8 = K_1 \oplus h(R_j^{new} \| T_5)$ and $SK = h(K_1 \| R_j^{new})$ and then sends $\langle M_7, M_8, T_4, T_5 \rangle$ to the drone D_j.

6. Upon receiving the messages $\langle M_7, M_8, T_4, T_5 \rangle$ from GWN_i, D_j checks the validation of time by $|T_6 - T_5| < \Delta T$, if it is not, the drone will cancel its further operations. Otherwise it computes $M_7^* = h(R_j \| T_2 \| T_4)$ and checks the equality of M_7^* and M_7, after successfully authentication of GWN_i and *Ser*, the drone D_j calculates $K_1 = M_8 \oplus h(R_j^{new} \| T_5)$ and $SK = h(K_1 \| R_j^{new})$.

The PRLAP-IoD key agreement and authentication procedure is shown in Figure 4.

3.5 Drone-Drone authentication phase

This section briefly explains how a drone establishes a secure communication link with another drone in the PRLAP-IoD. This communication, which aims to authenticate and create a secure communication channel, is established by exchanging a session key. Figure 5 illustrates the steps of this process.

1. The drone D_1 sends a connection request encrypted by SK_1 as $\{Req, DID_2\}_{SK1}$ to gateway GWN_i.

2. Upon receiving $\{Req, DID_2\}_{SK1}$, GWN_i checks the identity of D_2. Then, it sends the request to D_2 which is encrypted by the SK_2 established in drone D_2 login and authentication phase.

3. When the drone D_2 confirms the connection request, it returns permission to the GWN_i. After the GWN_i receives the permission, it will generate a SK_3 for the next communication between drones D_1 and D_2 and sends $\{SK_3\}_{SK_1}$ and $\{SK_3\}_{SK_2}$ to D_1 and D_2 respectively.

4. Finally, the new communication between D_1 and D_2 is established by a session key SK_3.

3.6 Updating password phase

In this phase, the PRLAP-IoD provides an optional function to update the old password. Only GWN_i can do this, not the Ser. The specific operation as follows:

1. The GWN_i asks the U_i to input the outdated login credentials (ID_i, PW_i^{old}).

2. Upon receiving (ID_i, PW_i^{old}) from U_i, GWN_i checks whether $A_3 = A_3^{old}$ is or not. For this purpose, it computes the random number $r_a^* = A_2 \oplus h(ID_i \| PW_i^{old})$, $HPW_i^{old} = h(PW_i^{old} \| r_a^*)$ and $A_3^{old} = h(ID_i \| HPW_i^{old})$. If $A_3 = A_3^{old}$ is not valid, the gateway will reject the update request. Otherwise, GWN_i believes the legitimacy of the user and requests a new password PW_i^{new} from U_i.

3. After receiving PW_i^{new} from user, GWN_i computes $HPW_i^{new} = h(PW_i^{new} \| r_a^*)$, $K_{GS}^i = W_i \oplus HPW_i^{old}$, $W_i^{new} = K_{GS}^i \oplus HPW_i^{new}$, $A_2^{new} = r_a^* \oplus h(ID_i \| PW_i^{new})$ and $A_3^{new} = h(ID_i \| HPW_i^{new})$. Finally, W_i^{new}, A_2^{new} and A_3^{new} are stored to GWN_i to replace the W_i, A_2 and A_3 respectively.

4 Security analysis of the PRLAP-IoD

In this section, we will perform a thorough informal security analysis of the PRLAP-IoD scheme and outline various attacks including secret value disclosure attack, integrity contradiction attack, session key disclosure attack, traceability attack and anonymity contradiction attack against it within the Dolev-Yao adversary model, as described below:

Dolev-Yao Adversary Model:

The Dolev-Yao adversary model [29] serves as a theoretical framework in cryptography that defines the capabilities of an adversary aiming to breach the security of cryptographic protocols. Within this model, the adversary possesses several critical abilities: s/he can control communication channels, enabling his/her to intercept, alter, or inject messages between legitimate parties; s/he can impersonate legitimate users, allowing his/her for identity spoofing; s/he can execute replay attacks by capturing and reusing valid messages; s/he may collaborate with other malicious actors to strengthen attack strategies; s/he can access the internal state of protocols or participating entities to collect sensitive information; and finally, s/he can conduct adaptive attacks, modifying their tactics based on the reactions of legitimate parties. These capabilities highlight the importance of the Dolev-Yao model in the design and evaluation of secure cryptographic systems. In this paper, we apply this adversary model in our analysis and validation of the security of the examined protocols.

4.1 Integrity contradiction attack

Integrity is a property that ensures that if a message is altered during transmission, the recipient can detect the modification and determine that the message is no longer trustworthy. In this section, we will demonstrate that the protocol in question does not guarantee integrity for most messages. To execute this attack, an adversary can follow these steps:

- If the adversary changes the message M_1 sent by the drone D_j to the user/gateway node to the desired value $M_1' = M_1 \oplus \Delta$, since M_1 is not checked by the user/gateway node, GWN_i computes $R_j^{new'}$ as $R_j^{new'} = M_1' \oplus M_5 \oplus h(R_1 \| T_4)$ that $M_1' \neq M_1$ as a result, $R_j^{new'}$ is not equal to R_j^{new}.

- The same issue applies to the message M_8. The drone does not verify the integrity of M_8, allowing an adversary to arbitrarily modify its value, such as changing it to $M_8' = M_8 \oplus \Delta'$. Since the integrity is not verified on the drone side, consequently, the drone calculates $K_1' = M_8' \oplus h(R_j^{new} \| T_5)$ which is different from the calculated value K_1 on the user side and as a result SK will be computed as $SK = h(K_1' \| R_j^{new})$ on the drone side which is different from the value of $SK = h(K_1 \| R_j^{new})$ on the user side. This can also eventually lead to desynchronization attack.

- During the Drone-Drone authentication phase, there is a potential integrity contradiction attack. As illustrated in Figure 5, the secret key SK_3 is created for communication between two UAVs and is encrypted with each UAV's secret key before being sent to them. If an attacker modifies $\{SK_3\}_{SK_1}$, the recipient will receive a tampered message. Upon decrypting it, the recipient will obtain a different key value, causing any messages encrypted with this incorrect key to remain undecryptable by the drones.

This attack is guaranteed to succeed with a single execution of the protocol, achieving a 100% success rate.

4.2 Session key disclosure attack

If the authorized user (who can be an insider attacker) wants to connect to the D_j drone, as shown in Figure 4, and sends his previous C_j instead of C_j^{new} (notice that the adversary has previous messages with C_j), in that case, $R_j^{new} = R_j$, that means, this user gets R_j^{new}.
Consider a scenario where the user U_i (the insider attacker) aims to establish a connection with the drone by transmitting C_j^{new}, A_1^*, T_1. Given that the insider attacker has acquired R_j, s/he can compute the value of R_j^{new} based on the drone's response, which consists of (M_1, M_2, T_2). This is calculated as $R_j^{new} = M_1 \oplus h(R_j^* \| T_2)$. Subsequently, the attacker can derive the value of K_1 from M_8 using the formula $K_1 = M_8 \oplus h(R_j^{new} \| T_5)$. With both R_j^{new} and K_1 in their possession, the adversary can compute the session key for another user with the drone using $SK = h(K_1 \| R_j^{new})$. The success probability of this attack is 100%, and the protocol's complexity involves executing the protocol twice.

4.3 Traceability attack and anonymity contradiction

In the Phan Traceability model[30], when analyzing the relayed messages associated with the drone D_0 , we can determine whether arbitrary relayed messages pertain to the same drone. This indicates that we have successfully traced D_0.
This section illustrates that the PRLAP-IoD is susceptible to traceability attacks. The primary vulnerability lies in the $HDID_j$, which is permanently assigned to the drone and cannot be changed. As a result, by analyzing the $HDID_j$ message transmitted over the insecure channel between the user and the server, it becomes possible to identify the specific drone involved.
Conversely, if the attacker captures and retains all the messages exchanged in Figure 4, s/he can XOR the values of A_1^*, $HDID_j$, and T_1 transmitted through the insecure channel. Using the equation $(C_j^* \| HID_i^*) = A_1^* \oplus HDID_j \oplus T_1$, the attacker can derive $C_j^* \| HID_i^*$, which is a fixed value that can be utilized for traceability purposes. Furthermore, the HID_i can be associated with the drone. As a result, in PRLAP-IoD, both the user/gateway node and the drone, as well as their combination, can be traced. By applying an XOR operation to certain exchanged messages, we can obtain a constant value linked to the identities of the drone and the user/gateway node. This attack only necessitates a single execution of the protocol and has a success rate of 100%.

4.4 Secret values disclosure attack

This section describes the secret values disclosure attack by an insider attacker. The insider attacker is an authorized user registered in the system with access to the secure channel and smart cards. The insider attacker observes the registration phase between the user and the server, obtains and stores the exchanged values, and has access to the values stored in the smart card. Due to the fact that ID and PW always have a short length and the amount of guesses for them are respectively $2^{|ID|}$ and $2^{|PW|}$ where $|ID|$ and $|PW|$ show the bit length of ID and PW respectively:

1. Since the insider adversary obtained the HPW_i from the channel during the user registration phase, s/he guesses a value for ID_i and calculates the A_3 with ID_i and HPW_i, if the calculated value of A_3 is the same as the value of A_3 obtained from the smart card, the guess is correct. Otherwise, s/he has to guess again.

2. After obtaining the correct ID_i, s/he guesses a value for PW_i and obtains the r_a from A_2 retrieved from the smart card as $r_a = A_2 \oplus h(ID_i \| PW_i)$. Then, the adversary puts the obtained r_a together with ID_i in the relation $h(ID_i \| r_a)$, If the result was equal to HID_i, then the obtained r_a is correct. Otherwise, s/he should guess another value for PW_i and repeat this step to get the r_a. In that case, the correct value of PW_i has been obtained at that stage. So ID_i and PW_i are obtained. The success probability of this attack equals to 1 and its complexity equals to $2^{|ID|} + 2^{|PW|}$.

5 Proposed protocols

In this section, we improve the PRLAP-IoD protocol in two scenarios, (a) and (b), to ensure that it is adequately protected against known attacks. Scenario (a) is for conditions where resistance to user traceability attack is not required, such as personal and recreational uses, which have a lower computational load. Scenario (b) is for traceability-sensitive conditions, such as military use, which obviously has a higher computational load. Subsequently, we use the Scyther security analysis tool to demonstrate that the proposed protocols resist known active and passive attacks.

Scyther is a tool developed for the automated verification or falsification of security protocols and is presented in its updated versions assuming perfect cryptography. This tool assumes that all cryptographic functions are perfect and correct, and the adversary will learn nothing from an encrypted message unless s/he knows the decryption key. Therefore, the Scyther tool can be used to identify design security problems in a protocol.

5.1 Assumptions

Proposed schemes use a noise-resistant and reliable PUF function capable of achieving 100 percent accuracy in complex environments (voltage fluctuations and temperature variations), as reported in[28]. In the proposed schemes, secure one-way hash functions and XOR operations are also utilized to enhance security and resistance against high-level attacks. The proposed authentication protocols same as their predecessor PRLAP-IoD include five phases: pre-deployment, registration, login, authentication, and password update.

The notations used in these protocols are also shown in Table 2. To address PRLAP-IoD protocol's security vulnerabilities, we focus solely on enhancing the user registration phase, the drone-to-drone authentication phase, the login and authentication phase and password update phase. As a result, the other phases remain largely similar to the PRLAP-IoD protocol, and we have chosen not to repeat them in order to avoid redundancy.

5.2 Fixing the user recognition flaw

In the proposed protocols, M_4 is computed as $M_4 = HDID_j \oplus h(HID_i \| K_{GS}^{i*} \| T_3)$ and instead of $HDID_j$, the gateway sends HID_i to the server, which based on it, the server understands which gateway it is communicating with and, based on retrieving $HDID_j^*$ at the server side as $M_4 \oplus h(HID_i \| K_{GS}^{i*} \| T_3)$, also understands which drone it is communicating with.

5.3 Fixing the secret value disclosure attack

To enhance PRLAP-IoD security and bolster resistance against secret disclosure attack presented in this paper, it is proposed that during the user registration phase, to calculate the pseudonymous identity and password and also A_3, wherever the user's identity is used, the user's password should also be concatenated to it. This increases the computational complexity and makes it more difficult for adversaries to guess the values. In fact, this modification prevents attackers from easily guessing the user's identity and password individually, forcing them to attempt both simultaneously. Figure 6 illustrates the improved user registration process, which consequently alters the values of HID_i, HPW_i, and A_3 in the enhanced authentication phase. In the figure 6, the changes compared to the user registration phase of the PRLAP-IoD protocol are highlighted in blue.

5.4 Fixing key disclosure attack

To strengthen the security of the enhanced protocols against key disclosure attacks, the drone receives a challenge C_j during the registration phase and computes its response. In the authentication phase, the gateway calculates $A_1^* = A_1 \oplus HPW_i^* \oplus h(C_j^{new} \| T_1 \| HDID_j)$ and sends (A_1^*, C_j^{new}, T_1) to the drone. The drone based on these values, retrieved $(C_j^* \| HID_i^*)$ as $A_1^* \oplus HDID_j \oplus h(C_j^{new} \| T_1 \| HDID_j)$. To initiate a new session, the new challenge C_j^{new} is compared with the previously stored C_j^*. If these two values differ, the authentication process can proceed. However, if they are the same, both the process and the session will be terminated. This calculation is illustrated in the authentication phase of the two improved protocols in Figures 7 and 8, respectively while the changes compared to the login and authentication phase of the PRLAP-IoD protocol are highlighted in blue.

5.5 Fixing traceability attack and anonymity violation

To break the mentioned traceability attack in the subsection 4.3, in the proposed protocols, the message A_1^* is changed to the value $A_1^* = A_1 \oplus HPW_i^* \oplus h(C_j^{new} \| T_1 \| HDID_j)$. Since in the proposed protocols, the $HDID_j$ is no longer sent in the channel, one can not retrieve $(C_j^* \| HID_i^*)$ by computing $A_1^* \oplus HDID_j \oplus h(C_j^{new} \| T_1 \| HDID_j)$ and so can not retrieve constant information for traceability attack. Proposed protocols are secure against the traceability attack presented in subsection 4.3, but due to the fact that HID_i is sent through public channel instead of $HDID_j$ and not updated in the proposed protocol (a), it does not provide resistance against user traceability attack whereas the proposed protocol (b) introduces improvements that secures it against user traceability attack also.

To implement this, we needed to refresh the pseudonymous identities of both the user and the drone for each session, which would

be utilized in the subsequent session. The new pseudonymous identity for the drone, denoted as $NHDID_j = h(HDID_j \| R_j^{new})$, is generated by the drone, and the same value is computed on the server side as $NHDID_j^* = h(HDID_j \| R_j^{new})$. This required generating R_j^{new} on the server, which has been accomplished. Additionally, since the user possesses R_j^{new}, he can easily compute $NHDID_j$. Both the old and new pseudonymous identities of the drones are stored on the server and in the gateway node, while the drone retains only the new identity due to memory constraints. The user's identity is updated by calculating $NHID_i = h(HID_i \| R_1)$ on their device, with the server also computing this value as $NHID_i^* = h(HID_i \| R_1^*)$. The old and new pseudonymous identities of the user are kept in the server database and on the user's device. The drone also needs to retain this value, to do so, the user first generates $M_9 = R_1 \oplus h(R_j^{new} \| NHDID_j)$ on their side and sends it to the drone. The drone then calculates R_1^* using the received M_9 and derives the user's new pseudonymous identity as $NHID_i = h(HID_i \| R_1^*)$, storing it in its memory. These processes are illustrated in Figure 8.

5.6 Fixing integrity contradiction attack

Addressing the integrity contradiction attack in the proposed protocols for the login and authentication phases (protocol (a) and protocol (b)) involves slight differences, which are detailed separately below.

The integrity of the transmitted values is also ensured by using hash functions, and if the adversary changes the values, the correct information will not be received on the recipient side and the receiver will notice the change.

In the gateway side of the proposed protocols, the message M_4 is also changed to $M_4 = HDID_j \oplus h(HID_i \| K_{GS}^{i*} \| T_3)$. For protocol (a), other suggested improvements include a modification where the new value of Z_1 is calculated as $Z_1 = h(M_1 \| M_2 \| T_2 \| R_j^{new})$ for the messages M_1 and M_2. This method safeguards the integrity of messages M_1, M_2, and T_2 by transmitting Z_1 to the user, who then computes a new value $Z_1^* = h(M_1 \| M_2 \| T_2 \| R_j^{new})$ and compares the two values. Since R_j^{new} is known only to the two parties and is not transmitted over the public channel, any potential adversary remains unaware of it. If the Z_1 message is tampered with, the user will identify this discrepancy upon receiving it and performing the calculation and comparison with Z_1^*, thereby halting any further communication and the authentication process.

In the protocol (a), the user computes $Z_2 = h(M_3 \| M_4 \| HID_i \| T_2 \| T_3 \| R_1)$ to verify the integrity of the M_3, M_4, HID_i, T_2 and T_3 messages. In Z_2, R_1 is a randomly generated number that is not sent over the public channel. The server derives R_1^* based on the received message and other pertinent information. Therefore, if an attacker modifies M_3, M_4, HID_i, T_2 and T_3 messages, the server will notice them when it receives Z_2 and calculates $Z_2^* = h(M_3 \| M_4 \| HID_i \| T_2 \| T_3 \| R_1^*)$, as the two values will not align. Similarly, Z_3 is calculated to ensure the integrity of message M_5, M_6, M_7 and T_4 in the protocol (a), but in this case, the server is the sender and the user is the receiver. The critical value used to compute Z_3 is the same random number generated by the user. The server computes $Z_3 = h(M_5 \| M_6 \| M_7 \| T_4 \| R_1^*)$ and sends it to the user, who then calculates $Z_3^* = h(M_5 \| M_6 \| M_7 \| T_4 \| R_1)$ and compares it with the received value. If they match, the user can confirm the integrity of the M_5, M_6, M_7 and T_4 messages.

The user also computes the value of $Z_4 = h(M_7 \| M_8 \| T_4 \| T_5 \| R_j^{new})$ to verify the integrity of the M_7, M_8, T_4 and T_5 messages before sending it to the drone. In this case, the key value used, similar to the calculation of Z_1, is R_j^{new}. The drone then calculates $Z_4^* = h(M_7 \| M_8 \| T_4 \| T_5 \| R_j^{new})$ and checks if it matches the received Z_4 value. If they are equal, it confirms the correctness and integrity of the M_7, M_8, T_4 and T_5 messages. These calculations take place during the authentication step, as illustrated in the proposed protocol (a) in Figure 7.

In the protocol (b), additional calculations are required to address traceability attacks and breaches of anonymity, leading to the computation of the user's new identity, referred to as $NHDID_j$ as $h(HDID_j \| R_j^{new})$. To ensure the integrity of this identity, the Z_1 message is computed as $Z_1 = h(M_1 \| M_2 \| R_j^{new} \| NHDID_j)$. Both the server and user verify the correctness of this message by checking it against Z_1^* and confirming its equality with the received Z_1. The user's device also generates the value $NHID_i = h(HID_i \| R_1)$, which is verified by calculating Z_2 as $Z_2 = h(M_3 \| M_4 \| R_1 \| NHID_i)$ and sending it to the server. The server computes $NHID_i^* = h(HID_i \| R_1^*)$ and then computes $Z_2^* = h(M_3 \| M_4 \| R_1^* \| NHID_i^*)$ for equality checking with Z_2, thereby ensuring message integrity. Additionally, the M_7 message on the server has been modified to $M_7 = h(R_j \| NHDID_j^* \| T_2 \| T_4)$ compared to the original protocol. $NHID_i^*$ is incorporated into the Z_3 message as $Z_3 = h(M_5 \| M_6 \| M_7 \| T_4 \| R_1^* \| NHID_i^*)$, allowing the user to verify messages integrity by calculating $Z_3^* = h(M_5 \| M_6 \| M_7 \| T_4 \| R_1 \| NHID_i^*)$ and checking its equality with Z_3. In the proposal (b), a new value, $M_9 = R_1 \oplus h(R_j^{new} \| NHDID_j)$, is also calculated in the gateway side, to determine R_1^* in the drone for obtaining the new user identity, $NHID_i$. In the proposed protocol (b), the value of $Z_4 = h(M_7 \| M_8 \| M_9 \| R_j^{new} \| T_4 \| T_5)$ is also added to provide the transmitted messages integrity, while in the drone side, it computes $Z_4^* = h(M_7 \| M_8 \| M_9 \| R_j^{new} \| T_4 \| T_5)$ and checks its equality with Z_4, thereby ensuring message integrity. Figure 8 illustrates the enhanced protocol (b).

Fixing integrity contradiction attack in the improved drone- drone authentication phase

To preserve integrity property, it is suggested that in the drone-drone authentication phase of proposed protocols the encryption of the secret key be accomplished as $\{SK_3 \| h(SK_3)\}_{SK_1/SK_2}$. This allows the recipient to verify the integrity and correctness of sent session key. Because the second part of the concatenated value is a hash of the first part. So, the receipt can verify the correctness of decrypted session key by computing its hash value and comparing with the retrieved second part. Figure 9 illustrates the improved drone- drone authentication phase while the changes compared to the drone-drone authentication phase

of the PRLAP-IoD protocol are highlighted in blue.

5.7 Key management and revocation strategy

In our proposed protocols, a new session key is generated with each authentication, which increases security by limiting the impact of possible key disclosure. The password PW value can be updated as needed as described below, allowing for timely changes to the secret values at the gateway side. To effectively manage device and key revocation, we also implemented a Centralized Revocation List (CRL) that is updated at the gateway side whenever a device is compromised, preventing the drone from communicating with another drone or server.

5.7.1 Updating password phase

In this phase, the proposed protocols (a) and (b) have an optional function to update the old password. Only GWN_i can do this, not the Ser as below:

1. The GWN_i asks the U_i to input the outdated login credentials (ID_i, PW_i^{old}).
2. Upon receiving (ID_i, PW_i^{old}) from U_i, GWN_i checks whether $A_3 = A_3^{old}$ is or not. For this purpose, it computes the random number $r_a^* = A_2 \oplus h(ID_i \| PW_i^{old})$, $HPW_i^{old} = h(PW_i^{old} \| r_a^* \| ID_i)$ and $A_3^{old} = h(ID_i \| HPW_i^{old} \| PW_i^{old})$, if it is not valid, the gateway will reject the update request. Otherwise, GWN_i believes the legitimacy of the user and requests a new password PW_i^{new} from U_i.
3. After receiving PW_i^{new} from user, GWN_i computes $HPW_i^{new} = h(PW_i^{new} \| r_a^* \| ID_i)$, $K_{GS}^i = W_i \oplus HPW_i^{old}$, $W_i^{new} = K_{GS}^i \oplus HPW_i^{new}$, $A_2^{new} = r_a^* \oplus h(ID_i \| PW_i^{new})$ and $A_3^{new} = h(ID_i \| HPW_i^{new} \| PW_i^{new})$. Finally, W_i^{new}, A_2^{new} and A_3^{new} are stored to GWN_i to replace the W_i, A_2 and A_3 respectively.

6 Proposed protocols security analysis

In this section, we will formally and informally analyze the security of the improved schemes. We will also evaluate their resilience against common attacks. We will employ the Scyther tool as a security protocol verifier.

6.1 Informal Security analysis

In this section, we will conduct a comprehensive informal security analysis of the improved authentication schemes and verify their resilience against common attacks in the Dolev-Yao adversary model which explained as below:

Dolev-Yao adversary model:

The Dolev-Yao adversary model[29] is a theoretical framework in cryptography that outlines the capabilities of an adversary attempting to compromise the security of cryptographic protocols. In this model, the adversary has several key capabilities:

- can control communication channels, allowing intercept, modify, or inject messages between legitimate parties;
- has the ability to impersonate legitimate users, facilitating identity spoofing;
- can perform replay attacks by capturing and reusing valid messages;
- may collaborate with other malicious entities to enhance their attack strategies;
- can access the internal state of protocols or participating entities, which helps gather sensitive information; and finally, can execute adaptive attacks, adjusting their tactics based on the responses from legitimate parties.

These capabilities make the Dolev-Yao model a critical consideration in the design and analysis of secure cryptographic systems. In this paper, we use this adversary model in the analysis and proof of the security of the proposed protocols.

Mutual authentication

The server is tasked with confirming the legitimacy of the messages exchanged between the drone and the user. In particular, the drone transmits M_2, to which the server replies with M_7 for mutual authentication. Likewise, the user sends M_4 to the server, who then responds with M_6 to verify their identities to one another. Ultimately, the drone and the user authenticate each other by exchanging M_1 and M_8 respectively. This process of mutual authentication guarantees that all participants are indeed who they assert they are.

Session key agreement

Following a successful protocol execution, a session key SK is established for future communication between the drone and the user. This key $SK = h(K_1 \| R_j^{new})$ incorporates the drone's response R_j^{new} to a challenge C_j^{new} generated by the user, along with a

random number K_1 generated by the user. This shared creation of the session key, involving both the user, drone, and server, ensures mutual authentication and data confidentiality.

Effective interception for illegal login

The proposed schemes utilize $A_3 = h(ID_i \| HPW_i \| PW_i)$, which is generated during the user registration phase. Upon receiving the ID_i and PW_i from the user, the GWN_i computes $r_a^* = A_2 \oplus h(ID_i \| PW_i)$ and $HPWi^* = h(PW_i \| r_a^* \| ID_i)$ to derive $A_3^* = h(ID_i \| HPW_i^* \| PW_i)$. Then A_3^* and A_3 are compared, and if they do not match, the login request is denied. This approach effectively reduces unnecessary communication and computational burden.

Resistance to user device loss attack

Many studies have shown that sensitive data stored in memory can be vulnerable to side-channel attacks[31], including power analysis[32] and electromagnetic analysis[33]. As a result, it is essential to take into account the potential risks associated with a compromised user device when designing authentication protocols. In the proposed schemes, the drone holds the messages B_1, $HDID_j$, and GID_i while the user's device contains the messages W_i, A_1, A_2, A_3, $HDID_j$, HID_i, and $COUNTER$. Even if an attacker manages to obtain these values, s/he would still be unable to retrieve the actual identity ID_i, password PW_i, and response R_j without access to the random nonce r_a, the challenge C_j, and the server's secret key X_{Ser}. Thus, the proposed schemes effectively defend against physical attacks, such as the loss of a user device.

Resistance to physical attack

In a physical capture attack on a drone, an adversary attempts to extract the values stored in the drone's memory. However, this is thwarted by a PUF within the drone. The PUF's challenge-response pairs are highly dependent on the device's physical characteristics and environmental conditions, making any attempt at tampering or replication futile. A compromised drone would produce verification values that are unrecognizable by the server. Even if the PUF is not tampered with but simulated, the lack of messages, such as C_j and A_1 prevents the adversary from calculating the desired information. Since the PUF is integrated into the physical circuitry during the drone's manufacturing process, its behavior cannot be replicated. Consequently, physical capture attacks on drones equipped with PUFs are rendered ineffective.

Resistance to impersonation attack

The proposed schemes are resistant to both user and drone impersonation attacks. To impersonate a user, an adversary would require access to the user's unique identifier ID_i and password PW_i, which is highly impractical. Similarly, to impersonate a drone, an attacker would need to replicate the unique response R_j generated by the PUF, and it must be provided to the server. However, due to the inherent unpredictability and physical nature of the PUF, it is virtually impossible to generate a matching response. Therefore, the system is robust against all kinds of impersonation attacks.

Resistance to privileged insider attack

A legitimate user submits a registration request to the authority with a message containing HID_i, HPW_i. If an insider attacker attempts to obtain the user's password PW_i, it is impossible due to the presence of r_a, ID_i and the hash function. Therefore, the proposed schemes are highly resistant to privileged insider attacks.

Resistance to de-synchronization attack

In the enhanced protocol (a), there is a requirement to retain the same secret data for each session, and the information does not need to be updated by the entity following successful authentication. Conversely, in the enhanced protocol (b), both the drone's pseudonymous identity and the user's pseudonymous identity are updated across all active entities in the session after a successful session. To mitigate the risk of a de-synchronization attack, we store both the new and old pseudonymous identity values. This means that in the improved protocol (b), if one party updates its information but the message fails to reach the other party for any reason, the other party will not update its values, leading to a state of de-synchronization. In this scenario, the previous pseudonymous identities can still provide protection against this issue, allowing communication within the protocol to proceed using those identities.

Forward /backward secrecy

Forward secrecy (also known as perfect forward secrecy) ensures that even if a long-term private key is compromised in the future, previously established session keys remain secure. This means that past communications cannot be decrypted, as each session uses a unique ephemeral key that is not derived from the long-term key only.

Backward secrecy, on the other hand, refers to the ability to ensure that if a long-term private key is compromised, it does not affect the security of future sessions. In other words, even if an attacker gains access to a long-term private key, they cannot use it to decrypt or compromise any future communications.

The established session key SK is composed of the response R_j^{new} and the random number K_1 of the user not long-term secret values. R_j^{new} and random number K_1 are different for each session. Therefore, even if a malicious adversary obtains long-term secret keys, the authenticity of the previous and future session keys will not be affected. Hence, proposed schemes have forward/backward secrecy features.

Table 3. The Scyther tool security claims and the notations used in SPDL representation of protocol

Notation	Description
G	User/Gateway role
S	Server role
D	Drone role
$Send(S,G,M)$	S sends a message M to G. This action may include encryption or other security measures.
$Receive(S,D,M)$	S receives the message M sent by D. This process may involve decryption and validation of the message.
$match(Y,X)$	The Scyther tool command that compares the value of Y with X and, if equal, replaces the value of Y with the value of X in all protocol specifications.
$claim(S,Secret,K_{GS})$	The Scyther tool security claim that ensures K_{GS} remains confidential and is not disclosed to unauthorized parties.
$claim(S,Alive)$	The Scyther tool security claim that indicates that S is active and capable of responding to messages during the protocol execution.
$claim(S,Niagree)$	The Scyther tool security claim that ensures that S and another party have mutually agreed on a message, providing non-repudiation of this agreement.
$claim(S,Nisynch)$	The Scyther tool security claim that indicates that S and another party are synchronized in their actions or messages within the protocol.
$claim(S,Weakagree)$	The Scyther tool security claim that represents a weaker form of agreement where S can agree on a message but may not have strong proof of this agreement.

6.2 Scheme validation using the Scyther tool

The Scyther tool is a robust and commonly utilized resource for analyzing and validating security protocols. It has been developed to operate on MAC, Linux, and Windows platforms. In this study, the Scyther Compromise-0.9.2 is employed to simulate the proposed protocols on Ubuntu 16.04.6 running on VMware® Workstation 17 Pro.

In this section, we have utilized the Scyther tool to implement and evaluate the proposed protocols. The Scyther tool uses Security Protocol Description Language (SPDL) as its format for accepting security protocols. Within SPDL, each entity in the protocol is assigned a specific role. In the SPDL representation of our proposed protocols, G denotes the User/Gateway role, D signifies the Drone role, and S indicates the Server role. A key aspect of the security analysis involves assessing the protocol's behavior against known attacks as outlined in the Scyther tool claims.The notations and security claims of the Scyther tool are described in Table 3.

Figure 10 showcases the Security Protocol Description Language (SPDL) source code for the proposed protocol (a). This code is a formal representation that meticulously outlines the interactions and behaviors of the entities involved in the protocol, thereby providing a clear framework for understanding its operational mechanics. The SPDL format not only specifies the messages exchanged between participants but also encapsulates the underlying logic governing these exchanges. Following this, the security verification results for proposed protocol (a) are illustrated in Figure 11, which details the outcomes of an extensive analysis conducted using the Scyther tool. This analysis assesses the protocol's resilience against various security threats, such as replay attacks and man-in-the-middle attacks, ensuring that the protocol adheres to established security standards.

In addition to the proposed protocol (a), Figure 12 presents the SPDL source code for the proposed protocol (b), which similarly delineates the key components and interactions essential for its functionality. Accompanying this source code, the corresponding security verification results for proposed protocol (b) are also provided as depicted in the Figure 13, offering insights into its security efficacy. These results stem from a thorough evaluation process aimed at identifying potential vulnerabilities and ensuring that the proposed protocol (b) maintains a robust defense against various attack vectors, thus validating its reliability for practical applications.

6.3 Security comparison of the proposed protocols with recent protocols

In this section, we evaluate the proposed protocols (a) and (b) against recent protocols[9–11,22–24,26,27], with the findings summarized in Table 4. Protocols[10] and[11] exhibit flaws related to forward secrecy and are vulnerable to attacks that can disclose secret values. Precisely, in protocol[22], a privileged insider attacker who gains access to the pseudonymous identity of a registered drone can establish session keys with other drones. Meanwhile, in[23], if a malicious registered user retains the F_i value from a previous authentication, they can utilize this value to compute the session key without needing to send authentication requests to the authentic server during the next authentication step. This undermines the necessity for the server's involvement in generating a new session key, exposing a security weakness in the session key agreement of[23]. Additionally, both protocols[22] and[23] are susceptible to physical capture attacks on the device, as they store long-term key values in the device's memory

Table 4. Security properties comparison of the proposed schemes with recent similar schemes.

Properties	[10]	[11]	[22]	[23]	[24]	[26]	[27]	[9]	Proposed protocol (a)	Proposed protocol (b)
SP_1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
SP_2	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
SP_3	-	-	✓	✓	✓	✓	✓	✓	✗	✓
SP_4	-	-	✓	✓	✓	✓	✓	✓	✓	✓
SP_5	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓
SP_6	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
SP_7	-	-	✗	✓	✓	✓	✓	✓	✓	✓
SP_8	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓
SP_9	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
SP_{10}	-	-	✓	✓	✓	✓	✓	✓	✓	✓
SP_{11}	✗	✗	-	-	✓	✓	✓	✗	✓	✓
SP_{12}	✓	-	-	-	✓	✓	✓	✗	✓	✓
SP_{13}	-	-	✓	-	✓	✓	✓	✗	✓	✓

✓: a protocol meets the property. ✗: a protocol does not meet the property. - not checking the property in the protocol. SP_1: Mutual authentication; SP_2: Session key agreement; SP_3: Anonymity and untraceability; SP_4: Resistant to device loss attack; SP_5: Resistant to physical attack; SP_6: Resistant to impersonation attack; SP_7: Resistant to privileged insider attack; SP_8: Forward and backward secrecy; SP_9: Resistant to replay attack; SP_{10}: Resistant to man-in-the-middle attack; SP_{11}: Resistant to secret values disclosure attack; SP_{12}: Resistant to integrity violation attack; SP_{13}: Resistant to session key disclosure attack.

for subsequent authentication. To the best of our knowledge, there have been no reported attacks or vulnerabilities for IoD authentication schemes of[24],[26] and[27].
Detailed explanations of the security issues in the[9]'s protocol can be found in Section 4. Consequently, the proposed protocols in this study offer enhanced security features compared to the other recent schemes analyzed.

7 Performance evaluation

When developing authentication protocols for the Internet of Drones (IoD) environment, it is essential to take into account the efficient utilization of resources. In this section, we will first compare the computational costs of the proposed protocols with those of protocols[9–11,22–24,26,27], followed by a comparison of their communication costs and storage costs.

7.1 Computational cost comparison

Table 5 shows the execution time of each operation based on the execution time used in[34]. Smart metering in[34] can be used as a reference to help us compare the computational cost of the proposed schemes with other schemes. The calculations were performed on a device equipped with a 798MHz processor and 256MB of RAM, and the server was simulated on an Ubuntu 12.04 virtual machine with a 2.6GHz i5-4300 processor. To consider PUF performance evaluation,[34] uses a 128-bit arbiter PUF circuit on an MSP430 micro controller device with a 798 MHz processor. The timing of cryptographic operations can vary significantly due to several factors related to the hardware and software environments. Key influences include hardware specifications, such as processing power and memory speed, which directly affect execution times; software implementation efficiency, where optimized or hardware-accelerated algorithms can enhance performance; environmental factors like network latency and bandwidth, impacting operations that rely on server communication; concurrent processes that may cause resource contention; and configuration settings, including buffer sizes and algorithm parameters. Understanding these factors helps clarify the variability in operation times across different environments, facilitating a better interpretation of results.

In [24], the communication cost at the drone is $2T_{PUF}+13T_h$ and at the user side is $12T_h$, and at the server side, is $17T_h$. The total execution time for the authentication phase in this scheme approximately is 1.077 milliseconds.
In [26], the communication cost at the drone $T_{PUF}+7T_h$ and at the user side is $8T_h$, and at the server side, is $2T_{PUF}+11T_h$. The total execution time for the authentication phase in this scheme approximately is 0.631 milliseconds.
In [27], the communication cost at the drone is $T_{PUF}+4T_{mp}+4T_h+2T_{pa}$ and and at the server side, they amount to $5T_{mp}+5T_h+T_{pa}$. The total execution time for the authentication phase in this scheme approximately is 24.424 milliseconds.
In the scheme[9], the drone executes six hash operations and two PUF operations. Therefore, the execution time on the drone is $2T_{PUF}+6T_h \approx 0.396\ ms$. The user executes eight hash operations. Therefore, the execution time on the user side is

Table 5. Execution time of different operations used in this paper (in ms)[34].

Notation	Description	Drone/Gateway side	Server side
T_h	Hash Function Execution time	0.026	0.011
T_{mp}	ECC Multiplication Execution Time	5.9	2.6
T_{PUF}	PUF Execution Time	0.12	-
T_{fe}	Fuzzy Extractor Execution Time	1.67	2.85
T_{pa}	ECC Add Execution Time	0.30	0.11
T_s	Symmetric Encryption/Decryption Execution Time	0.79	0.41
T_{cmap}	Chaotic Map Execution Time	0.92	0.66

Table 6. Comparison of computation cost (in ms) of proposed schemes with similar recent schemes

Schemes	$D_j/Device_j$	U_i/GWN_i	$Server$	Total
Tanveer et al.[10]	$3T_{cmap}+2T_s+4T_h$	$T_{fe}+5T_{cmap}+4T_s+6T_h$	$3T_{cmap}+T_s+3T_h$	16.453 *ms*
Irshad et al.[11]	$3T_{mp}+8T_h+1T_{pa}$	$T_{fe}+3T_{mp}+15T_h+T_{pa}$	$2T_s+3T_{mp}+9T_h+T_{pa}$	47.097 *ms*
Zhang et al.(2020)[22]	$7T_h$	$10T_h$	$7T_h$	0.519 *ms*
Gupta et al.[23]	$4T_h$	$7T_h$	$5T_h$	0.341 *ms*
Choi et al.[24]	$2T_{PUF}+13T_h$	$12T_h$	$17T_h$	1.077 *ms*
Modarres and Sarbishaei[26]	$T_{PUF}+7T_h$	$8T_h$	$2T_{PUF}+11T_h$	0.631 *ms*
Nyangaresi et al.[27]	$T_{PUF}+4T_{mp}+4T_h+2T_{pa}$	-	$5T_{mp}+5T_h+T_{pa}$	24.424 *ms*
Zhang et al.(2024)[9]	$6T_h+2T_{PUF}$	$8T_h$	$6T_h$	0.67 *ms*
Proposed scheme (a)	$8T_h+2T_{PUF}$	$12T_h$	$8T_h$	0.848 *ms*
Proposed scheme (b)	$11T_h+2T_{PUF}$	$15T_h$	$12T_h$	1.048*ms*

$8T_h \approx 0.208\ ms$. Similarly, the server performs six hash operations, with a time of $6T_h \approx 0.066ms$. Thus, the total execution time of the scheme[9] approximately is 0.67 *ms*.

In the proposed protocol (a), the drone executes eight hash operations and two PUF operations. Therefore, the execution time on the drone is $2T_{PUF}+8T_h \approx 0.448\ ms$. The user also executes 12 hash operations, resulting in an execution time of $12T_h \approx 0.312\ ms$ on the user side. Similarly, the server performs eight hash operations, with a time of $8T_h \approx 0.088\ ms$. Thus, the total execution time required for the computations in the proposed protocol (a) approximately is 0.848 *ms*.

In the proposed protocol (b), the drone executes eleven hash operations and two PUF operations. Therefore, the execution time on the drone is $2T_{PUF}+11T_h \approx 0.526\ ms$. The user also executes 15 hash operations, resulting in an execution time of $15T_h \approx 0.39ms$ on the user side. Similarly, the server performs twelve hash operations in a time of $12T_h \approx 0.132ms$. Thus, the total execution time required for the computations in the proposed scheme (b) approximately is 1.048 *ms*.

The execution time for the schemes[9–11,22,23] is calculated in a similar way, with the results presented in Table 6.

Figure 14 shows the computational cost comparison of the proposed schemes with recent similar schemes.

7.2 Communication cost comparison

First, we give the basic definition of communication cost, which is equal to the number of bits sent on the communication channels by entities present in each session. The lower the number of these bits, the lower the network traffic and the higher the transfer speed.

A comprehensive evaluation of communication costs of proposed protocols is conducted based on key metrics, namely the quantity and size of exchanged messages.

The number of exchanged messages for the proposed scheme and schemes[9–11,22–24,26,27] was counted as described below. In the Tahavor et al. protocol[35], we assumed $32-$bit timestamp B_t, $128-$bit Pseudonymous identity B_{id}, $128-$bit random numbers B_r, $160-$bit SHA-1 hash function B_h, $160-$bit PUF challenge-response pair B_p, and $128-$bit fuzzy extractor output B_f, and 320 bits for the elliptic curve point B_e. We also used conventional values for further calculations, including $256-$bit chaotic map B_{cmap}, $160-$bit private key B_{pr}, and $320-$bit public key B_{pu}. Additionally, based on calculations in[10], two values were uniquely used: $128-$bit ciphertext B_{ct} and a variable-length $T_{R_a}(SP_{11})$ that could be either 288 bits or 416 bits, depending on its content. The bit lengths of the parameters used in the communication cost comparison are shown in the Table 7.

[10]'s scheme involves three message exchanges for authentication and access to the drone. These messages have sizes of 576 bits, 608 bits, and 320 bits, resulting in a total message size of 1504 bits.[11]'s scheme involves three message exchanges as $\langle W_{u_i}, Cert^{+}_{u_i}, R'_1, SID^{*}_{u_i}, Q_{u_i}, TS_1\rangle$, $\langle W^{*}_{u_i}, X_i, Cert'_{CN_j}, R'_1, D_{u_i}, TS_1, TS_2\rangle$ and $\langle Cert^{+}_{SD_k}, N_{SD_k}, R'_2, SKV, W^{*}_{u_i}, TS_2, TS_3\rangle$. The first

Table 7. The bit lengths of the parameters used in the communication cost comparison

Parameter	Bit length (Bits)
Timestamp	32
Identity	160
Random Numbers	128
Hash Function	160
Fuzzy Extractor Output	128
Elliptic Curve Point	320
Chaotic Map Output	256
ECC Private Key	160
ECC Public Key	320
Challenge for PUF	160
Output of PUF	160
Symmetric Encryption/Decryption	128

message, comprising a $128-$bit ciphertext (W_{u_i}), $Cert_{u_i}^{+}$ comprises various components: a private key, two elliptic curve multiplications contributing 1120 bits in total, R_1' is an elliptic curve multiplication which occupying 320 bits, $SID_{u_i}^{*}$ is a pseudonymous identity with 160 bits, Q_{u_i} is the output of a hash function and is equivalent to 160 bits and finally TS_1 is a $32-$bit timestamp, resulting in a total message size of 992 bits. The second and third messages, respectively, require 1024 bits and 1184 bits. Thus, the overall size of all messages in[11]'s scheme is 3200 bits. In[24], the transmitted messages are $Msg1 = \{PID_i, M_1, M_2, V_1, T_1\}$, $Msg2 = \{M_3, M_4, V_2, T_2\}$, $Msg3 = \{M_5, M_6, V_3, T_3\}$, and $Msg4 = \{M7, V4, T4\}$. Therefore, the communication cost for this scheme equal with $Msg1 = 160+160+160+160+32 = 672$ bits, $Msg2 = (160+160)+160+160+32 = 672$ bits, $Msg3 = (160+160)+160+160+32 = 672$ bits, and $Msg4 = 160+160+32 = 352$ bits. Thus, the total communication overhead during authentication is $672+672+672+352 = 2368$ bits as shown in Table 8.

In[26], the transmitted messages are $M_1 = \{\text{PID}_u, C_g, X_1, X_2, T_1, I_1\}$, $M_2 = \{\text{PID}_d, \text{PID}_u, C_d, X_3, X_4, T_2, I_2\}$, and $M_3 = \{\text{PID}_d, X_5, T_3, I_3\}$. Therefore, the communication cost for this scheme equal with $M_1 = 160+160+160+160+32+160 = 832$ bits, $M_2 = 160+160+160+160+160+32+160 = 992$ bits, and $M_3 = 160+32+32+160 = 384$ bits. Thus, the total communication overhead during authentication is $832+992+384 = 2208$ bits, as shown in Table 8.

In[27], the transmitted messages are $D_{req} = \{PID_i, D_1, D_2, D_3\}$, $G_{Res} = \{G_2, G_4, G_5, G_6, T_1\}$, and $D_{Auth1} = \{PID_j, D_5, D_6, T_3\}$. So, the communication cost for this scheme is $D_{req} = 160+320+320+320 = 1120$ bits, $G_{Res} = 160+320+160+160+32 = 832$ bits, and $D_{Auth1} = 160+320+160+32 = 672$ bits. Thus, the total communication overhead during authentication is $1120+832+672 = 2624$ bits.

In[9]'s scheme, the drone sends one message and receives two messages, the user sends three messages and receives two messages, and the server sends one message and receives one message. These five messages are $\langle A_1^{*}, C_j^{new}, T_1\rangle$, $\langle M_1, M_2, T_2\rangle$, $\langle M_3, M_4, HDID_j, T_2, T_3\rangle$, $\langle M_5, M_6, M_7, T_4\rangle$, and $\langle M_7, M_8, T_4, T_5\rangle$, which include ten hash functions, an identity, seven time stamps, and a PUF function challenge. Therefore, the total size of these messages is $(10\times 160)+(1\times 160)+(7\times 32)+(1\times 160) =$ 2144 bits.

In the proposed scheme (a), the messages are $\langle A_1^{*}, C_j^{new}, T_1\rangle$, $\langle M_1, M_2, Z_1, T_2\rangle$, $\langle M_3, M_4, HID_i, Z_2, T_2, T_3\rangle$, $\langle M_5, M_6, M_7, Z_3, T_4\rangle$, and $\langle M_7, M_8, Z_4, T_4, T_5\rangle$. Notably, by including the values of Z_1, Z_2, Z_3 and Z_4, four additional hash functions have been incorporated into the transmitted items. The remaining values consist of an identity, seven timestamps, and a challenge for the PUF function. In total, the size of all messages exchanged in the proposed scheme (a) equals to $(14\times 160)+(1\times 160)+(7\times 32)+(1\times 160) = 2784$ bits.

In scheme (b), the messages exchanged include $\langle A_1^{*}, C_j^{new}, T_1\rangle$, $\langle M_1, M_2, Z_1, T_2\rangle$, $\langle M_1, M_2, M_3, M_4, Z_1, Z_2, HID_i, T_2, T_3\rangle$, $\langle M_5, M_6, M_7, Z_3, T_4\rangle$ and $\langle M_7, M_8, M_9, Z_4, T_4, T_5\rangle$. The total communication cost is calculated based on the transmission of 18 hash functions, one identity, seven timestamps, and one challenge for the PUF function. This results in a total of $(18\times 160)+(1\times 160)+(7\times 32)+(1\times 160) = 3424$ bits. The communication cost for schemes[22,23] is similarly computed, and the findings are presented in Table 8.

Figure 15 shows communication cost comparison (in bits) of proposed schemes with recent similar schemes.

After conducting security evaluations and comparing the proposed schemes with similar ones, we found that the schemes (a) and (b) incur higher communication and computational costs. Nevertheless, considering their robustness against various known attacks and the partial vulnerability of the other schemes to these threats, the rise in costs can be regarded as minimal.

Table 8. Communication cost comparison (in bits) of proposed schemes with similar recent schemes

Schemes	Rounds	Communication Cost (bits)
Tanveer et al.[10]	3	1504
Irshad et al.[11]	3	3200
Zhang et al.(2020)[22]	3	1472
Gupta et al.[23]	5	2080
Choi et al.[24]	4	2368
Modarres and Sarbishaei[26]	4	2208
Nyangaresi et al.[27]	3	2624
Zhang et al.(2024)[9]	5	2144
Proposed scheme (a)	5	2784
Proposed scheme (b)	5	3424

Table 9. The total storage cost comparison of proposed schemes with other similar schemes

Schemes	Storage cost (bits)
Tanveer et al.[10]	480
Irshad et al.[11]	1760
Zhang et al.(2020)[22]	320
Gupta et al.[23]	800
Choi et al.[24]	320
Modarres and Sarbishaei[26]	960
Nyangaresi et al.[27]	1440
Zhang et al.(2024)[9]	480
Proposed scheme (a)	480
Proposed scheme (b)	480

7.3 Storage cost comparison

This section compares the memory requirements for storing essential parameters in the devices (for the schemes[9,10,22,24,26,27] drone and IoT devices for schemes[11,23]) and the drone for our proposed schemes. Table 9 summarizes the results.

In[10]'s scheme, the data elements $\langle SID_j, DSK_j, GID_k \rangle$ are stored in the drone's memory where SID_j is a pseudonymous identity of 160 bits, DSK_j is a private key of 160 bits, and GID_k is another identity of 160 bits. Consequently, the total storage requirement for[10]'s scheme is also 480 bits.

[11]'s scheme stores the following values in IoT device memory: SID_{SD_k} and ID_{CN_j} are two $160-$bit pseudonyms for identification and authentication, N_{CN_j} is a $320-$bit elliptic curve point multiplication result, Pr_{SD_k} is a $160-$bit private key and Pub_{SD_k} is a $320-$bit public key, N_{SD_k} is another $320-$bit elliptic curve point multiplication result, $Cert_{SD_k}$ is a $160-$bit and K_{SD_k,CN_j} is a $160-$bit hash function output respectively. Based on these values, the total required storage is 1760 bits.

In[22]'s scheme, the $\langle \alpha_j, PID_j \rangle$ are stored in the drone's memory, which α_j is a $160-$bit hash function output and PID_j is a pseudonymous identity of a drone that is equal to 160 bits. Based on the specified values, the total required storage is $(2 \times 160) = 320$ bits.

[23]'s scheme stores the values in the wearable device as $\langle e_j, x_j, f_j, MI_u, MGID_i \rangle$ where e_j, x_j and f_j are three $160-$bit hash function outputs. $160-$bit pseudonymous identities of user and gateway are denoted as MI_u and $MGID_i$, respectively. So, based on the specified values, the total required storage is $(5 \times 160) = 800$ bits.

In[24], the drone stores only $\{PDID_j, b_j\}$, which is equal to $\{160+160=320 \text{ bits}\}$. The drone in[26], stores ID_g, ID_d, and MK, along with an emergency list of essential secrets, which is equal to: $\{2 \times (160+160+160) = 960 \text{ bits}\}$. In[27], the drone stores the values $\{PUF(.), ID_i, Ch_i, D_1, D_2, D_3, D_4\}$ in its memory, which is equal to $\{160+160+320+320+320+160 = 1440 \text{ bits}\}$. In the scheme presented in[9], the data $\langle B_1, HDID_j, GID_i \rangle$ are kept in the drone's memory, where B_1 is a hash function output of 160 bits, $HDID_j$ represents the drone's pseudonymous identity of 160 bits, and GID_i denotes a gateway node identity of 160 bits. This results in a total memory requirement of $(3 \times 160) = 480$ bits. Even with the enhanced security features of the improved protocol (a), the storage requirement for the drone remains the same as in[9]'s scheme. Likewise, the proposed protocol (b) also necessitates an equivalent amount of memory as both[9]'s scheme and improved protocol (a).

Figure 16 depicts the storage cost of the proposed schemes compared to other recent similar schemes.

8 Conclusion

In recent years, the Internet of Drones (IoD) has seen significant expansion and is increasingly utilized in both civilian and military sectors. Many researchers have developed authentication protocols tailored for IoD environments; however, numerous protocols have been found inadequate in defending against various attacks and have not managed to maintain low communication and computational costs.

Recently, Zhang et al. introduced an authentication method leveraging Physical Unclonable Functions (PUFs), incorporating unique identifiers and hash functions to enhance security in Internet of Devices (IoD) communications. However, our study revealed that this method is susceptible to multiple attacks, such as secret value exposure, integrity breaches, key extraction, traceability issues, and loss of anonymity. The attacks discussed were shown to have a success probability of one.

In this paper, we also presented two PUF-based authentication protocols specifically designed for IoD communications, focusing on drone-to-user and drone-to-drone interactions, to tackle these pressing challenges. The proposed protocols, both in terms of communication and computational costs, are higher than the previous scheme by Zheng et al. However, they have similar storage costs to that scheme. An important point to note is that these proposed protocols offer enhanced security against a wider range of attacks compared to their predecessor. Specifically, proposed protocol (a) is not secure against anonymity violation and user traceability attacks, making it suitable for applications where these properties are not critical. On the other hand, if anonymity and resistance to user traceability are essential, protocol (b) should be utilized, as it provides these features, albeit at a higher computational and communication cost compared to protocol (a). The suggested protocols (a) and (b) are also scalable since the data from the drones is kept on the server, which has no constraints. Each drone retains only its own data. In drone-drone communication, the default gateway generates the communication key, which is then encrypted using the long-term key that is shared between the drones and the gateway before being sent to the drones. As a result, each drone only needs to keep its own key with the default gateway and does not have to store separate communication keys for interactions with other drones.
Security evaluations and performance analyses indicated that the proposed protocols encompass all necessary security features essential for unmanned aerial vehicle networks, thereby ensuring the reliability of the systems.

Funding Declaration

Masoumeh Safkhani was supported by Shahid Rajaee Teacher Training University under grant number 1404/385773.

Conflict of interest

The authors declare no conflict of interest. The founding sponsors had no role in the design of the study; in the collection, analysis, or interpretation of data; in the writing of the manuscript, and in the decision to publish the results.

Availability of Data and Materials

All source codes used for the analysis and protocol validation in this work have been made publicly available at the following GitHub repository:
https://github.com/Masouemh-Safkhani/Two-Secure-Authentication-Protocols-for-Mitigating-Vulnerabilities-in-IoD

Authors Statement

Masoumeh Safkhani: Conception and design of study, Acquisition of data, Analysis and/or interpretation of data, Validation, Supervision, Formal analysis, Writing - original draft, Writing - review & editing. **Mahmoud Ghorbani Fard:** Conception and design of study, Acquisition of data, Analysis and/or interpretation of data, Software, Writing - original draft.

References

1. Kumar, V., Ahmad, M. & Kumari, A. A secure elliptic curve cryptography based mutual authentication protocol for cloud-assisted TMIS. *Telematics Informatics* **38**, 100–117 (2019).
2. Sharma, N. & Dhiman, P. A survey on iot security: challenges and their solutions using machine learning and blockchain technology. *Clust. Comput.* **28**, 313 (2025).
3. Sharma, N. & Dhiman, P. A secure addressing mutual authentication scheme for smart iot home network. *Multimed. Tools Appl.* **84**, 25111–25143 (2025).

4. Sharma, N. & Dhiman, P. Lightweight privacy preserving scheme for iot based smart home. *Recent Adv. Electr. & Electron. Eng. (Formerly Recent Patents on Electr. & Electron. Eng.* **17**, 763–777 (2024).
5. Sharma, N. & Dhiman, P. Design of secure and unique addressing with mutual authentication scheme in iot networks. *Peer-to-Peer Netw. Appl.* **18**, 50 (2025).
6. Gharibi, M., Boutaba, R. & Waslander, S. L. Internet of drones. *IEEE Access* **4**, 1148–1162 (2016).
7. Derhab, A. *et al.* Internet of drones security: Taxonomies, open issues, and future directions. *Veh. Commun.* **39**, 100552 (2023).
8. Svaigen, A. R., Boukerche, A., Ruiz, L. B. & Loureiro, A. A. Design guidelines of the internet of drones location privacy protocols. *IEEE Internet Things Mag.* **5**, 175–180 (2022).
9. Zhang, Z. *et al.* PRLAP-IoD: A PUF-based robust and lightweight authentication protocol for internet of drones. *Comput. Networks* **238**, 110118 (2024).
10. Tanveer, M., Alasmary, H., Kumar, N. & Nayak, A. SAAF-IoD: secure and anonymous authentication framework for the internet of drones. *IEEE Transactions on Veh. Technol.* (2023).
11. Irshad, A. *et al.* SUSIC: a secure user access control mechanism for sdn-enabled iiot and cyber–physical systems. *IEEE Internet Things J.* **10**, 16504–16515 (2023).
12. Mahmood, K. *et al.* A security enhanced chaotic-map based authentication protocol for internet of drones. *IEEE Internet Things J.* (2024).
13. Srinivas, J., Das, A. K., Wazid, M. & Vasilakos, A. V. Designing secure user authentication protocol for big data collection in IoT-based intelligent transportation system. *IEEE Internet Things J.* **8**, 7727–7744 (2020).
14. Turkanović, M., Brumen, B. & Hölbl, M. A novel user authentication and key agreement scheme for heterogeneous ad hoc wireless sensor networks, based on the internet of things notion. *Ad Hoc Networks* **20**, 96–112 (2014).
15. Porambage, P., Schmitt, C., Kumar, P., Gurtov, A. & Ylianttila, M. Two-phase authentication protocol for wireless sensor networks in distributed iot applications. In *2014 IEEE Wireless Communications and Networking Conference (WCNC)*, 2728–2733 (IEEE, 2014).
16. Alzahrani, B. A., Barnawi, A. & Chaudhry, S. A. A resource-friendly authentication protocol for UAV-based massive crowd management systems. *Secur. Commun. Networks* **2021**, 3437373 (2021).
17. Srinivas, J., Das, A. K., Kumar, N. & Rodrigues, J. J. TCALAS: temporal credential-based anonymous lightweight authentication scheme for internet of drones environment. *IEEE Transactions on Veh. Technol.* **68**, 6903–6916 (2019).
18. Zhou, Y., Liu, T., Tang, F. & Tinashe, M. An unlinkable authentication scheme for distributed iot application. *IEEE Access* **7**, 14757–14766 (2019).
19. Wazid, M., Das, A. K., Kumar, N., Vasilakos, A. V. & Rodrigues, J. J. Design and analysis of secure lightweight remote user authentication and key agreement scheme in internet of drones deployment. *IEEE Internet Things J.* **6**, 3572–3584 (2018).
20. Tanveer, M. *et al.* PAF-IoD: PUF-enabled authentication framework for the internet of drones. *IEEE Transactions on Veh. Technol.* (2024).
21. Wu, H. & Preneel, B. AEGIS: a fast authenticated encryption algorithm. In *Selected Areas in Cryptography–SAC 2013: 20th International Conference, Burnaby, BC, Canada, August 14-16, 2013, Revised Selected Papers 20*, 185–201 (Springer, 2014).
22. Zhang, Y., He, D., Li, L. & Chen, B. A lightweight authentication and key agreement scheme for internet of drones. *Comput. Commun.* **154**, 455–464 (2020).
23. Gupta, A., Tripathi, M., Shaikh, T. J. & Sharma, A. A lightweight anonymous user authentication and key establishment scheme for wearable devices. *Comput. Networks* **149**, 29–42 (2019).
24. Choi, J., Son, S., Kwon, D. & Park, Y. A puf-based secure authentication and key agreement scheme for the internet of drones. *Sensors* **25**, 982 (2025).
25. Sharma, M., Narwal, B., Anand, R., Mohapatra, A. K. & Yadav, R. Psecas: A physical unclonable function based secure authentication scheme for internet of drones. *Comput. Electr. Eng.* **108**, 108662 (2023).
26. Modarres, A. M. A. & Sarbishaei, G. Enhancing internet of drones security: A robust multi-factor authentication and key establishment protocol. *IEEE Transactions on Veh. Technol.* (2025).

27. Nyangaresi, V. O., Ahmad, M., Maghrabi, L. A. & Althaqafi, T. Cost-effective puf and ecc based authentication protocol for secure internet of drones communication. *IEEE Internet Things J.* (2025).

28. Chuang, K.-H. *et al.* A physically unclonable function using soft oxide breakdown featuring 0% native ber and 51.8 fj/bit in 40-nm cmos. *IEEE J. Solid-State Circuits* **54**, 2765–2776 (2019).

29. Dolev, D. & Yao, A. On the security of public key protocols. *IEEE Transactions on information theory* **29**, 198–208 (2003).

30. Phan, R. C.-W. Cryptanalysis of a new ultralightweight RFIDauthentication protocol—sasi. *IEEE Transactions on Dependable secure Comput.* **6**, 316–320 (2008).

31. Spreitzer, R., Moonsamy, V., Korak, T. & Mangard, S. Systematic classification of side-channel attacks: A case study for mobile devices. *IEEE communications surveys & tutorials* **20**, 465–488 (2017).

32. Messerges, T. S., Dabbish, E. A. & Sloan, R. H. Examining smart-card security under the threat of power analysis attacks. *IEEE transactions on computers* **51**, 541–552 (2002).

33. De Mulder, E. *et al.* Electromagnetic analysis attack on an FPGA implementation of an elliptic curve cryptosystem. In *EUROCON 2005-The International Conference on" Computer as a Tool"*, vol. 2, 1879–1882 (IEEE, 2005).

34. Gope, P. PMAKE: privacy-aware multi-factor authenticated key establishment scheme for advance metering infrastructure in smart grid. *Comput. Commun.* **152**, 338–344 (2020).

35. Tahavori, M. & Moazami, F. Lightweight and secure puf-based authenticated key agreement scheme for smart grid. *Peer-to-Peer Netw. Appl.* **13**, 1616–1628 (2020).

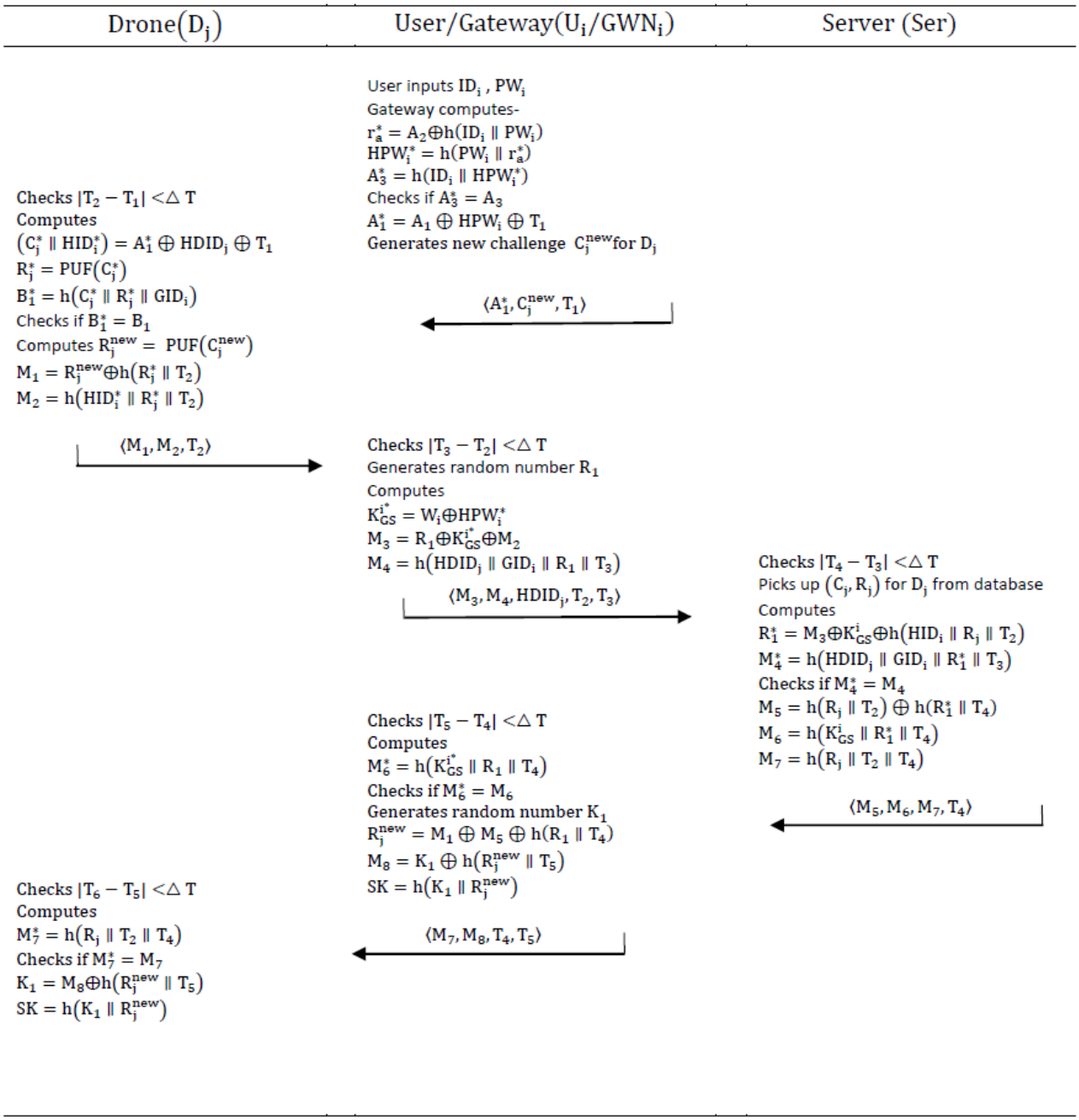

Figure 4. Login and authentication phase of PRLAP-IoD[9]

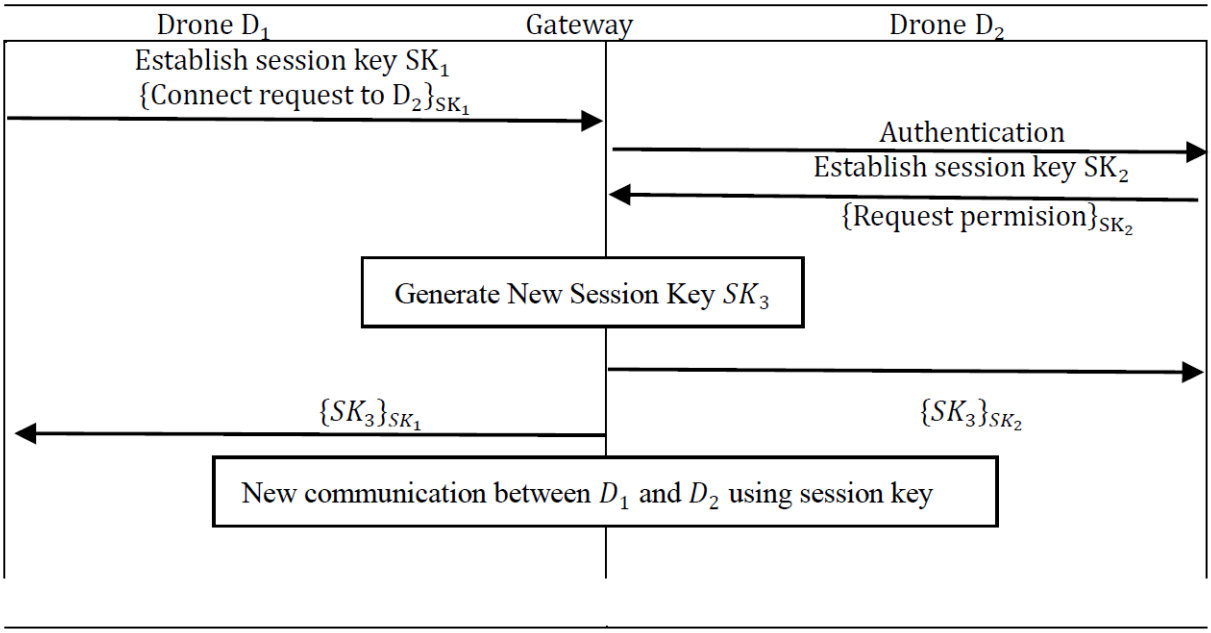

Figure 5. Drone-Drone authentication phase of PRLAP-IoD[9]

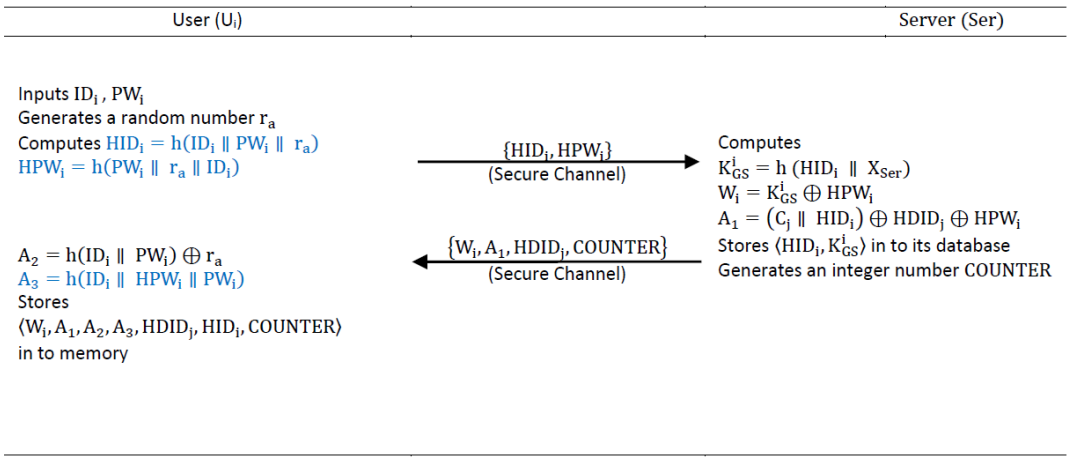

Figure 6. Improved user registration phase in the proposed protocols

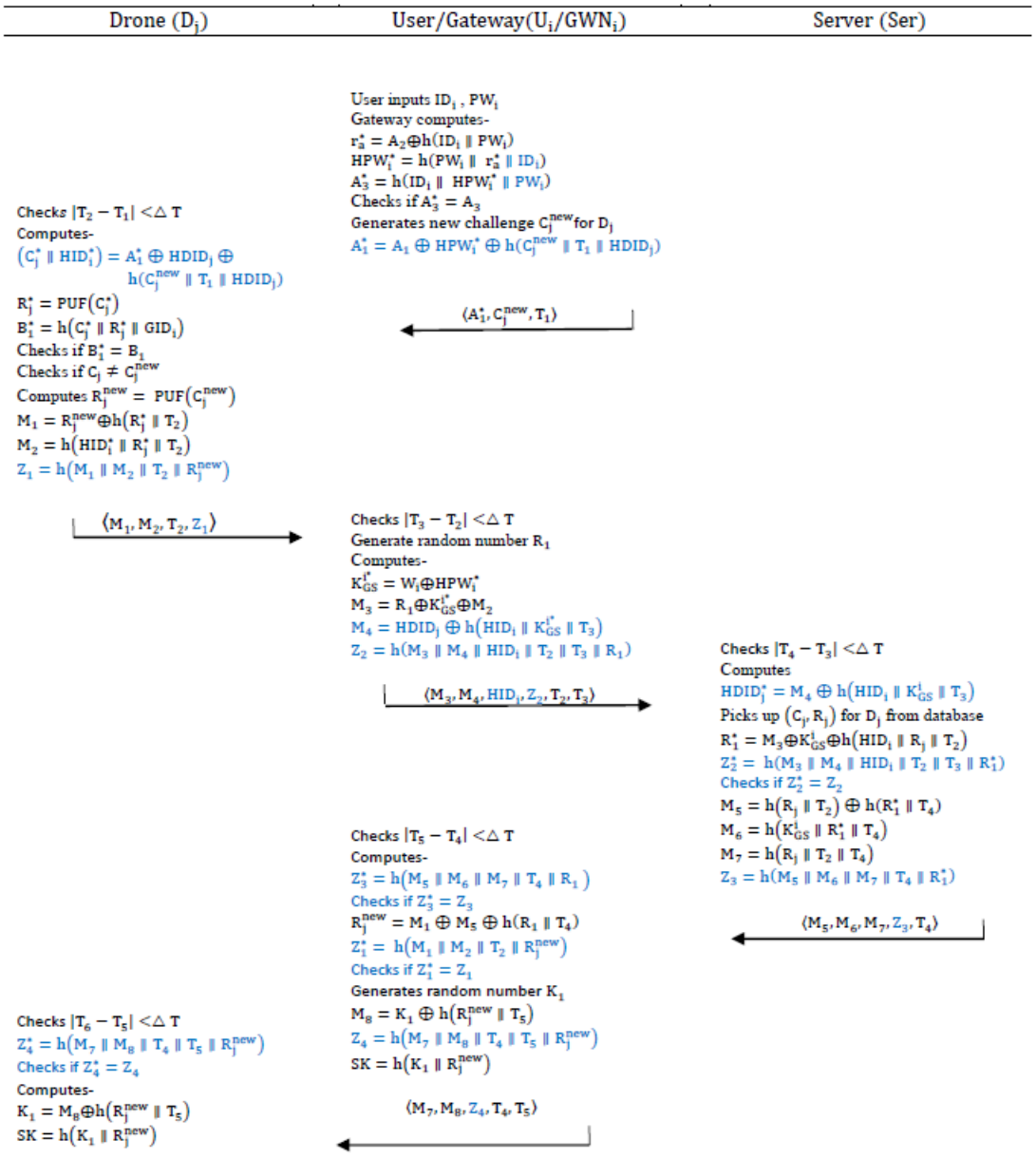

Figure 7. Improved login and authentication phase-protocol (a)

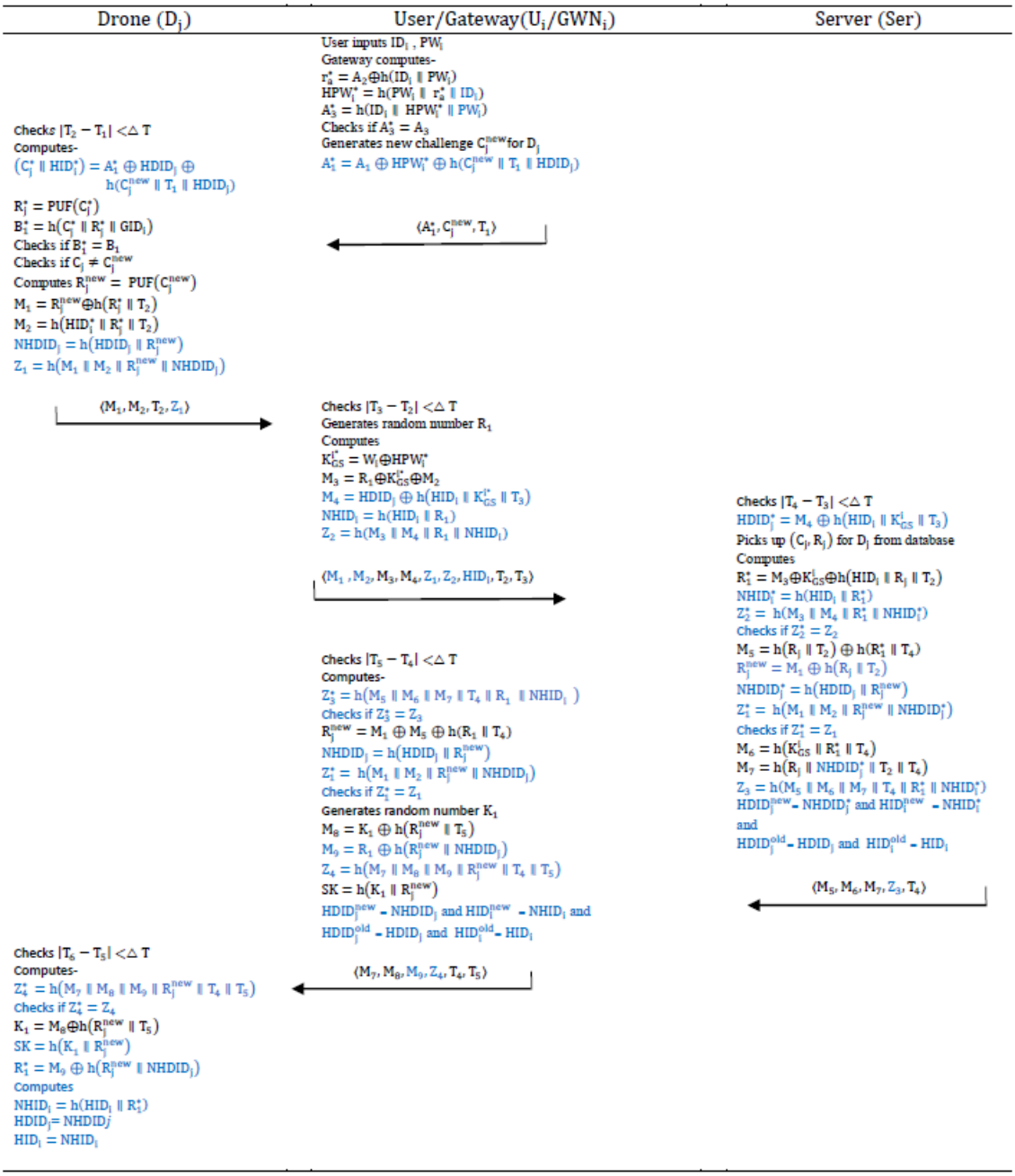

Figure 8. Improved login and authentication phase-protocol (b)

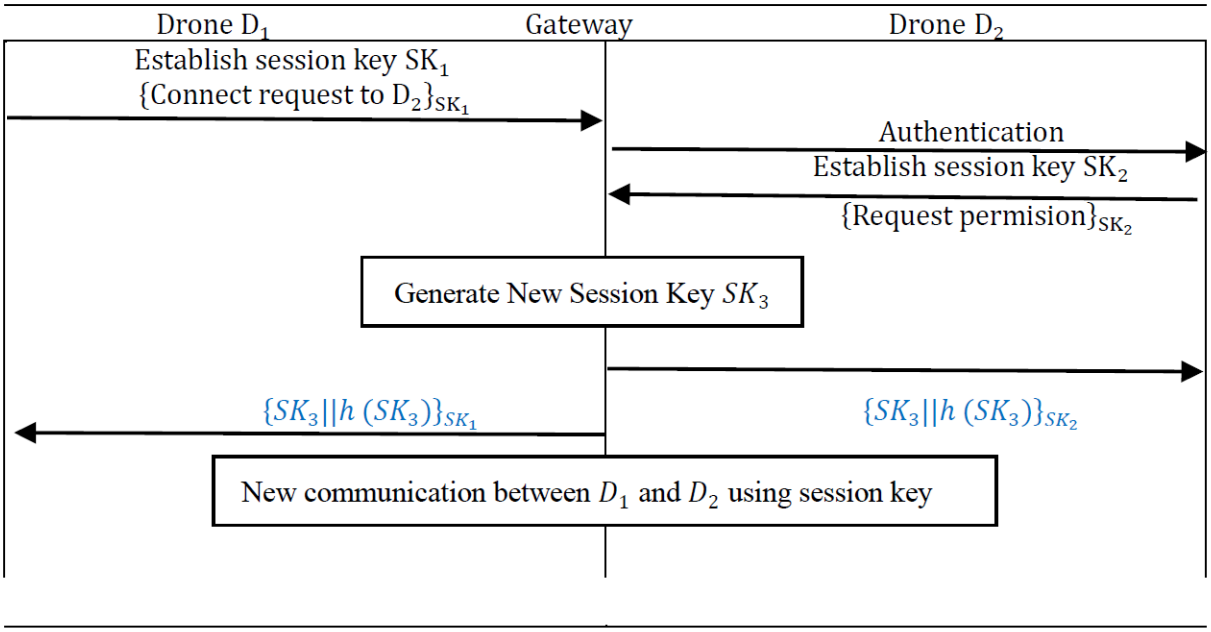

Figure 9. Improved Drone-Drone authentication phase

```
hashfunction H;
const xor: Function;
const con: Function;
const PUF: Function;
const F160f: Function;
const E160e: Function;
usertype Ticket;
usertype Timestamp;
const HDIDj;
secret HIDi;
secret IDi;
secret DIDi;
secret PWi;
secret A3;
secret A2;
secret A1;
secret Wi;
secret Cj;
secret GIDi;
const Cjnew;
secret Rj;
secret B1;
secret Xser;
protocol ImprovedProtocol-A (G,D,S){
// USER_GATEWAY LEVEL 1
macro rastar=xor(A2,H(con(IDi,PWi)));
macro HPWistar=H(con(con(PWi,rastar),IDi));
macro A3star=H(con(con(IDi,HPWistar),PWi));
macro A1star=xor(xor(A1,HPWistar),H(con(con(Cjnew,T1),HDIDj)));
// DRON LEVEL 1
macro
Cjstar=F160f(xor(xor(A1star,HDIDj),H(con(con(Cjnew,T1),HDIDj))));
macro HIDistar=
E160e(xor(xor(A1star,HDIDj),H(con(con(Cjnew,T1),HDIDj))));
macro Rjstar= PUF(Cjstar);
macro B1star= H(con(con(Cjstar,Rjstar),GIDi));
macro Rjnew=PUF(Cjnew);
macro M1= xor(Rjnew,H(con(Rjstar,T2)));
macro M2=H(con(con(HIDistar,Rjstar),T2));
macro Z1=H(con(con(con(M1,M2),T2),Rjnew));
// USER_GATEWAY LEVEL 2
macro KGSistar=xor(Wi,HPWistar);
macro Z1star= H(con(con(con(M1,M2),T2),Rjnew));
macro M3=xor(xor(R1,KGSistar),M2);
macro M4=xor(HDIDj, H(con(con(HIDi,KGSistar),T3)));
macro Z2=H(con(con(con(con(con(M3,M4),HIDi),T2),T3),R1));
// SERVER ONLY LEVEL
macro KGSi=H(con(HIDi,Xser));
macro
R1star=xor(xor(M3,KGSi),H(con(con(HIDi,Rj),T2)));
macro
Z2star=H(con(con(con(con(con(M3,M4),HIDi),T2),T3),R1
star));
macro HDIDjstar=xor(M4,H(con(con(HIDi,KGSi),T3)));
macro M5=xor(H(con(Rj,T2)),H(con(R1star,T4)));
macro M6=H(con(con(KGSi,R1star),T4));
macro M7=H(con(con(Rj,T2),T4));
macro Z3=H(con(con(con(con(M5,M6),M7),T4,R1star)));
// USER_GATEWAY LEVEL 3
macro Z3star=
H(con(con(con(con(M5,M6),M7),T4,R1)));
macro Rjnew=xor(xor(M1,M5),H(con(R1,T4)));
macro M8=xor(K1,H(con(Rjnew,T5)));
macro Z4= H(con(con(con(con(M7,M8),T4),T5),Rjnew));
macro SK=H(con(K1,Rjnew));
// DRON LEVEL 2
macro M7star=H(con(con(Rj,T2),T4));
macro Z4star=H(con(M8,Rjnew));
macro K1=xor(M8,H(con(Rjnew,T5)));
macro SK=H(con(K1,Rjnew));
role G{
fresh R1: Nonce;
fresh K1: Nonce;
fresh ra: Nonce;
fresh T1: Timestamp;
fresh T3: Timestamp;
fresh T5: Timestamp;
var T2: Timestamp;
var T4: Timestamp;
send_1 (G,D,A1star,Cjnew,T1);
var M1,M2,Z1;
recv_2 (D,G,M1,M2,Z1,T2);
match (Z1,Z1star);
send_3 (G,S,M3,M4,HIDi,Z2,T2,T3);
var M7,M6,M5;
recv_4 (S,G,M5,M6,M7,Z3,T4);
match(Z3,Z3star);
send_5 (G,D,M7,M8,Z4,T4,T5);
match (Z4,Z4star);
claim_G (G, Secret, GIDi);
claim_G (G, Secret, PWi);
claim_G (G, Secret, ra);
claim_G (G,SKR,SK);
claim_G (G,Niagree);
claim_G (G, Nisynch );
claim_G (G, Alive );
claim_G (G, Weakagree);
}
role D{
var R1: Nonce;
var K1: Nonce;
fresh T2: Timestamp;
fresh T6: Timestamp;
var T1: Timestamp;
var T4: Timestamp;
var T5: Timestamp;
var A1star,Cjnew;
recv_1
(G,D,A1star,Cjnew,T1);
send_2 (D,G,M1,M2,Z1,T2);
var M7,M8;
recv_5
(G,D,M7,M8,Z4,T4,T5);
claim_D (D, Secret, DIDi);
claim_D (D, Secret, A3);
claim_D (D,SKR,SK);
claim_D (D,Niagree);
claim_D (D, Nisynch );
claim_D (D, Alive );
claim_D (D, Weakagree);
}
role S{
var R1: Nonce;
var K1: Nonce;
var ra: Nonce;
fresh T4: Timestamp;
var T1: Timestamp;
var T2: Timestamp;
var T3: Timestamp;
var T5: Timestamp;
var HDIDjstar;
recv_3
(G,S,M3,M4,HIDi,Z2,T2,T3);
match (Z2,Z2star);
send_4 (S,G, M5, M6, M7,Z3,
T4);
claim_S (S, Secret, KGSi);
claim_S (S, Nisynch );
claim_S (S, Niagree );
claim_S (S, Alive );
claim_S (S, Weakagree);
}
}
```

Figure 10. The Scyther SPDL source code of the proposed protocol (a)

Claim				Status		Comments
ImprovedProtocol_A	G	ImprovedProtocol_A,G	Secret GIDi	Ok		No attacks within bounds.
		ImprovedProtocol_A,G1	Secret PWi	Ok		No attacks within bounds.
		ImprovedProtocol_A,G2	Secret ra	Ok		No attacks within bounds.
		ImprovedProtocol_A,G3	SKR H(con(xor(xor(K1,H(con(xor(xor(xor(PUF(Cjnew),...	Ok		No attacks within bounds.
		ImprovedProtocol_A,G4	Niagree	Ok		No attacks within bounds.
		ImprovedProtocol_A,G5	Nisynch	Ok		No attacks within bounds.
		ImprovedProtocol_A,G6	Alive	Ok		No attacks within bounds.
		ImprovedProtocol_A,G7	Weakagree	Ok		No attacks within bounds.
	D	ImprovedProtocol_A,D	Secret DIDi	Ok	Verified	No attacks.
		ImprovedProtocol_A,D1	Secret A3	Ok	Verified	No attacks.
		ImprovedProtocol_A,D2	SKR H(con(xor(xor(K1,H(con(xor(xor(xor(PUF(Cjnew),...	Ok		No attacks within bounds.
		ImprovedProtocol_A,D3	Niagree	Ok		No attacks within bounds.
		ImprovedProtocol_A,D4	Nisynch	Ok		No attacks within bounds.
		ImprovedProtocol_A,D5	Alive	Ok		No attacks within bounds.
		ImprovedProtocol_A,D6	Weakagree	Ok		No attacks within bounds.
	S	ImprovedProtocol_A,S	Secret H(con(HIDi,Xser))	Ok		No attacks within bounds.
		ImprovedProtocol_A,S1	Nisynch	Ok		No attacks within bounds.
		ImprovedProtocol_A,S2	Niagree	Ok		No attacks within bounds.
		ImprovedProtocol_A,S3	Alive	Ok		No attacks within bounds.
		ImprovedProtocol_A,S4	Weakagree	Ok		No attacks within bounds.

Done.

Figure 11. The security verification result of the proposed protocol (a) through the Scyther Compromise-0.9.2 tool

```
hashfunction H;
const xor: Function;
const con: Function;
const PUF: Function;
const F160f: Function;
const E160e: Function;
usertype Ticket;
usertype Timestamp;
const HDIDj;
secret HIDi;
secret IDi;
secret DIDi;
secret PWi;
secret A3;
secret A2;
secret A1;
secret Wi;
secret Cj;
secret GIDi;
const Cjnew;
secret Rj;
secret B1;
secret Xser;
protocol  ImprovedProtocol-B (G,D,S){
// USER_GATEWAY LEVEL 1
macro rastar=xor(A2,H(con(IDi,PWi)));
macro HPWistar=H(con(con(PWi,rastar),IDi));
macro A3star=H(con(con(IDi,HPWistar),PWi));
macro A1star=xor(xor(A1,HPWistar),H(con(con(Cjnew,T1),HDIDj)));
// DRON LEVEL 1
macro
Cjstar=F160f(xor(xor(A1star,HDIDj),H(con(con(Cjnew,T1),HDIDj))));
macro HIDistar=
E160e(xor(xor(A1star,HDIDj),H(con(con(Cjnew,T1),HDIDj))));
macro Rjstar= PUF(Cjstar);
macro B1star= H(con(con(Cjstar,Rjstar),GIDi));
macro Rjnew=PUF(Cjnew);
macro M1= xor(Rjnew,H(con(Rjstar,T2)));
macro M2=H(con(con(HIDistar,Rjstar),T2));
macro NHDIDj=H(con(HDIDj,Rjnew));
macro Z1=H(con(con(con(M1,M2),Rjnew),NHDIDj));
// USER_GATEWAY LEVEL 2
macro KGSistar=xor(Wi,HPWistar);
macro Z1star= H(con(con(M1,M2),Rjnew));
macro M3=xor(xor(R1,KGSistar),M2);
macro M4=xor(HDIDj, H(con(con(HIDi,KGSistar),T3)));
macro NHIDi=H(con(HIDi,R1));
macro Z2=H(con(con(con(M3,M4),R1),NHIDi));

// SERVER ONLY LEVEL
macro KGSi=H(con(HIDi,Xser));
macro R1star=xor(xor(M3,KGSi),H(con(con(HIDi,Rj),T2)));
macro M5=xor(H(con(Rj,T2)),H(con(R1star,T4)));
macro Rjnew=xor(xor(M1,M5),H(con(R1star,T3)));
macro NHIDistar= H(con(HIDi,R1star));
macro Z2star=H(con(con(con(M3,M4),R1star),NHIDistar));
macro HDIDjstar=xor(M4,H(con(con(HIDi,KGSi),T3)));
macro M6=H(con(con(KGSi,R1star),T4));
macro NHDIDjstar=H(con(HDIDj,Rjnew));
macro
Z1star=H(con(con(con(M1,M2),Rjnew),NHDIDjstar));
macro M7=H(con(con(con(Rj,NHDIDjstar),T2),T4));
macro
Z3=H(con(con(con(con(con(M5,M6),M7),T4),R1star)));
// USER_GATEWAY LEVEL 3
macro Z3star=
H(con(con(con(con(con(M5,M6),M7),T4),R1)));
macro Rjnew=xor(xor(M1,M5),H(con(R1,T4)));
macro NHDIDj=H(con(HDIDj,Rjnew));
macro Z1star=H(con(con(con(M1,M2),Rjnew),NHDIDj));
macro M8=xor(K1,H(con(Rjnew,T5)));
macro M9=xor(R1,H(con(Rjnew,NHDIDj)));
macro Z4=
H(con(con(con(con(con(M7,M8),M9),Rjnew),T4),T5));
macro SK=H(con(K1,Rjnew));
// DRON LEVEL 2
macro
Z4star=H(con(con(con(con(con(M7,M8),M9),Rjnew),T4)
,T5));
macro K1=xor(M8,H(con(Rjnew,T5)));
macro SK=H(con(K1,Rjnew));
macro R1star=xor(M9,H(con(Rjnew,NHDIDj)));
macro NHIDi= H(con(HIDi,R1star));
role G{
fresh R1: Nonce;
fresh K1: Nonce;
fresh ra: Nonce;
fresh T1: Timestamp;
fresh T3: Timestamp;
fresh T5: Timestamp;
var T2: Timestamp;
var T4: Timestamp;
send_1 (G,D,A1star,Cjnew,T1);
var M1,M2,Z1;
recv_2 (D,G,M1,M2,Z1,T2);
send_3 (G,S,M1,M2,M3,M4,Z1,Z2,HIDi,T2,T3);
var M7,M6,M5;
recv_4 (S,G,M5,M6,M7,Z3,T4);
match(Z3,Z3star);
match (Z1star,Z1);
send_5 (G,D,M7,M8,Z4,T4,T5);
match (Z4,Z4star);
claim_G (G, Secret, GIDi);
claim_G (G, Secret, PWi);
claim_G (G, Secret, ra);
claim_G (G,SKR,SK);
claim_G (G,Niagree);
claim_G (G, Nisynch );
claim_G (G, Alive );
claim_G (G, Weakagree);
}
role D{
var R1: Nonce;
var K1: Nonce;
fresh T2: Timestamp;
fresh T6: Timestamp;
var T1: Timestamp;
var T4: Timestamp;
var T5: Timestamp;
var T3:Timestamp;
var A1star,Cjnew;
recv_1
(G,D,A1star,Cjnew,T1);
send_2 (D,G,M1,M2,Z1,T2);
var M7,M8;
recv_5
(G,D,M7,M8,Z4,T4,T5);
claim_D (D, Secret, DIDi);
claim_D (D, Secret, A3);
claim_D (D,SKR,SK);
claim_D (D,Niagree);
claim_D (D, Nisynch );
claim_D (D, Alive );
claim_D (D, Weakagree);
}
role S{
var R1: Nonce;
var K1: Nonce;
var ra: Nonce;
fresh T4: Timestamp;
var T1: Timestamp;
var T2: Timestamp;
var T3: Timestamp;
var T5: Timestamp;
var HDIDjstar;
recv_3
(G,S,M1,M2,M3,M4,Z1,Z2,HI
Di,T2,T3);
match (Z2star,Z2);
macro
Rjnew=xor(xor(M1,M5),H(co
n(R1star,T3)));
match (Z1star,Z1);
send_4 (S,G, M5, M6, M7,Z3,
T4);
claim_S (S, Secret, KGSi);
claim_S (S, Nisynch );
claim_S (S, Niagree );
claim_S (S, Alive );
claim_S (S, Weakagree);
}
}
```

Figure 12. The Scyther SPDL source code of the proposed protocol (b)

Claim				Status		Comments
ImprovedProtocol_B	G	ImprovedProtocol_B,G	Secret GIDi	Ok		No attacks within bounds.
		ImprovedProtocol_B,G1	Secret PWi	Ok		No attacks within bounds.
		ImprovedProtocol_B,G2	Secret ra	Ok		No attacks within bounds.
		ImprovedProtocol_B,G3	SKR H(con(xor(xor(K1,H(con(xor(xor(xor(PUF(Cjnew),...	Ok		No attacks within bounds.
		ImprovedProtocol_B,G4	Niagree	Ok		No attacks within bounds.
		ImprovedProtocol_B,G5	Nisynch	Ok		No attacks within bounds.
		ImprovedProtocol_B,G6	Alive	Ok		No attacks within bounds.
		ImprovedProtocol_B,G7	Weakagree	Ok		No attacks within bounds.
	D	ImprovedProtocol_B,D	Secret DIDi	Ok	Verified	No attacks.
		ImprovedProtocol_B,D1	Secret A3	Ok	Verified	No attacks.
		ImprovedProtocol_B,D2	SKR H(con(xor(xor(K1,H(con(xor(xor(xor(PUF(Cjnew),...	Ok		No attacks within bounds.
		ImprovedProtocol_B,D3	Niagree	Ok		No attacks within bounds.
		ImprovedProtocol_B,D4	Nisynch	Ok		No attacks within bounds.
		ImprovedProtocol_B,D5	Alive	Ok		No attacks within bounds.
		ImprovedProtocol_B,D6	Weakagree	Ok		No attacks within bounds.
	S	ImprovedProtocol_B,S	Secret H(con(HIDi,Xser))	Ok		No attacks within bounds.
		ImprovedProtocol_B,S1	Nisynch	Ok		No attacks within bounds.
		ImprovedProtocol_B,S2	Niagree	Ok		No attacks within bounds.
		ImprovedProtocol_B,S3	Alive	Ok		No attacks within bounds.
		ImprovedProtocol_B,S4	Weakagree	Ok		No attacks within bounds.

Done.

Figure 13. The security verification result of the proposed protocol (b) through the Scyther Compromise-0.9.2 tool

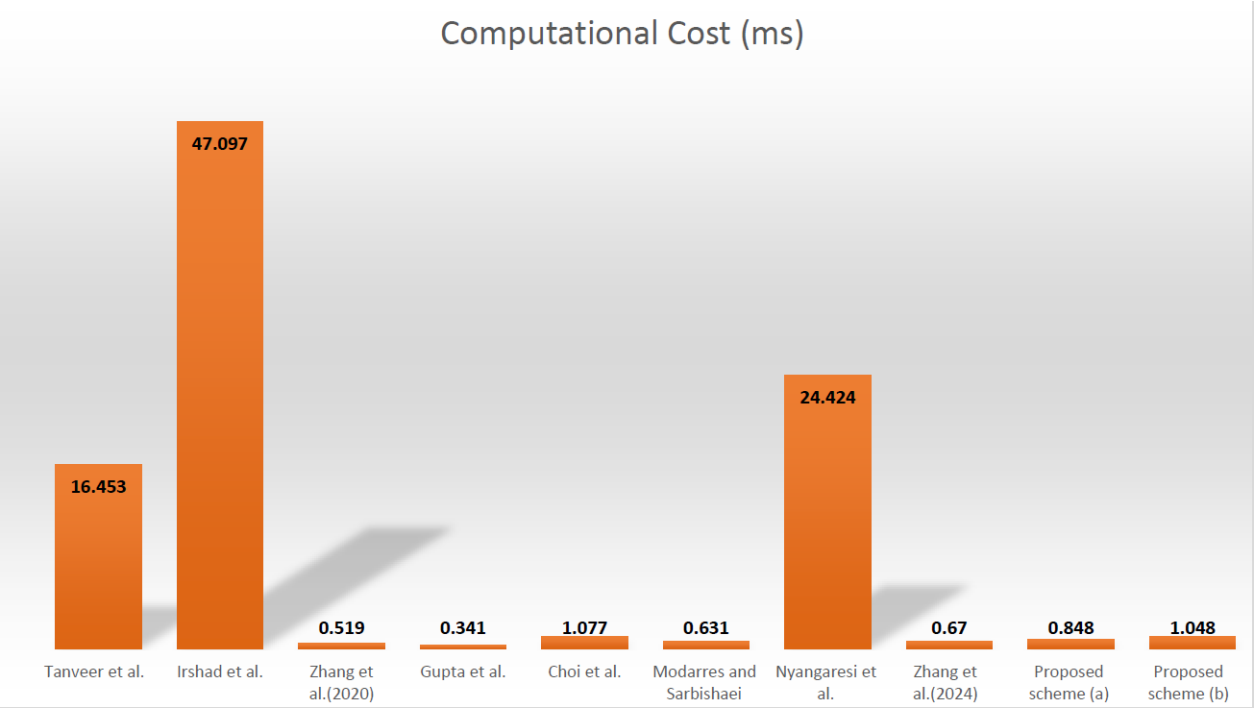

Figure 14. Graphical computational cost comparison of proposed schemes with similar recent schemes

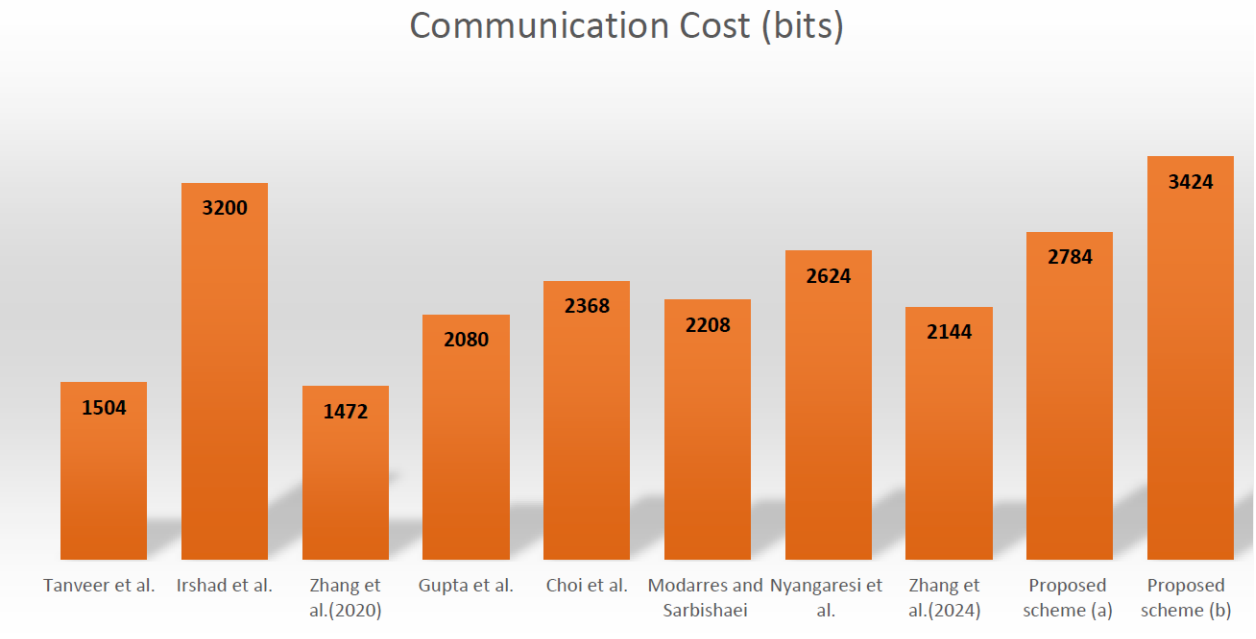

Figure 15. Graphical communication cost comparison (in bits) of proposed schemes with recent similar schemes

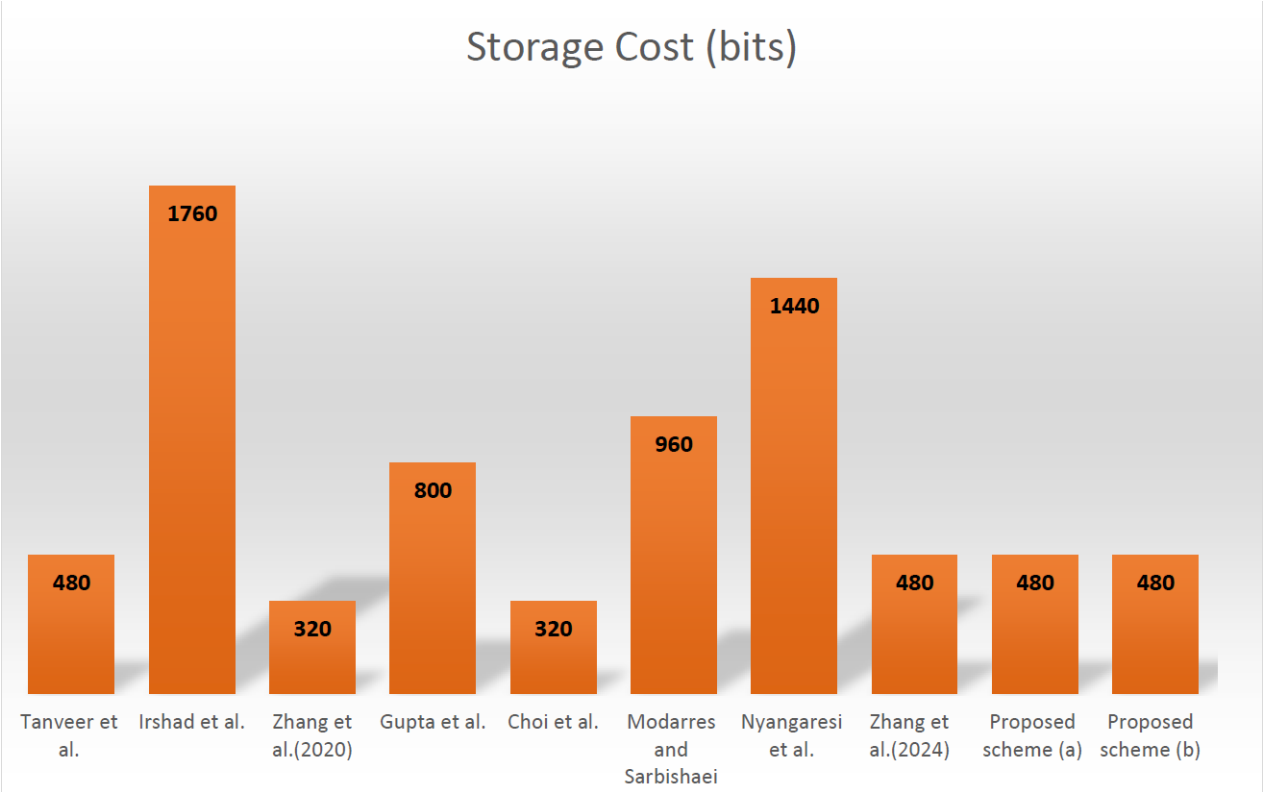

Figure 16. Graphical total storage cost comparison of proposed schemes with other similar schemes