RUAM-IoD: A Robust User Authentication Mechanism for the Internet of Drones

MUHAMMAD TANVEER1, AHMED ALKHAYYAT2, ALAMGIR NAUSHAD3, ABD ULLAH KHAN3, (Member, IEEE), NEERAJ KUMAR4,5,6, (Senior Member, IEEE), AND ABDULLAH G. ALHARBI7, (Member, IEEE)

1School of Systems and Technology (SST), University of Management and Technology, Lahore 54770, Pakistan
2Department of Computer Technical Engineering, College of Technical Engineering, Islamic University, Najaf 54001, Iraq
3Department of Computer Science, National University of Sciences and Technology (NUST) Balochistan Campus (NBC), Quetta 87300, Pakistan
4School of Computer Science, University of Petroleum and Energy Studies, Dehradun, Uttarakhand 248007, India
5Department of Computer Science and Information Engineering, Asia University, Taichung City 41354, Taiwan
6Department of Electrical and Computer Engineering, King Abdulaziz University, Jeddah 21589, Saudi Arabia
7Department of Electrical Engineering, Faculty of Engineering, Jouf University, Sakaka 42421, Saudi Arabia

Corresponding author: Abd Ullah Khan (abdullah@nbc.nust.edu.pk)

ABSTRACT The revolutionary advancement in the capabilities of hardware tools, software packages, and communication techniques gave rise to the Internet of Things-supported drone networks (IoD), thereby enabling smooth communication among devices and applications, and impacting drastically the various aspects of human lives. However, with the increasing sophistication in the infrastructure of IoD, new security threats arise that require novel algorithms and schemes as solutions. To this end, several schemes have recently been proposed. However, some schemes cannot perfectly address the novel security aspects associated with IoD environments, while others cannot provide computational or communication efficiency. Motivated by these research gaps in the existing literature, we leverage elliptic curve cryptography along with symmetric encryption and hash function, and propose a novel and robust user authentication mechanism for the IoD, called RUAM-IoD. We validate the security of the established SK formally through the random oracle model. Similarly, we provide informal security analysis to demonstrate the security capabilities of RUAM-IoD against different pernicious security attacks. Likewise, we establish a comparison of the RUAM-IoD with several state-of-the-art authentication schemes to show that RUAM-IoD acquires less storage, communication, and computational cost.

INDEX TERMS Internet of Drone, privacy, unmanned aerial vehicles, key exchange, security.

I. INTRODUCTION

Unmanned aerial vehicle (UAV) or drone is a versatile platform for communication that provides flexibility in altitude, line-of-sighting, mobility, and so on. Consequently, drones can be considered as broadband wireless access solutions for terrestrial network devices [1], [2]. The applications that benefit from UAVs include military, traffic management, tracing and tracking, disaster management, surveillance and monitoring, wireless communication, and so on [3]. This implies that drones can increasingly become capable of providing ubiquitous computing, on board processing, and wireless communication. This way, drones can serve as airborne base stations, thereby expanding the reachability of terrestrial networks. Moreover, as a flying base station, a drone is immune to damage caused by geographical disasters and calamities, consequently proving economically efficient [4]. The most important and unique feature of the smart UAVs is the provisioning of effective, dependable, and quick connection establishment in urban and rural areas, large roads, expanding regions, etc.

To impart these services, UAVs are dependent on the Internet of Things (IoT) networks, thereby leading to the Internet of Drones (IoD) networks [5]. Fig. 1 general overview of the UAV/IoD based communication system. Typically, the IoD networks contain a ground station (GS), several flying drones, and a certain number of remote users. The drones play the role of collecting information from the environment of interest and transmitting the collected information to the corresponding server residing within GS. The GS controls through wireless channels (by sending control commands) the type of information required to be collected by the drones, and...
the frequency of the information collection [6], [7]. Remote users are the beneficiary of the information collected by the drones and processed by the GS. This implies that remote users can access in real-time the information collected by drones using the Internet. However, using the public wireless channels for such information retrieval from drones poses security threats and vulnerabilities, leading to unauthorized information exposure. Given that the information collected by drones is sensitive and private most of the time, its security and privacy cannot be ignored. This implies that the secure exchange of information between users and drones in the IoD environment is a critical requirement for realizing the benefits of drones and their applications [9].

However, there are several challenges to deal with for perfectly exploiting the IoD for the applications mentioned above. These challenges include power consumption of the drones, optimization of a drone’s trajectorial motion, deployment of the remote user’s association, the communication protocol used, the throughput and latency improvement, the effective link establishment, and security and privacy [10], [11].

II. RELATED WORK
Secure with privacy-preserving communication mechanisms are imperative for the IoD networks. Security and privacy requirements in the IoD networks are reviewed in [9], [21]. Moreover, various user AKA schemes are presented in [17], [22]–[30] to enable encrypted and reliable communications in the IoD environment. However, a vast majority of these AKA schemes are prone to a variety of security attacks. Table 1 tabulates user AKA schemes with their limitations and techniques employed in these schemes. The authors in [5] presented a terrestrial credential-based AKA scheme and employed a random oracle model (ROM) to validate the session key’s (SK) security. The scheme checks the authenticity of the user before accessing the sensitive information from a specific drone in real-time and fixes a secret SK among the user and drone to communicate securely after establishing an SK. Wazid et al. [12] designed AKA scheme by employing SHA-160 and ECC to enable encrypted communication in the IoD networks. The scheme employs a drone and a user to establish an SK for indecipherable information exchange. Similarly, Tanveer et al. [34] proposed an AKA scheme by employing SHA-160 and ECC-based AKA scheme to enable indecipherable communication in the IoD networks. The scheme enables a drone and user to establish an SK for indecipherable information exchange. However, the scheme unable to resist the DSY attack. A user AKA scheme is suggested by Sajid et al. [27] for the IoD network that employs SHA-160 and ECC, which enables both the user and drone to communicate securely after establishing SK. Wazid et al. [12] designed AKA scheme employing ECC and SHA-160 to enable encrypted communication in the IoD environment. Nevertheless, the their scheme cannot provide resistance against certain attacks like UI and PI.

Aside from these works, Ali et al. [14] designed an AKA protocol to ensure the indecipherable communication between user and drone. The scheme is based on SHA-160 and XOR operation. However, it is found in [14] that the scheme is prone to PI, UI, SSD, forgery, and denial-of-service (DoS) attacks. Bera et al. [36] devised an AKA protocol for the IoD networks by employing cryptographic techniques, such as ECC and SHA-256. However, the authors couldn’t stop DSY attack in their scheme. Iqbal et al. [37] formulated an AKA scheme for smart home environment by employing SHA-160 and XOR. However, it is proved in [38] that Iqbal et al.’s scheme is exposed to SK disclosure, UI, and MITM attacks. Moreover, it is also found that the presented approach does not provide user anonymity (UA) and mutual authentication (MA). A puncturable pseudorandom function-based user AKA scheme is proposed in [39]. The scheme permits users and end-devices to establish an SK after achieving MA. Vinoth et al. [40] designed a multi-factor AKA scheme for the IoT environment using SHA-160 and AES. However, the scheme is prone to DoS, DSY, replay, and device capture attacks.
TABLE 1. A cursory review of eminent user AKA schemes.

| AKA schemes       | Environment Year | Limitations                                                                 | Techniques                              |
|-------------------|------------------|-----------------------------------------------------------------------------|-----------------------------------------|
| Wazid et al. [12] | IoT/IoD          | 2018 Prone to user impersonation (UI) and privilege insider (PI) attacks.       | SHA and Exclusive-OR (XOR)              |
| Gope and Sikdar [13] | IoT/GS          | 2019 Fragile to replay, man-in-the-middle (MITM), and CS impersonation (CSI) attacks. | XOR and secure hash algorithm (SHA)      |
| Jangtra et al. [5] | IoT/IoD          | 2019 Prone to user impersonation (UI) and privilege insider (PI) attacks.       | SHA and XOR                             |
| Ali et al. [14]   | IoT/IoD          | 2020 Does not ensure the indistinguishability of encryption under chosen plaintext attack (IND-CPA) and anonymity and untraceability features. | XOR, advanced encryption standard (AES), SHA |
| Ever [15]         | IoT/IoD          | 2020 Vulnerable to ephemeral secret leakage (ESL) and does not provide the new drone addition phase. In addition, it does not render untraceability and anonymity properties. | Elliptic curve cryptography (ECC), SHA, bi-linear pairing, AES |
| Cho et al. [16]   | IoD/IoT          | 2020 Does not resist ESL, anonymity and untraceability attacks. Does not render new drone addition phase. | ECC, SHA, ECC-based digital signature, and AES |
| Shelaaz et al. [17] | IoT/IoD          | 2021 Does not provide anonymity and prone to drone/device impersonation (DI), CSI, and ESL attacks. | ECC, XOR, and SHA |
| Wazid et al. [18] | IoT              | 2020 Unable to resist De-Synchronization (DSY) attack.                       | ECC, SHA, and XOR                       |
| Zhang et al. [19] | IoT              | 2020 Vulnerable to forgery and drone capture attacks.                       | SHA and XOR                             |
| Rena et al. [20]  | IoT              | 2021 Unable to render untraceability and anonymity features. Does not protect stolen smart device (SSD), PI, ESL, UI, and bio-metric and password change (BPC) attacks. | SHA and XOR                             |

A. MOTIVATION

Drones collect sensitive data from various environments of interest and dispatch the data to GS via a wireless channel. The channel is vulnerable and risky, and can be exploited. Moreover, oftentimes distant users need to collect the sensitive information, in real-time, directly from the deployed drone, rather than utilizing the data collected at the central server (CS) posted at GS. Therefore, it is imperative to allow only authorized users to obtain critical information directly from the drone. In addition, it is necessary to protect the communication between the drone and the remote user from being disclosed by an attacker or adversary. Above this, most of AKA schemes proposed so far for this purpose are insecure against the PI attack because, in these schemes, the secret information related to users are stored in plaintext form [31, 41]. An insider adversary can obtain the secret information associated with a specific user from CS and execute an attack on behalf of the user. Other AKA schemes proposed for providing secure communication with drones are vulnerable to BPC, SSD, UI, and MITM attacks. Thus, it is crucial to design a secure and reliable AKA scheme to ensure indecipherable communication for the IoD environment [9].

B. RESEARCH CONTRIBUTION

This paper has the following main contributions.

1) A novel and robust user authentication mechanism for the IoD, called RUAM-IoD, is presented, which is based on the AES-CBC-256 encryption, ECC, SHA-256 hash function, and XOR operation. The proposed RUAM-IoD authenticates the user prior to enabling him to access the network’s resources. In addition, RUAM-IoD establishes a private SK to accomplish the encrypted communication in the IoD network while ensuring the user’s anonymity during the execution of the AKA phase.

2) We perform informal security verification of RUAM-IoD that shows that RUAM-IoD is secure against a variety of security risks, including DSY, BPC, drone capture, and SSD attacks. Moreover, we employ ROM-based formal validation and prove the security strength of the established private SK. Furthermore, we perform Scyther-based security validation and demonstrate that RUAM-IoD is dependable against replay and MITM attacks. In addition, RUAM-IoD is able to prevent PI attack by storing, in encrypted form, the secret information associated with the users and drones in the memory of CS.

3) We compare RUAM-IoD with relevant AKA schemes and prove that RUAM-IoD is comparatively efficient in communication, storage, and computational costs. Moreover, we prove that RUAM-IoD renders enhanced security features than the related schemes.

C. PAPER ORGANIZATION

The remaining parts of the paper are arranged as follow. The system model, i.e., the network and threat model, is described in Section III. The preliminaries are discussed in Section IV. The functional phases of the RUAM-IoD scheme are explained in detail in Section V. In Section VI, informal security analysis is carried out, ROM based validation is performed, and Scyther-based formal security is discussed. Lastly, the comparative analysis is presented in Section VII, and the concluding remarks in Section VIII.
associated with \( U_e \) and \( D_s \). Moreover, \( D_x \) is deployed to monitor and collect critical information from a specific FZ and to disseminate the collected information to \( CS \) via a wireless channel.

In a specific IoD application, for instance, a smart city traffic management system, \( U_e \) needs to collect traffic congestion information directly from \( D_t \) to avoid the delay. Thus, it is necessary to protect the information exchanged between \( U_e \) and \( D_x \). Also, it is required to ensure that the attacker cannot modify the information communicated in the IoD environment. This implies that it is imperative to prevent the unauthorized \( U_e \) from accessing the IoD network resources. Therefore, an AKA scheme is necessary to ensure the encrypted communication between \( U_e \) and \( D_t \) after the authentication of \( U_e \) is carried out. RUAM-IoD ensures the encrypted communication after establishing SK between \( U_e \) and \( D_x \).

### B. THREAT MODEL

The network entities usually exchange information through a public communication channel exposed to various security risks. Thus, adversary \( A \) can compromise the information exchanged between the network entities by taking advantage of the public nature of the communication channel. In RUMP-IoD, we have contemplated the widely-accepted “Dolev-Yao (DY) threat model” [42], [43]. Under the DY threat model, \( A \) can capture and eavesdrop on all the communicated information or message of the network’s nodes. \( A \) can also alter, forge, delete, and manipulate information while communicating with the network entities. In addition to DY model, “Canetti and Krawczyk’s model (CK-adversary model)” is also applied on the proposed RUAM-IoD. According to the CK-adversary model, \( A \) can compromise short-term secret (STS), session states, and SKs by hijacking the sessions. Therefore, the composition of SK established between network entities should be based on both STS and long-term secrets (LTS) to resist the ESL attacks. Furthermore, \( A \) can physically compromise or capture some \( D_x \) and smart devices \( SD_s \). Consequently, \( A \) can extract the secret or confidential parameters, which are pre-loaded in the memory of these devices, by utilizing power analysis (PA) attacks. However, DRC is deemed as an entirely entrusted network entity in the IoD environment.

### IV. PRELIMINARIES

This section presents the preliminaries used in RUMP-IoD.

#### A. AES-CBC-256

AES-CBC-256 is stateless cipher block chaining mode of the AES algorithm, which satisfies the IND-CPA property. Logically, the encryption process of AES-CBC-256 defined as follows

\[
CT_x = E_{key}(IV, PT),
\]

where \( PT, IV, CT_x \), and denotes the plaintext, the initialization vector, and the ciphertext, respectively. Moreover, \( key \) denotes the encryption key of size 256. Furthermore, the decryption process using AES-CBC-256 is defined by

\[
PT = D_{key}(IV, CT_x),
\]

where \( PT \) shows the plaintext retrieved from the decryption mechanism. In the proposed RUAM-IoD, AES-CBC-256 is employed as encryption/decryption algorithm, which satisfies IND-CPA property. Formally IND-CPA can be defined as follows [44], [45].

**Definition 1:** Let single/multiple eavesdropper are denoted by SE/ME, respectively. Let \( OR_{key1}, OR_{key2}, \ldots, OR_{keyN} \) denote \( N \) distinct independent encryption oracles corresponding to encryption keys \( key1, key2, \ldots, keyN \), respectively. We denote the advantage function of SE/ME as

\[
\begin{align*}
A_{SE, \Omega}^{IND-CPA}(l) &= 2 \cdot Pb(SE \leftarrow OR_{key1}(\cdot)): (0, B1) \leftarrow SE; \\
\theta &\leftarrow [0, 1]; \gamma \leftarrow OR_{key1}(\cdot \theta); \\
SE(\gamma) &= \theta - 1,
\end{align*}
\]

\[
\begin{align*}
A_{ME, \Omega}^{IND-CPA}(l) &= 2 \cdot Pb(ME \leftarrow OR_{key1}, \ldots, OR_{keyN}); \\
(0, B1) &\leftarrow SE; \theta \leftarrow [0, 1]; \gamma_1 \leftarrow OR_{key1}(\cdot \theta); \\
\gamma &\leftarrow OR_{key1}(\cdot \theta); \gamma_N \leftarrow OR_{keyN} \\
(b_0) : ME(\gamma_1, \ldots, \gamma_N) &= \theta - 1.
\end{align*}
\]

Here, \( \Omega \) denotes an encryption algorithm (AES-CBC-256), which is IND-CPA secure in ME/SE eavesdropper setting and \( A_{ME, \Omega}^{IND-CPA}(l) \) is trivial for \( A \) in polynomial time (TPoly).

#### B. FUZZY EXTRACTOR

The fuzzy extractor (FE) mechanism is a broadly adopted tool to validate bio-metric authentication. FE is specified as a tuple \( \{B_m, Lth, ERT\} \) and comprises the subsequent two algorithms.

1) \( Gen(\cdot) \) : It takes user’s bio-metric information \( B_m \) as the input parameter and generates bio-metric key \( BK \in [0, 1]^{Lth} \), where \( Lth \) denotes the length of
Table 2. List of notations used in RUAM-IoD.

| Notation     | Description                              |
|--------------|------------------------------------------|
| IDu, PAWu    | Identity and password of user            |
| PIDu, SPu   | Pseudo-identity and secret parameter of user |
| PIDD, SPD  | Pseudo-identity and secret parameter of drone |
| Athen, Athm | If both Athen and Athm are equal         |
| EC(a, b), P | Elliptic curve with base point           |
| SKu, Pu     | Private and public key of CS             |
| SKd, Pd     | Private and public key of user           |
| Biin        | Bio-metric information of legitimate user|
| Gen(·), Rrp | PE biometric key generation and functions|
| ERT         | Error tolerance parameter of PE algorithm|
| BKU, rrp    | PE generated biometric key and reproduction parameters |
| BK           | Bio-metric key generated by PE and used in the login phase |
| k, kl, ku   | Secret keys used in the encryption process during the AKA process |
| Kkey(St)    | Represents the encryption process to encrypt “St” |
| Dkey(St)    | Represents the decryption process to decrypt “St” |
| Phx         | Represents the probability of event “x” |
| TPxy        | Polynomial-time                           |
| H, ⊕, ⊕     | Concatenation, adversary, hash function, XOR, and hash function, respectively |

BK and reproduction parameter rrp, i.e., Gen(Bin) = (BK, rrp). In addition, Gen(·) is a deterministic algorithm.

2) Rep(·) : It takes user’s noisy Bin and rrp as the input parameter and generate BK, as Rep(Bin, rrp) = BK with the necessary condition HD(Bin, Bin) ≤ ERT, where ERT is the error tolerance and HD is the hamming distance between Bin and Bin.

V. RUAM-IoD

This section presents our proposed AKA scheme, RUAM-IoD. The proposed RUAM-IoD comprises six phases: (i) System Initialization Phase, (ii) DRG Phase, (iii) User Registration (URG) Phase, (iv) AKA Phase, (v) BPC Phase, and (vi) Revocation Phase. These phases are elaborated in detail in the following sub-sections. Table 2 shows a list of notations employed in RUAM-IoD.

A. INITIALIZATION PHASE OF THE SYSTEM

DRC determines an elliptic curve, i.e., EC(α, β), of the form Y^2 = X^3 + αX + β (mod q) over GF(q), q being the big prime number, with the condition 4α^3 + 27β^2 ≠ 0 (mod q), with O as the the point of infinity. DRC then selects picks a EC(α, β) base or generation point P, such that P ∈ EC(α, β), of order say N, where N.P = P + P + P + ⋯ + P (N times).

DRC picks private key SKg ∈ Zq for CS and generate a public as PUg = SKg.P. Moreover, DRC stores the parameters {SKg, PUg, EC(α, β), P} in the tempered proof database of CS. Eventually, CS makes the parameters {PUg, EC(α, β), P} public in the IoD environment.

B. DRG PHASE

In DRG phase, DRC is responsible for registering D1 prior to its deployment in a particular FZ. In addition, DRC preloads some distinct secret parameters in the memory of D1. These secret parameters are used during the AKA process. DRC executes the following measures to position a D1 in a particular FZ.

1) STEP DRG-1

DRC determines a distinct random-number RD1 along with a pseudo-identity PIDD1 for a specific D1. In addition, DRC computes the secret parameter SPD1 for D1 as follows.

\[ G = H(RD_1 \parallel PID_{D1}). \]

2) STEP DRG-2

Eventually, DRC reserves the credential {PIDD1, P, SPD1} in D1’s memory. In addition, D1 has the access to all the public parameters of CS, such {Pu, EC(α, β), P}.

C. URG PHASE

In URG phase, Ue register itself with DRC before accessing the services from a specific D1, which is deployed the DSP. The DRC issues a smart device SDe with pre-loaded secret parameters. The assigned secret parameters are validated by CS during the AKA process to allow Ue to procure the sensitive information from a particular D1 in real-time. DRC register a user by performing the following necessary steps.

1) STEP URG-1

Ue selects its password PAWu and unique identity IDu. In addition, Ue marks its bio-metric Bin as follows.

\[ SD_{e} = H(ID_{u} \parallel PAW_{u}). \]

where SDe determines RM = H(IDu || PAWu) and contrives a registration request message M_{rrm} : [RM] and dispatches M_{rrm} to DRC via a secure communication channel.

2) STEP URG-2

After getting M_{rrm} from Ue, DRC selects a random-number IVreg, bearing 128 bits size, and determines the secret parameter SPDu and PIDu for Ue as follows.

\[ A = H(S_{kg} \parallel IV_{reg} \parallel M_{rrm}). \]

SPDu = (A1 ⊕ A2),

PIDu = (A1 ⊕ SPu).

where A1 and A2 are derived by splitting A into two strings or parts of the same size (128 bits each). Moreover, DRC determines

\[ Y = (PID_u \oplus SP_u \oplus PID_{D1}). \]

\[ K_g = (IV_{reg} \parallel Y). \]

\[ CT_g = E_{K_g}(IV_{reg}, PT_g). \]

where K_{g} is the secret key, which is used to encrypt plaintext

\[ PT_g = \{SP_u, SPD_u, PID_{D1}\} \] by employing AES-CBC-256...
and stores the parameters \( \{ PID_U, CT_g, IV_{reg} \} \) in the memory of CS. In the proposed RUAM-IoD, the sensitive information associated with \( U_e \) and \( D_x \) are stored in encrypted form. Finally, DRC constructs a registration responses message \( M_{rep} : \{ PID_U, SP_U, PID_D \} \) and dispatches \( M_{rep} \) to \( U_e \) through a secure channel.

3) STEP URG-3
After procuring \( M_{rep} \) from DRC, \( SD_e \) selects an initialization vector \( IV_a \) and computes
\[
AR = (BK_U \oplus ID_U), \quad (13)
\]
\[
KR = (AR \parallel PAW_U), \quad (14)
\]
\[
Athn = H(PAW_U \parallel ID_U \parallel BK_U \parallel Y), \quad (15)
\]
\[
CT_{st} = E_{KR}(IV_a), \quad (16)
\]
where \( AR \) is obtained by XORing bio-metric key and identity of \( U_e \), \( KR \) is the secret key used in the encryption process to encrypt the plaintext or the sensitive information associated with \( U_e \). \( CT_{st} \) is obtained by encrypting \( PT_{st} = \{ PID_U, SP_U, PID_D, Y \} \) by using AES-CBC-256. Athn is the authentication parameter, which is obtain by performing the hash operation on \( PAW_U, ID_U, BK_U \), and \( Y \). Finally, \( SD_e \) stores the credentials \( \{ CT_{st}, Athn, rpp, Gen(\cdot), ERT, Rep(\cdot), IV_a \} \) in its inherent memory.

D. AKA PHASE
In this phase, \( U_e \) achieves the local authentication by providing its secret credentials as the input to \( SD_e \). After performing local authentication, \( SD_e \) sends the AKA request to CS for the further validation of \( U_e \). To ensure encrypted communication in the future, both \( U_e \) and \( D_x \) to set up an SK with the help of CS. Following steps are needed to execute this phase of AKA.

1) STEP AKA-1
\( U_e \) achieves the local authentication by making use of its own secret parameters, including password \( PAW_U \) and identity \( ID_U \), and bio-metric \( B^l_{in} \). After receiving these secret parameters, \( SD_e \) computes the bio-metric key \( (BK^l_U) = \text{Rep}(B^l_{in}, rpp) \). To verify the authenticity of \( U_e \)’s secret parameters, \( SD_e \) computes
\[
AR^l = (BK^l_U \oplus ID_U), \quad (17)
\]
\[
KR^l = (AR^l \parallel PAW^l_U), \quad (18)
\]
\[
PT_{st} = D_{KR^l}(CT_{st}), \quad (19)
\]
\[
PT_{st} = \{ PID_U, SP_U, PID_D, Y \}, \quad (20)
\]
\[
Athn = H(PAW_U \parallel ID_U \parallel BK^l \parallel Y), \quad (21)
\]
and checks the condition \( Athn^l = Athn \). If it holds, \( SD_e \) passes the local authentication of \( U_e \) and proceeds the AKA process. Otherwise, \( SD_e \) discontinues the AKA process.

2) STEP AKA-2
After achieving the local authentication, \( SD_e \) chooses a timestamp \( T_A \), random-number \( R_{U_e} \), distinct secret key \( S_{ku} \), bearing 32, 128, 160 bits sizes, respectively, and calculates
\[
Pu_u = P \cdot S_{ku}, \quad (22)
\]
\[
k = S_{ku} \cdot Pu_u, \quad (23)
\]
\[
U = H(k \parallel T_A), \quad (24)
\]
where \( Pu_u \) denotes the public key of \( U_e \), \( k \) represents the shared secret, which is obtained after performing ECC point multiplication of \( S_{ku} \) and \( Pu_u \), and \( U \) is obtained after performing the hash operation on \( k \) and \( T_A \). In addition, \( SD_e \) computes
\[
Q_1 = (Y \parallel PID_U) \oplus U, \quad (25)
\]
\[
K_a = (Y \parallel SP_U), \quad (26)
\]
\[
IV_a = U_a \oplus U_b, \quad (27)
\]
\[
Q_2 = E_{K_a}(IV_a, R_{U_e}, PID_D), \quad (28)
\]
\[
Athn^1 = H(PID_U \parallel PID_D \parallel Y \parallel R_{U_e} \parallel T_A). \quad (29)
\]
where \( Q_1 \) is obtained after XORing \( U \) and concatenation of \( Y \) and \( PID_U \), \( K_a \) is the secret key of size 256 bits, used to encrypt \( R_{U_e} \) and \( PID_D \), \( IV_a \) denotes the initialization vector, \( Q_2 \) is obtained after preforming encryption using AES-CBC-256, and \( Athn^1 \) represents the authentication parameter, which will be verified at the destination. Furthermore, \( SD_e \) constructs the message \( MS_a : \{ T_A, Q_1, Q_2, Pu_u, Athn^1 \} \) and sends \( MS_a \) to CS for further verification via open channel.

3) STEP AKA-3
Upon procuring \( MS_a \), CS determines the freshness of \( MS_a \) after verifying the condition \( T_{DL} \geq T_r - T_A \). If \( MS_a \) is fresh or within the specified time delay limit, CS computes
\[
k_1 = S_{ku} \cdot Pu_u, \quad (30)
\]
\[
U_2 = H(k_1 \parallel T_A), \quad (31)
\]
\[
(Y \parallel PID_U) = Q_1 \oplus U_2. \quad (32)
\]
where \( k_1 \) denotes the shared secret between CS and \( U_e \) and \( U_2 \) is obtained after performing hash of \( k_1 \) and \( T_A \). Moreover, after retrieving \( Y \) and \( PID_U \) from \( Q_1 \), CS checks if \( PID_D \) exist in its database. If it is found, CS retrieves \( \{ CT_{st} \} \) related to \( PID_U \) from its own database. Furthermore, CS computes
\[
K_g = (IV_{reg}, Y), \quad (33)
\]
\[
PT_g = D_{K_g}(IV_{reg}, CT_{st}), \quad (34)
\]
where \( K_g \) is the secret key, which is used to decrypt the encrypted \( (CT_{st}) \) information stored and after successful decryption process CS retrieves the plaintext \( PT_g = \{ SP_U, PID_D, SP_D \} \). Additionally, CS computes
\[
K_d = (Y \parallel SP_U), \quad (35)
\]
\[
IV_{a2} = U_a^b \oplus U_b^b, \quad (36)
\]
\[
(R_{U_e}, PID_D) = D_{K_d}(IV_{a2}, Q_2), \quad (37)
\]
\[
Athn^{a2} = H(PID_U \parallel PID_D \parallel Y \parallel R_{U_e} \parallel T_A). \quad (38)
\]
where \( K_d \) is the secret key used to decrypt \( Q_2 \) and \( IV_{a2} \) denotes the initialization vector. Finally, to validate the
authenticity of MSb. CS validates the condition $\text{Athn}^{3,2} \overset{?}{=} \text{Athn}^{2}$. If it holds, CS accepts the received MSb. Otherwise, CS terminates the AKA procedure.

4) STEP AKA-4
After getting the validity of $U_e$ verified, CS chooses a timestamps $T_b$, random-number $R_G$ of size 32 and 128 bits, respectively. Moreover, CS computes

$$K_G = H(T_b \parallel SP_{D_i} \parallel PID_{D_i}).$$

$$R_G' = R_U \oplus PID_{U_e}.$$  \hspace{1cm} (39)

$$IV_G = K_G^{a} \oplus K_G^{b},$$  \hspace{1cm} (40)

$$Q_3 = E_K^{a}((IV_G, R_U, R_G').$$  \hspace{1cm} (41)

$$\text{Athn}^{3,3} = H(T_b \parallel R_U \parallel R_G' \parallel SP_{D_i} \parallel PID_{D_i}).$$  \hspace{1cm} (42)

where $K_G$ is the secret key used in the encryption process, $IV_G$ is the initialization vector, $Q_3$ is obtained after performing the encryption using AES-CBC-256, and $\text{Athn}^{3,3}$ is the authentication parameter. Finally, CS constructs the message $MS_b : \{T_b, Q_3, Pu, \text{Athn}^{3,3}\}$ and transmits $MS_b$ to $D_x$ via open channel.

5) STEP AKA-5
$D_x$ after receiving $MS_b$ verifies if the condition $T_{DL} \geq |T_e - T_b|$. If the condition holds, $D_x$ considers $MS_b$ as a licit message. Moreover, $D_x$ computes

$$K_D = H(T_b \parallel SP_{D_i} \parallel PID_{D_i}).$$  \hspace{1cm} (43)

$$IV_D = K_D^{a} \oplus K_D^{b},$$  \hspace{1cm} (44)

$$(R_U, R_G') = D_K^{a}((IV_D, Q_3).$$  \hspace{1cm} (45)

$$\text{Athn}^{4} = H(T_b \parallel R_U \parallel R_G' \parallel SP_{D_i} \parallel PID_{D_i}).$$  \hspace{1cm} (46)

where $K_D$ represents the secret key used in the decryption process, $IV_D$ is the initialization vector, and $\text{Athn}^{4}$ is the authentication parameter. In addition, $D_x$ validates the condition $\text{Athn}^{3,2} \overset{?}{=} \text{Athn}^{4}$ to verify the authenticity of $MS_b$. If it holds, $D_x$ consider the received message $MS_b$ as a licit message and continue the AKA process. Moreover, $D_x$ chooses a timestamps $T_C$ and random-number $R_D$ of size 32 ans 128 bits, respectively and calculates

$$Pu_d = P \cdot S_{id},$$  \hspace{1cm} (47)

$$k_d = S_{id} \cdot Pu,$$  \hspace{1cm} (48)

$$U_D = H(k_d \parallel T_C \parallel R_U).$$  \hspace{1cm} (49)

$$IV_{D1} = (U_D^{a} \oplus U_D^{b}),$$  \hspace{1cm} (50)

$$Q_4 = E_{U_D}((IV_{D1}), R_G' \oplus R_D).$$  \hspace{1cm} (51)

where $Pu_d$ represents the public key of $D_x$, $k_d$ denotes the shared secret between $D_x$ and $U_e$, $U_D$ signifies the secret key used in the encryption process, $IV_{D1}$ is the initialization vector, and $Q_4$ is obtained by employing AES-CBC-256. Moreover, $D_x$ calculates

$$\text{Athn}^{5} = H(H(R_{U_e} \parallel R_G' \parallel T_C) \parallel SK_{D_i} \parallel PID_{D_i}).$$  \hspace{1cm} (52)

Finally, $D_x$ fabricates a message $MS_c : \{T_C, Q_4, Pu_d, \text{Athn}^{5}\}$ and dispatches it to $U_e$ via open channel. Furthermore, for indecipherable the communication with $U_e$, $D_x$ computes $SK_{D_i}$ as

$$k_u = S_{ku} \cdot Pu_d,$$  \hspace{1cm} (53)

$$U_u = H(k_d \parallel T_C \parallel R_U),$$  \hspace{1cm} (54)

$$IV_{u1} = (U_u^{a} \parallel U_u^{b}),$$  \hspace{1cm} (55)

$$R_G' \parallel R_D = D_{U_u}((IV_{u1}), Q_4),$$  \hspace{1cm} (56)

where $k_u$ is shared secret between $U_e$ and $D_x$, $U_u$ is the secret key to perform the encryption, and $IV_{u1}$ is the initialization vector. In addition, $SD_e$ computes SK to achieve the indecipherable communication with $D_x$ and authentication parameter as follows

$$SK_{U_e} = H(H(R_{U_e} \parallel R_G' \parallel R_D) \parallel H(k_d \parallel T_C \parallel R_G' \parallel R_D) \times \parallel PID_{D_i})).$$  \hspace{1cm} (57)

$$\text{Athn}^{6} = H(H(R_{U_e} \parallel R_G' \parallel R_D) \parallel T_C \parallel SK_{D_i} \parallel PID_{D_i}),$$  \hspace{1cm} (58)

where $\text{Athn}^{6}$ is the authentication parameter. Finally, to validate the authenticity of $MS_c$, $U_e$ checks $\text{Athn}^{5} \overset{?}{=} \text{Athn}^{6}$, if holds, authentication is successfull. Furthermore, $SD_e$ replaces $IV_{u1}^{a}$ and $CT_{st}^{n}$ with $IV_{u1}$ and $CT_{st}$ in its own memory. The AKA process of RUAM-IoD is depicted in Fig. 3.

E. BPC PHASE
In the proposed RUAM-IoD, $U_e$ is allowed to change/update its bio-metric and password. To change/update the bio-metric and password information, $U_e$ needs to perform the necessary steps.

1) STEP BPC-1
After receiving old secret parameters, such as $PAW_{U_e}^{o}$ and $B_m^{o}$ (both new and old bio-metric information are same), $SD_e$ performs the following computation to update the secret parameters, such as $PAW_{U_e}^{n}$ and $B_m^{n}$

$$(B_m^{n})_U = \text{Rep}(B_m^{o}, rpp),$$  \hspace{1cm} (59)

$$AR^{n} = (B_m^{n} \oplus ID_{U_e}),$$  \hspace{1cm} (60)

$$KR^{n} = (AR^{n} \parallel PAW_{U_e}^{n}),$$  \hspace{1cm} (61)

$$PT_{st} = D_{KR^{n}}((IV_{u}), CT_{st}),$$  \hspace{1cm} (62)
imprints biometric $B_{n}^{U_e}$, computes $\text{Rep}(B_{n}^{U_e}, \text{rpp})$, $\text{AR}^{n} = (\text{BK}_{U_e}^{n}, \text{ID}_{U_e})$, $\text{KR}^{n} = (\text{AR}^{n} \parallel \text{PAW}_{U_e}^{n})$, $\text{Athn}^{n} = H(\text{PAW}_{U_e}^{n} \parallel \text{ID}_{U_e} \parallel \text{BK}_{U_e}^{n} \parallel Y)$), and checks $\text{Athn}^{n} = \text{Athn}$. If holds, $\text{SD}_{n}^{e}$ replaces new $(\text{CT}_{U_e}^{n}, \text{Athn}^{n}, \text{rpp}, \text{Gen}(), \text{ERT}, \text{Rep}(), IV_{a}^{n})$ in its own memory.

FIGURE 3. RUAM-IoD user AKA phase.

where $PT_{U_e} = \{\text{PID}_{U_e}, \text{SP}_{U_e}, \text{PID}_{D_{U_e}}, Y\}$. In addition, $SD_e$ computes $\text{Athn}^{n} = H(\text{PAW}_{U_e}^{n} \parallel \text{ID}_{U_e} \parallel \text{BK}_{U_e}^{n} \parallel Y)$ and checks the condition $\text{Athn}^{n} = \text{Athn}$. If it is valid, $SD_e$ notifies to $U_e$ to select its new/fresh secret parameters, such as new password $\text{PAW}_{U_e}^{n}$ and fresh $\text{Bio}_{U_e}^{n}$.

2) STEP BPC-2

After receiving $\text{PAW}_{U_e}^{n}$ and $\text{Bio}_{U_e}^{n}$ from $U_e$, $SD_e$ performs the following computations

$$(BK_{U_e}^{n}, \text{rpp}^{n}) = \text{Gen}(B_{n}^{U_e}), \quad (65)$$

$$(\text{AR}^{n} = (BK_{U_e}^{n}, \text{ID}_{U_e}), \quad (66)$$

$$(\text{KR}^{n} = (\text{AR}^{n} \parallel \text{PAW}_{U_e}^{n}), \quad (67)$$

$${\text{Athn}}^{n} = H(\text{PAW}_{U_e}^{n} \parallel \text{ID}_{U_e} \parallel \text{BK}_{U_e}^{n} \parallel Y), \quad (68)$$

$${\text{CT}}_{U_e}^{n} = E_{\text{KR}^{n}}((IV_{a}^{n}, PT_{U_e}^{n}). \quad (69)$$

Finally, $SD_e$ replaces the old stored credentials $(\text{CT}_{U_e}^{n}, \text{Athn}^{n}, \text{rpp}^{n}, \text{Gen}(), \text{ERT}, \text{Rep}(), IV_{a}^{n})$ with new credentials $(\text{CT}_{U_e}^{n}, \text{Athn}^{n}, \text{rpp}^{n}, \text{Gen}(), \text{ERT}^{n}, \text{Rep}(), IV_{a}^{n})$ in its own memory. The BPC phase is summarized in Fig. 4.

F. REVOCATION PHASE

It is assumed that a valid $U_e$ of the IoD environment lost its $SD_e$. However, $U_e$ can obtain new $SD_{e}$ with new/fresh credentials from $DRC$ and executes the following steps to perform the revocation (RvP) phase.

FIGURE 4. BPC phase.

1) STEP RvP-1

After getting new $SD_{e}$ from $DRC$, $U_e$ inputs its secret parameters, such as $\text{PAW}_{U_e}$ and $\text{ID}_{U_e}$ and computes $R_{\text{rpp}} = H(\text{ID}_{U_e} \parallel \text{PAW}_{U_e})$. $SD_{e}$ constructs a revocation message $M_{rpp} : \{R_{\text{rpp}}\}$ to $CS$ via a secure communication channel. After receiving $M_{rpp}$, $CS$ computes the following computation

$$AA = H(S_{kg} \parallel IV_{rpp} \parallel M_{rpp}), \quad (70)$$

$$SP_{U_e} = (AA_{1} \oplus AA_{2}), \quad (71)$$

$$\text{PID}_{U_e} = (AA_{1} \oplus \text{SP}_{U_e}), \quad (72)$$
CS checks the existence of $PID_{U_c}$ in its own database. If $PID_{U_c}$ is detected, CS removes the information associated with $PID_{U_c}$ and dispatches a message to $U_c$ for new registration.

2) STEP RvP-2
After getting the new registration request from $U_c$, DRC conducts the same procedure as accomplished in Step URG-2 under Section V-C. Subsequently, the new secret parameters are dispatched to $U_c$. $M_{ro3} : \{PID_{D_{x}}^{new}, SP_{D_{x}}^{new}, PID_{D_{x}}\}$. Upon receiving $M_{ro3}$, SD executes Step URG-3 of Section V-C and updates \{$CT_{x}^{new}$, $Athn_{x}^{new}$, $rpp_{x}^{new}$, $Gen(.)$, $ERT_{x}^{new}$, $Rep(.)$, $IV_{x}^{new}$\} in its own memory. In addition, CS stores the credentials \{$PID_{U_c}^{new}$, $CT_{g}^{new}$, $IV_{reg}^{new}$\} in its own memory.

G. DYNAMIC DRONE DEPLOYMENT PHASE
The proposed RUAM-IoD renders the functionality of dynamic drone deployment (DDD) phase. DRC following step to deploy a new $D_{x}^{new}$ drone in the target FZ.

1) STEP DDD-1
DRC chooses a random-number $R_{D_{x}}^{new}$ and a unique pseudo-identity $PID_{D_{x}}^{new}$ for $D_{x}^{new}$. Moreover, DRC calculates the secret parameter $SP_{D_{x}}^{new}$ for $D_{x}^{new}$ as follows
\[
G = H(R_{D_{x}}^{new} \parallel PID_{D_{x}}^{new}),
\]
\[
SP_{D_{x}}^{new} = (G_1 \oplus G_2),
\]
where $G_1$ and $G_2$ are obtained after dividing $G$ into two equal parts.

2) STEP DDD-2
Finally, DRC pre-loads the credentials \{$PID_{D_{x}}^{new}$, $P$, $SP_{D_{x}}^{new}$\} in the memory of $D_{x}^{new}$. In addition, $D_{x}^{new}$ has the access to all the public credentials of CS, such \{$Pu_g$, $EC((\alpha, \beta), P)$\}.

VI. SECURITY ANALYSIS
This section presents the informal analysis of RUAM-IoD to demonstrates its immunity/resistance against different pernicious security vulnerabilities, such as CSI, UI, SSD, DoS, and BPC attacks. Furthermore, ROM-based analysis is conducted to prove SK’s security, established between $U_c$ and $D_{x}$. Moreover, Scyther is employed to illustrate that RUAM-IoD can resist or protect replay and MITM attacks.

A. INFORMAL SECURITY ANALYSIS
In this subsection, informal analysis of RUAM-IoD is conducted to show its effectiveness against the succeeding attacks.

1) BPC ATTACK
After procuring the information \{$CT_{x}^{new}$, $Athn_{x}^{new}$, $rpp_{x}^{new}$, $Gen(.)$, $ERT_{x}^{new}$, $Rep(.)$, $IV_{x}^{new}$\}, which are pre-loaded in the memory of $SD_{x}$, A need to update the password of $U_c$. However, to update the password of $U_c$, A picks $PAW_{D_{x}}^{A}$ and $ID_{A}^{D_{x}}$, and $B_{in}^{A}$ on behalf of $U_c$ and perform the following computations $(BK_{U_c}^{A}) = Rep(B_{in}^{A}, rpp), AR^{A} = (BK_{U_c}^{A} \parallel ID_{U_c}), KR^{A} = (AR^{A} \parallel PAW_{D_{x}}^{A}), PT_{x}^{A} = H(D_{x}^{new} \parallel ID_{A}^{D_{x}} \parallel BK_{U_c}^{A} \parallel Y^{A})$. To check if the decryption process is successful, A verifies the condition $Athn_{x}^{A} = Athn_{x}$. Hovered, It is computationally infeasible for A to determine the secret credentials, such as $PAW_{U_c}$ and $ID_{U_c}$, and $B_{in}$ associated with $U_c$ simultaneously. Therefore, A cannot perform theses computation successfully without the knowledge of $PAW_{U_c}$ and $ID_{U_c}$, and $B_{in}$ and canot update the password of $U_c$. Thus, the proposed RUAM-IoD is secured against BPC attack.

2) SSD ATTACK
Assume that A can obtain the lost/stolen smart device $SD_{x}$ of $U_c$. A by employing PA attack can extricate the information \{$CT_{x}^{new}$, $Athn_{x}^{new}$, $rpp_{x}^{new}$, $Gen(.)$, $ERT_{x}^{new}$, $Rep(.)$, $IV_{x}^{new}$\}, which are pre-loaded in the memory of $SD_{x}$. A cannot gain any confidential or secret information related to $U_c$ because all the sensitive information are stored in the encrypted form. It is imperative for A to determine $KR = (AR \parallel PAW_{D_{x}})$, where $AR = (BK_{U_c} \parallel ID_{U_c})$ to make the encryption process successful. A requires to know $ID_{U_c}$, $PAW_{U_c}$, and $BK_{U_c}$, to compute $KR$. Computationally, it is impracticable for A to determine the bio-metric key $BK_{U_c}$, which is used in deriving KR. The secret key $KR$ used to encrypt the information retrieved from $SD_{x}$’s memory. Therefore, without knowing KR, Computationally, it is infeasible for A to extricate any sensitive information related to $U_c$ after retrieving information form $SD_{x}$. Thus, RUAM-IoD is secured with respect to SSD attack.

3) MITM ATTACK
According to DY model, the adversary, A, can capture, modify or compromise all the exchanged message, which are communicated over the wireless channel. During the AKA process, the communicated message are $MS_{a} : \{T_{a}, Q_{1}, Q_{2}$, $Pu_{A}$, $Athn_{a}^{A}\}$, $MS_{b} : \{T_{b}, Q_{3}$, $Pu_{B}$, $Athn_{b}^{A}\}$, and $MS_{c} : \{T_{c}, Q_{4}$, $Pu_{C}$, $Athn_{c}^{A}\}$ Now, A may attempt to alter the content of the transmitted messages to make the message receiving entity believe that the received messages are from the legitimate entity. If A tries to reconstruct $MS_{a}$, A requires to alter the contents of $Q_{1}, Q_{2}$, $Pu_{A}$, $Athn_{a}^{A}$, which requires the knowledge of $PID_{U_c}$, $SP_{U_c}$, and $S_{in}$. Moreover, to reconstruct $MS_{b}$, A requires the knowledge $SP_{D_{x}}$, and $S_{in}$. $S_{in}$ and $R_{in}$, are the necessitated parameters to regenerate $MS_{c}$. Therefore, it is impractical for A to regenerate a valid message without the knowing the secret credentials associated with a specific entity. Thus, RUAM-IoD can withstand MITM attack.

4) DoS ATTACK
Local authentication is necessary to prevent $U_c$ from sending too many AKA requests to CS. To accomplish the local authentication, $U_c$ requires to inputs its secret credentials, such as $ID_{U_c}$, $PAW_{U_c}$, and $BK_{U_c}$ at the interface of $SD_{x}$ and execute the following computations $(BK_{U_c}) = Rep(B_{in}^{A}, rpp)$. 

19844  
VOLUME 10, 2022
A may attempt to replay the messages to excerpt some estimable information from network entities involved in the AKA process. According to the DY model, A requires to reconstruct a message to make believe CS that this reconstructed message is from a legitimate UE of the system. However, to construct a licit \( MS_{a} \), A requires to know the secret credentials, such as \( SP_{U} \), and \( S_{lu} \). Moreover, it is computationally impracticable for A to procure the secret credentials of \( U_{e} \). Thus, A cannot effectuate the UI attack. To reconstruct \( MS_{B} \) and \( MS_{C} \), A needs to know the secret credentials of CS and \( D_{s} \). A cannot impersonate as a legitimate CS and \( D_{s} \) in the communication system without the knowledge of secret credentials of CS and \( D_{s} \). Therefore, RUAM-IoD is secured against UI, DI, and CSI attacks.

6) UA AND UNTRACEABILITY
A has the capability to expropriate all the exchanged messages, such as \( MS_{a} : \{ T_{A}, Q_{1}, Q_{2}, Pu_{A}, Athn_{a}^{n1} \} \), \( MS_{b} : \{ T_{B}, Q_{3}, Pu_{B}, Athn_{b}^{n3} \} \), and \( MS_{c} : \{ T_{C}, Q_{4}, Pu_{C}, Athn_{c}^{n5} \} \), which transmitted over the public communication channel during the AKA process. It is difficult for A to determine the real-identities of network entities from captured \( MS_{a} \), \( MS_{b} \), and \( MS_{c} \). Therefore, the proposed RUAM-IoD can resist the IG guessing attack. In addition, \( MS_{a} \), \( MS_{b} \), and \( MS_{c} \) are randomly generated because they incorporate the latest timestamp and fresh random-number. After capturing two \( D \) messages from different AKA sessions it is hard for A to determine any significant information by correlating these two messages. Thus, RUAM-IoD ensures UA and untraceability features.

7) REPLAY ATTACK
In RUAM-IoD, there are three messages are exchanged, i.e., \( MS_{a} : \{ T_{A}, Q_{1}, Q_{2}, Pu_{A}, Athn_{a}^{n1} \} \), \( MS_{b} : \{ T_{B}, Q_{3}, Pu_{B}, Athn_{b}^{n3} \} \), and \( MS_{c} : \{ T_{C}, Q_{4}, Pu_{C}, Athn_{c}^{n5} \} \) to accomplish the AKA process. \( MS_{a} \), \( MS_{b} \), and \( MS_{c} \) are exchanged over the public communication channel. According to the DY model, A can potentially capture, modify or compromise all the disseminated messages in the IoD environment. Now, A may attempt to replay the messages to excerpt some estimable information from network entities involved in the AKA process. Each message communicated during the AKA process is produced using the participant’s latest timestamp and a fresh random-number. Therefore, the message receiving entity checks the validity of the timestamp. In case of an invalid timestamp, the message is contemplated as replayed message, and the message receiving network entity declines the validation of the replayed messages, restricting A from effectuating the replay attack.

8) PI ATTACK
In RUAM-IoD, DRC is contemplated as a fully trusted network entity and CS is considered as semi-trusted network entity. A can obtain the secret credentials associated with the legitimate \( U_{e} \) and \( D_{s} \) of the communication system and can effectuate any malicious attacks on the behalf of \( U_{e} \). In RUAM-IoD, secret credentials related to \( U_{e} \) and \( D_{s} \) are stored in encrypted form and insider attacker cannot procure secret information related to \( U_{e} \) and \( D_{s} \) without knowing the secret key \( S_{ek} \) of CS, stored in temper proof database of CS. Therefore, RUAM-IoD is secured against PI attack.

9) DRONE CAPTURE ATTACK
In the IoD environment, it is tough to monitor the drone for all the time (24×7). Thus, A can capture some drones, which are deployed in the IoD environment. After capturing a drone A can extract the sensitive information, such as \( \{ PID_{U}, P, SP_{U} \} \). Since all the drones are assigned with distinct and unique secret parameters. Therefore, the secret parameters of compromised drones are not useful to derive SK, which is established between the non-compromised drone and \( U_{e} \). Thus, RUAM-IoD is secured against drone capture attack.

10) ESL ATTACK
Adversary under the CK-adversary model can compromise the secret credentials (LTS and STS) and session state, in addition to actions permitted under the DY model. In RUAM-IoD, the session key \( SK_{U_{e}}(=SK_{D_{s}})=H(H(R_{U_{e}} || R_{D_{s}} ^{G} + R_{P}) || H(k_{d} || T_{C} || R_{D_{s}} ^{G} + R_{P}) || PID_{D_{s}}) \) is constructed using both LTS and STS secret credentials. By compromising STS, A will not be able to construct the session key \( SK_{U_{e}}(=SK_{D_{s}}) \), which is established between \( U_{e} \) and \( D_{s} \). Similarly, by compromising LTS, A will be able to derive the session key \( SK_{U_{e}}(=SK_{D_{s}}) \). Therefore, to derive the session key \( SK_{U_{e}}(=SK_{D_{s}}) \), A requires to know both LTS and STS secret credentials, which is a computationally expensive task for A. Thus, RUAM-IoD is resistant to ESL attack.

B. FORMAL SECURITY VERIFICATION THROUGH ROM
In our proposed scheme, ROM-based formal method is employed to prove the security strength of the SK that is established during the AKA phase. It is worth noting that a total of three participants, i.e., \( U_{e} \), CS, and \( D_{s} \), play roles during the AKA process. Moreover, from theorem 1, we verify that A is unable to determine the SK, which is determined and set up between the network entities, \( U_{e} \) and \( D_{s} \), by means of CS. ROM has the following components, which are associated with the different queries, accessed by A.
1) PARTICIPANTS
The instances \( p_1, p_2, \) and \( p_3 \) of \( U_e, CS, \) and \( D_x \) are shown by \( \pi_{U_e}, \pi_{CS}, \) and \( \pi_{D_x} \), respectively, which are also deemed as random oracles.

2) FRESHNESS
\( \pi_{U_e} \) and \( \pi_{D_x} \) are deemed to be fresh if the SK established between \( U_e \) and \( D_x \) is not known to \( A \) when \( A \) performs \( Reveal \) query, as explained in Table 3.

3) ACCEPTED STATE
The instance \( \pi \) is deemed to be in accepted state when it receives the last expected message while carrying out the AKA process. In addition to this, \( Sid \) symbolizes the session identifier of \( \pi \) for the present AKA session. It is worth noting that \( Sid \) is created by concatenating the exchanged messages generated in sequence by \( \pi \).

4) PARTNERING
Two instances \( \pi_{p_1} \) and \( \pi_{p_2} \) are considered to be partners in case the three subsequent conditions are simultaneously fulfilled: 1) \( \pi_{p_1} \) and \( \pi_{p_2} \) need to exchange the common \( Sid \) after authenticating each other conjointly, 2) \( \pi_{p_1} \) and \( \pi_{p_2} \) need to be in accepted states, and 3) \( \pi_{p_1} \) and \( \pi_{p_2} \) need to be interdependent partners.

5) ADVERSARY
DY model stipulates that \( A \) has the capabilities to seize all the messages disseminated among the entities in the IoD environment. This implies that \( A \), by means of the queries defined in Table 3, can modify, inject, and delete the communicated messages.

Moreover, this also implies that \( A \) has the capability to access the hash function \( H(\cdot) \). It is worth noting that \( H(\cdot) \) is modeled as a random-oracle, say \( RSH \). Above this, the queries, which are defined in Table 3, are exploited by \( A \) to simulate an attack.

Definition 2: Elliptic Curve Discrete Logarithm Problem (ECDLP): For any \( Pu_g = S_{gk} \cdot P, Aq_{ECDLP}(TP_{oly}) \) is the for \( A \)'s advantage or the probability to derive \( Sk_g \) from \( Pu_g \) within polynomial-time \( TP_{oly} \). It is hard for \( A \) to determine \( Sk_g \) from \( Pu_g \) within polynomial-time, which makes \( Aq_{ECDLP}(TP_{oly}) \) trivial and defined as the elliptic curve discrete logarithm problem (ECDLP).

Definition 3 (Semantic Security): Let \( B \) is the correct bit and \( B' \) is the guessed by \( A \). If condition \( B = B' \) holds, \( A \) wins the game. If \( Pb[\text{Succ}] \) signify the probability of success, \( A \)'s advantage in breaching \( SK_g \)'s security, established while executing the AKA phase of RUAM-IoD is represented by \( Ad_{A}^{\text{RUAM-IoD}} = 2 \cdot Pb[\text{Succ}] - 1 \). RUAM-IoD is protected if \( Ad_{A}^{\text{RUAM-IoD}} \) is trivial under the ROM.

The proof of \( SK_g \)'s security of the proposed RUAM-IoD is presented in Theorem 1.

Theorem 1: Let \( Ad_{A}^{\text{RUAM-IoD}(TP_{oly})} \) is advantage of \( A \) running against the proposed RUAM-IoD in \( TP_{oly} \) to compromise the \( SK_g \)'s security, which is established between \( U_e \) and \( D_x \). If \( Q_0 \) designates SHA-256 quires, \( |RSH| \) indicates output size of SHA-256. \( Q_{CS} \) denotes the send queries, \( lth \) represents the size of \( BK_{U_e} \), \( |PSWD| \) symbolizes the dictionary of passwords, \( Aq_{IND-CPA}^{\Omega_{\text{A}}} \) (1) signifies \( A \)'s advantage to breach the security of AES-CBC-256 in \( TP_{oly} \) (Definition 1), and \( Aq_{ECDLP}^{\Omega_{\text{A}}} \) designates the advantage in compromising ECDLP (Definition 2). \( A \)'s advantage to compromise the \( SK_g \)'s security, which is set up between \( U_e \) and \( D_x \) while executing the proposed RUAM-IoD can be defined as:

\[
Ad_{A}^{\text{RUAM-IoD}(TP_{oly})} = \frac{H^2}{|RSH|} + \frac{|CRS|}{2^{|CPRK|}} + 2 \cdot Aq_{\text{IND-CPA}}^{\Omega_{\text{A}}} + 2 \cdot Aq_{ECDLP}^{\Omega_{\text{A}}}
\]

Proof: Following five games (\( GM_{x} \mid x = 0, 1, 2, 3, 4 \)) are utilized to prove Theorem 1. We follow the same method to prove Theorem 1 as in [12].

\( GM_0 \) : This game is associated with real attack, which is executed by \( A \) against RUAM-IoD in the ROM. It is imperative for \( A \) to select the bit \( B \) at the beginning of \( GM_0 \). The semantic security of RUAM-IoD renders the following:

\[
Ad_{A}^{\text{RUAM-IoD}(TP_{oly})} = 2 \cdot Pb[\text{Succ}] - 1
\]

\( GM_1 \) : An eavesdropping attack is effectuated in this game, in which \( A \) can eavesdrop all the messages, such as \( MS_{D} \) : \( \{T_A, Q_1, Q_2, Pu_D, Athn^1\} \), \( MS_b \) : \( \{T_B, Q_3, Pu_D, Athn^3\} \), and \( MS_c \) : \( \{T_C, Q_4, Pu_d, Athn^5\} \) by using the Execute query defined in Table 3, which are exchanged during execution of the AKA process. Upon the completion of this game, \( A \) required to make the \( Reveal \) query along with \( Test \) query to determine whether the derived \( SK_g \) is the correct key or a random key. In the proposed RUAM-IoD is computed as \( SK_{U_e} (= SK_{D_x}) = H(H(R_{U_e} \parallel R'_{G} \parallel R_D) \parallel H(k_d \parallel T_C \parallel R''_{G} \parallel R_D) \parallel PID_{D_x}) \), which is the amalgamation of both the LTS and STS parameters. Therefore, \( A \) requires knowing both STS parameters, such as \( R_{U_e}, R_G, R_D, S_{U_e}, \) and \( k_d \) and LTS parameters, such as \( PID_{D_x} \) and \( PID_{D_x} \) to construct a valid \( SK_{U_e} (= SK_D) \). Therefore, only by capturing the communicated message, such as \( MS_{a}, MS_b, \) and \( MS_c \), \( A \)'s winning
possibility/probability of $GM_1$ is not enhance at all. Thus, both the games $GM_1$ and $GM_2$ remains indistinguishable. So, we get

$$Pb[Succ0] = Pb[Succ1].$$  \hfill (77)

$GM_2$ : In this game, $A$ launches an active attack, which incorporates the Send and RSH oracles and attempts to convince a specific network entity to receive the modified message. In addition, $A$ can implement any number of queries to find a collision in the hash digest. However, all the exchanged messages are protected by the irreversible and collision-resistant SHA-256. Therefore, it is infeasible for $A$ to attain the collision in the output (hash digest) produced by SHA-256. Then, by birthday paradox, the succeeding result is achieved:

$$|Pb[Succ1] − Pb[Succ2]| ≤ \frac{H_2^2}{2 \cdot |RSH|}.$$  \hfill (78)

$GM_3$ : CorruptSD query is implemented in this game. Therefore, $A$ can extract all the sensitive information, such as $\{CT_i, \text{ Athn, rpp, Gen()}, \text{ ERT, Rep().},IV_a\}$, which are preloaded in the memory of $SD_e$ employing PA attack. $A$, from the extracted information cannot procure any useful information because the secret information assigned to $U_e$ are stored in the encrypted form. Therefore, $A$ need to decrypt $CT_i$ to procure the secret parameters. However, to make the decryption process successful, $A$ requires to compute the secret key $KR = (AR \parallel PAW_u,)$ which is used for the encryption process. The secret key $KR$ is the amalgamation of $ID_{u_i},PAW_{u_j}$, and $BK_{u_j}$. The guessing probability of the bio-metric key $BK_{u_j}$ is $\frac{2^{|E|}}{|E|}$, which is negligible. Thus, it is impractical for $A$ to get any secret parameter by extracting the information from the memory of $SD_e$. In addition, $U_e$ is permitted to make a restricted number of wrong password attempts. Under these conditions, $GM_2$ and $GM_3$ are indistinguishable in the exclusion of guessing attack; the subsequent result is procured:

$$|Pb[Succ2] − Pb[Succ3]| ≤ \frac{Q_{CS}}{2^{Lh}|PWD|}.$$  \hfill (79)

$GM_4$ : This is the final game, $A$ will try to derive the session key $SK_{U_e}(= SK_{D_e})$, which is establish between $U_e$ and $D_e$ by eavesdropping all the exchanged message, such as $MS_{A_i}, MS_{B_i}$, and $MS_{C_i}$. In the proposed RUAM-IoD, the session key is constructed as $SK_{U_e}(= SK_{D_e}) = H(H(R_{U_i} \parallel R_G \parallel R_D) \parallel H(k_l \parallel T_C \parallel R_G \parallel R_D) \parallel PID_{D_e})$, where $k_l = S_{U_l} \cdot P_{U_l}$. It is impractical for $A$ to derive $S_{U_l}$ from the public key of user $P_{U_l}$ and $S_{U_l}$ from public key of drone $P_{U_d}$ in polynomial time and is referred to ECDLP problem in ECC (Definition 2). In addition, the secret parameters, such as $R_{U_i}, R_G$, and $R_D$ are exchanged among the network entities in encrypted form. In RUAM-IoD, AES-CBC-256 is used as the encryption algorithm, which is secure (IND-CPA secure) to use and $A$ cannot breach the security of AES-CBC-256 in polynomial time (Definition 1. Therefore, it is hard for $A$ to derive $SK_{U_e}(= SK_{D_e})$. So, both the games $GM_3$ and $GM_4$ remain indistinguishable in the absence of breaching the security of AES-CBC-256 and solving the ECDLP. The following result can be achieved:

$$|Pb[Succ3] − Pb[Succ4]| ≤ Ad_{\Omega \cdot A}^{IND−CPA}(I) + Ad_{\Omega \cdot A}^{ECDLP}(TP_{oly}).$$  \hfill (80)

$A$ has accomplished all the queries. Therefore, $A$ requires to determine the bit $B_i$ in order to win the game after executing the Test query. It is then obvious that

$$Pb[Succ4] = 1/2.$$  \hfill (81)

From (76) and (77), we get

$$Ad_{\Omega \cdot A}^{RUAM−IoD}(TP_{oly}) = |Pb[Succ0] − \frac{1}{2}|.$$  \hfill (82)

From (82), we get

$$\frac{1}{2} \cdot Ad_{\Omega \cdot A}^{RUAM−IoD}(TP_{oly}) = |Pb[Succ1] − Pb[Succ4]|.$$  \hfill (83)

By using (81) and (83), we obtain

$$\frac{1}{2} \cdot Ad_{\Omega \cdot A}^{RUAM−IoD}(TP_{oly}) = |Pb[Succ1] − Pb[Succ4]|.$$  \hfill (84)

By utilizing the triangular inequality, following

$$|Pb[Succ1] − Pb[Succ4]| \leq |Pb[Succ1] − Pb[Succ2]| + |Pb[Succ2] − Pb[Succ4]| \leq |Pb[Succ1] − Pb[Succ2]| + |Pb[Succ2] − Pb[Succ3]| + |Pb[Succ3] − Pb[Succ4]|.$$  \hfill (85)

By utilizing (78), (79), (81), and (85), we get

$$\frac{1}{2} Ad_{\Omega \cdot A}^{RUAM−IoD}(TP_{oly}) = \frac{H_2^2}{2 \cdot |RSH|} + \frac{Q_{CS}}{2^{Lh−1}|PWD|} + Ad_{\Omega \cdot A}^{IND−CPA}(l) + Ad_{\Omega \cdot A}^{ECDLP}(TP_{oly}).$$  \hfill (86)

Hence, from equation (86), we get

$$Ad_{\Omega \cdot A}^{RUAM−IoD}(TP_{oly}) \leq \frac{H_2^2}{|RSH|} + \frac{Q_{CS}}{2^{Lh−1}|PWD|} + 2 \cdot Ad_{\Omega \cdot A}^{IND−CPA}(l) + 2 \cdot Ad_{\Omega \cdot A}^{ECDLP}(TP_{oly}).$$  \hfill (87)

**C. SECURITY EVALUATION USING SCYThER TOOL**

Fig. 5 exhibits the result generated through Scyther tool-based formal security validation. Scyther is utilized extensively to prove the security perspectives of any security protocol in an automated way. Compared to other security protocol validation tools, such as Pro-Verify and AVISPA, Scyther is more commonly employed by the researcher to validate the security of the proposed AKA schemes. One of the advantages of Scyther is that it is based on the DY
adversarial model and the simulation results it generates to ensure that the secret parameters are not disclosed while executing the AKA scheme.

Since Scyther uses the security protocol description language (SPDL), a python-like language, for the description of security protocols hence, RUAM-IoD is coded in SPDL. To this end, three roles are defined in the SPDL script, which are $U_e$, $CS$, and $D_x$. In addition, there are different claims in SPDL generated either manually or automatically. Scyther facilitates describing and verifying these claims. For instance, the “Alive claim” guarantees that a network entity has accomplished some events. “Nisynch claims” guarantees that all the communicated messages between two network entities are delivered successfully. “Weak-agree” ensures the AKA scheme is protected against the impersonation attack. All these automatically generated claims are verified according to the procedure shown in Fig. 5. In addition, the manually generated claim, such as $\text{claim}(UE, \text{Secret}, SKU)$ and $\text{claim}(DX, \text{Secret}, SKD)$ are also verified, which indicates that an attacker cannot determine the secret SK. Fig. 5 indicates that the proposed RUAM-IoD is safe and an attacker cannot find any vulnerability.

VII. PERFORMANCE EVALUATION

RUAM-IoD is compared with the existing AKA scheme, such as Wazid et al. [18], Sutrala et al. [35], and Jangirala et al. [32]. The performance of RUAM-IoD is measured in terms of computational, memory/storage, and communication costs. We utilize the widely-accepted “Multi-precision Integer and Rational Arithmetic Cryptographic Library (MIRACL)” to conduct the experimental evaluation for different cryptographic primitives. This will enable us to estimate the computational time of the cryptographic primitives on the succeeding two environments (platforms):

1) We consider the settings (platform ($PF - 1$)): Intel(R) Core(TM) i7-6700 with CPU: 3.40 GHz, RAM: 8 GB, OS: Ubuntu 16.04 LTS, 64-bit to simulate the server ($CS$) type environment.

2) The settings (platform ($PF - 2$)) are considered for simulating the drone ($D_x$) and user ($U_e$): Raspberry Pi (RP-3) with CPU: Quad-core@1.2 GHz (64 bits), RAM: 1 GB, and OS: Ubuntu 16.04 LTS (64-bit).

In the existing literature, the same environment is used to conduct experiments on resource-constricted devices [46], [47]. Each cryptographic (algorithm) primitive is executed for 100 time for $PF - 1$ and $PF - 2$ to procure the average computational time different cryptographic primitives. Table 4 provides the average computational time of various cryptographic primitives.

A. SECURITY FEATURES COMPARISON

In this subsection, we compare the security feature of RUAM-IoD with Wazid et al. [18], Sutrala et al. [35], and Jangirala et al. [32]. To this end, a comparative analysis of the security features of RUAM-IoD and the related scheme is presented in Table 5. It is shown in the table that the scheme of Sutrala et al. [35] cannot resist DSY attack, and the scheme of Wazid et al. [18] is susceptible to DSY attack and does not render ROM-based analysis and RvP phase. The scheme of Jangirala et al. [32] is susceptible to MITM, UI, parallel session, DI, and SK compromise attacks and does not render the anonymity and untraceablity features. In contrast, RUAM-IoD is secured against the DSY, UI, DI, and SK compromised attacks.

B. COMPUTATIONAL COST COMPARISON

Computational cost denotes the CPU time required by a security scheme to complete its AKA process. Thus, without losing the security features, minimizing the computational cost is a critical design goal of AKA or security schemes. Table 4 presents the computational cost of various cryptographic primitives, which are used to compute the computational cost of RUAM-IoD and the related AKA schemes. The computational cost at user side in the proposed RUAM-IoD is $8T_{HF} + 4T_{ENC} + 3T_{EPM} + T_{FE} \approx [15.825]$ ms, while
Sutrala et al. [35], Wazid et al. [18], and Jangirala et al. [32] require $16T_{HF} + 5T_{EPM} + 2T_{EPA} + T_{FE} \approx [22.804]$ ms, $19T_{HF} + 4T_{EPM} + T_{EPA} + T_{FE} \approx [20.66]$ ms, and $16T_{HF} + 5T_{EPM} + 2T_{EPA} + T_{FE} \approx [22.804]$ ms. RUAM-IoD requires less computational cost at the user side than the related AKA schemes, as shown in Fig. 6. The CS, stationed at the DRC of DSP, is a critical component in the IoT environment. So, it is desirable to reduce the computational cost at CS. The computational cost at CS side in the proposed RUAM-IoD is $5T_{HF} + 3T_{ENC} + T_{EPM} \approx [0.858]$ ms, while Sutrala et al. [35], Wazid et al. [18], and Jangirala et al. [32] require $9T_{HF} + 3T_{EPM} + 2T_{EPA} \approx [2.084]$ ms, $T_{HF} + 5T_{EPM} + T_{EPA} \approx [3.058]$ ms, and $11T_{HF} + 3T_{EPM} + T_{EPA} \approx [2.138]$ ms. So, RUAM-IoD incurs lesser computational cost than the related AKA schemes as shown in Fig. 6. Aside from this, Fig. 7 shows that the computational cost increases at CS, in all the schemes, as the number of authentication (user) requests increases at CS. However, RUAM-IoD reduces the computational cost in comparison to the other schemes. In the proposed RUAM-IoD, the computational cost at drone ($D_x$) or sensor node $6T_{HF} + 2T_{ENC} + 2T_{EPM} \approx [8.55]$ ms, while Sutrala et al. [35], Wazid et al. [18], and Jangirala et al. [32] require $8T_{HF} + 4T_{EPM} + T_{EPA} \approx [14.32]$ ms, $12T_{HF} + 4T_{EPM} + T_{EPA} \approx [15.87]$ ms, and $8T_{HF} + 3T_{EPM} + T_{EPA} \approx [11.40]$ ms. Fig. 6 also shows that RUAM-IoD needs less computational resources at the drone side than required by the related AKA schemes. This implies that RUAM-IoD is suitable for drone environment because drone being a resource-constrained device requires a reduced level of computational cost. In addition, Table 3 and Fig. 8 illustrate the total computational cost required to accomplish the AKA process of RUAM-IoD.

### C. COMMUNICATION COST COMPARISON

Communication cost signifies the number of communicated messages (bits) transmitted to perform the AKA process. Therefore, it is essential to reduce the communication cost required to accomplish AKA process without risking the security traits of a security scheme. In the proposed RUAM-IoD, during the AKA process, the communicated messages are $M_{Sa} = \{T_A, Q_1, Q_2, Pu_a, Athn^{at}\}$, $M_{Sb} = \{T_B, Q_3, Pu_b, Athn^{bs}\}$, and $M_{Sc} = \{T_C, Q_4, Pu_c, Athn^{ct}\}$. The length $M_{Sa}$, $M_{Sb}$, and $M_{Sc}$ is $32 + 256 + 160 + 256 = 960$ bits, $32 + 256 + 160 + 256 = 704$ bits, and $32 + 128 + 160 + 256 = 576$ bits, respectively. Thus, the total communication cost required by RUAM-IoD to accomplish the AKA phase is $[960 + 704 + 576] = [2240]$ bits. Contrarily, Wazid et al. [18], Sutrala et al. [35], and Jangirala et al. [32] require $[3360]$ bits, $[3200]$ bits, and $[2656]$ bits, respectively. So, it is evident from Table 7 and Fig. 9 that RUAM-IoD demands less communication cost than demanded by the related AKA protocols. Fig. 10 illustrates the communication cost incurred when multiple users need to obtain the real-time information from a particular $D_x$ concurrently.

### D. STORAGE COST COMPARISON

As drones are resource-constricted devices with limited storage/memory resources, diminishing its memory utilization is the pressing need when designing an AKA protocol.
TABLE 7. Communication cost comparison.

| Protocol/Scheme | Communicated Messages During AKA Phase | Total |
|-----------------|----------------------------------------|-------|
| Sutrala et al. [35] | $U_e, CS/GS, D_x/SN_x, U_e$ | [3200] bits |
| Wazid et al. [18] | $U_e, CS/GS, D_x/SN_x, U_e$ | [3184] bits |
| Jangirala et al. [32] | $U_e, CS/GS, D_x/SN_x, U_e$ | [2656] bits |
| RUAM-IoD | $U_e, CS/GS, D_x/SN_x, U_e$ | [2240] bits |

FIGURE 9. Communication cost required to execute the AKA phase (single user).

FIGURE 10. Communication cost at $D_x$ side with increasing user authentication requests.

In RUAM-IoD, three entities are involved in the accomplishment of AKA process. These entities include $U_e$, $CS$, and $D_x$. Moreover, in RUAM-IoD, $U_e$, $CS$, and $D_x$ are required to store\{CT_{st}, Athn, rpp, Gen(\text{)}, ERT, Rep(\text{)}, IV_e\}, \{PID_{U_e}, CT_g, IV_{reg}\}, and \{PID_{D_x}, SP_{D_x}, P\}$, respectively. $U_e$, $CS$, and $D_x$ have to store $\{512 + 256 + 160 + 8 + 128\} = [1064]$ bits, \{128 + 384 + 128\} = 640 bits, and \{256\} bits, respectively. This way, RUAM-IoD demands a storage/memory capacity of $\{1064 + 640 + 256\} = [1985]$ bits. Contrarily, the AKA scheme of Wazid et al. [18], Sutrala et al. [35], and Jangirala et al. [32] require to store $\{4696\}$ bits, $\{4320\}$ bits, and $\{1768\}$ bits, respectively. This comparison is more visibly illustrated in Fig. 11 wherein RUAM-IoD needs fewer memory/storage cost than Wazid et al. [18] and

Sutrala et al. [35] with a marginal increment in memory/storage cost compared to Jangirala et al. [32].

VIII. CONCLUSION

This paper has presented an AKA scheme, called RUAM-IoD, for securing the communication between a remote user and a drone. To this end, RUAM-IoD checks the authenticity of a remote user before allowing him to access, in real-time, the sensitive information from a drone deployed in a particular FZ. After validating the authenticity of the remote user, RUAM-IoD establishes an SK between the user and the drone to make their communication indecipherable. The effectiveness of RUAM-IoD is verified against various security attacks through informal analysis. Furthermore, the security of the established SK is validated using ROM-based formal analysis. In addition, Scyther-based validation is performed on RUAM-IoD that demonstrated that the RUAM-IoD is secure against various security attacks. Furthermore, the performance analysis demonstrated that RUAM-IoD requires less computational, storage, and communication cost without compromising the security features.

REFERENCES

[1] M. M. Azari, G. Geraci, A. Garcia-Rodriguez, and S. Pollin, “UAV-to-UAV communications in cellular networks,” IEEE Trans. Wireless Commun., vol. 19, no. 9, pp. 6130–6144, Sep. 2020.
[2] A. Noorwali, M. A. Javed, and M. Z. Khan, “Efficient UAV communications: Recent trends and challenges,” Comput., Mater. Continua, vol. 67, no. 1, pp. 463–476, 2021.
[3] H. Shakhatreh, A. H. Sawalmeh, A. Al-Faqaha, Z. Dou, E. Almaita, I. Khalil, N. S. Othman, A. Khreishah, and M. Guizani, “Unmanned aerial vehicles (UAVs): A survey on civil applications and key research challenges,” IEEE Access, vol. 7, pp. 48572–48634, 2019.
[4] J. Wang, Z. Na, and X. Liu, “Collaborative design of multi-UAV trajectory and resource scheduling for 6G-enabled Internet of Things,” IEEE Internet Things J., vol. 8, no. 20, pp. 15096–15106, Oct. 2021.
[5] J. Srinivas, A. K. Das, N. Kumar, and J. J. Rodrigues, “TCALAS: Temporal credential-based anonymous lightweight authentication scheme for Internet of Drones environment,” IEEE Trans. Veh. Technol., vol. 68, no. 7, pp. 6903–6916, Oct. 2019.
[6] M. Gharibi, R. Boutaba, and S. L. Waslander, “Internet of Drones,” IEEE Access, vol. 5, pp. 1148–1162, 2016.
[7] R. J. Hall, “An Internet of Drones,” IEEE Internet Comput., vol. 20, no. 3, pp. 68–73, May/Jun. 2016.
[8] S. Atoev, O.-H. Kwon, S.-H. Lee, and K.-R. Kwon, “An efficient SC-FDM modulation technique for a UAV communication link,” Electronics, vol. 7, no. 12, p. 352, Nov. 2018.
