Computer Networks 224 (2023) 109612

Contents lists available at ScienceDirect

Computer Networks

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

# LAKA-UAV: Lightweight authentication and key agreement scheme for cloud-assisted Unmanned Aerial Vehicle using blockchain in flying ad-hoc networks

Sungjin Yu [a,b], Joonyoung Lee [b], Anil Kumar Sutrala [c], Ashok Kumar Das [d], Youngho Park [b,*]

[a] *Electronics and Telecommunications Research Institute, Daejeon 34129, South Korea*
[b] *School of Electronics and Electrical Engineering, Kyungpook National University, Daegu 41566, South Korea*
[c] *CA Technologies – A Broadcom Company, Hyderabad 500 032, India*
[d] *Center for Security, Theory and Algorithmic Research, International Institute of Information Technology, Hyderabad 500032, India*

ARTICLE INFO

*Keywords:*
Unmanned Aerial Vehicle (UAV)
Cloud computing
Blockchain
Access control
Authentication
Key agreement

ABSTRACT

Unmanned Aerial Vehicle (UAV) can be employed in various applications, including traffic control, and delivery application. However, these services are susceptible to various security attacks because sensitive data are exchanged through an open channel. Thus, a secure authentication scheme is essential for UAV. UAV has limited storage resources and computing capabilities. To overcome these problems, cloud computing is considered as a promising solution. Cloud computing provides various properties such as storage availability and scalability. Unfortunately, the cloud server's database can be a major target for an adversary because it is a centralized system. If an adversary intrudes on the cloud server's database, he/she may attempt to intercept or learn the stored data. To mitigate these issues, we design a secure and lightweight authentication and key agreement scheme for cloud-assisted UAV using blockchain in flying ad-hoc networks, called LAKA-UAV. LAKA-UAV utilizes blockchain technology to ensure access control and data integrity using log transactions and the cloud server securely manages collected data from UAV. We prove the security of LAKA-UAV based on informal security analysis and formal security verification implementation. Based on testbed experiments using MIRACL, LAKA-UAV provides about 2.13 times more efficient performance on average compared with related schemes in terms of computation. We present the blockchain implementation using Hyperledger Sawtooth platform.

## 1. Introduction

With the development of "Flying Ad-Hoc Networks (FANET), 5G communication, and Internet of Things (IoT)", the unmanned aerial vehicles (UAV) have made the realization of smart cities possible in the future [1–3]. The smart cities are emerged as a new paradigm to alleviate the challenges of rapid and continuous urbanization and provide better facilities and better quality of life for citizens. However, the main issues in smart city are the processing of large amounts of data from hundreds of thousands of sensors and IoT integrated into smart objects. UAV combined with the "infrastructure, IoT and FANET" are considered as a promising solution to solve these problems. UAV has been rapidly evolved over the past few years owing to useful features such as environmental monitoring, disaster management, traffic monitoring, search and rescue, goods transportation, data collection, and tracking [4]. UAV allows various services anytime and anywhere owing to their multiple benefits of flexible mobility and fast deployment. The UAV environments consist of the mobile user, the drone, and the ground station server (GSS). UAV is deployed in various fields to collect a large amount of data from different applications and send the collected data to the GSS. The GSS monitors and analyzes the collected data to provide useful services for legitimate users. However, the significant issues of UAV are that is difficult to collect and deliver the data in real-time because it has limited storage resources and computing capabilities. If there are unexpected and sudden demands for storage resources and computing power, UAV should ensure sufficient capacities.

In the past few years, many studies for UAV have adopted cloud computing to address storage and computing problems [5–7]. As an important technique to provide efficient UAV services, cloud computing can serve as a platform for data sharing between the mobile user and the drone. The user's requests and the various types of collected data can be processed and stored by cloud computing, allowing the

* Corresponding author.
*E-mail address:* parkyh@knu.ac.kr (Y. Park).

https://doi.org/10.1016/j.comnet.2023.109612
Received 19 September 2022; Received in revised form 11 January 2023; Accepted 2 February 2023
Available online 6 February 2023

extraction of valuable information that can be in enhancing the quality of safety, experience, and life in the smart city. However, the database of the cloud server cannot resolve problems such as bottlenecks and single point of failure because it is a centralized system. To resolve these problems, blockchain technology has emerged as a promising solution.

Recently, numerous blockchain-based UAV systems have been presented [8–10] to resolve the problems of cloud computing. Blockchain is a network technology that guarantees data integrity and decentralization by sharing information with multiple distributed nodes. Blockchain is considered a trusted distributed ledger that maintains transactions in a chain of chronological blocks linked via hash values. Moreover, blockchain offers properties, including decentralization, data integrity, and so on. Although these blockchain-based UAV systems have received great attention, the authentication mechanism and cloud computing for secure data sharing have not been addressed. Therefore, we design a "robust and lightweight authentication and key agreement scheme for the cloud-assisted unmanned aerial vehicle using blockchain in FANET (LAKA-UAV)" to ensure integrity and decentralization functionalities for data sharing. LAKA-UAV uses the cloud technique to achieve sufficient storage resources and computing capabilities, and the data in each block only stores metadata to improve block construction and minimize distributed storage waste. Moreover, LAKA-UAV utilizes blockchain technology such as Hyperledger Fabric to ensure efficient access control, data integrity, and decentralization through log transactions. LAKA-UAV proves that ensures efficient computation cost and a high-security level by performing testbed experiments and blockchain implementation.

### 1.1. Significances and motivations

Recently, the applications of UAV provide several advantages and benefits to smart cities. The UAVs are difficult to collect and deliver data in real-time because they are resource-constrained with regard to low computing and storage capabilities. To address these issues, cloud computing is considered a promising solution. However, the data worker and requester of a previous cloud-assisted UAV system in FANET communicate over an insecure channel. Thus, there are chances that the transmitted messages may be compromised, leaked, or altered by an adversary. Due to the existing weaknesses of a cloud-assisted UAV system in FANET, sensitive information can be leaked or it may cause other potential security threats. Therefore, it becomes very necessary to ensure an effective solution to enhance the privacy and security issues of the cloud-assisted UAV system in FANET. The blockchain can also play an important role as it is a tamper-resistant technology that can resist the most serious attacks and make the system more decentralized and robust. Moreover, there are still not many of these approaches in the presented literature survey of cloud-assisted UAV systems using blockchain in FANET. These facts have motivated us to propose a new lightweight authentication and key agreement scheme for cloud-assisted UAV using blockchain that resolves the potential security threats and provides cost-effective performance and necessary security functionalities.

### 1.2. Contributions

The major contributions of LAKA-UAV are as follows:

- We propose a "robust and lightweight authentication and key agreement scheme for cloud-assisted UAV using blockchain in FANET". In LAKA-UAV, the sensing data collected by the drones in the flying zones from deployed IoT smart objects are securely transmitted to the GSS. The transactions and signatures are utilized by GSS for generation blocks and these are added after block verification by other GSS in P2P networks using the consensus algorithm.
- We evaluate the "formal (mathematical) security analysis using the Real-or-Random (ROR) oracle model" [11] to evaluate the session key security of LAKA-UAV. Moreover, we perform the "formal (simulation) security of LAKA-UAV using the Automated Verification of Internet Security Protocols and Applications (AVISPA)" [12,13], which evaluates the security to "man-in-the-middle (MITM) and replay" attacks. Thus, LAKA-UAV is shown to resist potential security attacks needed in cloud-assisted UAV using blockchain in FANET through formal and informal security analyses.
- We perform the "testbed experiments for cryptographic primitives using the broadly-utilized Multiprecision Integer and Rational Arithmetic Cryptographic Library (MIRACL)" [14]. Based on the results of testbed experiments, the performance is verified whether the computation time required for cryptographic primitives in the proposed protocol is suitable for practical servers and mobile devices.
- A practical demonstration of LAKA-UAV through blockchain implementation is also presented to observe its impact on the system performance on computation time. We perform the blockchain implementation for computation times of a varied number of blocks mined and a varied number of transactions per block in blockchain using Hyperledger Sawtooth platform [15].
- We perform a performance comparative analysis of LAKA-UAV with the relevant existing schemes. Thus, LAKA-UAV demonstrates the superiority of the proposed scheme over other relevant existing schemes in terms of computation cost and security features.

## 2. Related works

In the last few decades, many authentication and key agreement schemes [16,17] have been presented to provide effective services in UAV environments. Wazid et al. [18] presented a "secure and lightweight remote user authentication and key agreement scheme in UAV environments". They discussed some security challenges and requirements for UAV. Unfortunately, Alladi et al. [19] demonstrated that Wazid et al.'s scheme [18] could not provide "perfect backward secrecy and formal proof". Srinivas et al. [20] proposed a "temporal credential-based lightweight authentication scheme in UAV environments". However, Ali et al. [21] proved that their scheme [20] could not prevent "impersonation attacks and also provide user anonymity". Thus, Ali et al. [21] presented a "lightweight authentication mechanism for UAV in smart city environments" to improve the security problems of Srinivas et al.'s scheme [20]. However, according to the information given in [22], Ali et al. [21] still is not resistant to session key disclosure and denial of service (DoS) attacks. Thus, these schemes [18–21] should ensure the security and privacy of data collected from the UAV, and also consider outsourcing the data to the cloud. If there is a sudden and unexpected demand for storage and resources, existing schemes for UAV have to provide sufficient capacities.

In the past few decades, many cloud-assisted schemes [23–25] for UAV have been presented to improve storage overload problems. Koubaa et al. [23] presented a "cloud-based real-time object tracking system for UAV". Their scheme [23] ensures an efficient UAV system to provide storage availability and extensibility through the cloud. Hong et al. [24] proposed a "cloud-assisted control system architecture to control and monitor for multiple UAV". Their scheme [24] provides sufficient storage resources for users to provide efficient UAV services. Koubaa et al. [25] presented a "service-oriented cloud-assisted management system for UAV". Their scheme [25] overcomes the limitations of the computing and storage resource for the UAV because intensive computations are not executed on-board, but rather offloaded to the cloud. However, these schemes [23–25] are essentially a centralized system so that they do not resolve problems such as bottlenecks and single point of failure. Thus, blockchain technology with decentralized

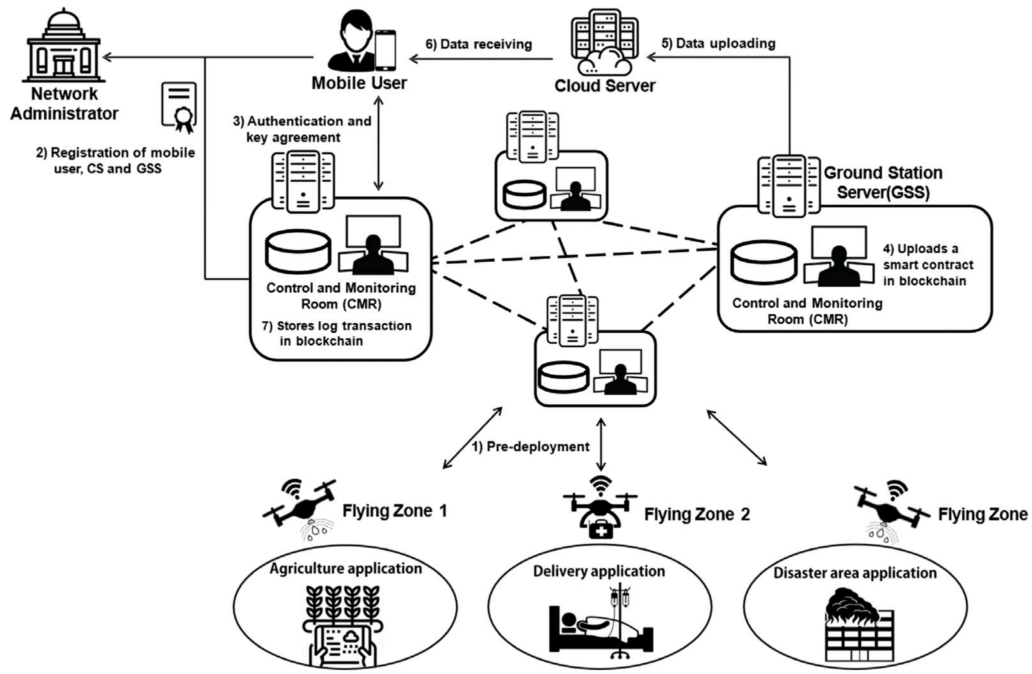


**Fig. 1.** Proposed system model.

properties is considered as a promising solution to resolve the problems of centralized systems.

Recently, many blockchain-based schemes for UAV have been proposed [26–28] to resolve the problems associated with cloud computing. Mehta et al. [26] presented a review on security and privacy issues to provide useful services for users in blockchain-assisted UAV environments. Moreover, they discussed the limitations of the existing UAV and the proposed solutions for next-generation UAV. Ch et al. [27] proposed a "security and privacy of UAV data using blockchain technology" to ensure effective services. Their scheme [27] provides confidentiality, integrity, and availability of UAV data using blockchain. Yazdinejad et al. [28] presented a "blockchain-based system architecture" to provide users with a transparent and efficient mechanism for secure communication. Their scheme [28] ensures low latency and secure authentication of UAVs using blockchain. Although the existing blockchain-based UAV systems [26–28] received a lot of attention, these schemes do not perform testbed experiments for blockchain technology and have not been presented more detailed cloud and blockchain-based protocol model for UAV. Thus, we design a "detailed robust and lightweight authentication and key agreement protocol for the cloud-assisted UAV using blockchain in FANET" and also present the implementation for blockchain based on Hyperledger Sawtooth platform and the security verification based on AVISPA simulation.

## 3. Preliminaries

This section introduces preliminaries to improve the readability of this paper.

### 3.1. Threat model

We introduce "Dolev and Yao (DY) model" [29] to evaluate the security of LAKA-UAV. According to the DY model, a malicious adversary can "forge, inject, delete, eavesdrop, or modify" transmitted messages over an insecure channel. A malicious adversary can also steal a legitimate user's mobile device (MD) and extract secret credentials stored in memory by utilizing power analysis [30]. After obtaining secret credentials of MD, a malicious adversary may attempt security threats such as "forgery, stolen mobile device, offline password guessing, and impersonation" attacks. Moreover, a malicious adversary is infeasible to guess the real identity and password of the legitimate user simultaneously.

We introduce "Canetti and Krawczyk (CK) model" [31,32], which is more powerful than the DY model. In the CK model, a malicious adversary can compromise session states and secret credentials via a session-hijacking attack. Therefore, the session key between each entity should be dependent on both "short-term secrets and long-term secrets".

### 3.2. Hyperledger fabric

Recently, numerous researchers have been presented blockchain technology to provide decentralized properties and data integrity. In particular, digital currency represented by Bitcoin has been a huge success and has attracted worldwide attention to blockchain technology. However, public blockchain like Bitcoin has some disadvantages. The public blockchain consumes a large amount of computational costs because all nodes participate in the consensus process, and it may cause scalability problems. To enhance these problems of the public blockchain, LAKA-UAV utilizes Hyperledger Fabric [33] one of the consortium blockchains.

Hyperledger Fabric [33] was proposed as an open-source blockchain by the Linux Foundation. The goals of Hyperledger Fabric are to promote cross-industry cooperation by utilizing blockchain. Unlike Bitcoin, Hyperledger Fabric does not require digital currency and offers various advantages such as reliability, scalability, and blockchain performance. This technology implements a practical byzantine fault-tolerant (PBFT) consensus mechanism [34,35]. Thus, we apply the PBFT mechanism in LAKA-UAV for better "access control and decentralized" properties of the sharing information.

## 4. System model for cloud-assisted UAV using blockchain

This section introduces a cloud-assisted UAV system model using Hyperledger Fabric in Fig. 1. The proposed model consists of fifth entities: the network administrator, mobile user, unmanned aerial vehicle, ground station server, and cloud server. The detailed descriptions of each entity are as below.

- Network Administrator (NA): This entity is a trusted authority and is responsible for the registration of participants. Moreover, a NA manages a permissioned blockchain.
- Ground Station Server: GSS is responsible for managing the deployed UAVs in their respective FZs. It collects the real-time data from UAV and records them in the form of transactions, and then makes them available to the blockchain network. Moreover, GSS encrypts the data collected from the UAV using a session key and then sends it to the cloud server. GSS acts with a trusted control and monitoring room (CMR) to monitor and analyze all the activities and data taking place in UAV environment. This process is apparently effective to draw some useful conclusions from the stored, processed and analyzed data. For example, it will give predictions about the chances of the disaster condition, road condition, and so on. GSS is a trusted entity and assumed that it is not compromised by the malicious adversary.
- Cloud Server: The cloud server has sufficient storage capacity and computing power. The cloud server manages and stores the collected data from UAV to ensure secure storage resources and data sharing. The cloud server manages the encrypted UAV data received from the GSS and sends them to the mobile user requesting the UAV services.
- Unmanned Aerial Vehicle: The airspace comprises multiple flying zones (FZ) and multiple UAVs can be deployed in the specific FZ to monitor various environments. UAV comprises several IoT sensors such as "laser sensor", "range imaging sensor", "ultrasonic sensor", "orientation sensor", and "thermal sensor". UAV deployed in a particular FZ collects data in the various environments through the sensors and transmits the gathered information to the GSS.
- Mobile User: The mobile user authenticates the cloud server and GSS to receive UAV services. After authentication, a session key is established between each entity for secure communications in the future.

The communication flows of the LAKA-UAV are presented in Fig. 1, where all steps are described as follows:

1. Each drone is assigned a unique identity with the help of the GSS and then is registered in the GSS before being deployed in the flying zone.
2. The mobile user, cloud server, and ground station server register the unique identities with the help of the NA to access UAV services.
3. The mobile user and ground station server authenticate each other and establish a session key for secure communication in the future.
4. After each participant is mutually authenticated successfully, GSS receives the messages for a smart contract from each drone using a pre-shared key and generates a smart contract and uploads it to blockchain.
5. GSS encrypts the collected data of the drone using a session key and then re-encrypts the data using a pre-shared secret credential and transmits it to the cloud server. After that, the cloud server decrypts the re-encrypted UAV data and stores it in the database.
6. When the mobile user wants to access the UAV data in the system, the mobile user sends the data request message to the cloud server. Then, the cloud server encrypts the corresponding UAV data using a pre-shared secret credential and transmits it to the mobile user. After getting the message, the mobile user obtains various UAV services using a session key successfully.
7. Finally, the GSS generates a log transaction, including the mobile user's masked identity, the drone's identity, the access time, location, destination address of UAV data, and GSS's certificate. After that, GSS uploads the block of log transactions to the blockchain.

## 5. Proposed scheme

In FANET environments, UAVs may have limited access to shared resources due to a large number of unnecessary requests. This will cause the system to overload and might result in the rejection of some or all legitimate requests to be fulfilled [36]. UAVs have limited resources in terms of computation overhead and low computing power which makes it difficult to perform high computation overheads. Thus, UAVs are not suitable to apply asymmetric cryptography that requires high computation overheads [37]; otherwise, the efficiency of the authentication will be greatly affected. Moreover, UAVs are difficult to collect and deliver data in real-time because it has limited low computing and storage capability. Besides, smart devices (e.g., smart phones) have sufficient resource capabilities to perform symmetric key cryptography and public key cryptography, but it is efficient to use lightweight cryptographic primitives to provide effective communication in FANET environments. To address these problems, we design a "secure and lightweight authentication and key agreement scheme for cloud-assisted UAV using Hyperledger Fabric in FANET". In the proposed protocol, the smart device and drone utilize lightweight cryptographic primitives such as hash function and XOR operation during the authentication phase. In addition, the cloud server and ground station server have sufficient computing and computation capabilities and also generate a public key to participate in the Hyperledger Fabric and utilize it for certificates and credentials to prove that it is the authorized node by NA in the Hyperledger Fabric. Therefore, LAKA-UAV allows only authenticated participants to outsource UAV data, and each operation for outsourcing data is integrated into the blockchain as a transaction.

LAKA-UAV comprises six phases: "pre-deployment, registration, login and authentication, UAV data uploading, UAV data requesting, and block construction and verification". Before performing the registration process, a network administrator ($NA$) initialize system public parameters to build Hyperledger Fabric using elliptic curve cryptosystems (ECC) [38]. $NA$ first chooses a base group $G$ over an elliptic curve $E_p$ with order $p$ that is a prime number. $P$ of the order $q$ is one of generators of $G$, where $q$ is a prime number. After that, $NA$ selects a private key $K_{NA}$ and generates a public key $PK_{NA} = K_{NA} \cdot G$. Then, $NA$ shares policies and the network configuration with all network participants. $NA$ publishes system public parameters $(G, P, PK_{NA}, p, q)$. Table 1 presents the notations in LAKA-UAV.

### 5.1. Pre-deployment phase

Initially, each remote drone ($D_j$) is registered with $GSS$ for pre-deployment. $GSS$ assigns an identity $ID_{D_j}$ of each $D_j$ before placing them into any area partitioned as $n_{FZ}$ FZs with a $CID_k$. Then, $GSS$ computes pre-shared key $K_{GD} = h(CID_k \| ID_{D_j} \| K_{GSS})$ and stores $\{CID_k, K_{GD}\}$ in the memory of $D_j$ and $\{CID_k, ID_{D_j}, K_{GD}\}$ in its own database.

### 5.2. Registration phase

In LAKA-UAV, the registration phase comprises "user registration, cloud server registration, and GSS registration". $MU_i$, $CS$, and $GSS$ must register with $NA$ to participate in the network. The overall steps for this phase are presented below.

#### 5.2.1. User registration phase

When $MU_i$ wants to access UAV services, $MU_i$ must register with $NA$. This phase is shown in Fig. 2 and overall steps for this phase are presented below.

- **UR 1:** $MU_i$ chooses an unique identity $ID_i$ and password $PW_i$. After that, $MU_i$ selects a random number $r_i$ and calculates $HID_i = h(ID_i \parallel PW_i)$. Then, $MU_i$ sends $\{HID_i, r_i\}$ to $NA$ over a secure channel.

**Table 1**
Notations.

| Symbol | Meaning |
|---|---|
| $MU_i$ | Mobile user |
| $D_j$ | Drone |
| $GSS_j$ | Ground station server |
| $CS$ | Cloud server |
| $r_{CS}, PK_{CS}$ | ECC key pair of $CS$ |
| $r_{GSS}, PK_{GSS}$ | ECC key pair of $GSS_j$ |
| $K_{NA}$ | Master key of $NA$ |
| $Cert_j$ | Certificate of $GSS_j$ |
| $E_p(a,b)$ | A nonsingular elliptic curve $y^2 = x^3 + ax + b \pmod p$ |
| $n_{FZ}$ | The number of drones to be placed in the flying zone |
| $D_{data}$ | Collected data by $D_j$ |
| $T_i$ | Timestamp |
| $T_{up}$, $T_{ac}$ | Uploading and accessing time of $D_{data}$ |
| $D_{loc}$ | Location of $D_j$ |
| $D_{da}$ | Destination address of $D_j$ |
| $SK$ | Session key between $MU_i$ and $GSS_j$ |
| $h(\cdot)$ | Collision-resistant cryptographic hash function |
| $\oplus$ | XOR operation |
| $\|\|$ | Concatenation operation |

| Mobile user $(MU_i)$ | Network administrator $(NA)$ |
|---|---|
| Selects identity $ID_i$,<br>high-entropy password $PW_i$.<br>Generates random number $r_i$.<br>Computes<br>$HID_i = h(ID_i\|\|PW_i)$<br>$\underline{\{HID_i, r_i\}}$<br>(via secure channel) | |
| | Generates a random number $a_i$<br>Computes<br>$X_i = h(HID_i\|\|K_{NA}\|\|a_i)$<br>$Z_i = h(HID_i\|\|a_i)$<br>$A_i = Z_i \oplus h(HID_i\|\|X_i)$<br>Stores $\{HID_i, a_i\}$ in secure database<br>$\underline{\{X_i, A_i\}}$<br>(via secure channel) |
| Computes $HPW_i = h(PW_i\|\|r_i)$<br>$B_i = r_i \oplus h(PW_i\|\|HID_i)$<br>$C_i = X_i \oplus h(HID_i\|\|HPW_i\|\|r_i)$<br>$D_i = h(HID_i\|\|HPW_i\|\|X_i\|\|r_i)$<br>Stores $\{A_i, B_i, C_i, D_i\}$ in MD | |

**Fig. 2.** Summary of user registration Phase.

- **UR 2:** $NA$ generates a random number $a_i$ and calculates $X_i = h(HID_i \| K_{NA} \| a_i)$, $Z_i = h(HID_i \parallel a_i)$ and $A_i = Z_i \oplus h(HID_i \parallel X_i)$. Then, $NA$ stores $\{HID_i, a_i\}$ in database and sends $\{X_i, A_i\}$ to $MU_i$ over a secure channel.
- **UR 3:** $MU_i$ computes $HPW_i = h(PW_i \parallel r_i)$, $B_i = r_i \oplus h(PW_i \parallel HID_i)$, $C_i = X_i \oplus h(HID_i \| HPW_i \| r_i)$, and $D_i = h(HID_i \| HPW_i \| X_i \parallel r_i)$ and stores $\{A_i, B_i, C_i, D_i\}$ in MD.

### 5.2.2. Cloud server registration phase

To provide secure UAV services for $MU_i$, $CS$ must register with $NA$. This phase is shown in Fig. 3 and the overall steps for this phase are as follows.

- **CSR 1:** $CS$ chooses an unique $ID_{CS}$ and generates a private key $r_{CS}$ and a public key $PK_{CS} = r_{CS} \cdot P$. Then, $CS$ sends $\{ID_{CS}\}$ to $NA$ over a secure channel.
- **CSR 2:** $NA$ retrieves $\{HID_{list}, a_i\}$ in the database. After that, $NA$ generates a random number $b_i$ and calculates $Z_i = h(HID_{list} \parallel a_i)$ and $S_i = h(ID_{CS} \parallel b_i)$. Finally, $NA$ stores $\{ID_{CS}, S_i\}$ in secure database and sends $\{HID_{list}, Z_i, S_i\}$ to $CS$ over a secure channel.
- **CSR 3:** $CS$ computes $Q_i = (Z_i \parallel HID_{list}) \oplus r_{CS}$ and $W_i = S_i \oplus h(r_{CS} \parallel Z_i)$. After that, $CS$ stores $\{Q_i, W_i\}$ in database.

| Cloud server $(CS)$ | Network administrator $(NA)$ |
|---|---|
| Selects identity $ID_{CS}$<br>Generates random number $r_{CS}$<br>Generates public key $PK_{CS} = r_{CS} \cdot P$<br>$\underline{\{ID_{CS}\}}$<br>(via secure channel) | |
| | Retrieves $\{HID_{list}, a_i\}$ in secure database<br>Generates a random number $b_i$<br>Computes<br>$Z_i = h(HID_{list}\|\|a_i)$<br>$S_i = h(ID_{CS}\|\|b_i)$<br>Stores $\{ID_{CS}, S_i\}$ in secure database<br>$\underline{\{HID_{list}, Z_i, S_i\}}$<br>(via secure channel) |
| Computes<br>$Q_i = (Z_i\|\|HID_{list}) \oplus r_{CS}$<br>$W_i = S_i \oplus h(r_{CS}\|\|Z_i)$<br>Stores $\{Q_i, W_i\}$ in database | |

**Fig. 3.** Summary of cloud server registration phase.

| Ground station server $(GSS_j)$ | Network administrator $(NA)$ |
|---|---|
| Selects identity $ID_{GSS}$<br>Generates random number $r_{GSS}$<br>Generates public key $PK_{GSS} = r_{GSS} \cdot P$<br>$\underline{\{ID_{GSS}\}}$<br>(via secure channel) | |
| | Retrieves $\{HID_{list}, ID_{CS}, S_i, a_i\}$<br>in secure database<br>Generates a random number $c_i$<br>Computes<br>$Cert_j = h(ID_{GSS}\|\|c_i) + K_{NA} \cdot PK_{GSS}$<br>$X_i = h(HID_{list}\|\|K_{NA}\|\|a_i)$<br>$\underline{\{Cert_j, HID_{list}, ID_{CS}, X_i, S_i\}}$<br>(via secure channel) |

**Fig. 4.** Summary of ground station server registration phase.

### 5.2.3. Ground station server registration phase

To provide efficient UAV services for $MU_i$, $GSS_j$ must register with $NA$. This phase is shown in Fig. 4 and the overall steps for this phase are presented below.

- **GSR 1:** $GSS_j$ chooses an unique $ID_{GSS}$ and generates a private key $r_{GSS}$ and a public key $PK_{GSS} = r_{GSS} \cdot P$. Then, $GSS_j$ sends $\{ID_{GSS}\}$ to $NA$ over a secure channel.
- **GSR 2:** $NA$ retrieves $\{HID_{list}, ID_{CS}, S_i, a_i\}$ in secure database. Then, $NA$ generates a random number $c_i$ and computes $Cert_j = h(ID_{GSS} \parallel c_i) + K_{NA} \cdot PK_{GSS}$ and $X_i = h(HID_{list} \| K_{NA} \| a_i)$. Finally, $NA$ sends $\{Cert_j, HID_{list}, ID_{CS}, X_i, S_i\}$ to $GSS_j$ over a secure channel.
- **GSR 3:** $GSS_j$ stores $\{Cert_j, HID_{list}, ID_{CS}, X_i, S_i\}$ in secure database.

## 5.3. Login and authentication phase

Fig. 5 shows that $MU_i$ and $GSS_j$ authenticate with the help of $CS$ and generate the session key to access useful UAV services and the overall steps for this phase are presented below.

- **AP 1:** $MU_i$ inputs an unique $ID_i$ and $PW_i$. $MU_i$ calculates $HID_i = h(ID_i \parallel PW_i)$, $r_i = B_i \oplus h(PW_i \parallel HID_i)$, $HPW_i = h(PW_i \parallel r_i)$, $X_i = C_i \oplus h(HID_i \| HPW_i \| r_i)$, $Z_i = A_i \oplus h(HID_i \parallel X_i)$, and $D_i^* = h(HID_i \| HPW_i \| X_i \parallel r_i)$, and verifies $D_i^* \stackrel{?}{=} D_i$. If it is invalid, the current session is terminated, otherwise; $MU_i$ selects a random nonce $RN_i$ and a timestamp $T_1$ and calculates $U_1 = RN_i \oplus h(X_i \parallel HID_i)$, $U_{M-CS} = h(HID_i \parallel Z_i)$, and $U_{M-GSS} = h(HID_i \| RN_i \| X_i)$. Then, $MU_i$ sends $\{HID_i, U_1, U_{M-CS}, U_{M-GSS}, T_1\}$ to $CS$ via a public channel.
- **AP 2:** $CS$ checks a freshness of $T_1$ and then retrieves $\{Q_i, W_i\}$ in database. After that, $CS$ computes $(Z_i \parallel HID_{list}) = Q_i \oplus r_{CS}$, $S_i = W_i \oplus h(r_{CS} \parallel Z_i)$, and $U_{M-CS}^* = h(HID_i \| Z_i \| T_1)$

| Mobile user ($MU_i$) | Cloud server ($CS$) | Ground station server ($GSS_j$) |
|---|---|---|

Inputs $ID_i$ and $PW_i$
Computes
$HID_i = h(ID_i||PW_i)$
$r_i = B_i \oplus h(PW_i||HID_i)$
$HPW_i = h(PW_i||r_i)$
$X_i = C_i \oplus h(HID_i||HPW_i||r_i)$
$Z_i = A_i \oplus h(HID_i||X_i)$
$D_i^* = h(HID_i||HPW_i||X_i||r_i)$
Checks $D_i^* \stackrel{?}{=} D_i$
Generates a random nonce $RN_i$
and a timestamp $T_1$
$U_1 = RN_i \oplus h(X_i||HID_i)$
$U_{M-CS} = h(HID_i||Z_i||T_1)$
$U_{M-GSS} = h(HID_i||RN_i||X_i||T_1)$

$\{HID_i, U_1, U_{M-CS}, U_{M-GSS}, T_1\}$

Checks a freshness of $T_1$
Retrieves $\{Q_i, W_i\}$ in database
Computes
$(Z_i||HID_{list}) = Q_i \oplus r_{CS}$
$S_i = W_i \oplus h(r_{CS}||Z_i)$
$U_{M-CS}^* = h(HID_i||Z_i||T_1)$
Checks $U_{M-CS}^* \stackrel{?}{=} U_{M-CS}$
Generates a random nonce $RN_{CS}$ and a timestamp $T_2$
Computes
$CM_1 = RN_{CS} \oplus h(S_i||T_2)$
$CM_{CS-GSS} = h(RN_{CS}||S_i||T_2)$

$\{HID_i, U_1, CM_1, CM_{CS-GSS}, U_{M-GSS}, T_1, T_2\}$

Checks a freshness of $T_2$
Computes
$RN_{CS} = CM_1 \oplus h(S_i||T_2)$
$CM_{CS-GSS}^* = h(RN_{CS}||S_i||T_2)$
Checks $CM_{CS-GSS}^* \stackrel{?}{=} CM_{CS-GSS}$
Checks a freshness of $T_1$
Computes
$RN_i = U_1 \oplus h(X_i||HID_i)$
$U_{M-GSS}^* = h(HID_i||RN_i||X_i||T_1)$
Checks $U_{M-GSS}^* \stackrel{?}{=} U_{M-GSS}$
Generates a random nonce $RN_{GSS}$
and a timestamp $T_3$
$G_1 = RN_{GSS} \oplus h(HID_i||RN_i||X_i)$
$G_2 = RN_{GSS} \oplus h(RN_{CS}||T_3)$
$HID_i^{new} = h(HID_i||RN_{GSS})$
Matches and stores $HID_i$ with $HID_i^{new}$ in DB
$SK = h(RN_i||RN_{GSS}||X_i)$
$G_{GSS-CS} = h(HID_i||S_i||RN_{CS}||T_3)$
$G_{GSS-M} = h(HID_i||X_i||SK||T_3)$

$\{G_1, G_2, G_{GSS-CS}, G_{GSS-M}, T_3\}$

Checks a freshness of $T_3$
Computes
$G_{GSS-CS}^* = h(HID_i||S_i||RN_{CS}||T_3)$
Checks $G_{GSS-CS}^* \stackrel{?}{=} G_{GSS-CS}$
Computes
$RN_{GSS} = G_2 \oplus h(HID_i||RN_{CS})$
$HID_i^{new} = h(HID_i||RN_{GSS})$
Replaces $HID_i$ with $HID_i^{new}$
Generates a timestamp $T_4$
$CM_{CS-M} = h(Z_i||RN_i||HID_i||G_1||T_4)$

$\{G_1, CM_{CS-M}, G_{GSS-M}, T_3, T_4\}$

Computes
$C_{CS-M}^* = h(Z_i||RN_i||HID_i||G_1||T_4)$
Checks $CM_{CS-M}^* \stackrel{?}{=} CM_{CS-M}$
Computes
$RN_{GSS} = G_1 \oplus h(HID_i||RN_i||X_i)$
$SK = h(RN_i||RN_{GSS}||X_i)$
$G_{GSS-M}^* = h(HID_i||X_i||SK||T_3)$
Checks $G_{GSS-M}^* \stackrel{?}{=} G_{GSS-M}$
$HID_i^{new} = h(HID_i||RN_{GSS})$
Replaces $HID_i$ with $HID_i^{new}$

Both $MU_i$ and $GSS_j$ shares the common session key $SK$

**Fig. 5.** Summary of login and authentication phase.

and checks $U_{M-CS}^* \stackrel{?}{=} U_{M-CS}$. If it is not equal, the current session is incorrect, otherwise; $CS$ generates a random nonce $RN_{CS}$ and a timestamp $T_2$ and computes $CM_1 = RN_{CS} \oplus h(S_i \parallel T_2)$ and $CM_{CS-GSS} = h(RN_{CS} \| S_i \| T_2)$. Then, $CS$ sends $\{HID_i, U_1, CM_1, CM_{CS-GSS}, U_{M-GSS}, T_1, T_2\}$ to $GSS_j$.

- **AP 3:** $GSS_j$ checks a freshness of $T_2$ and computes $RN_{CS} = CM_1 \oplus h(S_i \parallel T_2)$, and $CM_{CS-GSS}^* = h(RN_{CS} \| S_i \| T_2)$ and verifies $CM_{CS-GSS}^* \stackrel{?}{=} CM_{CS-GSS}$. If it is incorrect, the current session is terminated, otherwise; $GSS_j$ calculates $RN_i = U_1 \oplus h(X_i \parallel HID_i)$, and $U_{M-GSS}^* = h(HID_i \| RN_i \| X_i \parallel T_1)$ and checks $U_{M-GSS}^* \stackrel{?}{=} U_{M-GSS}$. If it is invalid, the current session is terminated, otherwise; $GSS_j$ generates a random nonce $RN_{GSS}$ and a timestamp $T_3$ and computes $G_1 = RN_{GSS} \oplus h(HID_i \| RN_i \| X_i)$, $G_2 = RN_{GSS} \oplus h(RN_{CS} \parallel T_3)$, $HID_i^{new} = h(HID_i \parallel RN_{GSS})$. Then, $GSS_j$ matches $HID_i$ and $HID_i^{new}$ and then stores them in database. $GSS_j$ computes $SK = h(RN_i \| RN_{GSS} \| X_i)$, $G_{GSS-CS} = h(HID_i \| S_i \| RN_{CS} \parallel T_3)$, and $G_{GSS-M} = h(HID_i \| X_i \| SK \parallel T_3)$. Finally, $GSS_j$ sends $\{G_1, G_{GSS-CS}, G_{GSS-M}, T_3\}$ to $CS$.
- **AP 4:** $CS$ checks a freshness of $T_3$ and computes $G_{GSS-CS}^* = h(HID_i \| S_i \| RN_{CS} \parallel T_3)$ and verifies $G_{GSS-CS}^* \stackrel{?}{=} G_{GSS-CS}$. If it

is invalid, the current session is terminated, otherwise; $CS$ calculates $CM_{CS-M} = h(Z_i \| RN_i \| HID_i \| G_1 \| T_4)$. Finally, $CS$ sends $\{G_1, CM_{CS-M}, G_{GSS-M}, T_3, T_4\}$ to $MU_i$.

- **AP 5:** $MU_i$ checks a freshness of $T_4$ and computes $CM^*_{CS-M} = h(Z_i \| RN_i \| HID_i \| G_1 \| T_4)$ and checks $CM^*_{CS-M} \stackrel{?}{=} CM_{CS-M}$. If it is not equal, the current session is terminated, otherwise; $MU_i$ calculates $RN_{GSS} = G_1 \oplus h(HID_i \| RN_i \| X_i)$, $SK = h(RN_i \| RN_{GSS} \| X_i)$, and $G^*_{GSS-M} = h(HID_i tvert X_i \| SK \| T_3)$ and verifies $G^*_{GSS-M} \stackrel{?}{=} G_{GSS-M}$. If the condition is invalid, the current session is terminated, otherwise; $MU_i$ and $GSS_j$ establish a session key $SK$ successfully.

### 5.4. UAV data uploading phase

After getting the corrected information by $D_j$, $GSS_j$ generates the smart contract and uploads it in the blockchain. Then, $GSS_j$ generates encrypted UAV data and stores it in $CS$. The overall steps for this phase are as follows.

- **DUP 1:** $D_j$ computes $DID_j = h(ID_{D_j} \| K_{GD})$, $M_{DG} = E_{K_{GD}}(CID_k, D_{data})$, and $M_{SC} = h(DID_j \| CID_k \| K_{GD})$. Then, $D_j$ sends $\{DID_j, M_{DG}, M_{SC}\}$ to $GSS_j$.
- **DUP 2:** $GSS_j$ decrypts $D_{K_{GD}}(M_{DG})$ using the pre-shared secret key $K_{GD}$ and retrieves $\{ID^*_{D_j}\}$ in the database. Then, $GSS_j$ computes $DID^*_j = h(ID^*_{D_j} \| K_{GD})$ and checks $DID^*_j \stackrel{?}{=} DID_j$. If it is valid, $GSS_j$ computes $M^*_{SC} = h(DID_j \| CID_k \| K_{GD})$ and then verifies $M^*_{SC} \stackrel{?}{=} M_{SC}$. If it is correct, $GSS_j$ analyzes the data collected from $D_j$. For example, it will give predictions about the chances of the disaster condition, road condition, and so on. Then, $GSS_j$ generates a smart contract $SC = (ID_{D_j}, CID_k, Cert_j, D_{data})$ and publishes $SC$ in the blockchain.
- **DUP 3:** Then, $GSS_j$ computes $EDI = E_{SK}(D_{data}, ID_{D_j}, CID_k, T_{up})$, $M_{up} = E_{S_i}(EDI)$ and $M_{GC} = h(EDI \| S_i)$. $GSS_j$ sends $\{M_{up}, M_{GC}\}$ to the $CS$.
- **DUP 4:** After obtaining the messages from $GSS_j$, the $CS$ decrypts $D_{S_i}(M_{up})$ and computes $M^*_{GC} = h(EDI \| S_i)$ and verifies $M^*_{GC} \stackrel{?}{=} M_{GC}$. If it is correct, $CS$ stores $EDI$ in the database.

### 5.5. UAV data requesting phase

When $MU_i$ wants to access the UAV data in the system, $MU_i$ sends data request messages to $CS$. The overall steps for this phase are presented below.

- **DRP 1:** $MU_i$ generates request message $RE$ and computes $M_{req} = E_{Z_i}(RE \| HID_i)$ and $M_{MC} = h(RE \| HID_i)$. After that, $MU_i$ sends $\{M_{req}, M_{MC}\}$ to $CS$.
- **DRP 2:** After obtaining the messages from the $MU_i$, the $CS$ decrypts $D_{Z_i}(M_{req})$, computes $M^*_{MC} = h(RE \| HID_i)$ and verifies $M^*_{MC} \stackrel{?}{=} M_{MC}$. If it is correct, $CS$ retrieves $EDI$ corresponding request message. Then, $CS$ computes $M_{res} = E_{Z_i}(EDI)$, $M_{CM} = h(RE \| EDI \| HID_i)$ and sends $\{M_{res}, M_{CM}\}$ to $MU_i$.
- **DRP 3:** $MU_i$ decrypts $D_{Z_i}(M_{res})$ and computes $M^*_{CM} = h(RE \| EDI \| HID_i)$ and checks $M^*_{CM} \stackrel{?}{=} M_{CM}$. If it is valid, $MU_i$ decrypts $D_{SK}(EDI)$. Thus, $MU_i$ obtains various and useful UAV services.

### 5.6. Block construction and verification phase

We present the detailed descriptions for creating a block by a ground station server $GSS_j$ and then verifying that block by the blockchain networks after executing PBFT consensus algorithm.

| Block Header | |
|---|---|
| Block Version | $BVer$ |
| Previous Block Hash | $PBHash$ |
| Merkle Tree Root | $MTR$ |
| Timestamp | $TS$ |
| Owner of Block | $GSS_j$ |
| Public Key of Owner | $PK_{GSS_j}$ |
| **Block Payload (Transactions)** | |
| Transaction #1 | $T_{x_1}$ |
| Transaction #2 | $T_{x2}$ |
| ⋮ | ⋮ |
| Transaction #$n$ | $T_{x_n}$ |
| Current Block Hash | $CBHash$ |
| ECDSA Signature on Block | $ECDSA.Sig_{T_x}$ |

**Fig. 6.** Structure of a block on the transactions by $GSS_j$.

#### 5.6.1. Block formation phase

In this section, $CS$ transmits $EDI = E_{SK}(D_{data}, ID_{D_j}, CID_k, T_{up})$ to the $MU_i$ and generates the time $T_{ac}$ and then $MU_i$ has accessed to $EDI$ and sends $T_{ac}$ to $GSS_j$. After that, $GSS_j$ securely corrects $n$ number of information, filters those data, and also forms $n$ number of log transactions, note that $T_x = (HID_i, ID_{D_j}, T_{ac}, D_{loc}, D_{da}, Cert_j)$, which uses to the log transactions pool ($T_{x_{pool}}$). After that, $GSS_j$ computes the Merkle tree root (MTR) on these log transactions $T_{x_n}$ and calculates ECDSA-based signature on the transactions $T_{x_n}$ as $ECDSA.Sig_{T_{x_n}} = ECDSA.Sig_{gen}(T_{x_n})$. Once the number of log transactions in $T_{x_{pool}}$ attains to the pre-defined threshold value, a leader $GSS_j$ is elected based on a round-robin fashion from blockchain networks and the leader $GSS_j$ composes a detailed block ($Block_n$) as Fig. 6.

#### 5.6.2. Block verification and addition phase

After the formation of a $Block_n$ by the leader $GSS_j$ in blockchain networks, this phase is performed a voting-based PBFT consensus algorithm [34,35] to upload it to the blockchain. $GSS_j$ publishes $Block_n$ to all peer nodes in the blockchain networks for block verification. After getting the block, other peer nodes verifies $Block_n$ with existing $T_{x_{pool}}$. If all the log transactions presented in the $Block_n$ are verified with $T_{x_{pool}}$, the peer nodes put a vote into a commitment message pool ($CMP$). $GSS_j$ verifies $CMP$ and then if it attains to a minimal approval ($Min.App$) threshold for the $Block_n$ in the blockchain, where $Min.App = 2 * (n_{GSS_j} - 1)/3 + 1$ with $n_{GSS_j}$ is the number of peer nodes in blockchain networks, the newly generated $Block_n$ is added in blockchain. Simultaneously, other peer nodes add the $Block_n$ in their ledgers. The overall process of the block verification and addition phase is presented in Algorithm 1.

## 6. Security analysis

We prove the security of LAKA-UAV by performing "informal and formal security analyses such as ROR oracle model and AVISPA simulation".

### 6.1. Formal security analysis using ROR oracle model

This section evaluates "session key (SK) security of LAKA-UAV from the active/passive adversary $MA$ using the ROR oracle model [11]". We briefly introduce the ROR oracle model prior to evaluating SK security proof for the LAKA-UAV.

**Algorithm 1** PBFT Consensus for Block Verification and Addition in Blockchain

**Input**: Transaction pool ($T_{x_{pool}}$) threshold ($T_x = n$), number of peer nodes: $n_{GSS_j}$ minimal approval ($Min.App$), where $Min.App = 2 * (n_{GSS_j} - 1/3 + 1)$
**Output**: Commitment message pool (CMP) and block addition status
A leader ($GSS_j$) is elected by the round-robin fashion from blockchain networks
$GSS_j$ generates a block $Block_n$ with $T_{x_{pool}}$ as shown in Fig. 6.
$GSS_j$ sets $CMP \leftarrow \Phi$ (empty) and transmits $Block_n$ to other peer nodes $GSS_j (j = 1, 2, ..., n_{GSS_j})$ for voting request
$GSS_j$ receives $Block_n$ and validates it with $T_{x_{pool}}$
  **for** Each peer node $GSS_j$ **do**
**if** (($T_{x_i} = Valid$) and ($MTR = Valid$) and ($ECDSA.Sig_{T_x} = Valid$) and ($CBHash = Valid$)) **then**
  Set $CMP = CMP + 1$
**end if**
  **end for**
**if** ($|CMP| \geq Min.App$) **then**
  Add the block $Block_n$ to the blockchain
  Broadcast commitment message to the blockchain
**end if**

Table 2
Queries and descriptions.

| Query | Purpose |
|---|---|
| $Execute(\mathcal{P}^{t_1}_{MU}, \mathcal{P}^{t_2}_{CS}, \mathcal{P}^{t_3}_{GSS})$ | It is modeled that $MA$ performs the well-known attacks by eavesdropping exchanged messages between $MU$, $CS$, and $GSS$ over a public channel. |
| $CorruptMD(\mathcal{P}^{t_1}_{MU})$ | It denotes that $MA$ can extract secret credentials stored in the mobile device by performing power-analysis attack. |
| $Reveal(\mathcal{P}^t)$ | This query denotes that $MA$ can reveal the $SK$ created by its participant in the current session. |
| $Send(\mathcal{P}^t, M)$ | Using this query, $MA$ can send message $M$ to the instance $P^t$ and also receive the response message from $P^t$. |
| $Test(\mathcal{P}^t)$ | This query denotes the security of the session key among $MU$, $CS$, and $GSS$ following the ROR model. An unbiased coin $c$ is tossed prior to the game starts, and then the result is utilized to decide the output of the $Test$ query. If $MA$ performs this query and the session key $SK$ between $MU$ and $GSS$ is fresh, $P^t$ returns $SK$ if the condition $c = 1$ or a random number when $c = 0$. Otherwise, it returns the null value $(\perp)$. |

In LAKA-UAV, there are three participants: "the mobile user $P^{t_1}_{MU}$, the cloud server $P^{t_2}_{CS}$, and the ground station server $P^{t_3}_{GSS}$, where $P^{t_1}_{MU}$, $P^{t_2}_{CS}$, and $P^{t_3}_{GSS}$ are instances $t_1^{th}$ of $MU$, $t_2^{th}$ of $CS$, and $t_3^{th}$ of $GSS$", respectively. In Table 2, we define queries for the ROR oracle model such as $Corrupt$, $Execute$, $Send$, $Test$ and $Reveal$ to perform formal (mathematical) analysis. Moreover, a collision-resistant hash function $h(\cdot)$ is modeled as a random oracle $Hash$. We use Zipf's law [39] to evaluate SK security of LAKA-UAV.

**Theorem 1.** *Assume that $Adv^{LAKA-UAV}_{MA}$ is the advantage function of $MA$ in order to break SK security of LAKA-UAV. Then, we derive the following:*

$$Adv^{LAKA-UAV}_{MA} \leq \frac{q_h^2}{|Hash|} + 2\{C \cdot q^s_{send}\}$$

*where $q_h$, $q_{send}$, and $|Hash|$ are the "range space of hash function, the number of $Send$ query and the number of $Hash$", respectively. Moreover, $C$ and $s$ are parameters for the Zipf's law [39].*

**Proof.** The sequences of four games are denoted by $GM_i$, where ($i \in [0, 3]$). The advantage of $MA$ for winning the game $GM_i$ is denoted by $Adv^{LAKA-UAV}_{MA,GM_i} = Pr[Succ^{MA}_{GM_i}]$, where $Pr[E]$ is the probability of a random event $E$. The detailed descriptions of four games are shown in Game 0–3.

**Game** $GM_0$: The first game $GM_0$ is considered as an actual attack executed by $MA$ in LAKA-UAV. The bit $c$ is picked up randomly before the beginning of $GM_0$. According to this game $GM_0$, we obtain the following:

$$Adv^{LAKA-UAV}_{MA} = |2 \cdot Adv^{LAKA-UAV}_{MA,GM_0} - 1| \tag{1}$$

**Game** $GM_1$: This $GM_1$ denotes that $MA$ simulates an eavesdropping attack, in which exchanged messages are intercepted between $MU$, $CS$, and $GSS$ during the authentication phase using the $Execute$ query. In $GM_1$, $MA$ sends $Reveal$ and $Test$ queries. The output of the $Reveal$ and $Test$ queries decide if $MA$ obtains random nonces and $SK = h(RN_i \| RN_{GSS} \| X_i)$ between $MU$ and $GSS$. To derive $SK$, $MA$ needs random nonces ($RN_i, RN_{GSS}$), and secret credential ($X_i$). Thus, $MA$'s probability of winning $GM_1$ by eavesdropping on the transmitted messages do not increase. We get the following:

$$Adv^{LAKA-UAV}_{MA,GM_1} = Adv^{LAKA-UAV}_{MA,GM_0} \tag{2}$$

**Game** $GM_2$: This $GM_2$ is modeled as an active attack by $Hash$ and $Send$ queries. In the game $GM_2$, $MA$ can intercept all transmitted messages $\{HID_i, U_1, U_{M-CS}, U_{M-GSS}\}$, $\{HID_i, CM_1, CM_{CS-GSS}, U_{M-GSS}\}$, $\{G_1, G_{GSS-CS}, G_{GSS-M}\}$, and $\{G_1, CM_{CS-M}, G_{GSS-M}\}$ during the authentication phase. The random nonces $RN_i$, $RN_{CS}$, and $RN_{GSS}$ are not derived from intercepted messages because it is protected by hash function $h(\cdot)$. Thus, $GM_1$ and $GM_2$ are indistinguishable because the collision probability is negligible when $MA$ transmits $Send(P^t, M)$ query. By applying the birthday paradox [40], we obtain the following:

$$|Adv^{LAKA-UAV}_{MA,GM_1} - Adv^{LAKA-UAV}_{MA,GM_2}| \leq \frac{q_h^2}{2|Hash|} \tag{3}$$

**Game** $GM_3$: In this final game, $GM_3$ implements the simulation of the $CorruptMD$ query. $MA$ can extract secret credentials $\{A_i, B_i, C_i, D_i\}$ in the memory of the MD performing power analysis attacks. Note that, $A_i = Z_i \oplus h(HID_i \| X_i)$, $B_i = r_i \oplus h(PW_i \| HID_i)$, $C_i = X_i \oplus h(HID_i \| HPW_i \| r_i)$, and $D_i = h(HID_i \| HPW_i \| X_i \| r_i)$. However, $GM_3$ is computationally infeasible for $MA$ to derive the $PW_i$ through the $Send$ query without "real identity $ID_i$, random number $r_i$, and secret credential $X_i$". $GM_2$ and $GM_3$ are indistinguishable if the password guessing attack is not implemented. Based on Zipf's law on passwords [39], we get the following:

$$|Adv^{LAKA-UAV}_{MA,GM_2} - Adv^{LAKA-UAV}_{MA,GM_3}| \leq C' \cdot q^s_{send} \tag{4}$$

When all the games are executed, $MA$ tries to guess the exact bit $c$ to win the game using $Test$ query. Thus, we obtain the following:

$$Adv^{LAKA-UAV}_{MA,GM_3} = \frac{1}{2} \tag{5}$$

Eqs. (1), (2), and (5) provide

$$\begin{aligned} \frac{1}{2} Adv^{LAKA-UAV}_{MA} &= |Adv^{LAKA-UAV}_{MA,GM_0} - \frac{1}{2}| \\ &= |Adv^{LAKA-UAV}_{MA,GM_1} - \frac{1}{2}| \\ &= |Adv^{LAKA-UAV}_{MA,GM_1} - Adv^{LAKA-UAV}_{MA,GM_3}| \end{aligned} \tag{6}$$

Furthermore, Eqs. (4), (5), and (6) help to derive

$$\begin{aligned} \frac{1}{2} Adv^{LAKA-UAV}_{MA} &= |Adv^{LAKA-UAV}_{MA,GM_1} - Adv^{LAKA-UAV}_{MA,GM_3}| \\ &\leq |Adv^{LAKA-UAV}_{MA,GM_1} - Adv^{LAKA-UAV}_{MA,GM_2}| \\ &\quad + |Adv^{LAKA-UAV}_{MA,GM_2} - Adv^{LAKA}_{MA,GM_3}| \\ &\leq \frac{q_h^2}{2|Hash|} + C' \cdot q^s_{send} \end{aligned} \tag{7}$$

We obtain the following inequality as follows:

$Adv^{LAKA-UAV}_{MA} \leq \frac{q_h^2}{|Hash|} + 2C' \cdot q^s_{send}$. Since Eq. (7) is equivalent to Theorem 1, we prove the semantic security of LAKA-UAV

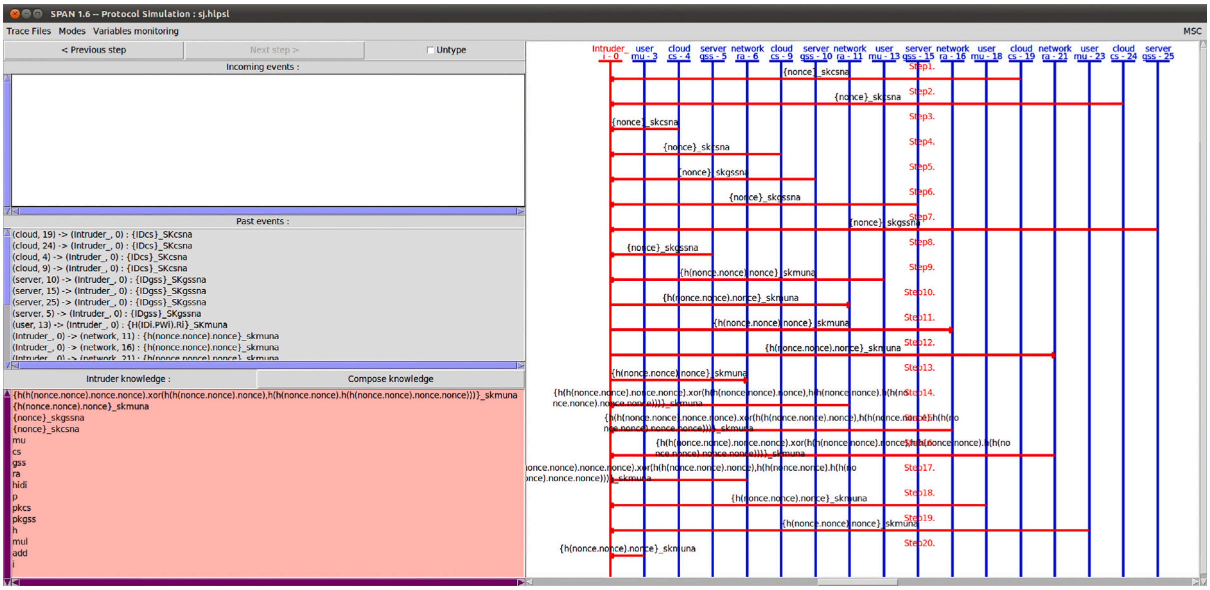


**Fig. 7.** AVISPA implementation results using SPAN.

| | |
|---|---|
| SUMMARY<br>**SAFE** | SUMMARY<br>**SAFE** |
| DETAILS<br>BOUNDED_NUMBER_OF_SESSIONS<br>TYPED_MODEL | DETAILS<br>BOUNDED_NUMBER_OF_SESSIONS |
| PROTOCOL<br>/home/span/span/testsuite/results/sjIoD.if | PROTOCOL<br>/home/span/span/testsuite/results/sjIoD.if |
| GOAL<br>As Specified | GOAL<br>As_specified |
| BACKEND<br>CL-AtSe | BACKEND<br>OFMC |
| COMMENTS<br>STATISTICS<br>Analysed : 3 states<br>Reachable : 3 states<br>Translation: 0.10 seconds<br>Computation: 0.01 seconds | COMMENTS<br>STATISTICS<br>parseTime: 0.00s<br>searchTime: 3.82s<br>visitedNodes: 412 nodes<br>depth: 9 plies |

**Fig. 8.** AVISPA evaluation results using CL-AtSe and OFMC.

### 6.2. Formal security analysis using AVISPA

We perform AVISPA simulation to prove security of LAKA-UAV against replay and MITM attacks. AVISPA simulation is implemented using "High-Level Protocol Specification Language (HLPSL) [41] to generate input format (IF)". There are four back-ends related to AVISPA simulation, including "Constraint Logic-based Attack Searcher (CL-AtSE), Tree automata based on Automatic Approximations for Analysis of Security Protocol (TA4SP), SAT-based Model Checker (SATMC), and On-the-Fly Model Checker (OFMC)". The output format (OF) is indicated the security of LAKA-UAV. More details for HLPSL and AVISPA specifications can be found in [12,13]. The HLPSL is implemented the various roles, including specification roles for $MU_i$, $CS$, and $GSS_j$, and mandatory roles for goal, environment, and session (see Fig. 7).

AVISPA simulation adopts the DY model, and can verify MITM and replay attacks. Based on the HLPSL, we implemented the security simulation for LAKA-UAV using the "security protocol animator (SPAN)". AVISPA evaluation results for the security of LAKA-UAV using "CL-AtSe and OFMC back-ends" are as shown in Fig. 8. Consequently, we prove that LAKA-UAV is resilient to various security attacks based on DY model because the simulation results are output as SAFE.

### 6.3. Informal security analysis

We present the "informal security analysis" to evaluate the security of LAKA-UAV. We demonstrated that LAKA-UAV can resist various security attacks and achieve authentication and anonymity.

#### 6.3.1. Impersonation attack

Suppose that $MA$ tries to impersonate by intercepting exchanged messages via a public channel. However, $MA$ cannot calculate the authentication request message $\{HID_i, U_1, U_{M-CS}, U_{M-GSS}, T_1\}$ because $MA$ does not obtain $MU_i$'s real identity $ID_i$, password $PW_i$, secret credentials $X_i$, and random nonce $RN_i$. Thus, LAKA-UAV prevents impersonation attacks.

#### 6.3.2. Replay attack

Assume that $MA$ attempts replay attacks using exchanged messages of the previous session via a public channel. However, all messages in LAKA-UAV are protected with random nonces $\{RN_i, RN_{CS}, RN_{GSS}\}$. Moreover, LAKA-UAV checks the timestamps to verify the freshness of messages. Therefore, even if $MA$ generates and sends new authentication messages, LAKA-UAV is resilient to replay attacks since the timestamp's freshness is invalid.

#### 6.3.3. MITM attack

Suppose that $MA$ can intercept exchanged messages over a nopen channel, then MITM attack may be possible. However, $MA$ cannot generate the authentication request/response messages since $MA$ cannot obtain secret credentials $\{X_i, Z_i, S_i\}$ and $NA$'s master key $K_{NA}$. Moreover, $MA$ cannot generate a session key $SK = h(RN_i \| RN_{GSS} \| X_i)$ without the random nonces $\{RN_i, RN_{CS}, RN_{GSS}\}$. Therefore, LAKA-UAV prevents MITM attacks.

#### 6.3.4. Session key disclosure attack

In the LAKA-UAV, $MA$ must obtain secret keys (long-term secret) $\{r_{CS}, r_{GSS}, K_{NA}\}$ and random nonces (short-term secret) $\{RN_i, RN_{CS}, RN_{GSS}\}$ to generate a correct session key $SK = h(RN_i \| RN_{GSS} \| X_i)$. However, $MA$ cannot calculate because $SK$ is encrypted with random nonces $\{RN_i, RN_{GSS}\}$ and secret credential $\{X_i\}$ using hash function. Thus, LAKA-UAV is secure against session key disclosure attacks.

*6.3.5. Mobile device stolen attack*

Referring to Section 3.1, we assume that $MA$ can steal the MD and extract the secret credentials $\{A_i, B_i, C_i, D_i\}$ in the MD using power analysis. However, $MA$ cannot obtain real identity $ID_i$, secret credential $X_i$, password $PW_i$, and random nonce $RN_i$ of a legitimate user using the secret credentials stored in the MD. Therefore, LAKA-UAV prevents mobile device stolen attacks.

*6.3.6. Off-line password guessing attack*

Referring to Section 3.1, we suppose that $MA$ can eavesdrop exchanged messages and extract secret credentials stored in the MD. In this attack, $MA$ attempts to off-line password guessing attacks to guess the $PW_i$ of the legitimate user. However, in the LAKA-UAV, the $PW_i$ is contained in $HPW_i = h(PW_i \parallel r_i)$. Thus, it is computationally infeasible for the $MA$ to guess the $PW_i$ without knowing the "real identity $ID_i$ and random number $r_i$". Consequently, LAKA-UAV is secure against off-line password guessing attacks.

*6.3.7. Ephemeral secret leakage (ESL) attack*

Based on CK model [31,32], we assume that $MA$ can compromise the session states and secret credentials such as random numbers apart from all the activities permitted under the DY model [29]. In LAKA-UAV, if only short-term secrets $\{RN_i, RN_{CS}, RN_{GSS}\}$ are exposed the session key between $MU_i$ and $GSS_j$ computed as $SK = h(RN_i \| RN_{GSS} \| X_i)$ is not compromised. On the other hand, if only long-term secrets $\{r_{CS}, r_{GSS}, K_{NA}\}$ are compromised, the session key $SK$ is not still exposed due to computationally infeasibility of Elliptic Curve Decisional Diffie–Hellman Problem (ECDDHP) [42]. In the LAKA-UAV, the session key $SK$ can only be revealed in a situation if $MA$ exposes both "short-term secret" and "long-term secret" credentials. Thus, LAKA-UAV is secure against ESL attacks.

*6.3.8. Desynchronization attack*

This attack is when $MA$ can block and eavesdrop the exchanged messages to make $MU_i$, $CS$, and $GSS_j$ unable to authenticate in the future. We assume that $MU_i$ does not receive the response message $\{G_1, G_{GSS-M}, T_3\}$ from $GSS_j$ via $CS$ because of malicious attacks or unexpected termination. However, $MA$ cannot perform this attack since $LAKA-UAV$ verifies whether $U^*_{GSS-M} \stackrel{?}{=} h(HID_i \| X_i \| SK \parallel T_3)$. If it is not equal, the current session is terminated. In a similar method, $CS$ will update $HID_i$ with $HID_i^{new}$. Moreover, if one of the following messages does not reach it to $MU_i$, $GSS_j$ and $MU_i$ will make desynchronized; they will be utilizing different parameters of $HID$. However, $GSS_j$ matches the $HID_i^{old}$ with $HID_i^{new}$ and then stores them in the database. Thus, LAKA-UAV is secure against desynchronization attacks.

*6.3.9. Known session-specific temporary information (KSSTI) attack*

Based on the CK model [31,32] This is an attack that assumes session random nonces are revealed and verify the session key security. In this attack, $MA$ can obtain random nonces $RN_i$, $RN_{CS}$, and $RN_{GSS}$ which are session-specific temporary information. $MA$ tries to generate a session key $SK = h(RN_i \| RN_{GSS} \| X_i)$ using random nonces. However, a session key $SK$ cannot be revealed since $MA$ cannot obtain the secret credential $X_i$. Therefore, LAKA-UAV prevents KSSTI attacks.

*6.3.10. Anonymity and untraceability*

Referring to Section 3.1, we assume that $MA$ can extract secret parameters stored in the MD, and intercept transmitted messages in the authentication phase. However, $MA$ cannot obtain real identities $\{ID_i, ID_{GSS}\}$ of all participants because the exchanged messages are encrypted with random number $\{r_{CS}, c_i\}$, $NA$'s master key $K_{NA}$, and password $PW_i$ using hash and XOR operations. Thus, LAKA-UAV guarantees secure anonymity of all participants.

Moreover, the random nonces and timestamps are different in any session, that is the exchanged messages in each session are dynamic and unique, so $MA$ cannot trace among $MU_i$, $CS$, and $GSS_j$ from different sessions. Consequently, LAKA-UAV achieves untraceability for $MU_i$, $CS$, and $GSS_j$.

*6.3.11. Mutual authentication*

In LAKA-UAV, all participants perform mutual authentication successfully. On receiving the login request message $\{U_{M-CS}\}$, $CS$ verifies $U^*_{M-CS} \stackrel{?}{=} h(HID_i \| Z_i \| T_1)$. If it is correct, $CS$ authenticates $MU_i$. After getting the authentication request messages from $MU_i$ and $CS$, $GSS_j$ checks $U^*_{M-GSS} \stackrel{?}{=} h(HID_i \| RN_i \| X_i \parallel T_1)$ and $CM^*_{CS-GSS} \stackrel{?}{=} h(RN_{CS} \| S_i \| T_2)$. If it is valid, $GSS_j$ authenticates $MU_i$ and $CS$. On receiving the authentication message $\{G_{GSS-CS}\}$, $CS$ verifies $G^*_{GSS-CS} \stackrel{?}{=} h(HID_i \| S_i \parallel RN_{CS} \| T_3)$. If it is correct, $CS$ authenticates $GSS_j$. After getting the authentication messages from $CS$ and $GSS_j$, $MU_i$ checks $C^*_{CS-M} \stackrel{?}{=} h(X_i | RN_i \| HID_i \| G_1 \parallel T_4)$ and $G^*_{GSS-M} \stackrel{?}{=} h(HID_i \| X_i \| SK \parallel T_3)$. If it is valid, $MU_i$ authenticates $CS$ and $GSS_j$. Thus, LAKA-UAV successfully provides secure mutual authentication among each entity.

## 7. Blockchain implementation

In this section, we discuss the simulation results of LAKA-UAV using the Hyperledger Sawtooth platform [15]. Hyperledger Sawtooth is an open-source Blockchain-as-a-Service (BaaS) platform under the Hyperledger Fabric [33] umbrella with highly modular architecture and supports a variety of consensus algorithms including "PBFT and Proof of Elapsed Time (PoET)". The nodes which provide consensus in the Sawtooth network are called validator nodes. The Sawtooth framework abstracts the business logic from its core system using transaction processors. A transaction processor validates a batch of transactions and updates state based on the rules defined by the application. The marquee feature of Sawtooth lies in its pluggable nature of transaction processors into validator nodes facilitating to run customized smart contracts, agnostic to technology and also without needing to know the entire core system. The core system allows different applications to co-exist on the same blockchain, selects transaction rules, selects the necessary permissive mechanism, and defines the consensus algorithms that are used to finalize the working of the digital ledger in a way that best supports the needs of an enterprise.

In the proposed LAKA-UAV simulation, we have made use of PBFT consensus mechanism for validating transactions. The structure of a block used in the simulation is presented in Fig. 6. We have considered that the parameters $BVer$, $PBHash$, $MTR$, $TS$, $GSS_j$, $PK_{GSS_j}$, $CBHash$, $BSign$ are of sizes 32, 256, 256, 32, 160, 320, 256 and 320 bits, respectively. Additionally, we have assumed that the size of a transaction $T_x$ is of atmost 1024 bits in size which results the size of a block to $(1632 + 1024 * n_t)$ bits, where $n_t$ is the number of transactions in a block. In our simulation, we have used 5 validator nodes each having the configuration: "Ubuntu 18.04, Intel(R) Core(TM) i9-9880H CPU @ 2.30 GHz, 2 GB RAM" and the performance of the proposed LAKA-UAV is evaluated for the following two cases.

- **Scenario 1:** Computation time in seconds versus number of transactions per block by having 5 validator nodes and mining 30 blocks.
- **Scenario 2:** Computation time in seconds versus number of blocks mined by having 5 validator nodes and 60 transactions per block.

In this simulation study, the synthetic data has been considered only. The data securely is gathered by the $GSS$s and then the data is used to form blocks. The simulation results for Scenario 1 and Scenario 2 are provided in Fig. 9 and Fig. 10, respectively.

It is worth noticing that the proposed consensus algorithm is based on the voting-based PBFT consensus algorithm [34,35] to upload the blocks to the blockchain. A PBFT-based voting algorithm requires the communication cost of $O(msg_c^{n_f})$, where $msg_c$ denotes the number of messages exchanged in the pre-prepare, prepare and commit phases executed by the $n_f$ servers participating in the consensus process. If we increase the number of nodes in the peer-to-peer (P2P) blockchain network (as shown in Fig. 10), it leads to increased time to reach

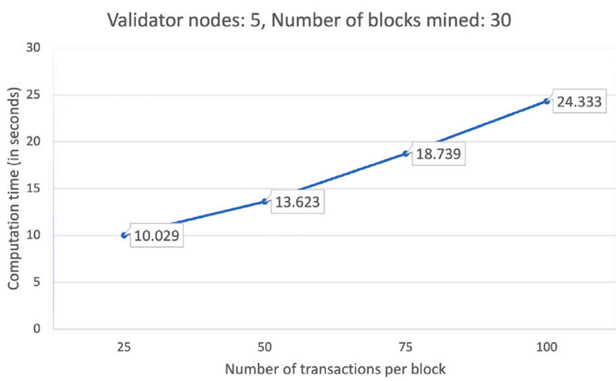


**Fig. 9.** Blockchain simulation—scenario 1.

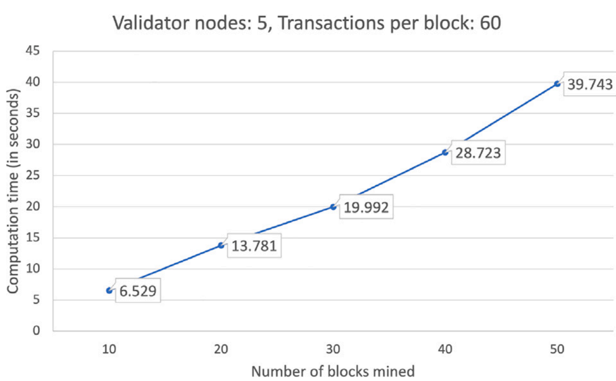


**Fig. 10.** Blockchain simulation—scenario 2.

consensus. This cost will be in addition to the communication cost computed for Algorithm 1. In Fig. 9, it is observed that when there is an increase in the number of transactions, it leads to an increase in the consensus computational time. This is primarily due to the fact that when the number of transactions in a block increases, each node in the P2P blockchain network needs to verify the signature, block hash and Merkle tree root for the increased number of transactions, which adds to the overall time taken by the consensus.

From both Figs. 9 and 10, it is worth observing that when the number of blocks miner or the number of transactions per block is increased, the computational time also increases.

## 8. Testbed experiments using MIRACL

In this section, we provide the "testbed experiments for measuring the computational time required for necessary cryptographic primitives used in the proposed scheme" and other relevant compared schemes using the MIRACL [14].

We use two scenarios for measuring the computational times of cryptographic primitives. We define $T_h$, $T_{ecm}$, $T_{fe}$, $T_{senc}$ and $T_{sdec}$ to estimate the execution time needed for a "hash function (for example, Secure Hash Algorithm (SHA-256) [43]), an elliptic curve point multiplication, a fuzzy extractor [44], a symmetric key encryption and a symmetric key decryption (for example, Advanced Encryption Standard (AES) [45])", respectively. It is assumed that $T_{fe} \approx T_{ecm}$.

- **Scenario 1:** We have considered a server setting as follows: "Model: MacBook Pro (2019), CPU Architecture: 64-bit, Processor: 2.3 GHz, Intel Core i9, Memory: 32 GB with OS: macOS Mojave 10.14.6". All cryptographic primitive has run for 100 times. After that the maximum, minimum and average time in milliseconds are observed for each primitive. The experimental results under server setting are tabulated in Table 3.

**Table 3**
Execution time (in milliseconds) using MIRACL for a server setting.

| Operation | Max. time (ms) | Min. time (ms) | Average time (ms) |
|---|---|---|---|
| $T_h$ | 0.053 | 0.023 | 0.024 |
| $T_{fe}$ | 0.540 | 0.337 | 0.382 |
| $T_{senc}$ | 0.002 | 0.001 | 0.001 |
| $T_{sdec}$ | 0.002 | 0.001 | 0.001 |

**Table 4**
Execution time (in milliseconds) using MIRACL for a Raspberry PI 3 setting.

| Operation | Max. time (ms) | Min. time (ms) | Average time (ms) |
|---|---|---|---|
| $T_h$ | 0.643 | 0.274 | 0.309 |
| $T_{fe}$ | 4.532 | 2.206 | 2.288 |
| $T_{senc}$ | 0.038 | 0.017 | 0.018 |
| $T_{sdec}$ | 0.054 | 0.009 | 0.014 |

- **Scenario 2:** We have considered Raspberry PI setting as "Model: Raspberry PI 3 B+ Rev 1.3, CPU Architecture: 64-bit, Processor: 1.4 GHz Quad-core, 4 cores, Memory(RAM): 1 GB with OS: Ubuntu 20.04 LTS, 64-bit". Similar to Scenario I, all cryptographic primitive has run for 100 times. The experimental results under Raspberry PI setting are provided in Table 4.

## 9. Comparative analysis

In this section, we perform a comparative analysis of LAKA-UAV in terms of "security features, computation costs, communication costs" with the baseline schemes of Wazid et al. [18], Srinivas et al. [20], and Ali et al. [21].

In [18], a "lightweight user authentication mechanism" was proposed in an Internet of Drones (IoD) environment. In this scheme, a legitimate user in IoD environment will be able to access the real-time data directly from the desirable drones only when that user is authorized to access the data from those drones. For this purpose, a mutual authentication between a user and the accessed drones is performed and a session key is established for secure data transfer. Since the scheme is centralized in nature, it is prone to a "single point of failure".

Srinivas et al. [20] suggested a "temporal credential based anonymous lightweight user authentication mechanism for IoD environment", known as TCALAS. In this approach, a legal registered external party or user will be interested in accessing the "real-time data from the designated drones residing in a particular fly zone" in the IoD environment. Their approach deals with a user authentication mechanism when a user can securely access the data directly from the accessed drones in the network. However, their scheme could not prevent "impersonation attacks" and it does also provide "user anonymity" [21].

Ali et al. [21] then proposed a "lightweight authentication scheme for UAVs communication in a smart city environment". However, their scheme fails to resist to "session key disclosure" and "denial of service (DoS)" attacks. In the following subsections, we explain the differences between these discussed baseline compared schemes and the proposed scheme.

### 9.1. Computation costs

We compare the computation costs of LAKA-UAV with existing schemes [18,20,21]. We use the "testbed experimental results for a server setting and a Raspberry PI 3 setting, which are measured for the computational time needed for various cryptographic primitives in Section 8". The experimental results for the average computational time needed for cryptographic primitives under $GSS$ or cloud server environment are considered with a server setting (as shown in Table 3). In this scenario, we have taken "$T_h \approx 0.024$ ms, $T_{fe} \approx 0.382$ ms,

**Table 5**
Comparative on computation costs.

| Scheme | Mobile user/Drone | GSS/Cloud server | Total computation costs |
|---|---|---|---|
| Wazid et al. [18] | $T_{fe} + 23T_h \approx 9.395$ ms | $8T_h \approx 0.192$ ms | 9.587 ms |
| Srinivas et al. [20] | $T_{fe} + 21T_h \approx 8.777$ ms | $9T_h \approx 0.216$ ms | 8.993 ms |
| Ali et al. [21] | $T_{fe} + 17T_h + T_{senc} \approx 7.559$ ms | $7T_h + T_{senc} + T_{sdec} \approx 0.170$ ms | 7.729 ms |
| Proposed (LAKA-UAV) | $13T_h \approx 4.017$ ms | $15T_h \approx 0.360$ ms | 4.377 ms |

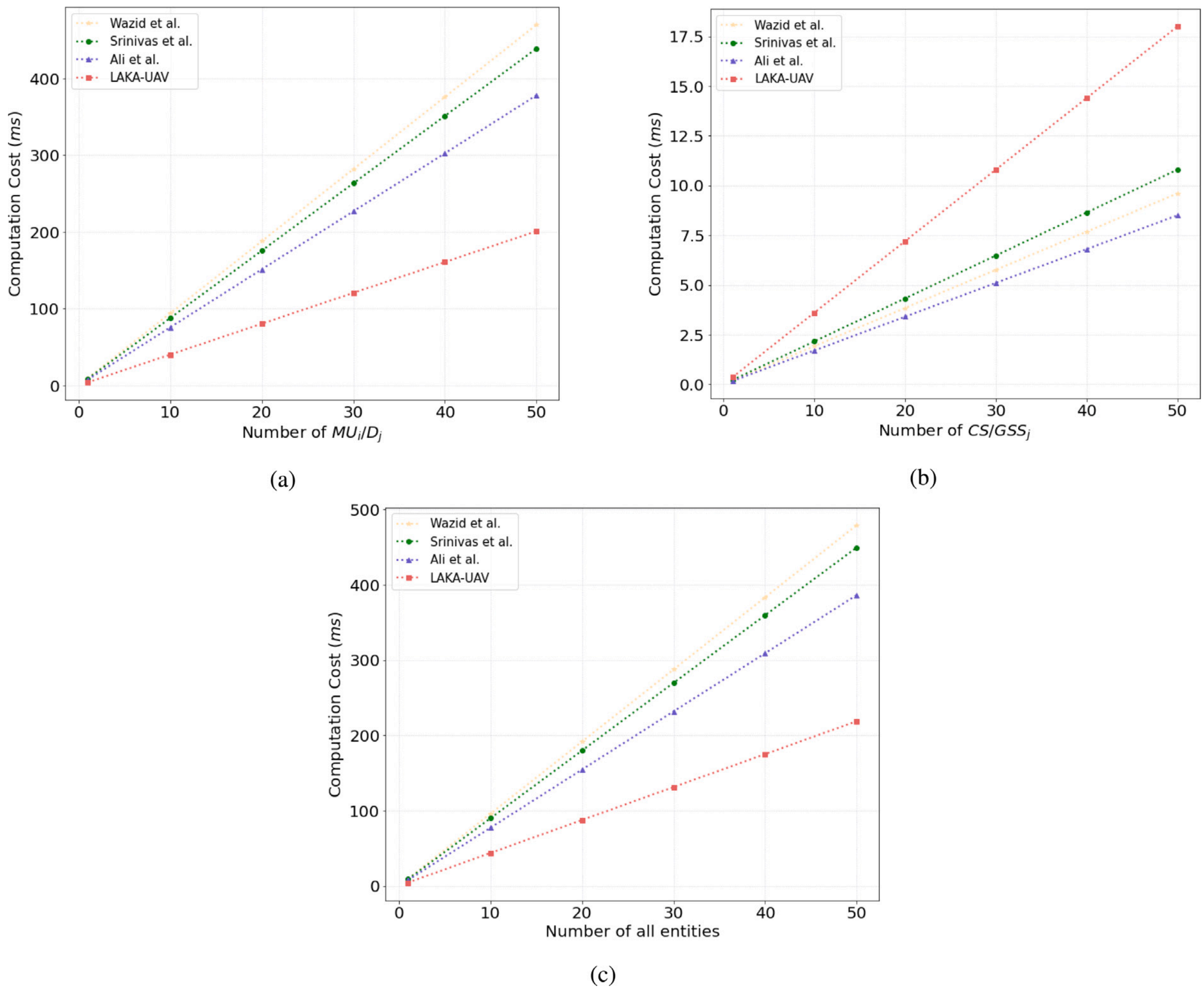


**Fig. 11.** Computation cost comparison of (a) mobile user/drone (b) GSS/Cloud Server (c) all entities.

$T_{senc} \approx 0.001$ ms and $T_{sdec} \approx 0.001$ ms". On the other side, we use the experimental results for the average computational time needed for cryptographic primitives under mobile user or drone environment with a Raspberry PI 3 setting (as shown in Table 4). Under this scenario, we have taken "$T_h \approx 0.309$ ms, $T_{fe} \approx 2.288$ ms, $T_{senc} \approx 0.018$ ms and $T_{sdec} \approx 0.014$ ms". We present the results of the computation cost comparison in Table 5 and Fig. 11. Consequently, LAKA-UAV ensures a more lightweight computation cost compared with the existing schemes.

### 9.2. Communication costs

We evaluate the communication costs of LAKA-UAV with related schemes [18,20,21] during the login and authentication phase in which the messages are exchanged by the registered participants. Referring to [21], we define that the bit lengths of "a random nonce, a timestamp, an identity, a hash (for example, SHA-256 hashing algorithm) and an ECC point are 160 bits, 32 bits, 160 bits, 256 bits, and 320 bits", respectively. Moreover, for symmetric encryption (for example, AES-128), plaintext and ciphertext blocks are considered as 128 bits. In the authentication phase of LAKA-UAV, the transmitted messages "$\{HID_i, U_1, U_{M-CS}, U_{M-GSS}, T_1\}$, $\{HID_i, U_1, CM_1, CM_{CS-GSS}, U_{M-GSS}, T_1, T_2\}$, $\{G_1, G_{GSS-CS}, G_{GSS-M}, T_3\}$, and $\{G_1, CM_{CS-M}, G_{GSS-M}, T_3, T_4\}$ require (256 + 256 + 256 + 256 + 32 = 1056 bits), (256 + 256 + 256 + 256 + 256 +32 + 32 = 1344 bits), (320 + 256 + 256 + 32 = 864 bits), and (320 + 256 + 256 + 32 + 32 = 896 bits)", respectively. Consequently, the total communication costs of LAKA-UAV are 4160 bits. Although LAKA-UAV has a higher communication cost than related schemes [18,20,21] and it offers cost-effective computation and storage costs than related schemes. We present the results of the communication costs comparison in Table 6 and Fig. 12.

### 9.3. Storage costs

We evaluate the storage costs of LAKA-UAV with related schemes [18,20,21]. We define that the bytes lengths of "the random nonce, identity, hash, ECC algorithm, fuzzy extractor algorithm, and

**Table 6**
Comparative on communication costs.

| Scheme | Number of messages | Total communication cost |
|---|---|---|
| [18] | 3 messages | 2656 bits |
| [20] | 3 messages | 2400 bits |
| [21] | 3 messages | 3296 bits |
| LAKA-UAV | 4 messages | 4160 bits |

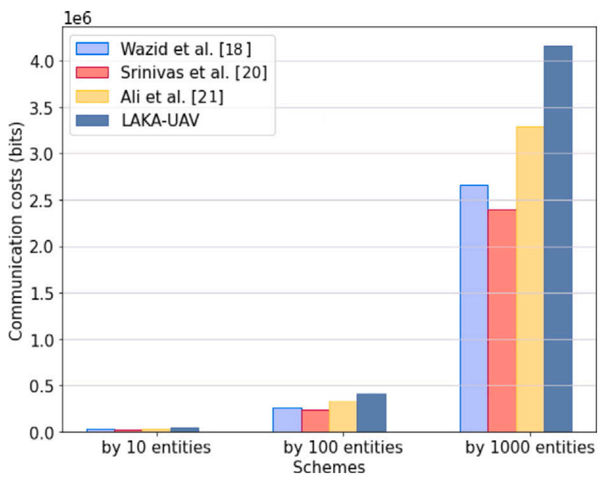


**Fig. 12.** Communication cost comparison.

**Table 7**
Comparative on storage costs.

| Scheme | Stored data($MU_i$) | Stored data($D_j$) | Stored data($GSS_j$) |
|---|---|---|---|
| [18] | 224 bytes | 112 bytes | 200 bytes |
| [20] | 168 bytes | 92 bytes | 132 bytes |
| [21] | 128 bytes | 92 bytes | 112 bytes |
| LAKA-UAV | 128 bytes | 52 bytes | 144 bytes |

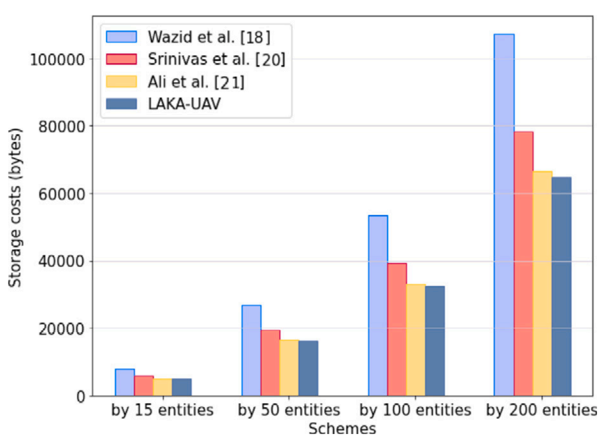


**Fig. 13.** Storage cost comparison.

AES algorithm are 20, 20, 32, 40, 40, and 16 bytes", respectively. In LAKA-UAV, stored data $\{A_i, B_i, C_i, D_i\}$ for the $MU_i$, $\{CID_k, K_{GD}\}$ for the $D_j$, and $\{Cert_j, HID_{list}, ID_{CS}, X_i, S_i\}$ for the $GSS_j$ require (32 + 32 + 32 + 32 = 128 bytes), (20 + 32 = 52 bytes), and (40 + 20 + 20 + 32 + 32 = 144 bytes). We present the analysis results for storage cost comparison in Table 7 and Fig. 13. Consequently, LAKA-UAV provides a more lightweight storage cost compared with existing schemes [18,20,21].

**Table 8**
Comparative on security features.

| Feature | Wazid et al. [18] | Srinivas et al. [20] | Ali et al. [21] | LAKA-UAV |
|---|---|---|---|---|
| $SFP_1$ | ∘ | ∘ | ∘ | ∘ |
| $SFP_2$ | ∘ | × | ∘ | ∘ |
| $SFP_3$ | ∘ | ∘ | ∘ | ∘ |
| $SFP_4$ | ∘ | ∘ | × | ∘ |
| $SFP_5$ | ∘ | ∘ | ∘ | ∘ |
| $SFP_6$ | ∘ | ∘ | ∘ | ∘ |
| $SFP_7$ | ∘ | ∘ | × | ∘ |
| $SFP_8$ | ∘ | ∘ | ∘ | ∘ |
| $SFP_9$ | ∘ | ∘ | × | ∘ |
| $SFP_{10}$ | ∘ | ∘ | ∘ | ∘ |
| $SFP_{11}$ | ∘ | ∘ | ∘ | ∘ |
| $SFP_{12}$ | × | × | ∘ | ∘ |
| $SFP_{13}$ | ∘ | ∘ | ∘ | ∘ |
| $SFP_{14}$ | ∘ | × | ∘ | ∘ |
| $SFP_{15}$ | ∘ | ∘ | ∘ | ∘ |
| $SFP_{16}$ | ∘ | ∘ | ∘ | ∘ |
| $SFP_{17}$ | × | ∘ | × | ∘ |

$SFP_1$: "Mobile device stolen attack"; $SFP_2$: "Impersonation attack"; $SFP_3$: "Offline password guessing attack"; $SFP_4$: "Session key disclosure attack"; $SFP_5$: "Replay attack"; $SFP_6$: "MITM attack"; $SFP_7$: "ESL attack under CK model"; $SFP_8$: "Desynchronization attack"; $SFP_9$: "DoS attack"; $SFP_{10}$: "KSSTI attack"; $SFP_{11}$: "Untraceability"; $SFP_{12}$: "Single point of failure"; $SFP_{13}$: "Mutual authentication"; $SFP_{14}$: "User anonymity"; $SFP_{15}$: "Perfect backward secrecy"; $SFP_{16}$: "Bottleneck"; $SFP_{17}$: "Formal (mathematical) analysis".

### 9.4. Security features

We present the "security features of LAKA-UAV compared to those of the related schemes [18,20,21]". Referring to Section 6, we proved that the proposed scheme is resistant to various security attacks and also guarantees necessary security requirements by performing formal and informal security analyses such as the ROR model and AVISPA simulation. According to Section 2, the previous literature shows that the related schemes are vulnerable to "various security attacks", and also their schemes cannot provide the necessary security requirements. In contrast, LAKA-UAV prevents "various security attacks" and also ensures "anonymity, mutual authentication, and single point of failure". Consequently, LAKA-UAV offers necessary security functionalities compared with existing schemes in Table 8.

## 10. Discussion and limitation

Our proposed scheme presents an authentication and key agreement mechanism for cloud-assisted UAV using blockchain in FANET that reduces the computation and communication overheads and provides a high-security level. Despite the fact that PBFT increases the communication cost between $GSS_j$, they have a stable wired connexion and an efficient consensus protocol that can reduce the computation cost and enhance scalability. Our scheme shows that it is suitable for a decentralized ledger. However, the transmission for ledger data still requires to depend on $GSS_j$. In future work, we will design a blockchain-based authentication and key agreement mechanism for UAV to reduce the dependency on the $GSS_j$ and enhance the communication overhead, which presents a system suitable for fully decentralized and more realistic scenarios.

## 11. Conclusions

We designed a lightweight and robust authentication and key agreement scheme for cloud-assisted UAV using blockchain in FANET to enhance data integrity, access control, storage overload, and security. We demonstrated that LAKA-UAV is secure against security attacks, and also ensures anonymity, bottlenecks, mutual authentication, and single point of failure. We then proved the session key security of LAKA-UAV by performing the ROR oracle model and demonstrated that LAKA-UAV is secure against replay and MITM attacks by performing AVISPA simulation. We presented the performance comparative analysis of LAKA-UAV with related schemes based on the testbed experiments using Raspberry 3-based MIRACL. Although the proposed LAKA-UAV has a higher communication cost than related schemes, it ensures lightweight computation and storage costs and also offers superior security features as compared to existing related schemes. Furthermore, the Blockchain implementation of LAKA-UAV is performed using the Hyperledger Sawtooth platform and the simulation results show that when the number of blocks miner or the number of transactions per block is increased, the computational time is also efficient.

## CRediT authorship contribution statement

**Sungjin Yu:** Conceptualization, Methodology, Writing – original draft, Formal analysis. **Joonyoung Lee:** Software, Validation. **Anil Kumar Sutrala:** Software, Formal analysis. **Ashok Kumar Das:** Writing – review & editing, Validation. **Youngho Park:** Writing – review & editing, Supervision.

## Declaration of competing interest

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

## Data availability

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

## Acknowledgments

This work was supported in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education under Grant 2020R1I1A3058605, and in part by the Electronics and Telecommunications Research Institute (ETRI) Grant funded by the Korean Government (Core Technology Research on Trust Data Connectome) under Grant 20ZR1300.

## References

[1] N.H. Motlagh, T. Taleb, O. Arouk, Low-altitude unmanned aerial vehicles-based internet of things services: Comprehensive survey and future perspectives, IEEE Internet Things J. 3 (6) (2016) 899–922.
[2] S.J. Yu, Y.H. Park, A robust authentication protocol for wireless medical sensor networks using blockchain and physically unclonable functions, IEEE Internet Things J. (2022) http://dx.doi.org/10.1109/JIOT.2022.3171791.
[3] S.H. Alsamhi, O. Ma, M.S. Ansari, F.A. Almalki, Survey on collaborative smart drones and internet of things for improving smartness of smart cities, IEEE Access 7 (2019) 128125–128152.
[4] M. Gharibi, R. Boutaba, S.L. Waslander, Internet of drones, IEEE Access 4 (2016) 1148–1162.
[5] C. Lin, D. He, N. Kumar, K.K.R. Choo, A. Vinel, X. Huang, Security and privacy for the internet of drones:Challenges and solutions, IEEE Commun. Mag. 56 (1) (2018) 64–69.
[6] Y.J. Chen, L.C. Wang, Privacy protection for internet of drones: A network coding approach, IEEE Internet Things J. 6 (2) (2019) 1719–1730.
[7] S. Zaidi, M. Atiquzzaman, C.T. Calafate, Internet of flying things (IoFT): A survey, Comput. Commun. 165 (1) (2021) 53–74.
[8] P. Mehta, R. Gupta, S. Tanwar, Blockchain envisioned UAV networks: Challenges, solutions, and comparisons, Comput. Commun. 151 (1) (2020) 518–538.
[9] T. Alladi, V. Chamola, N. Sahu, M. Guizani, Applications of blockchain in unmanned aerial vehicles: A review, Veh. Commun. 23 (2020) 100249–100273.
[10] M.A. Cheema, M.K. Shehzad, H.K. Qureshi, S.A. Hassan, H.J. Jung, A drone-aided blockchain-based smart vehicular network, IEEE Trans. Intell. Transp. Syst. (2020) http://dx.doi.org/10.1109/TITS.2020.3019246.
[11] M. Abdalla, P.A. Fouque, D. Pointcheval, Password-based authentication key exchange in the three-party setting, in: Public Key Cryptography, Les Diablerets, Switzerland, 2005, pp. 65–84.
[12] AVISPA, Automated validation of internet security protocols and applications, 2020, Available online: http://www.avispa-project.org/. (Accessed 16 June 2020).
[13] SPAN: A Security protocol animator for AVISPA, 2020, Available online: http://www.avispa-project.org/. (Accessed 16 June 2020).
[14] MIRACL cryptographic SDK: multiprecision integer and rational arithmetic cryptographic library, 2020, [Online]. Available: https://github.com/miracl/MIRACL. (Accessed April 2020).
[15] Hyperledger sawtooth architecture guide, 2020, Available online: https://sawtooth.hyperledger.org/docs/core/releases/1.1/architecture.html. (Accessed 17 December 2020).
[16] Y. Zhang, D. He, L. Li, B. Chen, A lightweight authentication and key agreement scheme for internet of drones, Comput. Commun. 154 (15) (2020) 455–464.
[17] B.D. Deebak, F.A. Turjman, A smart lightweight privacy preservation scheme for IoT-based UAV communication systems, Comput. Commun. 162 (1) (2020) 102–117.
[18] M. Wazid, A.K. Das, N. Kumar, A.V. Vasilakos, J.J.P.C. Rodrigues, Design and analysis of secure lightweight remote user authentication and key agreement scheme in internet of drones deployment, IEEE Internet Things J. 6 (2) (2019) 3572–3584.
[19] T. Alladi, V. Chamola, Naren, N. Kumar, PARTH: A two-state lightweight mutual authentication protocol for UAV surveillance networks, Comput. Commun. 160 (1) (2020) 81–90.
[20] J. Srinivas, A.K. Das, N. Kumar, J.J.P.C. Rodrigues, TCALAS: Temporal credential-based anonymous lightweight authentication scheme for internet of drones environment, IEEE Trans. Veh. Technol. 68 (7) (2019) 6903–6916.
[21] Z. Ali, S.A. Chaudhry, M.S. Ramzan, F.A. Turjman, Securing smart city surveillance: A lightweight authentication mechanism for unmanned vehicles, IEEE Access 8 (2020) 43711–43724.
[22] B.D. Deebak, F.A. Turjman, A smart lightweight privacy preservation scheme for IoT-based UAV communication systems, Comput. Netw. 162 (2020) 102–117.
[23] A. Koubaa, B. Qureshi, DroneTrack: Cloud-based real-time object tracking using unmanned aerial vehicles over the internet, IEEE Access 6 (2018) 13810–13824.
[24] C. Hong, D. Shi, A cloud-based control system architecture for multi-UAV, in: ICRCA'18: Proceedings of the 3rd International Conference on Robotics, Control and Automation, New York, United States, 2018, pp. 25–30.
[25] A. Koubaa, B. Qureshi, M.F. Sriti, A. Allouch, Y. Javed, M. Alajlan, O. Cheikhrouhou, M. Khalgui, E. Tovar, Dronemap planner: A service-oriented cloud-based management system for the internet-of-drones, Ad Hoc Netw. 86 (1) (2019) 46–62.
[26] P. Mehta, R. Gupta, S. Tanwar, Blockchain envisioned UAV networks: Challenges, solutions, and comparisons, Comput. Commun. 151 (1) (2020) 518–538.
[27] R. Ch, G. Srivastava, T.R. Gadekallu, P.K.R. Maddikunta, S. Bhattacharya, Security and privacy of UAV data using blockchain technology, J. Inform. Secur. Appl. 55 (2020) 102670–102681.
[28] A. Yazdinejad, R.M. Parzi, A. Dehghantanha, H. Karimipour, G. Srivastava, M. Aledhari, Enabling drones in the internet of things with decentralized blockchain-based security, IEEE Internet Things J. (2020) http://dx.doi.org/10.1109/JIOT.2020.3015382.
[29] D. Dolev, A.C. Yao, On the security of public key protocols, IEEE Trans. Inform. Theory 29 (2) (1983) 198–208.
[30] P. Kocher, J. Jaffe, B. Jun, Differential power analysis, in: Advances in Cryptology, CRYPTO, Berlin, Germany, 1999, pp. 388–397.
[31] R. Canetti, 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, Netherlands, 2002, pp. 337–351.
[32] J.Y. Lee, C.H. Kim, A.K. Das, Y.H. Park, Secure and efficient honey list-based authentication protocol for vehicular ad hoc networks, IEEE Trans. Netw. Sci. Eng. 8 (2021) 2412–2425.
[33] Hyperledger: open source blockchain technologies, 2020, Available online: https://www.hyperledger.org/. (Accessed 17 December 2020).
[34] M.H. Kim, S.J. Yu, J.Y. Lee, Y.H. Park, Y.H. Park, Design of secure protocol for cloud-assisted electronic health record system using blockchain, Sensors 20 (2020) 2913–2934.
[35] N. Lu, Y. Zhang, W. Shi, S. Kumari, K.K.R. Choo, A secure and scalable data integrity auditing scheme based on hyperledger fabric, Comput. Secur. 92 (2020) 101741–101757.
[36] V. Hassija, V. Chamola, A. Agrawal, A. Goyal, N.C. Luong, D. Niyato, F.R. Yu, M. Guizani, Fast, reliable, and secure drone communication: A comprehensive survey, IEEE Commun. Surv. Tutor. 23 (4) (2021) 2802–2832.
[37] S.J. Yu, A.K. Das, Y.H. Park, P. Lorenz, SLAP-IoD, Secure and lightweight authentication protocol using physical unclonable functions for internet of drones in smart city environments, IEEE Trans. Veh. Technol. 71 (10) (2022) 10374–10388.
[38] N. Koblitz, Elliptic curve cryptosystems, Math. Comput. 48 (177) (1987) 203–209.
[39] D. Wang, H. Cheng, P. Wang, X. Huang, G. Jian, Zipf's law in passwords, IEEE Trans. Inf. Forensics Secur. 12 (11) (2017) 2776–2791.
[40] V. Boyko, P. Mackenzie, S. Patel, Provably secure password-authenticated key exchange using Diffie–Hellman, in: Proc. Int. Conf. Theory Appl. Crypto. Tech. Adv. Cryptol, EUROCRYPT, Bruges, Belgium, 2000, pp. 156–171.

[41] D.V. Oheimb, The high-level protocol specification lanuage HLPSL developed in the EU project AVISPA, in: Proc. of the APPSEM 2005 Workshop, Tallinn, Finland, 2005, pp. 1–17.

[42] S. Kumari, K. Renuka, Design of a password authentication and key agreement scheme to access E-healthcare services, Wirel. Pers. Commun. (2019) http://dx.doi.org/10.1007/s11277-019-06755-7.

[43] Secure Hash Standard, FIPS PUB 180-1, National Institute of Standards and Technology (NIST), U.S. Department of Commerce, 1995, http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf. (Accessed January 2021).

[44] Y. Dodis, L. Reyzin, A. Smith, Fuzzy extractors: how to generate strong keys from biometrics and other noisy data, in: Proceedings of the Advances in Cryptology, Eurocrypt'04, in: LNCS, vol. 3027, 2004, pp. 523–540.

[45] Advanced Encryption Standard (AES), FIPS PUB 197, National Institute of Standards and Technology (NIST), U.S. Department of Commerce, 2001, http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf. (Accessed January 2021).


**Sungjin Yu** "received the B.S. degree in electronics engineering from Daegu University, and also the M.S. degree in electronics engineering from Kyungpook National University, Daegu, South Korea, in 2017, and 2019, respectively, where he is currently pursuing the Ph.D. degree with electronics and electrical engineering. He is currently a researcher with the Electronics and Telecommunications Research Institute (ETRI), Daejeon, South Korea. His research interests include blockchain, authentication, information security, VANET, FANET, Internet of Vehicles, Internet of Drones, AI security, and Metaverse security".

**Joonyoung Lee** "received the B.S. and M.S. degrees in electronic and electrical engineering from Kyungpook National University, Daegu, South Korea, in 2018 and 2020, respectively, where he is currently pursuing the Ph.D. degree with the School of Electronics Engineering. His research interests include information security, Internet of Things, and authentication".

**Anil Kumar Sutrala** "received his Ph.D. degree in computer science and engineering from the International Institute of Information Technology (IIIT), Hyderabad, India, in 2018 and also M.C.A. from the University of Hyderabad, India, in 2006. He is currently working as a principal software engineer with the CA Technologies – A Broadcom Company, Hyderabad 500 032, India. His research interests include cryptography and network security. He has published several journal articles in his research areas".

**Ashok Kumar Das** "received a Ph.D. degree in computer science and engineering, an M.Tech. degree in computer science and data processing, and an M.Sc. degree in mathematics from IIT Kharagpur, India. He is currently an Associate Professor with the Center for Security, Theory and Algorithmic Research, International Institute of Information Technology, Hyderabad, India. His current research interests include cryptography, system and network security including security in vehicular ad hoc networks, smart grids, smart homes, Internet of Things (IoT), Internet of Drones, Internet of Vehicles, Cyber–Physical Systems (CPS) and cloud computing, intrusion detection, blockchain and AI/ML security. He has authored over 295 papers in international journals and conferences in the above areas, including over 250 reputed journal papers". He was a recipient of the Institute Silver Medal from IIT Kharagpur. He is on the editorial board of IEEE Systems Journal, Journal of Network and Computer Applications (Elsevier), Computer Communications (Elsevier), Journal of Cloud Computing (Springer), Cyber Security and Applications (Elsevier), IET Communications, KSII Transactions on Internet and Information Systems, and International Journal of Internet Technology and Secured Transactions (Inderscience), and has served as a Program Committee Member in many international conferences. He also severed as one of the Technical Program Committee Chairs of the first International Congress on Blockchain and Applications (BLOCKCHAIN'19), Avila, Spain, June 2019, International Conference on Applied Soft Computing and Communication Networks (ACN'20), October 2020, Chennai, India, and second International Congress on Blockchain and Applications (BLOCKCHAIN'20), L'Aquila, Italy, October 2020. His Google Scholar h-index is 62 and i10-index is 188 with over 11,600 citations".

**Youngho Park** "received his B.S., M.S., and Ph.D. degrees in electronic engineering, Kyungpook National University, Daegu, Korea in 1989, 1991, and 1995, respectively. He is currently a professor at School of Electronics Engineering, Kyungpook National University. In 1996–2008, he was a professor at School of Electronics and Electrical Engineering, Sangju National University, Korea. In 2003–2004, he was a visiting scholar at School of Electrical Engineering and Computer Science, Oregon State University, USA. His research interests include information security, computer networks, and multimedia".