BioKA-ASVN: Biometric-Based Key Agreement Scheme for Air Smart Vehicular Networks Using Blockchain Service

Basudeb Bera, Abhishek Bisht, Ashok Kumar Das, *Senior Member, IEEE*, Bharat Bhargava, *Fellow, IEEE*, David K. Y. Yau, *Senior Member, IEEE*, Pascal Lorenz, *Senior Member, IEEE*, Biplab Sikdar, *Senior Member, IEEE*

***Abstract*—Air Smart Vehicular Networks (ASVNs) have become increasingly essentials in military and commercial industries in recent years, where drones are deployed to interact among each other via wireless medium in airspace due to their agility and versatility. ASVNs can form a closed loop from data perception to final execution by integrating communication devices, computation tools, and control modules. However, the used unencrypted and publicly available navigational signals like Global Positioning System (GPS) and communication over (public) insecure Internet connections, including wireless networks and WiFi, make ASVNs vulnerable to security breaches. Attackers can easily gain access to a drone's configuration and take control remotely, by launching various attacks such as GPS spoofing, false sensor data injection, maldrone malware injection, eavesdropping, man-in-the-middle, Denial-of-Service (DoS), replay, forgery, and Skyjack attacks. This paper proposes a robust and efficient multi-factor biometric-based security mechanism using blockchain as a service to address the security and privacy breaches in ASVNs. A comparative study demonstrates that the proposed scheme provides superior security and better functionality features with low communication and computation overheads as compared to the existing analogous schemes. The feasibility of the proposed scheme in real-life drone applications is also demonstrated through a real-time testbed and blockchain simulation. Furthermore, a detailed security analysis using the automated software validation tool, namely Scyther, verifies the proposed scheme's significant level of security.**

***Index Terms*—Air smart vehicular network (ASVN), authentication, security, blockchain, Scyther, testbed experiments.**

I. Introduction

In an air smart vehicular network (ASVN), drones communicate over the wireless medium using airspace with minimum human involvement. A drone, commonly known as an "unmanned aerial vehicle (UAV)", is an aircraft without any human pilot and controlled by remotely. The number of UAV applications has been increasing steadily, ranging from basic entertainment systems to high-tech medical emergencies due to its agility, affordability, and ease of deployment [1]. According to the Federal Aviation Administration (FAA) report, sales of larger commercial UAVs have risen from 0.6 million in 2016 to 2.7 million by 2020, with an increase from 2.5 million in 2016 to 7 million in 2020 [2].

An ASVN system is powered by a variety of onboard sensors (e.g., global positioning system (GPS), accelerometer, etc.), which provide sensor-related information to the associated inbuilt flight controller, which can then forward these readings to the corresponding operator via a communication channel (mostly wireless networks and WiFi). The operator sends the control signals to the flight controller based on the received information. In such a case, a malicious hacker can easily access to a drone's configuration and he/she can hijack it by applying an open-source drone-hijack application (for example, Skyjack). As a result, the hacker can take control of the wireless device remotely [3]. Serious harm can be done in the case of a drone's application for military purposes if a military drone is hijacked. The drone is not only physically damaged or its physical components are compromised in this case, but the attacker may also be able to use the drone as a weapon. Thus, an ASVN needs to provide a secure communication from data perception through information exchange to final execution by integrating embedded systems with communication devices, computation tools, and control modules [1], [4].

To keep the system in a correct state, four essential components need to be interacted and functional. The hardware, software, sensors, and communication link are among these elements. The ASVN system may crash and become grounded if any components fail [5]. The Internet of Drones (IoD) technology relies on a variety of communication methods to connect the drones to each other and also to ground-based control systems. The security of these communication channels is essential to ensure the safe and reliable operations in ASVN systems. In this article, we consider only the communication-level security. In the context of communication-level security measures, the past literature has extensively utilized encryption, authentication, access control, and more, as evidenced by various works [6]–[11], [3], [12]. Among these security measures, we specifically emphasize the significance of the "authentication and key agreement security approach" in en-

Basudeb Bera and Biplab Sikdar are with the "Department of Electrical and Computer Engineering, National University of Singapore, Singapore 117583 (e-mail: b.bera26@nus.edu.sg, bsikdar@nus.edu.sg)."

Abhishek Bisht and Ashok Kumar Das are with the "Center for Security, Theory and Algorithmic Research, International Institute of Information Technology, Hyderabad 500032, India (e-mail: abhishek.bisht@research.iiit.ac.in, iitkgp.akdas@gmail.com, ashok.das@iiit.ac.in)."

Bharat Bhargava is with the "Department of Computer Sciences, Purdue University, West Lafayette, Indiana, 47907, USA (e-mail: bbshail@purdue.edu)."

David K. Y. Yau is with the "Information Systems Technology and Design, Singapore University of Technology and Design, Singapore, 487372 (e-mail: david_yau@sutd.edu.sg)."

Pascal Lorenz is with the "University of Haute Alsace, France (e-mail: lorenz@ieee.org). *(Corresponding authors: Ashok Kumar Das; Pascal Lorenz)*"

suring communication-level security. This approach plays a critical role in preventing unauthorized access, data breaches, and other security threats that could potentially jeopardize the safety of drones [13]. In the realm of ASVN, drones typically employ wireless communication technologies such as Wi-Fi, Bluetooth, or cellular networks to establish connections with other devices or fellow drones. However, these communication methods are vulnerable to interception or eavesdropping by unauthorized parties if appropriate security measures are not in place. Drones operate under constraints, including limited computing resources, memory capacity, and battery life. These constraints pose challenges when implementing complex cryptographic techniques. Furthermore, drones' high mobility can adversely affect the quality and stability of wireless communication channels, making it difficult to maintain a secure key exchange. In addition to mobility challenges, drones are often deployed in hostile environments where they may encounter jamming, eavesdropping, or physical capture attacks. These threats can compromise the security of the key exchange process. Furthermore, drones are frequently used in time-sensitive applications, such as battlefield scenarios, where delays in the key exchange can have severe consequences. Given these unique characteristics and challenges, any designed security protocol for drones must prioritize both speed and security, as well as lightweight implementation, to maximize the operational lifespan of these devices [14], [15]

To enhance security, numerous authentication and key agreement protocols have emerged over the past decade [6]–[11]. However, many of these protocols, despite incorporating two or three-factor authentication, fall short of achieving the desired security level in challenging environments. This underscores the significance of both the number of factors involved and their implementation in determining authentication's reliability and security [16]. In ASVN, where data is voluminous and diverse, storage presents a unique challenge. A single server is impractical due to the risk of a single point of failure. To address this issue, blockchain technology emerges as a promising solution, offering cryptographically secure and decentralized distribution [17]. Consequently, our objective is to design a blockchain-based multi-factor authentication system for ASVNs, incorporating a user's mobile device, password, personal biometrics, and real identity as the four crucial factors to achieve the highest level of security.

A. System Models

We explain both the network and threat models that are required for the proposed approach in this section.

1) Network Model: The network model for the proposed scheme (BioKA-ASVN) is provided in Fig. 1. In this network architecture, we consider the communication entities as a) user (U_i), b) drone (DR_j), and c) a ground server (also considered as an authentication server) (GS). GS has a responsibility to register other entities in the network and is assumed to be a fully trusted registration authority. A user U_i can register with the GS by providing minimal information securely, and at the end of the registration process, GS gives some secret credentials for future communication and authentication. The GS registers a drone with unique and distinct credentials for each DR_j. Once the registration is over, the entities are deployed into their respective working areas, and GS is placed under a physical locking system. DR_j detects information from a drone's airspace and sends it to the associated GS, which is forwarded to an attached peer-to-peer (P2P) cloud server (CS) network, also known as a blockchain center. The data is finally stored in a blockchain for secure storage.

2) Adversary Model: In the proposed BioKA-ASVN, all the network entities communicate over the insecure wireless media. A user U_i can send a drone access request via GS, and the drone DR_j can also send the sensing information to the associated GS via a public channel (e.g., Wi-Fi or wireless media). The majority of the information are private and confidential in the drone environment (for example, battle field, smart agriculture, and boarder surveillance). Since the information are exchanged over the insecure channel, there is a security concern. According to the Dolev-Yao (DY) threat model [18], an unauthorized user (also called an adversary, $\mathcal{A}$) not only can eavesdrop on the communication messages, but can also delete, modify, and inject malicious contents into the communication channel. We also adopt the *de-facto* and widely-recognized Canetti and Krawczyk (CK)-threat model [19], where $\mathcal{A}$ has more provision than the DY threat model. Under the CK-adversary threat model, $\mathcal{A}$ is not restricted to intercepting, modifying, deleting, or inserting incorporated messages of various entities (for example, U_i, DR_j, GS, and CS) engaged in the network as indicated in the DY model. Additionally, $\mathcal{A}$ is also capable of capturing long-term secrets, short-term keys, and session states by hijacking a session if these credentials are loaded inside an insecure memory of communicating parties during the mutual authentication and key agreement phase.

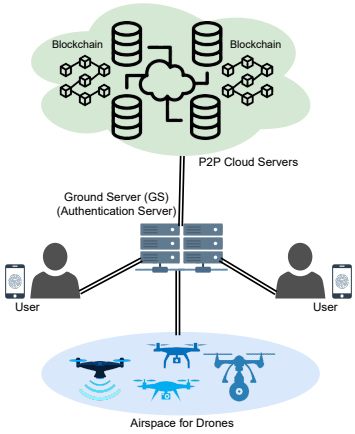


Fig. 1. Blockchain-based network model for air smart vehicular networks.

Due to hostile environment, $\mathcal{A}$ can physically capture a drone using any of the techniques: 1) "shoot it down with a gun", 2) "use anti-drone drones", 3) "use net-firing anti-drone guns", 4) "jam the drone's radio signal", and 5) "use trained eagles to capture drones" [20]. $\mathcal{A}$ may then attempt to launch further attacks, such as identity disclose, impersonation attacks and so on, using the retrieved secret information saved

in the physically kidnapped DR_j and side-channel attacks such as power analysis attacks [21]. In this paper, since GS is considered as a fully trusted authority and responsible for generating all public and private keys as well as certificates, all the generated public keys are authentic. Thus, it is not required to consider any other public key infrastructure (PKI) for public key authentication. In this work, we consider identity (ID), password (Pw), bio-metric template (B), and mobile device (MD)'s stored information of a registered user as the four factors for designing our authentication protocol. The real identity (ID) is hidden during communication over the public channel. We assume that the adversary $\mathcal{A}$ can guess a factor at a time in polynomial time, like a password or a real-identity of the user. However, guessing two factors (biometrics and real-identity, real-identity and password) will not be done in polynomial time by $\mathcal{A}$, and hence, it is a difficult task for $\mathcal{A}$. Moreover, based on observations presented in "practical threshold multi-factor authentication" [22], in practice, if a threshold number of factors is available to a user out of registered multi-factors, the user must be allowed to login to the system. Thus, if revealing any two factors cannot reveal the other factors, the disclosure of a session key based on the multi-factors is impossible.

B. Motivation

In an ASVN, drones rely on onboard sensors (i.e., gyroscopes, Global Positioning System (GPS), cameras and accelerometers) to transmit sensory data to their flight controllers, which then relay the information to operators through communication channels. Drones employ unencrypted navigational signals and public networks to communicate across an insecure Internet, creating privacy and security concerns. Malicious actors can gain access to the drone's configuration and manipulate it using an open-source drone-hijack application, like Skyjack, leading to unauthorized remote control [3], [5].

Several attacks have been reported, particularly in an IoD network, such as GPS spoofing, false sensor data injection, maldrone malware injection, eavesdropping, man-in-the-middle, Denial-of-Service (DoS), replay, forgery, and Skyjack attacks. Existing security solutions (e.g., access control, authentication, digital signature, and key agreement) [6]–[11] are unable to adequately protect against vulnerabilities and threats in drone communications, thus failing to ensure data confidentiality, message authentication, data availability, and message integrity [23], [24]. For instance, in 2023, Irshad *et al.* [25] proposed a three-factor authenticated key exchange mechanism in IIoT environments. However, their scheme incurs significant computation and communication costs, making it impractical for real-world IoT applications, especially in resource-constrained devices. In 2023, Zhang *et al.* [26] also proposed a three-factor authentication and key agreement scheme for an IoT-enabled e-healthcare application. Their proposed scheme is vulnerable to replay attacks and lacks to support for blockchain solutions and dynamic drone addition. Ali *et al.*'s scheme [9] exhibits vulnerabilities, such as smart card theft and masquerade attacks. It is also susceptible to ESL attacks under the Canetti and Krawczyk (CK)-threat model [19] due to flaws in constructing session keys using public information and less secure information, as detailed in [27]. Similarly, Jan *et al.*'s scheme [28] cannot ensure user anonymity and untraceability, as user real identities are revealed through the communicated messages over the public channel. Furthermore, their scheme is not resilient against privilege-insider attacks because the insider user, being an attacker, can reveal users' personal secret due to their unawareness when transmitting this sensitive data.

Although two-factor authentication provides some level of security, multi-factor authentication for air smart vehicular network is needed to extend the security level to fulfill the desired requirements. In three-factor authentication, sometimes any two factors (e.g., a password and a smart card) can be compromised, which increases the advantage of compromising the third factor (e.g., identity, biometrics, etc.). To overcome these security flaws, we propose a multi-factor authentication scheme in this potential application. The proposed scheme is a four-factor authentication protocol, which uses a user's password, real-identity, personal biometrics and mobile device as four factors. If any two factors are compromised, the adversary cannot compromise the other two factors simultaneously [22].

Designing a four-factor authentication protocol for air-smart vehicular networks may introduce specific challenges compared to a three-factor authentication system. Some challenges associated with four-factor authentication in the context of the air-smart vehicular networks are: 1) "limited payload and processing power" of the deployed drones, 2) "managing cryptographic keys for four-factor authentication is critical for security", 3) "privacy concerns" where the drones may capture sensitive information during their operations. Thus, introducing additional authentication factors, especially those involving biometrics, may raise privacy concerns, and 4) "scalability" when the number of drones in the network increases, the authentication protocol must scale to handle a growing network. However, in the proposed scheme, we handle user authentication with the deployed drones with the help of the GS in the network. Only the user's four factors (password, real-identity, personal biometrics and mobile device) are used in the user authentication process, whereas the deployed drones do not use any factors and they only use their pre-loaded secret credentials for authentication with the authorized users. As a result, "limited payload and processing power" of the deployed drones do not affect the scalability of the proposed scheme irrespective of the number of deployed drones. Moreover, introducing an additional authentication factor as compared to three-factors in authentication process does not increase the computational burden during the login, and authentication and key agreement phases of the proposed scheme (see Sections III-C and III-D). On the other side, our multi-factor authentication scheme achieves such a level of security. In addition, making secure storage is a major concern due to the risk of single server failure. A possible solution to this problem is to adopt blockchain technology, which provides (cryptographically) secure, decentralization, and transparency, and once the data is stored on the blockchain, an unauthorized entity cannot modify it due to the immutability it provides. To provide a secure communication as well as secure storage

in ASVN, a new blockchain-based authentication and key agreement system has been proposed.

C. Research Contributions

The article makes the following major contributions:

• The proposed protocol offers a mechanism for secure mutual authentication utilizing multi-factors, such as a user's password, real-identity, personal biometrics and mobile device as four factors, to establish the session keys among the communicating parties (here, drones and users). In the proposed scheme, a major advantage is that limited payload and processing power of the deployed drones do not affect the scalability of the proposed scheme irrespective of the number of deployed drones. At the same time, a drone establishes a session key with the ground station (GS) for secret sharing. Here, a user U_i can not only access the real time data from a deployed drone DR_j securely, but the GS also aggregates the same data securely with a session key with each other for storing into a blockchain center.

• A detailed formal security verification using broadly-accepted Scyther automated software validation tool and the informal (non-mathematical) security analysis show the proposed scheme's robustness against various potential attacks needed for air-smart vehicular networks.

• A real-time testbed experiment for various cryptographic primitives as well as for the proposed authentication protocol is used to show the feasibility, the testbed uses Raspberry Pi 4 (model B) devices with cryptographic standard library (cryptography 37.0.2). Moreover, a blockchain simulation on the real data (image type) is also presented for this protocol which identifies the uniqueness of the scheme.

In this article, our primary focus is to tackle the security concerns surrounding the communication channels between drones and users.

D. Paper Outline

In the next section, we elaborate on the literature on existing authentication and key agreement schemes for the drone applications. Section III discusses different phases of the proposed scheme. While Section IV presents the security analysis, Section IV-B also shows the formal security verification under the broadly accepted Scyther tool and Section V describes the real-time testbed experiments of the proposed scheme. Section VI presents a detailed comparative study of the proposed scheme with existing schemes to show its scalability and feasibility as compared to other competing schemes. Finally, Section VII draws some important concluding remarks on the proposed scheme.

II. Related Work

In 2018, Ali and Pal [6] proposed a three-factor user authentication protocol using the services of a remote server in a multi-server environment. The authors considered three network entities: a) users, b) registration center (RC), and c) servers, which communicate via wireless media. In their scheme, a user and a server register with the RC, and the user first proves his/her legitimacy to the server with a smart card and establishes a session key prior to accessing the services. Wu *et al.* [29] and Luo *et al.* [30] demonstrated that the security solution of Ali and Pal [6] suffers from some serious security issues, such as 1) user and server impersonation attacks, 2) smart card stolen attacks, 3) replay attacks, 4) user anonymity leakage, 5) insider attacks, and 6) lack of forward secrecy. In 2019, Adavoudi-Jolfaei *et al.* [7] designed a user authentication and access control mechanism based on three factors for wireless sensor network (WSN). In their scheme, users have the provision to access the sensor's data through a GW with a session key. Ryu *et al.* [31] emphasized that the scheme in [7] fails to protect against stolen smart card attacks, insider attacks, user impersonation attacks, and ESL attacks under the Canetti and Krawczyk (CK)-threat model [19]. In 2019, Ma *et al.* [8] proposed a user authentication protocol for vehicular ad-hoc networks. In their scheme, vehicle user, fog server, and cloud server mutually authenticate themselves and establish a session key for sharing information. Whereas, Eftekhari *et al.* [32] reveal that the security solution provided by Ma *et al.* [8] is also vulnerable to insider, known session specific temporary information, and stolen smart card attacks.

In 2019, Srinivas *et al.* [33] designed a scheme to provide secure and lightweight authentication in an IoD network for users and drones, while also protecting the privacy of drones and users by hiding their original identities during communications. In 2019, Li *et al.* [34] proposed a mutual-healing group key distribution mechanism for UAV networks using blockchain. In their scheme, a network entity, a ground control station, is treated as a server, responsible for constructing a blockchain to record the distribution of group keys. Additionally, this blockchain is used to manage the membership certificates of UAVs with a dynamic list. In 2020, Ali *et al.* [9] offered an improved three-factor smart-card based user authentication and key management scheme in multi-server environments. In their scheme, user accesses the data from the server through the RC by establishing a session key, whereas Yu and Park [27] showed that their scheme is not resistant against the man-in-the-middle (MiTM), smart card stolen, and masquerade attacks. Furthermore, their scheme is also vulnerable to the ESL attack under the CK-adversary model, and fails to offer mutual authentication.

In 2020, Li *et al.* [10] also developed a three-factor authentication protocol for wireless medical sensor networks (WMSNs) based on *honey_list* techniques. In their scheme, user can legally access the data after the three-party mutual authentication with the GW and the sensor node. In their protocol, the session key is constructed based on a random number and public information. As a result, their scheme is vulnerable to the ESL attack under the CK-adversary model, insider, replay, and MiTM attacks. Saleem *et al.* [44] highlighted that the scheme in [10] does not protect against a sensor node impersonation attack and it makes no provision for the anonymity of the user. In 2020, Ever [11] suggested an authentication scheme for UAVs which can act as mobile-sinks in an IoD environment. Their network entities communicate with a session key based on bilinear pairing, which requires a large amount of computational power. Furthermore, their

TABLE I
CRYPTOGRAPHIC METHODS, ADVANTAGES AND LIMITATIONS OF EXISTING SCHEMES IN DRONE ENVIRONMENTS

Scheme	Year	Cryptographic methods	Advantages	Drawbacks/Limitations
Ali and Pal [6]	2018	* Cryptographic hash function * ECC point multiplication * ECC point addition	* User authentication * Key agreement	* User and server impersonation attacks * Fails to resist smart card stolen, replay, and insider attacks * User anonymity leakage and lack of forward secrecy.
Adavoudi-Jolfaei *et al.* [7]	2019	* Cryptographic hash function	* User authentication * Key agreement	* Stolen smart card attacks * Insider and user impersonation attacks * ESL attacks under CK-threat model
Ma *et al.* [8]	2019	* Cryptographic hash function * ECC point multiplication	* User authentication	*Insider and stolen smart card attacks * Leakage known session specific information
Ali *et al.* [9]	2020	* Cryptographic hash function * Symmetric key encryption/decryption	* User authentication * Key management	* Man-in-the-middle (MiTM) * Smart card stolen and masquerade attacks found * ESL attack under the CK-adversary model * Fails to offer mutual authentication
Li *et al.* [10]	2020	* ECC point multiplication * Cryptographic hash function	* User authentication * Key negotiation * *Honey_list* techniques	* ESL attack under the CK-adversary model * Insider, replay, and MiTM attacks * Sensor node impersonation attack * Does not support anonymity of the user
Ever [11]	2020	* Bilinear pairing * Cryptographic hash function * Map-to-point function * ECC point multiplication	* Authentication * Key management	* Huge computational costs * ESL attack under the CK-adversary model * Does not support anonymity and blockchain * Does not support dynamic node addition
Shin and Kwon [35]	2020	* Cryptographic hash function * ECC point multiplication	* Authentication * Key agreement	* ESL attack under the CK-adversary model * Does not support dynamic node addition * Blockchain solution does not support
Tanveer *et al.* [36]	2020	* Cryptographic hash function * AEGIS encryption/decryption	* Authentication * Key agreement	* Does not support blockchain security
Jan *et al.* [28]	2021	* Cryptographic hash function * Bit-wise XOR	* Authentication * Key agreement	* Vulnerable to user anonymity and untraceability * Cannot resist privilege-insider attack * More communication costs
Zhang *et al.* [37]	2021	* Cryptographic hash function * Symmetric key encryption/decryption, * Hash-based message authentication code	* Authentication * Key establishment	* Cannot resist replay attack * Fails to provide user anonymity
Chang *et al.* [38]	2021	* ECC, * Fuzzy extractor function, * One way hash function	* Authentication * Key agreement	* ESL attacks under the CK-adversary model, * Does not resist replay and privileged insider attacks
Javed *et al.* [39]	2022	* Hyper-elliptic curve cryptography * Hash function	* Authentication * Key establishment	* Fails to drone revocation * Cannot resist replay attack
Feng *et al.* [40]	2022	* Bilinear pairing, * Symmetric key encryption/decryption, * One-way hash function	* Mutual authentication * Session key agreement	* Vulnerable to ESL attack under the CK-adversary model
Tan *et al.* [41]	2022	* Simple hash function * ECC	* Authentication * Key agreement	* Fails to support forward and backward secrecy * Does not support dynamic drone addition
Ryu *et al.* [42]	2022	* ECC * One-way hash function	* Authentication * Key agreement	* Vulnerable to message substitution and privileged insider attacks, * Found MITM, impersonation, and session key disclosure attacks
Mirsaraei *et al.* [43]	2022	* ECC * One-way hash function	* Authentication * Key agreement	* Vulnerable to ESL attacks under the CK-adversary model
Zhang *et al.* [26]	2023	* Symmetric key encryption/decryption, * One-way hash function	* Authentication * Key agreement	* Vulnerable to replay attack, * Does not support blockchain solution * Does not support dynamic drone addition
Irshad *et al.* [25]	2023	* ECC * One-way hash function	* Mutual authentication * Key agreement	* High computation and communication costs * Does not support blockchain solution * Does not support dynamic drone addition

scheme is not secure against ESL attack under the CK-adversary model and does not support anonymity, blockchain, and dynamic node addition functionality features.

In 2020, Shin and Kwon [35] presented an authentication and key agreement scheme for WSN in 5G-enabled IoT environment. In their scheme, U and GW mutually authenticate each other by relying on the authentication server (AAS) and then construct a common session key via the AAS. After that, based on the access privilege, U can access the real-time sensory information from the WSN with the help of an established session key. However, ESL attack under the CK-adversary model is possible in their scheme as the session key construction based on less secure parameters. In 2020, Tanveer *et al.* [36] also presented an authenticated key exchange mechanism for an IoD environment, where a mobile user validates its authenticity and establishes a session key with a drone through a management server that stores all the credentials. However, their scheme does not support blockchain security. In 2020, Wang *et al.* [45] proposed an authentication protocol for wireless sensor networks (WSNs) to ensure real-time data access while maintaining a high security level. Their protocol uses a combination of multi-factors, namely, password, smart card, identity, and biometric, for establishing a session key between a user and an accessed sensor node for real-time data access in WSNs. In 2020, Sun *et al.* [46] proposed a secure storage and access scheme based on a ciphertext policy attribute-based encryption system using blockchain for e-healthcare applications. They built an attribute-based encryption mechanism to securely store electronic medical records in an InterPlanetary File System (IPFS) storage environment, combined with blockchain technology.

In 2021, Jan *et al.* [28] proposed a key agreement protocol for IoD environments, where the users can access the drone's data from a ground station through authentication. Their proposal fails to resist various attacks, such as insider attacks, untraceability, and user anonymity, as well as the large communication overhead required to establish a session key. In 2021, Zhang *et al.* [37] proposed a "three-factor user authentication protocol" for IoD based on FourQ curves with the Boyko-Peinado-Venkatesan (BPV) pre-calculation techniques to ensure confidentiality of data communication. However, their protocol has high computation and communication overheads, cannot resist replay attack and fails to provide user anonymity as pointed out by Park *et al.* [47]. In 2021, Chang *et al.* [38] proposed a three-factor authentication system for IoT applications, wherein a user establishes a session key with the sensor nodes through a gateway node. In their scheme, the session key is constructed based on real identities and a random number, making it vulnerable to ESL attacks under the CK-adversary model. Furthermore,

their scheme does not provide resistance against replay attacks and privileged insider attacks. In 2021, Hussain *et al.* [48] proposed an authentication scheme based on ECC for securing communication between drones and a user. Their protocol uses three factors (a password, biometrics, and a mobile device) to secure the user-drone communication.

In 2022, Javed *et al.* [39] proposed a "hyper-elliptic curve cryptography (HECC)-based authentication scheme" in IoD applications by utilizing a blockchain as certificate authority and various transactions being considered certificates. Their proposed protocol cannot protect against replay attacks and fails to achieve drone revocation. In 2022, Feng *et al.* [40] proposed a "cross-domain authentication and key agreement protocol" for drone environments based on the consortium blockchain. In their proposed model, drones establish the session keys among themselves after smart contract-based mutual authentication, but an established session key is not secure because of vulnerability to an ESL attack under the CK-adversary model. In 2022, Tan *et al.* [41] proposed an "authentication approach for industrial UAVs" based on blockchain, where the established session key does not maintain forward and backward secrecy, and their scheme does not support dynamic drone addition.

In 2022, Ryu *et al.* [42] proposed a three-factor authentication protocol for a telecare medical information system, wherein the user (patient) establishes a session key with the telecare server through a registration center. However, Kumar *et al.* [49] have pointed out that their proposed scheme is vulnerable to various attacks, including message substitution, privileged insider, MITM (Man-in-the-Middle), impersonation, and session key disclosure attacks. In 2022, Mirsaraei *et al.* [43] proposed a three-factor authentication scheme for an IoT application in which a user/device authenticates and establishes a session key with a server. In their scheme, the session is constructed using public information and a random number. As a result, this scheme is vulnerable to ESL attacks under the CK-adversary model. In 2022, two separate proposals for secure storage in IoT applications using blockchain technology were put forth by Ullah *et al.* [50] and Bataineh *et al.* [51]. In [50], the authors employed an attribute-based access control (A-BAC) policy by utilizing the Ethereum blockchain as an auditable access control layer. In contrast, in [51], the authors adopted the Ethereum Blockchain infrastructure in a rich-thin client IoT approach. In this model, limited resource devices are thin clients, while higher resource devices are designated as rich clients. Both clients can access the blockchain, but only rich clients can execute the mining process.

In 2023, Irshad *et al.* [25] introduced a three-factor authenticated key exchange mechanism named SUSIC for SDN-based IIoT environments. In their scheme, users and smart devices authenticate with the controller node to establish a session key for data sharing. However, their scheme incurs significant computation and communication costs, making it impractical for real-world IoT applications, especially in resource-constrained devices. Additionally, their scheme lacks support for dynamic node addition and does not incorporate blockchain solutions. In 2023, Zhang *et al.* [26] proposed a three-factor authentication and key agreement scheme for an IoT-enabled e-healthcare application. In their scheme, a user, who is a patient with a smart card, authenticates to the server and establishes a session key for data sharing. However, their proposed scheme is vulnerable to replay attacks and lacks support for blockchain solutions and dynamic drone addition.

Various cryptographic methods, as well as the advantages and limitations of existing competing key agreement/authentication schemes in drone environments are summarized in Table I.

TABLE II
NOTATIONS AND THEIR MEANINGS

Notation	Significance
$E_q(a,b)$	A non-singular elliptic curve of the form: "$y^2 = x^3 + ax + b \pmod q$ with $4a^3 + 27b^2 \neq 0 \pmod q$"
G	A base point in $E_q(a,b)$ whose order is η as big as q
ID_i, Pw_i, B_i	User (U_i)'s identity, password, and bio-metric, respectively
HID, $THID$	Hashed and temporal hashed identity of U_i
LC	U_i's long-term secret of U_i
PSB	Pseudo bio-metric value of U_i
GS	Ground server (or authentication server)
MK_g	Master secret key of GS
r_g, $Pub_g = r_g \cdot G$	Private and public key of GS
DR_j	j^{th} drone
ID_j, r_j, ck_j	Identity, random secret, and certificate key of DR_j
$Cert_j$	Certificate of DR_j
TS_i	"Current timestamps for $i = 1, 2, 3, 4$"
ΔT	"Maximum message transmission delay"
$h(\cdot)$	"Collision-resistant cryptographic one-way hash function"
$SK_1 (= SK_1')$, SKV_1	Session key between GS and DR_j and session key verifier
$SK_2 (= SK_2')$, SKV_2	Session key between U_i and DR_j and session key verifier

III. THE PROPOSED PROTOCOL

This section explains in detail all the required phases needed in the proposed scheme. The list of the significant symbols and their descriptions are provided in Table II. The designed authentication and key agreement protocol is based on multi-factors, namely a registered user's real-identity (ID), password (Pw), biometric template (B), and mobile device (MD) as four factors. If an adversary $\mathcal{A}$ can guess any two factors out of four factors (see the adversary model in Section I-A2), he/she cannot derive other factors in polynomial time.

A. System Initialization Phase

The ground server, GS, which is considered a fully trusted authority, has the responsibility of setting up the system parameters as follows.

- A finite field $GF(q)$ is chosen at random, where q is a large odd prime (e.g., q of at least 160 bits is required to make computational elliptic curve problems like "elliptic curve discrete logarithm problem (ECDLP)" and "elliptic curve decisional Diffie-Hellman problem (ECDDHP)" intractable).
- GS selects a "non-singular elliptic curve $E_q(a,b)$ of the form $E_q(a,b) : y^2 = x^3 + ax + b \pmod q$ with the chosen constants $a, b \in Z_q$ such that $4a^3 + 27b^2 \neq 0 \pmod q$". GS then picks its own private key $r_g \in Z_q^* = \{1, 2, \cdots, q-1\}$ and computes corresponding public key as $Pub_g = r_g \cdot G$, which is an elliptic curve point (scalar) multiplication, where

$r_g \cdot G = G + G + \cdots + G$ (r_g times) and G in $E_q(a, b)$ is a base point whose order is η as big as q.

• GS also chooses its own master secret key, MK_g. Finally, GS publishes the public credentials as $\{Pub_g,\ E_q(a, b),\ G\}$ while keeping the private key r_g secret.

B. Registration Phase

The fully trusted registration authority with the ground server GS registers each drone (DR_j) and user (U_i).

1) User Registration Phase: A user U_i can register to GS via a secure channel using a mobile device, say MD_i with the following steps:

Step URP1: U_i selects an unique identity ID_i, password Pw_i, and imprints personal biometric template B_i at the sensor of some specific terminal (mobile device). Next, U_i executes a "fuzzy extractor probabilistic generation function" $Gen(\cdot)$ [52] to generate a bio-metric secret key $\sigma_i \in \{0, 1\}^{l_b}$ of length l_b bits and a public reproduction parameter τ_i corresponding to the bio-metric input B_i, that is, $Gen(B_i) = (\sigma_i, \tau_i)$. Furthermore, U_i computes a hashed identity $HID = h(ID_i)$, selects a random secret $r_i \in Z_q^*$, and computes pseudo bio-metric value as $PSB = h(ID_i\ ||\sigma_i\ ||r_i)$. U_i then sends the registration request $\{HID, PSB\}$ to GS via secure channel (GS is also considered as the authentication server).

Step URP2: Using a master secret key MK_g and chosen temporary hashed identity $THID$, GS computes $RPSB = h(PSB\ ||r_g)$, $PHID = h(HID\ ||MK_g)$, and $x_1 = h(RPSB\ ||MK_g) \oplus PHID$. Next, GS sends the registration information $\{x_1,\ PHID,\ THID,\ RPSB\}$ to U_i via secure channel. GS then stores the U_i's registration information $\{PHID,\ THID,\ RPSB\}$ into its secure database. In addition, the server GS is also protected by a physical locking system as in [53].

Step URP3: After receiving the information $\{x_1,\ PHID,\ THID,\ RPSB\}$, U_i computes $TC = h(Pw_i\ ||\sigma_i\ ||ID_i)$, $x_2 = r_i \oplus TC$, $x_3 = x_1 \oplus h(r_i\ ||Pw_i\ ||\sigma_i)$, $PHID^* = PHID \oplus h(r_i\ ||\sigma_i\ ||TC)$, $RPSB^* = RPSB \oplus h(\sigma_i\ ||ID_i\ ||r_i)$, and $x_4 = h(\sigma_i\ ||Pw_i\ ||ID_i\ ||r_i)$. After successful registration, U_i saves the information $\{(THID,\ PHID^*),\ RPSB^*,\ HID,\ x_2,\ x_3,\ x_4,\ E_q(a, b),\ G\}$ in MD_i's memory.

2) Drone Registration Phase: A drone DR_j can be registered by the trusted server GS with the following steps:

Step DRP1: For DR_j, GS chooses a unique and distinct identity ID_j, a random secret r_j, and a certificate key ck_j. Then, GS computes $Pub_j = r_j \cdot G$, $Pub_{ck_j} = ck_j \cdot G$, as well as a certificate $Cert_j = ck_j + h(Pub_g\ ||ID_j\ ||Pub_j) * r_g \pmod q$ and a long-term secret $LC = h(r_j\ ||ID_j\ ||ck_j\ ||MK_g\ ||RTS_j)$, where RTS_j is the registration timestamp.

Step DRP2: After successfully registering DR_j, GS loads the registration credentials $\{(r_j, Pub_j),\ Pub_g,\ Pub_{ck_j},\ LC,\ Cert_j, E_q(a, b), G\}$ into DR_j's memory, and deletes all secret keys $\{r_j,\ ck_j,\ LC\}$ form its database. After that, DR_j can be deployed to a flying zone.

C. User Login Phase

During this phase, a registered user, U_i, with a mobile device (for example, MD_i), validates himself/herself by associating MD_i with the registered credential received from the ground server (GS). Once it is completed, U_i will have access to MD_i and will be able to negotiate a session key with the smart device MD_i for future communication between U_i and a drone DR_j using the associated GS.

Step LI: To login, U_i enters the identity ID_i^*, password Pw_i^*, and imprints bio-metric B_i^* at the sensor of the mobile device MD_i. Next, MD_i computes $HID^* = h(ID_i^*)$, and checks $HID^* = HID$? If it matches, MD_i derives the bio-metric secret key σ_i^* corresponding to the inputs B_i^* and τ_i using the widely adopted "fuzzy extractor deterministic reproduction function $Rep(\cdot)$" [52] as $\sigma_i^* = Rep(B_i^*, \tau_i)$ with the condition that $Hd(B_i^*, B_i) \leq et$, where et is the pre-defined error tolerance threshold value and $Hd(\cdot)$ denotes the "Hamming distance between registered bio-metric template B_i and current bio-metric template B_i^*".

Step L2: MD_i then computes $TC^* = h(Pw_i^*\ ||\sigma_i^*\ ||ID_i^*)$, $r_i^* = x_2 \oplus TC^*$, $x_4^* = h(\sigma_i^*\ ||Pw_i^*\ ||ID_i^*\ ||r_i^*)$, and checks $x_4^* = x_4$? If this condition holds, the login request to MD_i is accepted. This means that $ID_i^* = ID_i$, $Pw_i^* = Pw_i$, $r_i^* = r_i$, and $B_i^* = B_i$ ($\sigma_i^* = \sigma_i$). Following this, U_i executes the authentication and key agreement protocols to securely share information with the drone DR_j via GS by establishing a session key.

D. Mutual Authentication and Key Establishment Phase

In this phase a session key is establish followed by a mutual authentication via GS between a user U_i and a drone DR_j. At the end of this process, two session keys are established: 1) first one (SK_1) is between GS and the drone DR_j and 2) second one (SK_2) is between the user U_i and the drone DR_j. The following stages are performed:

Step MAKE1: U_i initiates the process for establishing a session key with a DR_j. To do so, U_i derives $x_1 = x_3 \oplus h(r_i\ ||Pw_i\ ||\sigma_i)$, $PHID = PHID^* \oplus h(r_i\ ||\sigma_i\ ||TC)$, and $RPSB = RPSB^* \oplus h(\sigma_i\ ||ID_i\ ||r_i)$. U_i selects a random nonce $r_1 \in Z_q^*$, a current timestamp TS_1, and computes $V_1 = r_1 \cdot G$, $P = r_1 \cdot Pub_g = (P_x, P_y)$. After that, U_i calculates $V_2 = x_1 \oplus h(RPSB\ ||P_y\ ||TS_1) \oplus PHID$ and $V_3 = THID \oplus h(V_1\ ||V_2\ ||P_x\ ||TS_1)$. Following these, U_i creates an authentication request message, $Msg_1 = \{V_3,\ V_2,\ V_1,\ TS_1\}$ and sends it to the associated ground server GS, which serves as an authentication server, via an open channel.

Step MAKE2: After receiving the message Msg_1 at a timestamp TS_1^* from U_i, GS verifies the authenticity of U_i as well as Msg_1. GS first checks freshness of Msg_1 by the condition $|TS_1^* - TS_1| < \Delta T$ where ΔT is the "maximum transmission delay". If this holds, GS computes $P' = r_g \cdot V_1 = (P_x', P_y')$ and derives $THID = V_3 \oplus h(V_1\ ||V_2\ ||P_x'\ ||TS_1)$. Next, GS checks whether $THID$ exists or not, and if it does, GS then fetches $RPSB$ and $PHID$, and derives $x_1^* = h(RPSB\ ||MK_g) \oplus PHID$ and $V_2^* = x_1^* \oplus h(RPSB\ ||P_y'\ ||TS_1) \oplus PHID$. Next, GS verifies if $V_2^* = V_2$? If this condition is verified, it means that U_i successfully authenticated to GS. Further, GS picks a random nonce $r_2 \in Z_q^*$ and current timestamp TS_2, and computes $Q = r_g \cdot Pub_j = (Q_x, Q_y)$, and $T_1 = h(x_1^*\ ||r_2\ ||TS_1\ ||P_y'\ ||V_1) \oplus h(Q_x\ ||ID_j\ ||TS_2)$, a session key for sharing

information with DR_j as $SK_1 = h(h(x_1^* \,||r_2\, ||TS_1\, ||P'_y\, ||V_1)\, ||Q_y\, ||TS_2)$, and the session key verifier as $SKV_1 = h(SK_1\, ||T_1\, ||TS_2)$. GS creates an authentication request message $Msg_2 = \{V_1,\, TS_1,\, T_1,\, SKV_1,\, TS_2\}$ and sends it to the drone DR_j having the identity ID_j, via an open channel.

Step MAKE3: Once the message Msg_2 is received by DR_j from GS at a timestamp TS_2^*, DR_j checks its freshness by $|TS_2^* - TS_2| < \Delta T$. If it is so, DR_j computes $Q' = r_j \cdot Pub_g = (Q'_x, Q'_y)$, and derives $h(x_1^* \,||r_2\, ||TS_1\, ||P'_y\, ||V_1) = T_1 \oplus h(Q'_x\, ||ID_j\, ||TS_2)$. Now, DR_j computes the session key for sharing information with GS by $SK'_1 = h(h(x_1^* \,||r_2\, ||TS_1\, ||P'_y\, ||V_1)\, ||Q'_y\, ||TS_2)$, and verifies if $SKV_1 = h(SK'_1\, ||T_1\, ||TS_2)$? If it is verified successfully, both establish the same session key $SK_1 (= SK'_1)$. This session key is utilized for secure communication between GS and DR_j. Next, DR_j calculates $V_4 = r_j \cdot V_1 = (V_{4x}, V_{4y})$, and selects a random nonce $r_d \in Z_q^*$ and current timestamp TS_3 to compute $T_2 = h(LC\, ||r_d\, ||ID_j\, ||TS_2\, ||TS_3) \oplus V_{4x}$ and another session secret key $SK_2 = h(h(x_1^* \,||r_2\, ||TS_1\, ||P'_y\, ||V_1)\, ||h(LC\, ||r_d\, ||ID_j\, ||TS_2\, ||TS_3)\, ||V_{4y})$ to be shared with the user U_i. DR_j covers up its own certificate $Cert_j$ by $Cert_j^* = Cert_j \oplus h(Q'_y\, ||SK'_1)$, and computes a session key verifier as $SKV_2 = h(SK_2\, ||TS_3)$ and $ID_j^* = ID_j \oplus h(Q'_x\, ||TS_3)$. After that, DR_j constructs a message as $Msg_3 = \{Cert_j^*,\, SKV_2,\, T_2,\, ID_j^*,\, TS_3\}$ and sends it to GS as a response message via open channel.

Step MAKE4: After receiving Msg_3 from DR_j at timestamp TS_3^*, GS checks its novelty by verifying the condition $|TS_3^* - TS_3| < \Delta T$? If it is verified, GS derives $ID_j = ID_j^* \oplus h(Q_x\, ||TS_3)$, and checks the existence of ID_j. GS derives $Cert_j = Cert_j^* \oplus h(Q_y\, ||SK_1)$, and verifies the certificate by $Cert_j \cdot G = Pub_{ck_j} + h(Pub_g\, ||ID_j\, ||Pub_j) \cdot Pub_g$. If this condition is met, DR_j is considered an authenticated drone to send the sensing data. Following this, GS picks a current timestamp TS_4, and computes $T'_1 = h(x_1^* \,||r_2\, ||TS_1\, ||P'_y\, ||V_1) \oplus h(P'_x\, ||TS_4)$. After this, GS generates a message $Msg_4 = \{T'_1,\, SKV_2,\, T_2,\, TS_2,\, TS_3,\, TS_4\}$ and sends it to U_i as a response message to the request via open channel.

Step MAKE5: Suppose U_i receives the message Msg_4 at a timestamp TS_4^* from GS. U_i verifies the timestamp by $|TS_4^* - TS_4| < \Delta T$. If it is so, U_i derives $h(x_1\, ||r_2\, ||TS_1\, ||P'_y\, ||V_1) = T'_1 \oplus h(P_x\, ||TS_4)$, computes $V'_4 = r_1 \cdot Pub_j = (V'_{4x}, V'_{4y})$, $h(LC\, ||r_d\, ||ID_j\, ||TS_2\, ||TS_3) = T_2 \oplus V'_{4x}$ and the session key $SK'_2 = h(h(x_1\, ||r_2\, ||TS_1\, ||P_y\, ||V_1)\, ||h(LC\, ||r_d\, ||ID_j\, ||TS_2\, ||TS_3)\, ||V'_{4y})$ shared with DR_j. U_i verifies the session key with the received session key verifier by $SKV_2 = h(SK'_2\, ||TS_3)$. If this is correct, U_i believes that U_i and DR_j have the same session secret key $SK'_2 (= SK_2)$. Finally, they save this session key for secret communications in future.

A summary of the above authentication and key establishment phase is shown in Fig. 2.

E. Password and Bio-metric Updation Phase

Biometric data (such as fingerprints, facial features, iris patterns, voiceprints, and DNA) refers to a unique physical or behavioral characteristic that is used to identify individuals. There may be several reasons why someone might need to change their biometric data. For example, if an individual's biometric data has been compromised or stolen, they may need to change their biometric data to prevent identity theft or other forms of fraud. Another reason is medical, because certain medical conditions or procedures can affect an individual's biometric data, such as cosmetic surgery that may alter their facial features or injuries that affect their fingerprints. In such cases, it may be necessary to update their biometric data for identification purposes. Similarly, passwords need to be updated to avoid a security breach [54]. Although the biometrics do not change frequently over time as compared to the passwords of a user, it is upto a user whether to update the personal biometric every time when a password is updated. In that scenario, the user may keep his or her old personal biometrics.

This phase allows a registered user U_i to replace his or her old password and/or bio-metrics with a new password and bio-metrics. Once U_i has successfully completed a local login to MD_i (as described in Section III-C), the old password Pw_i^o and bio-metric template B_i^o are considered as the registered U_i's authentic credentials. After a successful login, U_i may replace the old password Pw_i^o and bio-metric template B_i^o with a new password Pw_i^n and bio-metric B_i^n. To do so, U_i requires to execute the following steps:

Step PBUP1: U_i enters the identity ID_i, new password Pw_i^n, and new bio-metric B_i^n. U_i, and generates a new bio-metric secret key σ_i^n and a new public reproduction parameter τ_i^n corresponding to the input B_i^n, that is, $Gen(B_i^n) = (\sigma_i^n, \tau_i^n)$. Next, U_i computes a new pseudo bio-metric value as $PSB^n = h(ID_i\, ||\sigma_i^n\, ||r_i)$. Following that, U_i sends the password and bio-metric update request $\{HID,\, THID,\, PSB^o,\, PSB^n\}$ to GS via secure channel.

Step PBUP2: GS computes $RPSB^o = h(PSB^o\, ||r_g)$ and $PHID^o = h(HID\, ||MK_g)$ corresponding to $THID$. After that, GS verifies the conditions: $RPSB^o = RPSB$ and $PHID^o = PHID$ with its stored information $\{PHID,\, THID,\, RPSB\}$. Once the verification is completed successfully, GS accepts the update request as a legitimate request.

Step PBUP3: GS computes $RPSB^n = h(PSB^n\, ||r_g)$, $PHID^n = h(HID\, ||MK_g)$, and $x_1^n = h(RPSB^n\, ||MK_g) \oplus PHID^n$. GS then sends new updated credentials $\{x_1^n,\, PHID^n,\, THID,\, RPSB^n\}$ to U_i via secure channel. GS stores U_i's updated information $\{PHID^n, THID, RPSB^n\}$ into its secure memory.

Step PBUP4: Once the information is received, U_i computes $TC^n = h(Pw_i^n\, ||\sigma_i^n\, ||ID_i)$, $x_2^n = r_i \oplus TC^n$, $x_3^n = x_1^n \oplus h(r_i\, ||Pw_i^n\, ||\sigma_i^n)$, $PHID^* = PHID^n \oplus h(r_i\, ||\sigma_i^n\, ||TC^n)$, $RPSB^* = RPSB^n \oplus h(\sigma_i^n\, ||ID_i\, ||r_i)$, and $x_4^n = h(\sigma_i^n\, ||Pw_i^n\, ||ID_i\, ||r_i)$. U_i finally saves the updated information $\{(THID,\, PHID^*),\, RPSB^*,\, HID,\, x_2^n,\, x_3^n,\, x_4^n,\, E_q(a,b),\, G\}$ in MD_i's memory.

F. Dynamic Drone Addition Phase

This phase allows the addition of a new drone DR_j^n into the drone network with the following steps:

Step NDAP1: For the new drone DR_j^n, GS picks a unique and distinct identity ID_j^n, a random secret r_j^n, and a certificate

User as U_i	GS	**Drone** as DR_j
Stored: $\{(THID, PHID^*), RPSB^*, HID, x_2, x_3, x_4, E_q(a,b), G\}$	**Stored**: $\{MK_g, E_q(a,b), G, PHID, THID, RPSB, r_g, ID_j\}$	**Stored**: $\{(r_j, Pub_j), Pub_g, Pub_{ck_j}, ID_j, LC, Cert_j, E_q(a,b), G\}$
Enter ID_i^*, Pw_i^*, B_i^* Compute $\sigma_i^* = Rep(B_i^*, \tau_i)$, $HID^* = h(ID_i^*)$. Check $HID^* = HID$? If yes, compute $TC^* = h(Pw_i^* \|\|\sigma_i^* \|\|ID_i^*)$, $r_i^* = x_2 \oplus TC^*$, $x_4^* = h(\sigma_i^* \|\|Pw_i^* \|\|ID_i^* \|\|r_i^*)$ Check $x_4^* = x_4$? If yes, login is accepted, i.e., $ID_i^* = ID_i$, $Pw_i^* = Pw_i$, $r_i^* = r_i$, and $B_i^* = B_i$ ($\sigma_i^* = \sigma_i$)		
Derive $x_1 = x_3 \oplus h(r_i \|\|Pw_i \|\|\sigma_i)$, $PHID = PHID^* \oplus h(r_i \|\|\sigma_i \|\|TC)$, and $RPSB = RPSB^* \oplus h(\sigma_i \|\|ID_i \|\|r_i)$ Select $r_1 \in Z_q^*$, timestamp TS_1, compute $V_1 = r_1 \cdot G$, $P = r_1 \cdot Pub_g = (P_x, P_y)$, $V_2 = x_1 \oplus h(RPSB \|\|P_y \|\|TS_1) \oplus PHID$, $V_3 = THID \oplus h(V_1 \|\|V_2 \|\|P_x \|\|TS_1)$ $\{V_3, V_2, V_1, TS_1\}$ $\rightarrow$	Check $\|TS_1^* - TS_1\| < \Delta T$? $P' = r_g \cdot V_1 = (P'_x, P'_y)$ $THID = V_3 \oplus h(V_1 \|\|V_2 \|\|P'_x \|\|TS_1)$ Check if $THID$ exist or not, if yes, fetch $RPSB$ and $PHID$, derive $x_1^* = h(RPSB \|\|MK_g) \oplus PHID$, $V_2^* = x_1^* \oplus h(RPSB \|\|P'_y \|\|TS_1) \oplus PHID$ Verify $V_2^* = V_2$? (user auth. succ.), select $r_2 \in Z_q^*$,	
	timestamp TS_2, compute $Q = r_g \cdot Pub_j = (Q_x, Q_y)$, $T_1 = h(x_1^* \|\|r_2 \|\|TS_1 \|\|P'_y \|\|V_1) \oplus h(Q_x \|\|ID_j \|\|TS_2)$ Session key $SK_1 = h(h(x_1^* \|\|r_2 \|\|TS_1 \|\|P'_y \|\|V_1)$ $\|\|Q_y \|\|TS_2)$, verifier $SKV_1 = h(SK_1 \|\|T_1 \|\|TS_2)$ $\{V_1, TS_1, T_1, SKV_1, TS_2\}$ $\rightarrow$	Check $\|TS_2^* - TS_2\| < \Delta T$?, compute $Q' = r_j \cdot Pub_g = (Q'_x, Q'_y)$, derive $h(x_1^* \|\|r_2$ $\|\|TS_1 \|\|P'_y \|\|V_1) = T_1 \oplus h(Q'_x \|\|ID_j \|\|TS_2)$, compute session key $SK'_1 = h(h(x_1^* \|\|r_2 \|\|TS_1$ $\|\|P'_y \|\|V_1) \|\|Q'_y \|\|TS_2)$, and verify $SKV_1 = h(SK'_1$ $\|\|T_1 \|\|TS_2)$?, if yes, both establish same session key, and compute $V_4 = r_j \cdot V_1 = (V_{4x}, V_{4y})$ Pick random $r_d \in Z_q^*$, timestamp TS_3, and compute $T_2 = h(LC \|\|r_d \|\|ID_j \|\|TS_2 \|\|TS_3) \oplus V_{4x}$, and another session key $SK_2 = h(h(x_1^* \|\|r_2 \|\|TS_1$
	Check $\|TS_3^* - TS_3\| < \Delta T$?, and derive $ID_j = ID_j^* \oplus h(Q_x \|\|TS_3)$, and check existence of ID_j Derive certificate $Cert_j = Cert_j^* \oplus h(Q_y \|\|SK_1)$, verify $Cert_j \cdot G = Pub_{ck_j} + h(Pub_g \|\|ID_j \|\|Pub_j) \cdot Pub_g$?	$\|\|P'_y \|\|V_1) \|\|h(LC \|\|r_d \|\|ID_j \|\|TS_2 \|\|TS_3) \|\|V_{4y})$, $Cert_j^* = Cert_j \oplus h(Q'_y \|\|SK'_1)$, session key verifier $SKV_2 = h(SK_2 \|\|TS_3)$, $ID_j^* = ID_j \oplus h(Q'_x \|\|TS_3)$ $\{Cert_j^*, SKV_2, T_2, ID_j^*, TS_3\}$ $\leftarrow$
Check $\|TS_4^* - TS_4\| < \Delta T$? If so, derive $h(x_1 \|\|r_2 \|\|TS_1 \|\|P'_y \|\|V_1) =$ $T'_1 \oplus h(P_x \|\|TS_4)$ Compute $V'_4 = r_1 \cdot Pub_j = (V'_{4x}, V'_{4y})$, $h(LC \|\|r_d \|\|ID_j \|\|TS_2 \|\|TS_3) = T_2 \oplus V'_{4x}$, session key $SK'_2 = h(h(x_1 \|\|r_2 \|\|TS_1 \|\|P_y$ $\|\|V_1) \|\|h(LC \|\|r_d \|\|ID_j \|\|TS_2 \|\|TS_3) \|\|V'_{4y})$, and verify $SKV_2 = h(SK'_2 \|\|TS_3)$? If valid, store $SK'_2 (= SK_2)$ with drone	Select current timestamp TS_4, and compute $T'_1 = h(x_1^* \|\|r_2 \|\|TS_1 \|\|P'_y \|\|V_1) \oplus h(P'_x \|\|TS_4)$ $\{T'_1, SKV_2, T_2, TS_2, TS_3, TS_4\}$ $\leftarrow$	
	Store $SK_1 (= SK'_1)$ with drone	Store $SK'_1 (= SK_1)$ with GS Store $SK_2 (= SK'_2)$ with user

Fig. 2. Summary of authentication and key agreement between U_i and DR_j.

key ck_j^n to calculate public keys as $Pub_j^n = r_j^n \cdot G$ and $Pub_{ck_j}^n = ck_j^n \cdot G$ corresponding to the secret keys r_j^n and ck_j^n, respectively. Next, GS generates a certificate $Cert_j^n = ck_j^n + h(Pub_g \|ID_j^n \|Pub_j^n) * r_g \pmod q$ and a secret value $LC^n = h(r_j^n \|ID_j^n \|ck_j^n \|MK_g \|RTS_j^n)$, where RTS_j^n is a registration timestamp for DR_j^n.

Step NDAP2: Once the registration is over, GS loads the registration credentials $\{(r_j^n, Pub_j^n),\ Pub_g,\ Pub_{ck_j}^n,\ LC^n,\ Cert_j^n,\ E_q(a,b),\ G\}$ into DR_j^n's memory, and deletes all secret keys $\{r_j^n,\ ck_j^n,\ LC^n\}$ form its database. After that, DR_j^n can be deployed to a flying zone.

G. Secure Data Aggregation by Ground Server

During the authentication and key agreement phase, the ground server GS and a drone DR_j establish a secure session key ($SK_1 = SK'_1$) for a session in the proposed scheme. This key is utilized to securely share information between GS and DR_j using the following steps:

Step 1. Suppose DR_j has the data, say $DATA_{DR_j}$. DR_j encrypts the data using the shared session key SK_1 as $EncData_{DR_j} = E_{SK_1}[Data_{DR_j} || TS_{current}]$, where $TS_{current}$ is current timestamp and $E_K[M]$ is the symmetric encryption (say, Advanced Encryption Standard (AES-128) algorithm) of the plaintext M using the key K. Next, DR_j sends the message $\{EncData_{DR_j}, TS_{current}\}$ to the GS.

Step 2. After receiving the message $\{EncData_{DR_j}, TS_{current}\}$ from DR_j, the GS checks the validity of the received timestamp $TS_{current}$. If it is valid, the GS uses the shared session key SK_1 to retrieve the data as $(Data_{DR_j} || TS'_{current}) = D_{SK_1}[EncData_{DR_j}]$, where $D_K[C]$ is the symmetric decryption (say, Advanced Encryption Standard (AES-128) algorithm) using the key K on the ciphertext C. The GS further checks if $TS'_{current} = TS_{current}$, and if it is valid, then the GS treats the data $DATA_{DR_j}$ as authentic.

In this way, with the help of the session key SK_1, the GS collects the sensing information from DR_j, and as a result, the sharing information cannot be revealed to the adversary, because the adversary does not know this key. Moreover, construction of a fake SK_1 is computationally infeasible task, because it requires the long-term secrets as well as short-term secrets of both GS and DR_j. As a result, after aggregation of the data, the transactions can be stored in blocks and then the blocks can be added into a blockchain for future analysis.

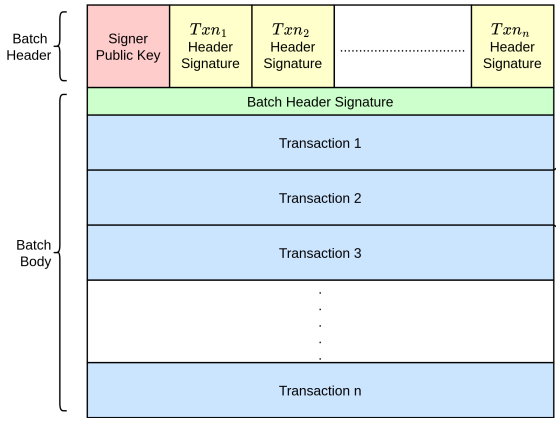


Fig. 3. Block structure.

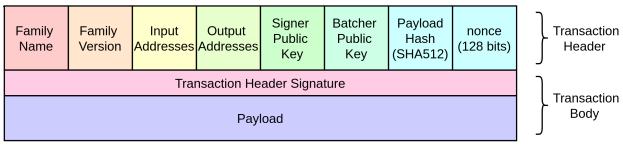


Fig. 4. Transaction format.

H. Block Construction and Addition by Cloud Servers

After receiving data from the associated GS in form of transactions, the CS in a P2P network constructs a block for addition to a blockchain. The blocks can be mined into the chain by executing a voting-based consensus algorithm, say using the "Practical Byzantine Fault Tolerance (PBFT)" method [55]. Note that in this work, the blockchain has been used as a service to store the data so that immutability, decentralization and transparency can be achieved as compared to storing the data simply in the semi-trusted cloud platform. The block structure and its associated transaction format as shown in Figs. 3 and 4, respectively, are used to store all the transactions.

Remark 1. *In the proposed scheme, two types of session keys are established. The session key SK_1 is used to secure communication between GS and a drone DR_i. The securely aggregated data from the drone DR_j using the session key SK_1 is used to form transactions, and later the transactions are used to create blocks and put into the blockchain. The session key SK_2 between a user U_i and an accessed drone DR_j is used to securely access the real-time data from DR_j by user U_i from any place and at any time.*

IV. Security Analysis

In this section, we provide the informal security analysis and formal security verification using the automated software validation tool to substantiate that the proposed scheme can resist several potential attacks against both passive and active adversaries.

A. Informal Security Analysis

1) Replay Attack: In the proposed scheme (BioKA-ASVN), all the communicated messages $Msg_1 = \{V_3,\ V_2,\ V_1,\ TS_1\}$, $Msg_2 = \{V_1,\ TS_1,\ ,SKV_1,\ T_1,\ TS_2\}$, $Msg_3 = \{Cert_j^*,\ SKV_2,\ T_2,\ ID_j^*,\ TS_3\}$ and $Msg_4 = \{T_1',\ SKV_2,\ T_2,\ TS_2,\ TS_3,\ TS_4\}$ contain fresh timestamps. Therefore, once an older message is retransmitted, it can be easily detected by verifying the attached timestamp. Hence, BioKA-ASVN resists the replay attack under the DY threat model.

2) Man-in-the-Middle(MiTM) Attack: In this attack, an attacker $\mathcal{A}$ may eavesdrop the user authentication request message $Msg_1 = \{V_3,\ V_2,\ V_1,\ TS_1\}$ under the DY threat model, and try to generate another valid message Msg_1^*. To do so, $\mathcal{A}$ can pick r_1^* and timestamp TS_1^*, and calculate $V_1^* = r_1^* \cdot G$, $P^* = r_1^* \cdot Pub_g = (P_x^*, P_y^*)$. After that, $\mathcal{A}$ attempts to derive $V_2^* = x_1 \oplus h(RPSB\ ||P_y^*\ ||TS_1^*) \oplus PHID$ and $V_3^* = THID \oplus h(V_1^*\ ||V_2^*\ ||P_x^*\ ||TS_1^*)$. Since $\mathcal{A}$ has no idea about the secret information $\{x_1,\ RPSB,\ PHID,\ THID\}$, $\mathcal{A}$ cannot derive valid V_2^* and V_3^*. Similarly, $\mathcal{A}$ also cannot generate other messages $\{Msg_2,\ Msg_3,\ Msg_4\}$. Therefore, BioKA-ASVN is secure against MiTM attacks.

3) Offline/Online Password Guessing Attack: Since the communicated messages $\{Msg_1,\ Msg_2,\ Msg_3,\ Msg_4\}$ do not contain any secret information (such as passwords, biometrics, and identities) in plaintext, $\mathcal{A}$ cannot obtain the password, biometric, and identities by employing an online guessing attack on the transmitted messages.

For offline password guessing attack, we consider that $\mathcal{A}$ has access to a registered user's mobile device MD_i and MD_i is not tamper resistant. $\mathcal{A}$ can pull out all the stored information $\{(THID,\ PHID^*),\ RPSB^*,\ HID,\ x_2,\ x_3,\ x_4,\ E_q(a,b),\ G\}$ from stolen/lost MD_i's memory. For correctly guessing the password, say Pw_g using $PHID^*$ and $RPSB^*$, $\mathcal{A}$ must have the correct identity ID_i and biometric secret σ_i. Since the biometric information cannot be guessed easily by $\mathcal{A}$, he/she cannot guess correct password offline. Hence, BioKA-ASVN resists password guessing attacks.

4) Stolen Mobile Device Attack: In this attack, we assume that the mobile device MD_i is stolen or found by $\mathcal{A}$. Then, $\mathcal{A}$ can access the stored data $\{(THID,\ PHID^*),\ RPSB^*,\ HID,\ x_2,\ x_3,\ x_4,\ E_q(a,b),\ G\}$ from MD_i's memory. As discussed above, $\mathcal{A}$ cannot guess the password, biometric, or identity of the user and these information are not stored as plaintext. Since the information is stored with the "collision resistance one-way cryptographic hash function", $\mathcal{A}$ cannot reveal secret credentials of user U_i. Thus, BioKA-ASVN is secure against the sensitive information leakage even if the legitimate U_i's MD_i is stolen.

5) Drone Impersonation Attack: In this attack, $\mathcal{A}$ attempts to pose as a genuine entity on behalf of a registered drone DR_j. After that, $\mathcal{A}$ tries to generate a legitimate message $Msg_3' = \{Cert_j^*,\ SKV_2,\ T_2,\ ID_j^*,\ TS_3\}$. To achieve this goal, $\mathcal{A}$ needs to select a random nonce r_d' and a timestamp TS_3', and try to compute $T_2' = h(LC\ ||r_d'\ ||ID_j\ ||TS_2\ ||TS_3') \oplus V_{4x}$. But, without knowledge of the secret information such as LC and ID_j, $\mathcal{A}$ cannot construct the message Msg_3'. Therefore, $\mathcal{A}$ cannot impersonate as a legitimate drone. This implies that BioKA-ASVN is protected against the drone impersonation attack.

6) GS Impersonation Attack: During the "authentication and key agreement phase", GS communicates the messages $Msg_2 = \{V_1,\ TS_1,\ T_1,\ SKV_1,\ TS_2\}$ and $Msg_4 = \{T_1',\ SKV_2, T_2, TS_2, TS_3, TS_4\}$ over the public channel, with the relevant entities. In this attack, $\mathcal{A}$ behaves like an authorized server, and attempts to construct a legal message Msg_2'. To continue this process, $\mathcal{A}$ selects a random nonce r_2' and a timestamp TS_2', and tries to compute $Q = r_g \cdot Pub_j = (Q_x, Q_y)$, $T_1' = h(x_1^*\ ||r_2'\ ||TS_1\ ||P_y'\ ||V_1) \oplus h(Q_x\ ||ID_j\ ||TS_2')$, and $SKV_1^* = h(SK_1\ ||T_1'\ ||TS_2')$. Since $\mathcal{A}$ has no knowledge of the secret information r_g, x_1^*, and ID_j, he/she cannot compute valid Q, T_1', and SKV_1^*. Therefore, $\mathcal{A}$ cannot impersonate GS, and as a result, he/she cannot communicate with the other entities on behalf GS on the fly. Thus, BioKA-ASVN is resistant to GS impersonation attack.

7) Privileged-Insider Attack: During the registration phase, neither a drone nor a user sends secret information to the

trusted registration authority (GS) in a plaintext format. Therefore, GS has no idea about the user's secret data. Instead, GS generates some secret information for a drone and loads these credentials into the drone's memory in offline mode only. After that, GS deletes all such secret credentials related to the drone from its memory. Hence, BioKA-ASVN is secure against the privileged-insider attack.

8) Physical Drone Capture Attack: Due to hostile environment, a drone can be physically captured by an adversary $\mathcal{A}$ using any of the techniques: 1) "shoot it down with a gun", 2) "use anti-drone drones", and 3) "use net-firing anti-drone guns" [20]. After that, $\mathcal{A}$ can extract the loaded information from the physically captured drone's memory using "power analysis attacks" [21]. It is worth noting that the stored credentials for a drone are distinct and unique for each registered drone. Therefore, if a drone is captured by $\mathcal{A}$, it does not reveal any secret credentials for other non-captured drones. Hence, BioKA-ASVN is resistant against physical drone capture attacks.

9) Ephemeral Secret Leakage (ESL) Attack: In the proposed BioKA-ASVN, the session key is established between U_i and DR_j as $SK_2 = h(h(x_1^* \,||r_2\, ||TS_1\, ||P_y'\, ||V_1)\, ||h(LC\, ||r_d\, ||ID_j\, ||TS_2\, ||TS_3)\, ||V_{4y})(= SK_2')$, where $x_1^*(= x_1) = h(RPSB\, ||MK_g) \oplus PHID$, $RPSB = h(PSB\, ||r_g)$, and $PHID = h(HID\, ||MK_g)$. It is important to note that the session key is constructed with the composition of the pair of long-term secret (for example, user password or biometric, GS's master private key, and drone's private key) as well as short-term credentials or session specific (called ephemeral) credentials (for instance, random nonce). Therefore, the session key can be revealed if and only if $\mathcal{A}$ is able to expose both the long-term as well as short-term secrets. Under the CK-adversary model, even though a session key is leaked for a given session, it will not threaten other session keys in previous or forthcoming sessions as the constructed session keys are distinct in different sessions due to used timestamps and random secrets, in addition to the long-term secrets. Similarly, for the session key established among GS and DR_j ($SK_1 = SK_1'$), it is also computationally infeasible to construct SK_1 by $\mathcal{A}$. Thus, BioKA-ASVN is secure against the ESL attack under the CK-adversary model.

10) Anonymity and Untraceability: Under anonymity preservation, the real identities of the network members (here, drones, users, GS, and CS) cannot be determined by any entity, even not by the CS (except GS). In other words, the anonymity of an entity in the system or network (here, ASVN) is achieved if other parties besides the GS are not allowed to know the participants' true identities [56]. The participants' (for example, U_i) real identities are never used during the entire process of authentication or data transmission, while a pseudo-identity of U_i, called $THID$, is utilized for user authentication to the fully trusted GS. Moreover, U_i's identity is not revealed to other participants, such as DR_j and CS (or any unauthorized party $\mathcal{A}$, who tries to eavesdrop on the communications) because U_i's real identity is not sent in plaintext format via communicated messages. Therefore, the proposed scheme achieves U_i's anonymity.

Although a drone DR_j's real identity is used for calculating a session key by hiding it through a "one-way cryptographic hash function" $h(\cdot)$, it is not disclosed by any of the communicated messages $Msg_1 = \{V_3,\, V_2,\, V_1,\, TS_1\}$, $Msg_2 = \{V_1,\, TS_1,\, T_1,\, SKV_1,\, TS_2\}$, $Msg_3 = \{Cert_j^*,\, SKV_2,\, T_2,\, ID_j^*,\, TS_3\}$ and $Msg_4 = \{T_1',\, SKV_2,\, T_2,\, TS_2,\, TS_3,\, TS_4\}$. Therefore, the property of $h(\cdot)$ prevents U_i and CS (or $\mathcal{A}$) from recovering the genuine identity of DR_j from the session key. Thus, the anonymity of a drone DR_j is also preserved in the proposed scheme. In each session, the communicated messages $\{Msg_1,\, Msg_2,\, Msg_3,\, Msg_4\}$ are generated with random nonces and current timestamps, which make the messages dynamic in nature. So, the messages are different and unique for different sessions. As a result, $\mathcal{A}$ cannot track the recipients of messages. Thus, BioKA-ASVN preserves the untraceability property too.

11) Three-Factor Security: A multi-factor authentication system (MFA) employs multiple factors during the authentication process to establish a highly secure interface. MFA comes in various categories, including two-factor, three-factor, and multi-factor (or n-factor) authentication. In a three-factor authentication system, the factors can be classified as either user-related information, such as a password, personal identification number (PIN) or other personal details, something possessed physically, like a one-time-use token, mobile device, smart card, or another similar item, and biometric data, such as fingerprint, iris, facial recognition, or speech patterns. A three-factor security analysis is a method used to assess and enhance security by considering three different factors or elements that contribute to overall security.

Assume that $\mathcal{A}$ guesses in offline/online the password Pw and real-identity ID of a legal user U, in order to successfully login to the system. To generate a correct (valid) session key (or to validate a session key), he/she needs to know at the same time U's biometric B and access to U's mobile device MD, which is infeasible. In another case, if $\mathcal{A}$ has access to U's MD and guesses the password, deriving U's biometric B and ID is also computationally infeasible as these values are hidden through cryptographic hash function. Similarly, if $\mathcal{A}$ has knowledge of B and Pw, in order to successfully generate a genuine session key, he/she needs to know the values of ID and access to U's MD, which is again impractical. Thus, the proposed BioKA-ASVN offers three-factor security.

B. Formal Security Verification under Scyther Tool: Simulation Study

Scyther is an automated security protocol verification tool used for formal analysis of security protocols under the assumption of perfect cryptography. It is used to find vulnerabilities in protocols resulting from the way they are constructed. It uses the Dolev-Yao (DY) threat model [18] as its predefined security model, and thus the user does not have to formalize the adversary's powers. In this model, an adversary can eavesdrop on the messages sent over the communication channel and can extract more information from the messages he/she has learned. The Scyther tool describes the protocols using its own specification language. It allows the user to describe the various roles in the protocol among which communication

occurs and also allows the sending and receiving of messages between roles using the "sent" and "recv" functions, each of which has a format of "sent_" or "recv_". The label is used to distinguish one send and receive pair from others.

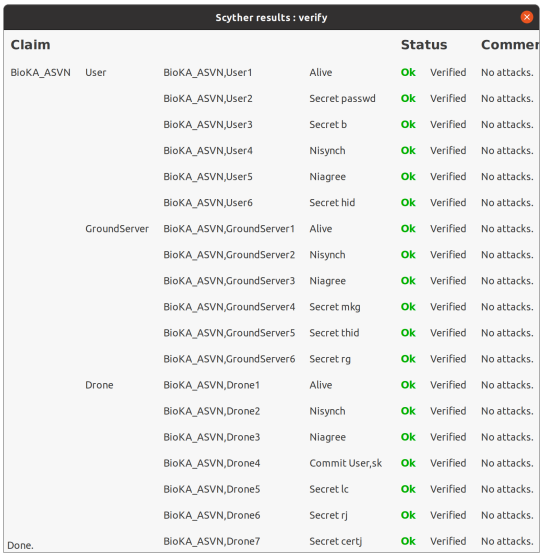


Fig. 5. Simulation results using Scyther tool.

The following characteristics of the proposed scheme (BioKA-ASVN) are analyzed by the Scyther tool simulation: a) secrecy, b) man-in-the-middle, c) replay, and d) reflection attack resistance. The results using the Scyther specification language are shown in Fig. 5. There are three roles corresponding to user (U_i), ground server (GS) and drone (DR_j). Since Scyther does not have a mechanism to preserve states, we have merged the registration and login phases together and passed them so that when the protocol is run, the user first registers and then logs in simultaneously. Any parameter that requires a random value uses the fresh keyword provided by Scyther. Security requirements in Scyther are described using 'claim' events. A claim can be of different types, for example, if the claim is of type "secret", then Scyther will treat the value provided in the claim event as being secret and verify the claim against the adversary as per its security model. Similarly, Nisynch describes non-injective synchronization and Niagree describes non-injective agreement. A detailed description can be found in the Scyther manual [57]. Therefore, the findings in Fig. 5 demonstrate that the Scyther did not identify any vulnerabilities or potential threats to the proposed scheme.

V. Experimental Setup and Results

In this section, we first perform the testbed experiments using Raspberry PI and server platforms. Next, we provide the testbed experiments for the entire proposed scheme. Finally, we present the blockchain implementation of our scheme.

A. Testbed Experiment Using Raspberry PI

The execution time of several cryptographic primitives has been measured using the widely-accepted cryptographic standard library, called "cryptography 37.0.2". It gives Python programmers access to cryptographic recipes and primitives, including both high-level and low-level interfaces to common cryptographic techniques like symmetric ciphers, message digests, and key derivation functions. Let T_{senc}, T_{sdec} T_{ecm}, T_{eca}, T_h, T_{bp}, and T_{mtp} denote the time needed for "Advanced Encryption Standard (AES-128) encryption", "AES-128 decryption", "elliptic curve point multiplication", "elliptic curve point addition", "one-way hash function using Secure Hash Algorithm (SHA-256) algorithm", "bilinear pairing operation" and "message to elliptic curve point function" respectively. A non-singular elliptic curve, namely secp256r1 of the form: "$y^2 = x^3 + ax + b \pmod p$" (for more details see RFC5480) is considered for the "elliptic curve point addition and multiplication". For calculating the timings for bilinear pairing operations, we utilized Tate bilinear pairing on super-singular elliptic curve of the form "$y^2 = x^3 - x + 1$ over the Galois field" [58], whereas the T_{mtp} for message to point (MTP) function has been calculated using the Koblitz's method [59]. We consider the following two cases for computing the execution time needed for various cryptographic primitives under a server and a drone/smart device. The experiments on each cryptographic primitive are also performed for 500 times. We then calculated the maximum, minimum and average runtime (in milliseconds) for each cryptographic primitive, and the average time is considered for the comparative study. The experimental results are reported in Table III.

Case 1. In this case, we consider a platform for computations with a server setting as follows: "Ubuntu 22.04 LTS, with memory: 16 GB, processor: Intel Core i7-9750H CPU @ 2.60GHz processor with 6 cores; 12 threads, OS type: 64-bit and disk type: SSD 256 GB".

TABLE III
Average execution time (in milliseconds) for cryptographic primitives using python

Primitive	Average time (ms) for server	Average time (ms) for Raspberry PI 4
T_h	0.0424	0.3187
T_{senc}	0.0173	0.0926
T_{sdec}	0.0163	0.0945
T_{ecm}	0.1590	1.0712
T_{eca}	0.0229	0.1509
T_{bp}	746.0845	3938.8026
T_{mtp}	0.6627	7.7039

Case 2. In this case, we have considered the user mobile device as a Raspberry PI setup with the configuration: "Raspberry PI 4 Model B, with CPU: 64-bit, Processor: 1.4 GHz Quad-core, 4 cores, Memory (RAM): 1GB, and OS: Ubuntu 20.04 LTS, 64-bit".

B. Testbed Implementation of Proposed Scheme

In this section, we present an implementation setup of our proposed authentication protocol. In this testbed experiment, we utilize three devices for the implementation of the authentication protocol: 1) a personal computer (PC) which is considered as a ground server (GS) with configuration: "Ubuntu 22.04 LTS over Intel core i7-9750H CPU @ 2.60GHz processor with 6 cores; 12 threads with 16 GB RAM and Solid State Drive 256 GB", 2) one Raspberry PI 4 (Pi-1) projected as

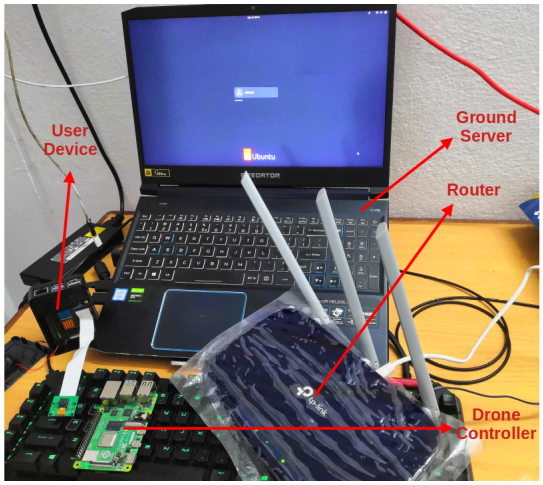


Fig. 6. Testbed experimental setup used for the proposed scheme.

a user mobile device (MD_i) with "Model B, 64-bit, 1.4 GHz Quad-core CPU and RAM 1GB along with 16 GB micro SD as the disk", and 3) another Raspberry PI 4 (Pi-2) with same configuration considered as a drone DR_j.

The three devices are connected via a local wireless network that has been setup by using a WiFi router. The router acts an access point and all the devices connect to the router as stations. The Internet works as the medium for connecting the devices together. All the communication or message passing takes place over the "Hypertext Transfer Protocol (HTTP)". The ground server GS has a HTTP server running at all times that keeps listening for requests from the user device (Pi-1) and the drone (Pi-2). The drone runs a HTTP server that keeps listening for any requests sent by the user via the GS. Once the mutual authentication is over, user (Pi-1) and drone (Pi-2) establish a session key which is shown in Fig. 7.

The left side of Fig. 7 displays the logs from a client device. It shows two choices for the user: 1) register and 2) login. In this diagram, the user selects login option and then inputs user id, password, bio-template and the public key of the drone with which the user wants to establish a secure connection. After entering the information, the client device sends a request to the GS to authenticate the user and establish the shared secret between the user and selected drone. Upon receiving the request, the GS authenticates the user and then forwards the request to the drone. The drone generates the shared secret and sends $\{Cert_j^*, SKV_2, T_2, ID_j^*, TS_3\}$ to the GS. The ground server (GS) forwards these information to the user after verification. The final information received by the user contains $\{T_1', SKV_2, T_2, TS_2, TS_3, TS_4\}$. The user validates these information upon receiving them and generates the same secret that is present with the drone. After this step, the user and the drone can directly communicate with each other securely using a standard symmetric encryption technique. It can be seen in the diagram that the established session keys on both user and drone sides are same, which means that they negotiated on the same session key for secure communication.

C. Blockchain Simulation Using Hyperledger Sawtooth

In this section, we use the Hyperledger Sawtooth framework [60] for simulation of our proposed blockchain network.

Hyperledger is an umbrella project of open source blockchain that was started by Linux Foundation in December 2015. Sawtooth is one of the frameworks developed and maintained by Hyperledger. It is an enterprise solution for the requirement of a blockchain network that is modular and flexible. Each node in the Sawtooth network consists of *Validator*, *REST API*, *Consensus Engine*, and *Transaction Processors*. The validator is responsible for verifying transactions and adding them as new blocks to the chain. It receives instructions from the consensus engine on when to add a new block. The REST API is used for communication with the user. All transactions are submitted through the REST API and the state of the system is also determined through it. Upon receiving a transaction, the validator sends the transaction to a registered transaction processor for transactions of that type. This module is programmed by the developer based on the needs of the application. The entire Sawtooth framework is highly modularized, and even the consensus algorithm can be changed in real-time without restarting the system.

In the proposed BioKA-ASVN, we implemented transaction processor and performed simulations using it. The entire simulation was run on a system set: "Ubuntu 22.04 LTS over Intel Core i7-9750H CPU @ 2.60GHz processor with 6 cores and 12 threads; 16 GB of RAM as primary memory and 256 GB solid state drive as secondary memory". The consensus algorithm used in this work is the "Practical Byzantine Fault Tolerance (PBFT) algorithm" [55]. The blockchain network stores image type data which are sent by the drone camera in the form of individual transactions. We incorporated the same block structure as shown in Fig. 3.

In this simulation, we considered a real time image dataset [61] captured by drone in a real scenarios. The blockchain simulations are run for two different cases and each of them is presented below. In both cases, the transactions were submitted with a delay of four seconds. This was done in order to prevent overflow of the transaction buffer of the validator. In the final timing results, these delays have been properly accounted for.

- **Case 1:** Fig. 8(a) shows this case in which the number of transactions per block are fixed to 10 and the number of nodes in the network is set to 10. The number of blocks mined are varied and it can be seen that the computation time increases linearly as the number of blocks mined is increased.
- **Case 2:** Fig. 8(b) shows this case in which the number of blocks mined is fixed to 10 and number of nodes in the network is 10. The number of transactions per block is varied and the calculated time shows a linear increase with the increase in the number of transactions per block.

VI. Comparative Analysis

This section compares the proposed BioKA-ASVN's performance with other related competing schemes, such as the schemes suggested by Adavoudi-Jolfaei *et al.* [7], Ma *et al.* [8], Ali *et al.* [9], Li *et al.* [10], Shin and Kwon [35], Ali and Pal [6], Ever [11], Irshad *et al.* [25], Zhang *et al.* [37], and Chang *et al.* [38].

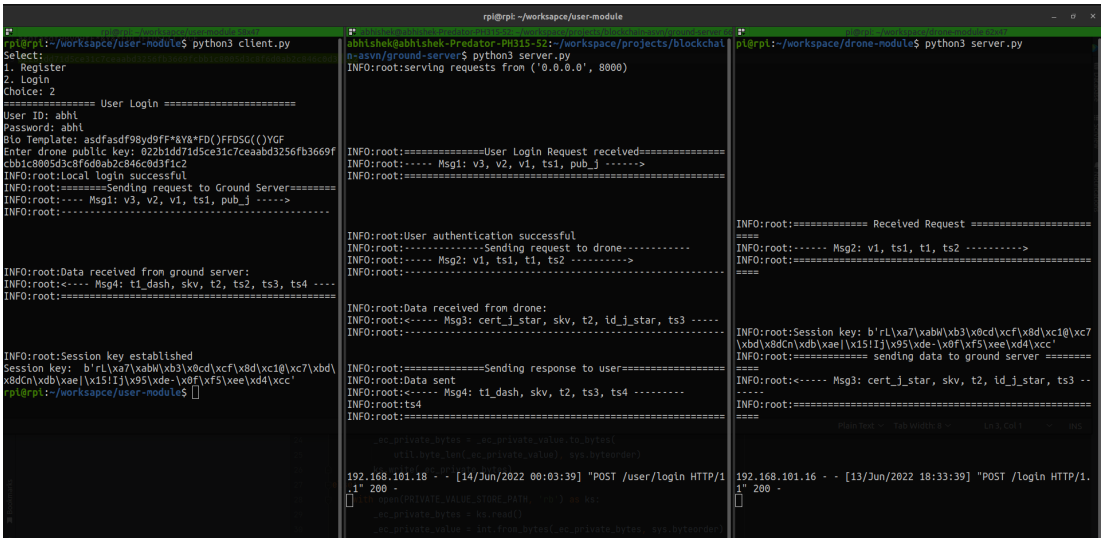
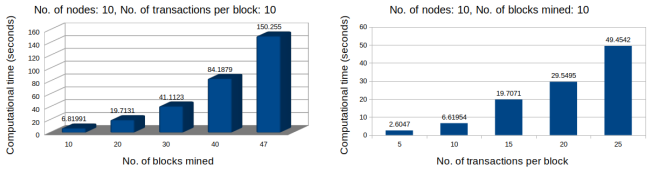


Fig. 7. Testbed experimental results for the proposed scheme.



(a) (b)

Fig. 8. Blockchain simulation results for (a) Case 1 (b) Case 2

A. Communication Costs Comparison

For calculating the communication and computation costs, we consider the authentication and key agreement phase as described in Section III-D between a user U_i and a drone DR_j via the GS. For the purpose of determining the cost of communication, it is considered that "identity", "pseudo-identity", "random number", "elliptic curve point" (for example, $G = (G_x, G_y) \in E_q(a, b)$ where G_x and G_y represent the x and y coordinates of the point G, respectively, hash digest (SHA-256 hashing algorithm), and timestamp are 160, 160, 160, (160 + 160) = 320, 256 and 32 bits, respectively.

TABLE IV
COMPARATIVE STUDY ON COMMUNICATION COSTS

Scheme	No. of messages	Total cost (in bits)
Ali and Pal [6]	4	4608
Adavoudi-Jolfaei *et al.* [7]	4	3552
Ma *et al.* [8]	4	5664
Ali *et al.* [9]	3	3040
Li *et al.* [10]	4	3584
Shin and Kwon [35]	4	4480
Ever [11]	6	5344
Zhang *et al.* [37]	4	4160
Chang *et al.* [38]	4	3712
Irshad *et al.* [25]	3	4128
Proposed (BioKA-ASVN)	4	3584

In BioKA-ASVN, the four messages $Msg_1 = \{V_3, V_2, V_1, TS_1\}$, $Msg_2 = \{V_1, TS_1, T_1, , SKV_1, TS_2\}$, $Msg_3 = \{Cert_j^*, SKV_2, T_2, ID_j^*, TS_3\}$ and $Msg_4 = \{T_1', SKV_2, T_2, TS_2, TS_3, TS_4\}$ require (256 + 256 + 320 + 32) = 864 bits, (320 + 32 + 256 + 256 + 32) = 896 bits, (160 + 256 + 256 + 256 + 32) = 960 bits, and (256 + 256 + 256 + 32 + 32 + 32) = 864 bits respectively, which in total need 3584 bits. From Table IV, it is seen that our proposed BioKA-ASVN has significantly lower communication cost as compared to the other related schemes, such as the schemes of Ma *et al.* [8], Shin and Kwon [35], Ali and Pal [6], Ever [11], Chang *et al.* [38], Irshad *et al.* [25], and Zhang *et al.* [37].

TABLE V
COMPARATIVE STUDY ON COMPUTATION COSTS

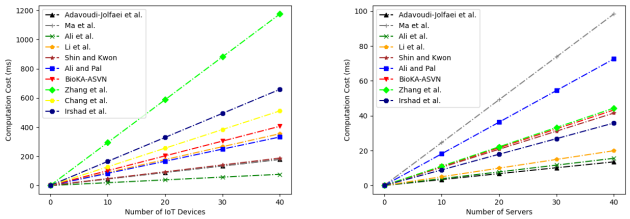
Scheme	User/Smart device/Drone	Server
Adavoudi-Jolfaei *et al.* [7]	$14T_h$ ≈ 4.4618 ms	$8T_h$ ≈ 0.3392 ms
Ma *et al.* [8]	$4T_h + 3T_{ecm}$ ≈ 4.4884 ms	$13T_h + 12T_{ecm}$ ≈ 2.4592 ms
Ali *et al.* [9]	$6T_h$ ≈ 1.9122 ms	$8T_h + 3T_{senc}/T_{sdec}$ ≈ 0.3896 ms
Li *et al.* [10]	$11T_h + 5T_{ecm}$ ≈ 8.8617 ms	$8T_h + T_{ecm}$ ≈ 0.4982 ms
Shin and Kwon [35]	$8T_h + 2T_{ecm}$ ≈ 4.692 ms	$17T_h + 2T_{ecm}$ ≈ 1.0388 ms
Ali and Pal [6]	$4T_h + 6T_{ecm}$ $+4T_{eca}$ ≈ 8.3056 ms	$8T_h + 8T_{ecm}$ $+6T_{eca} + 4T_{senc}/T_{sdec}$ ≈ 1.8158 ms
Ryu *et al.* [42]	$7T_h + 4T_{ecm} +$ $2T_{senc}/T_{sdec} \approx 6.59054$ ms	$9T_h + 4T_{ecm} +$ $4T_{senc}/T_{sdec} \approx 1.0848$ ms
Ever [11]	$9T_h + 2T_{bp} +$ $2T_{mtp} + 3T_{ecm}$ ≈ 7899.0949 ms	$6T_h + 3T_{bp} +$ $2T_{mtp} + 3T_{ecm}$ ≈ 2240.3103 ms
Zhang *et al.* [37]	$9T_h + 3T_{ecm} + 3T_{mtp} +$ $2T_{senc} \approx 29.371$ ms	$6T_h + T_{ecm} + T_{mtp} +$ $2T_{senc}/T_{sdec} \approx 1.1094$ ms
Chang *et al.* [38]	$30T_h + 3T_{ecm}$ ≈ 12.776 ms	–
Irshad *et al.* [25]	$20T_h + 9T_{ecm} + 3T_{eca}$ ≈ 16.4675 ms	$8T_h + 3T_{ecm} + 2T_{eca} +$ $2T_{senc}/T_{sdec} \approx 0.8956$ ms
Proposed (BioKA-ASVN)	$16T_h + 5T_{ecm}$ ≈ 10.4552 ms	$11T_h + 4T_{ecm} + T_{eca}$ ≈ 1.1253 ms

B. Computation Costs Comparison

We assume that T_h, T_{ecm}, T_{eca}, T_{senc} and T_{sdec} signify the time required to perform a "one-way cryptographic hash function", an "elliptic curve point multiplication" and an "elliptic curve point addition", a "symmetric encryption" and a "symmetric decryption", respectively. To compute the computation cost, we consider the authentication and key agreement phase of the proposed scheme as described in Section III-D. Therefore, a user U_i requires the computation cost of $8T_h$ $+3T_{ecm} \approx 5.7632$ ms, a drone DR_j requires the computation

cost of $8T_h$ $+2T_{ecm}$ $\approx$ 4.692 ms in total $16T_h$ $+5T_{ecm}$ $\approx$ 10.4552 ms for both, and a ground server GS needs the computation cost of $11T_h$ $+4T_{ecm}$ $+T_{eca}$ $\approx$ 1.1253 ms.

The experimental findings for various cryptographic primitives with respect to a drone DR_j and the GS (server) are then used, as described in Section V-A. Based on these findings, a comparison of computation costs between the proposed BioKA-ASVN and other existing schemes is shown in Table V. The comparison results for computation of sensor nodes (drones and mobile devices) as well as the ground servers among the proposed BioKA-ASVN and other existing scheme are also presented in Fig. 9(a) and Fig. 9(b), which demonstrate that BioKA-ASVN has comparable computing costs for drone/mobile device with the existing schemes, such as the scheme of Ever [11], Chang *et al.* [38], Irshad *et al.* [25], and Zhang *et al.* [37]. In [37], the execution time of two functions, $KDF(\cdot)$ and $BPV(\cdot)$ is taken as T_{mtp}.



(a) (b)

Fig. 9. Computational costs for (a) IoT devices (drones or mobile devices) (b) ground server

TABLE VI
COMPARATIVE STUDY ON FUNCTIONALITY & SECURITY ATTRIBUTES

Attribute (FASA)	[7]	[9]	[10]	[35]	[6]	[11]	[8]	BioKA-ASVN
$FASA_1$	✓	✓	×	✓	×	✓	✓	✓
$FASA_2$	✓	×	×	✓	✓	✓	✓	✓
$FASA_3$	✓	×	✓	✓	✓	✓	✓	✓
$FASA_4$	✓	✓	✓	✓	✓	✓	✓	✓
$FASA_5$	×	✓	×	✓	×	✓	✓	✓
$FASA_6$	✓	✓	✓	✓	×	✓	✓	✓
$FASA_7$	✓	✓	✓	✓	✓	✓	✓	✓
$FASA_8$	✓	✓	×	✓	✓	✓	✓	✓
$FASA_9$	×	✓	✓	✓	✓	×	×	✓
$FASA_{10}$	×	×	×	×	✓	×	×	✓
$FASA_{11}$	✓	✓	✓	×	×	×	×	✓
$FASA_{12}$	×	×	×	×	×	×	×	✓
$FASA_{13}$	×	×	✓	✓	×	×		✓
$FASA_{14}$	×	✓	×	✓	×	✓	×	✓
$FASA_{15}$	×	×	×	✓	×	N/A	×	✓

$FASA_1$: Replay attack; $FASA_2$: Man-in-the-middle attack; $FASA_3$: Mutual authentication; $FASA_4$: Key agreement; $FASA_5$: Device/drone impersonation attack; $FASA_6$: GSS/server impersonation attack; $FASA_7$: Malicious device deployment attack; $FASA_8$: Drone/device physical capture attack; $FASA_9$: Formal security verification using Scyther/AVISPA/Proverif tool; $FASA_{10}$: ESL attack under the CK-adversary model; $FASA_{11}$: Dynamic drone/device addition phase; $FASA_{12}$: Blockchain-based solution; $FASA_{13}$: Anonymity leakage; $FASA_{14}$: Privileged-insider attack; $FASA_{15}$: Smart card/mobile device stolen attack.
✓: A scheme is secure or it supports an attribute; ×: A scheme is insecure or it does not support an attribute; N/A: Not applicable.

C. Functionality & Security Attributes

Table VI shows that the proposed BioKA-ASVN satisfies all the security and functionality features that are needed to provide a stronger security solution in an air smart vehicular network, whereas other existing related solution do not fully satisfy the desired level of security. For example, the scheme [7] does not resist user smart cad stolen, insider, user impersonation attacks, along with ESL attack. Similarly, the scheme [10] is vulnerable to replay and session leakage attacks, and it does not support blockchain solution.

VII. CONCLUSION

We designed a generic bio-metric based multi-factor security mechanism for drone-assisted ASVN, called BioKA-ASVN. The proposed scheme supports the blockchain technology for providing secure data storage and services. BioKA-ASVN not only offers secure communication between the users and drones, it can also facilitate secure data sharing among the drones and GS by establishing the secret session keys. The blockchain formation and addition to blocks are executed by the P2P cloud server network for securing the information. Real-time test-bed experiments were performed to show the feasibility of the proposed scheme in reality. Finally, the performance evaluation shows the efficiency of the proposed scheme as compared to other competing schemes.

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers and the Associate Editor for their valuable feedback.

REFERENCES

[1] H. Wang, H. Zhao, J. Zhang, D. Ma, J. Li, and J. Wei, "Survey on Unmanned Aerial Vehicle Networks: A Cyber Physical System Perspective," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1027–1070, 2020.

[2] M. Zolanvari, R. Jain, and T. Salman, "Potential Data Link Candidates for Civilian Unmanned Aircraft Systems: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 292–319, 2020.

[3] M. Yahuza, M. Y. I. Idris, I. B. Ahmedy, A. W. A. Wahab, T. Nandy, N. M. Noor, and A. Bala, "Internet of Drones Security and Privacy Issues: Taxonomy and Open Challenges," *IEEE Access*, vol. 9, pp. 57 243–57 270, 2021.

[4] T. Alladi, Naren, G. Bansal, V. Chamola, and M. Guizani, "SecAuthUAV: A Novel Authentication Scheme for UAV-Ground Station and UAV-UAV Communication," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 15 068–15 077, 2020.

[5] Y. Mekdad, A. Aris, L. Babun, A. E. Fergougui, M. Conti, R. Lazzeretti, and A. S. Uluagac, "A survey on security and privacy issues of UAVs," *Computer Networks*, vol. 224, p. 109626, 2023.

[6] R. Ali and A. K. Pal, "An efficient three factor-based authentication scheme in multiserver environment using ECC," *International Journal of Communication Systems*, vol. 31, no. 4, p. e3484, 2018.

[7] A. Adavoudi-Jolfaei, M. Ashouri-Talouki, and S. F. Aghili, "Lightweight and anonymous three-factor authentication and access control scheme for real-time applications in wireless sensor networks," *Peer-to-Peer Networking and Applications*, vol. 12, no. 1, pp. 43–59, 2019.

[8] M. Ma, D. He, H. Wang, N. Kumar, and K.-K. R. Choo, "An Efficient and Provably Secure Authenticated Key Agreement Protocol for Fog-Based Vehicular Ad-Hoc Networks," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8065–8075, 2019.

[9] Z. Ali, S. Hussain, R. H. U. Rehman, A. Munshi, M. Liaqat, N. Kumar, and S. A. Chaudhry, "ITSSAKA-MS: An Improved Three-Factor Symmetric-Key Based Secure AKA Scheme for Multi-Server Environments," *IEEE Access*, vol. 8, pp. 107 993–108 003, 2020.

[10] X. Li, J. Peng, M. S. Obaidat, F. Wu, M. K. Khan, and C. Chen, "A Secure Three-Factor User Authentication Protocol With Forward Secrecy for Wireless Medical Sensor Network Systems," *IEEE Systems Journal*, vol. 14, no. 1, pp. 39–50, 2020.

[11] Y. K. Ever, "A secure authentication scheme framework for mobile-sinks used in the internet of drones applications," *Computer Communications*, vol. 155, pp. 143 – 149, 2020.

[12] Z. Lv, "The security of Internet of drones," *Computer Communications*, vol. 148, pp. 208–214, 2019.

[13] W. Li, X. Li, J. Gao, and H. Wang, "Design of Secure Authenticated Key Management Protocol for Cloud Computing Environments," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 3, pp. 1276–1290, 2021.

[14] M. W. Akram, A. K. Bashir, S. Shamshad, M. A. Saleem, A. A. AlZubi, S. A. Chaudhry, B. A. Alzahrani, and Y. B. Zikria, "A Secure and Lightweight Drones-Access Protocol for Smart City Surveillance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 19 634–19 643, 2022.

[15] L. Zhang and J. Chakareski, "UAV-Assisted Edge Computing and Streaming for Wireless Virtual Reality: Analysis, Algorithm Design, and Performance Guarantees," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 3, pp. 3267–3275, 2022.

[16] J. Yu, G. Wang, Y. Mu, and W. Gao, "An Efficient Generic Framework for Three-Factor Authentication With Provably Secure Instantiation," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 12, pp. 2302–2313, 2014.

[17] A. Yazdinejad, R. M. Parizi, A. Dehghantanha, H. Karimipour, G. Srivastava, and M. Aledhari, "Enabling Drones in the Internet of Things With Decentralized Blockchain-Based Security," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6406–6415, 2021.

[18] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198–208, 1983.

[19] R. Canetti and H. Krawczyk, "Universally Composable Notions of Key Exchange and Secure Channels," in *International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'02)*, Amsterdam, The Netherlands, 2002, pp. 337–351.

[20] A. N. Khan, "How to disable a drone on your property? #5 is my favorite," 2021, https://flythatdrone.com/blog/how-to-disable-drone-on-your-property/. Accessed on April 2022.

[21] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Examining smart-card security under the threat of power analysis attacks," *IEEE Transactions on Computers*, vol. 51, no. 5, pp. 541–552, 2002.

[22] W. Li, H. Cheng, P. Wang, and K. Liang, "Practical Threshold Multi-Factor Authentication," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 3573–3588, 2021.

[23] A. Fotouhi, H. Qiang, M. Ding, M. Hassan, L. G. Giordano, A. Garcia-Rodriguez, and J. Yuan, "Survey on UAV Cellular Communications: Practical Aspects, Standardization Advancements, Regulation, and Security Challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3417–3442, 2019.

[24] V. Hassija, V. Chamola, A. Agrawal, A. Goyal, N. C. Luong, D. Niyato, F. R. Yu, and M. Guizani, "Fast, Reliable, and Secure Drone Communication: A Comprehensive Survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2802–2832, 2021.

[25] A. Irshad, G. A. Mallah, M. Bilal, S. A. Chaudhry, M. Shafiq, and H. Song, "SUSIC: A Secure User Access Control mechanism for SDN-enabled IIoT and Cyber Physical Systems," *IEEE Internet of Things Journal*, pp. 1–1, 2023, doi: 10.1109/JIOT.2023.3268474.

[26] L. Zhang, Y. Zhu, W. Ren, Y. Zhang, and K.-K. R. Choo, "Privacy-Preserving Fast Three-Factor Authentication and Key Agreement for IoT-Based E-Health Systems," *IEEE Transactions on Services Computing*, vol. 16, no. 2, pp. 1324–1333, 2023.

[27] S. Yu and Y. Park, "Comments on "ITSSAKA-MS: An Improved Three-Factor Symmetric-Key Based Secure AKA Scheme for Multi-Server Environments"," *IEEE Access*, vol. 8, pp. 193 375–193 379, 2020.

[28] S. U. Jan, I. A. Abbasi, and F. Algarni, "A Key Agreement Scheme for IoD Deployment Civilian Drone," *IEEE Access*, vol. 9, pp. 149 311–149 321, 2021.

[29] T.-Y. Wu, L. Yang, Z. Lee, C.-M. Chen, J.-S. Pan, and S. H. Islam, "Improved ECC-Based Three-Factor Multiserver Authentication Scheme," *Security and Communication Networks*, vol. 2021, p. 6627956, 2021.

[30] M. Luo, A. Sun, D. He, and X. Li, "An efficient and secure 3-factor user-authentication protocol for multiserver environment," *International Journal of Communication Systems*, vol. 31, no. 14, p. e3734, 2018.

[31] J. Ryu, D. Kang, H. Lee, H. Kim, and D. Won, "A Secure and Lightweight Three-Factor-Based Authentication Scheme for Smart Healthcare Systems," *Sensors*, vol. 20, no. 24, pp. 1–25, 2020.

[32] S. A. Eftekhari, M. Nikooghadam, and M. Rafighi, "Security-enhanced three-party pairwise secret key agreement protocol for fog-based vehicular ad-hoc communications," *Vehicular Communications*, vol. 28, p. 100306, 2021.

[33] J. Srinivas, A. K. Das, N. Kumar, and J. J. P. C. Rodrigues, "TCALAS: Temporal Credential-Based Anonymous Lightweight Authentication Scheme for Internet of Drones Environment," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 7, pp. 6903–6916, 2019.

[34] X. Li, Y. Wang, P. Vijayakumar, D. He, N. Kumar, and J. Ma, "Blockchain-Based Mutual-Healing Group Key Distribution Scheme in Unmanned Aerial Vehicles Ad-Hoc Network," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 11, pp. 11 309–11 322, 2019.

[35] S. Shin and T. Kwon, "A Privacy-Preserving Authentication, Authorization, and Key Agreement Scheme for Wireless Sensor Networks in 5G-Integrated Internet of Things," *IEEE Access*, vol. 8, pp. 67 555–67 571, 2020.

[36] M. Tanveer, A. H. Zahid, M. Ahmad, A. Baz, and H. Alhakami, "LAKE-IoD: Lightweight Authenticated Key Exchange Protocol for the Internet of Drone Environment," *IEEE Access*, vol. 8, pp. 155 645–155 659, 2020.

[37] N. Zhang, Q. Jiang, L. Li, X. Ma, and J. Ma, "An efficient three-factor remote user authentication protocol based on BPV-FourQ for internet of drones," *Peer-to-Peer Networking and Applications*, vol. 14, no. 5, pp. 3319–3332, 2021.

[38] Y.-F. Chang, W.-L. Tai, P.-L. Hou, and K.-Y. Lai, "A Secure Three-Factor Anonymous User Authentication Scheme for Internet of Things Environments," *Symmetry*, vol. 13, no. 7, 2021.

[39] S. Javed, M. A. Khan, A. M. Abdullah, A. Alsirhani, A. Alomari, F. Noor, and I. Ullah, "An Efficient Authentication Scheme Using Blockchain as a Certificate Authority for the Internet of Drones," *Drones*, vol. 6, no. 10, 2022.

[40] C. Feng, B. Liu, Z. Guo, K. Yu, Z. Qin, and K.-K. R. Choo, "Blockchain-Based Cross-Domain Authentication for Intelligent 5G-Enabled Internet of Drones," *IEEE Internet of Things Journal*, vol. 9, no. 8, pp. 6224–6238, 2022.

[41] Y. Tan, J. Wang, J. Liu, and N. Kato, "Blockchain-Assisted Distributed and Lightweight Authentication Service for Industrial Unmanned Aerial Vehicles," *IEEE Internet of Things Journal*, vol. 9, no. 18, pp. 16 928–16 940, 2022.

[42] J. Ryu, J. Oh, D. Kwon, S. Son, J. Lee, Y. Park, and Y. Park, "Secure ECC-Based Three-Factor Mutual Authentication Protocol for Telecare Medical Information System," *IEEE Access*, vol. 10, pp. 11 511–11 526, 2022.

[43] A. Ghafouri Mirsaraei, A. Barati, and H. Barati, "A secure three-factor authentication scheme for IoT environments," *Journal of Parallel and Distributed Computing*, vol. 169, pp. 87–105, 2022.

[44] M. A. Saleem, S. Shamshad, S. Ahmed, Z. Ghaffar, and K. Mahmood, "Security Analysis on "A Secure Three-Factor User Authentication Protocol With Forward Secrecy for Wireless Medical Sensor Network Systems"," *IEEE Systems Journal*, vol. 15, no. 4, pp. 5557–5559, 2021.

[45] D. Wang, P. Wang, and C. Wang, "Efficient Multi-Factor User Authentication Protocol with Forward Secrecy for Real-Time Data Access in WSNs," *ACM Trans. Cyber-Phys. Syst.*, vol. 4, no. 3, 2020.

[46] J. Sun, X. Yao, S. Wang, and Y. Wu, "Blockchain-Based Secure Storage and Access Scheme For Electronic Medical Records in IPFS," *IEEE Access*, vol. 8, pp. 59 389–59 401, 2020.

[47] Y. Park, D. Ryu, D. Kwon, and Y. Park, "Provably Secure Mutual Authentication and Key Agreement Scheme Using PUF in Internet of Drones Deployments," *Sensors*, vol. 23, no. 4, 2023.

[48] S. Hussain, S. A. Chaudhry, O. A. Alomari, M. H. Alsharif, M. K. Khan, and N. Kumar, "Amassing the Security: An ECC-Based Authentication Scheme for Internet of Drones," *IEEE Systems Journal*, vol. 15, no. 3, pp. 4431–4438, 2021.

[49] C. Madan Kumar, R. Amin, and M. Brindha, "Cryptanalysis of Secure ECC-Based Three Factor Mutual Authentication Protocol for Telecare Medical Information System," *Cyber Security and Applications*, vol. 1, p. 100013, 2023.

[50] Z. Ullah, B. Raza, H. Shah, S. Khan, and A. Waheed, "Towards Blockchain-Based Secure Storage and Trusted Data Sharing Scheme for IoT Environment," *IEEE Access*, vol. 10, pp. 36 978–36 994, 2022.

[51] M. R. Bataineh, W. Mardini, Y. M. Khamayseh, and M. M. B. Yassein, "Novel and Secure Blockchain Framework for Health Applications in IoT," *IEEE Access*, vol. 10, pp. 14 914–14 926, 2022.

[52] Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data," in *Advances in Cryptology - EUROCRYPT 2004*. Interlaken, Switzerland: Springer Berlin Heidelberg, 2004, pp. 523–540.

[53] M. Wazid, A. K. Das, V. Odelu, N. Kumar, and W. Susilo, "Secure Remote User Authenticated Key Establishment Protocol for Smart Home Environment," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 2, pp. 391–406, 2020.

[54] S. Shukla and S. J. Patel, "A novel ECC-based provably secure and privacy-preserving multi-factor authentication protocol for cloud computing," *Computing*, vol. 104, no. 5, pp. 1173–1202, 2022.

[55] M. Castro and B. Liskov, "Practical Byzantine fault tolerance and proactive recovery," *ACM Transactions on Computer Systems*, vol. 20, no. 4, pp. 398–461, 2002.

[56] Y. Jiang, K. Zhang, Y. Qian, and L. Zhou, "Anonymous and Efficient Authentication Scheme for Privacy-Preserving Distributed Learning,"

IEEE Transactions on Information Forensics and Security, vol. 17, pp. 2227–2240, 2022.

[57] C. J. F. Cremers, "The Scyther Tool: Verification, Falsification, and Analysis of Security Protocols," in *Computer Aided Verification*, Princeton, NJ, USA, 2008, pp. 414–418.

[58] B. Lynn, "The Tate Pairing," 2001, https://crypto.stanford.edu/pbc/notes/ep/tate.html.

[59] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of computation*, vol. 48, no. 177, pp. 203–209, 1987.

[60] "Core repository for Sawtooth Distributed Ledger," 2016, Accessed on June 2022. [Online]. Available: https://github.com/hyperledger/sawtooth-core

[61] P. Zhu, L. Wen, D. Du, X. Bian, H. Fan, Q. Hu, and H. Ling, "Detection and Tracking Meet Drones Challenge," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 11, pp. 7380–7399, 2022.