Received July 3, 2019, accepted July 21, 2019, date of publication July 24, 2019, date of current version August 12, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2930774

BUS: A Blockchain-Enabled Data Acquisition Scheme With the Assistance of UAV Swarm in Internet of Things

ANIK ISLAM AND SOO YOUNG SHIN, (Senior Member, IEEE)
Kumoh National Institute of Technology, Gumi 39177, South Korea

Corresponding author: Soo Young Shin (wdragon@kumoh.ac.kr)

This research was supported by the MSIT (Ministry of Science, ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2019-2014-1-00639) supervised by the IITP (Institute for Information & communications Technology Planning & Evaluation).

ABSTRACT This paper presents a blockchain enabled secure data acquisition scheme utilizing an unmanned aerial vehicle (UAV) swarm where data are collected from the Internet of Things (IoT) devices and subsequently, forwarded to the nearest server through the UAV swarm. Before initiating data acquisition, the UAV swarm shares a shared key with the IoT devices in order to maintain communications. However, prior to transmitting data, the IoT devices encrypt the data and forward it to the UAV swarm. Upon receiving the data, the UAV swarm implements a two-phase validation utilizing the π-hash bloom filter and the digital signature algorithm to validate the sender; in addition, prior to forwarding data to the nearest server, it performs encryption. However, before adding data in blockchain, consent from all validators is required. Finally, the data are stored in blockchain with the approval of validators. A security analysis is performed to demonstrate the feasibility of the proposed scheme. Finally, the effectiveness of the proposed scheme is manifested through the implementation and simulation. The security analysis and the performance results show that UAV assist the IoT devices both in terms of connectivity and energy consumption, and provides security against the threats mentioned in the paper.

INDEX TERMS Blockchain, data acquisition, security, IoT, UAV swarm.

I. INTRODUCTION

Internet of Things (IoT) is an emerging technology that connects objects (termed as “things”), consisting of sensors, actuators, software, etc., via Internet. IoT devices can sense the environment as well as perform actions based on sensors data [1]. As the popularity of IoT increases, it has been estimated that approximately 30 billion IoT devices will be attached to this network by 2020 [2]. IoT brings great benefits across various sectors, including agriculture [3]–[6], healthcare [7]–[11], smart grids [12]–[16], smart homes [17]–[19], and others [20]. However, challenges such as connectivity, low computation power, and insufficient battery life still exist for IoT devices [21].

Unmanned Aerial Vehicles (UAVs) have drawn tremendous attention both in the industry and academia [22]. Although UAVs were initially designed for military purposes, they have been exploited to civil applications by virtue of their promising functionalities such as ease of deployment, low maintenance cost, and availability [23], [24]. Moreover, UAVs are currently equipped with a high computational power, sensors, actuators, which can not only collect but also process data, make decisions and perform actions following a decision [25]–[27]. Moreover, forming a swarm of UAVs does not just increase the quality of service, but also extends the application area for UAVs [28]. In a UAV swarm, if one of the UAVs becomes compromised, others can continue the mission [29]. Moreover, a UAV swarm can cover more region than a single UAV and perform longer than a single UAV [30]. Employing a UAV swarm in the consideration of the issues above can be a potential solution. A UAV swarm can extend network connectivity to IoTs for the non-line-of-sight region. Moreover, a UAV swarm can conserve the energy of IoTs by serving as a relay to forward the data and can be readily substituted in the event of a fault, while

The associate editor coordinating the review of this manuscript and approving it for publication was Usama Mir.

providing assistance to IoTs. Furthermore, a UAV swarm can be deployed to remote locations to assist IoT devices when deploying manpower is very difficult. Besides, a UAV swarm covers larger areas, which can assist in reducing deployment costs. However, communications between the UAV and the IoT, and between the UAV and the server are fraught with cyber threats including the man in the middle attacks, replay attacks, etc. Moreover, the accumulated data may undergo illegal alterations in the server which may raise integrity issue. Therefore, a scheme is required to ensure a secure data acquisition process and to maintain data integrity.

Among the existing researches, [28] proposed a UAV-Edge-Cloud computing model utilizing a UAV swarm to provide real-time support to the users and finally end data are stored in the cloud server. In [31], an open source UAV swarming platform (termed as "EasySwarm") was proposed with LoRa for the communication and a low-latency channel access protocol at the MAC layer. A UAV-based communication architecture for assisting body area networks (BAN) was proposed in [32]. Here, a UAV collected data from multiple BANs and transmitted to different servers (i.e. medical servers, physicians, and cloud servers). In [27], a cloud-based UAV platform was proposed, where emergency data were forwarded to the cloud. Reference [33] overviewed flying ad hoc networks (FANET) schemes, different algorithms for distributed gateway-selection, and cloud-oriented stability-control for multi-UAV in conjunction with future directions for future research. In [34], a multiantenna UAV relayed data acquisition scheme from a cluster of single-antenna IoT devices was proposed considering a hybrid channel model together with the large-scale and the small-scale channel fading. Reference [35] proposes a UAV assisted IoT network utilizing 5G for the future smart city. In [35], a hierarchy based UAV architecture is utilized in which lower UAVs collected data and transmitted it to the leader UAV which is used as a data fusion center. Reference [36] proposed a two-stage joint hovering altitude and power control for UAV in the consideration of a space-air-ground communication network to assist IoT applications. In [37], a UAV-assisted sensor search based on the content was proposed to handle multiple-content queries in an efficient way. A cost-effective drone based communication scheme was investigated in order to perform power transfer and communications simultaneously in [38]. In [39], a UAV-based fog computing platform was presented in order to provide support IoT application in the consideration of remote or challenging location. An analysis regarding the utilization of signal UAV vs. multiple UAV in the assistance of a wireless sensor network for collecting data was proposed in [40]. However, none of the aforementioned schemes considered any security concerns while utilizing UAVs in data acquisition from IoTs, as summarized in Table 1. In addition, they did not provide any information concerning the security of data after its collection from IoT, which may raise the issue of its integrity.

Blockchain is a digital ledger that is distributed among peers, with each peer holding the same copy of the data [41]. Satoshi Nakamoto first explains blockchain in his white paper "Bitcoin: A Peer-to-Peer Electronic Cash System" in 2008 [42]. A smart contract is a computerized script that can validate and execute instructions automatically [43]. However, blockchain adopts asymmetric encryption (private/public keys) to maintain communication. In blockchain, a public key is used as the identity of the user and a private key is utilized to validate the data [44]. Moreover, blockchain employs a Merkle tree[1] in blocks. In the Merkle tree, when a value is changed, the hash of the whole tree also changes. However, before adding data into blockchain, every miner has to come to an agreement over the validity of the data [46]. After getting acceptance from everyone, data are included in blockchain. After adding data to blockchain, no modification can be made. If anyone tries to make a change in the block, the hash of the block also changes and breaks the chain of blocks. To reconstruct the chain, all the validators must agree on this change. Thus, the data in blockchain remain secured. These blockchain features can be a potential solution to the aforementioned security threats (i.e., cyber attacks, data integrity issue).

In this paper, a blockchain enabled secure data acquisition scheme is proposed, in which data are collected from IoT devices employing a UAV swarm and stored in blockchain at the server. Providing security and data integrity utilizing blockchain in data acquisition of IoT devices via a UAV swarm has not been explored yet to the best of our knowledge. The major contributions of this paper are compiled as follows.

- A blockchain-enabled secure data acquisition scheme utilizing a UAV swarm is proposed.
- A two-phase device validation mechanism, utilizing a π-hash bloom filter[2] and a digital signature[3] algorithm is proposed.
- A security analysis based on various vulnerabilities is discussed.
- The effect of applying the π-hash bloom filter in the proposed scheme is simulated using both MATLAB (in the server) and Python (in the UAV). The performance of the π-hash bloom filter is discussed in terms of processing time, expected transmission of data, validation time, and energy consumption.
- An experimental setup is discussed, and the performance of the experiment is investigated in terms of throughput, processing time, and energy consumption. Blockchain is implemented on the top of Ethereum and the performance of blockchain is examined in terms of throughput, read, and latency.

The remainder of this paper is organized as follows: Section II illustrates the blockchain-enabled UAV swarm

[1]A Merkle tree is a hash tree that holds the hash of data in every leaf and parent of these leaf holds the hash of the child leaf [45].

[2]A bloom filter is a probabilistic and memory-efficient data structure applied to verify whether an element exists in a set [47].

[3]The digital signature is a virtual signature that is generated by the hash of the message utilizing the sender's private key and that signature is only verified by the public key of the sender [48].

TABLE 1. Summary of the existing researches along with their drawbacks.

Existing schemes	Details	Issues
Chen et al. [28]	A UAV-Edge-Cloud utilized UAV swam for data acquisition.	None of the existing researches considered security issues (i.e., cyber threats and data integrity issue).
Yuan et al. [31]	A UAV swarming platform used LoRa for communication.	
Sana et al. [32]	A UAV-assisted data acquisition scheme to collect health information from body area networks.	
Datta et al. [33]	A cloud-based UAV scheme that stored data to the cloud.	
Wang et al. [34]	A survey containing FANET schemes, different gateway-selection algorithms, and stability-control mechanisms for multi-UAV.	
Feng et al. [35]	A multiantenna UAV relayed data acquisition scheme for IoT devices.	
Qi et al. [36]	A UAV assisted IoT network utilizing 5G for the future smart city.	
Wang et al. [37]	A two-stage joint hovering altitude and power control scheme for UAV networks.	
Sharma et al. [38]	A UAV-assisted sensor search based scheme on IoT devices.	
He et al. [39]	A cost-effective drone based communication that transmits power and data at the same time.	
Mohamed et al. [40]	A UAV-based fog computing platform to support IoT devices.	
Wei et al. [41]	Performance analysis for data collection from WSN.	

assisted data acquisition scheme. In Section III, a secure UAV swarm assisted data acquisition mechanism is discussed in detail. A security analysis of the proposed scheme is represented in Section IV. A discussion of the simulation and experimental set-up along with the results obtained is described in Section V. Finally, Section VI concludes with future works.

II. PROPOSED UAV SWARM ASSISTED DATA ACQUISITION SCHEME

A blockchain-enabled data acquisition scheme (termed as ''BUS'') is devised to support the data acquisition from IoT Devices with the assistance of the UAV swarm, as demonstrated in FIGURE 1.

A. COMPONENTS OF BUS

BUS consists of the following components:

- IoT devices: IoT devices collect data from the environment and transmit this data to the server with the assistance of the UAV swarm. IoT Devices contain sensors and actuators that assists with the collection of data from the environment and also assists with performing an action based on the sensor data.
- UAV Swarm: the UAV swarm collects data from IoT devices and transmits this data to the nearest server. However, during the data acquisition mission, UAV swarm creates a hierarchy in roles to improve the quality of service. The hierarchy is depicted as follows:
  - Minion UAV (MUAV): MUAV stays close to IoT devices and directly collects data from them. Each MUAV has its own area to monitor and perform data acquisition. Each of the MUAVs maintains its own shared key in order to expedite communication with IoT devices and decrease power consumption when performing the security mechanism (i.e., sign, verify, encrypt, and decrypt).
  - Emissary UAV (EUAV): EUAV stays at the top of the hierarchy. EUAV collects data from MUAVs and forwards this data to the server. Prior to transmitting, the EUAV validates the authenticity of the sender. Upon successful validation, the EUAV forwards the data to the server. The EUAV transmits not only the IoT data but also the flight information along with information on the role of each UAV.
- Server: the server holds the data of IoT devices and each data is stored in blockchain. The server also maintains information on the UAV and IoT device along with mission details. BUS utilizes various types of servers as described below.
  - Mobile edge computing server (MECS): mobile edge computing (MEC) is an emerging technology that brings the cloud server into the proximity of the user. [49]. MEC utilizes a radio access network to maintain communication with the user [50]. With the assistance of the MECS, data from IoT devices can be collected in real-time. However, BUS also utilizes the MECS to collect and store data when the mobile network is available. In BUS, the MECS stores data in the base stations BSs) after obtaining acceptance from the validators. Each MECS server acts as a validator in BUS.
  - Ground control station (GCS): the GCS is a control station for managing the UAV swarm. The GCS can be connected via mobile network or satellite. The GCS may act as a validator based on the connection it utilizes.
  - Private cloud (PC): Private servers stay in the PC. The PC is connected with other validators through backhaul networks. The PC also acts as a validator in the network.
- Satellite: BUS utilizes satellites for data acquisition from IoT devices in locations where the mobile network is not available. The UAV swarm may maintain communication with the PC via satellite.

B. BASIC IDEA

The purpose of BUS is to collect data from IoT devices securely and to store this data into a server maintaining its integrity. In BUS, every participant including IoT devices and UAVs has to register with the server before participating in

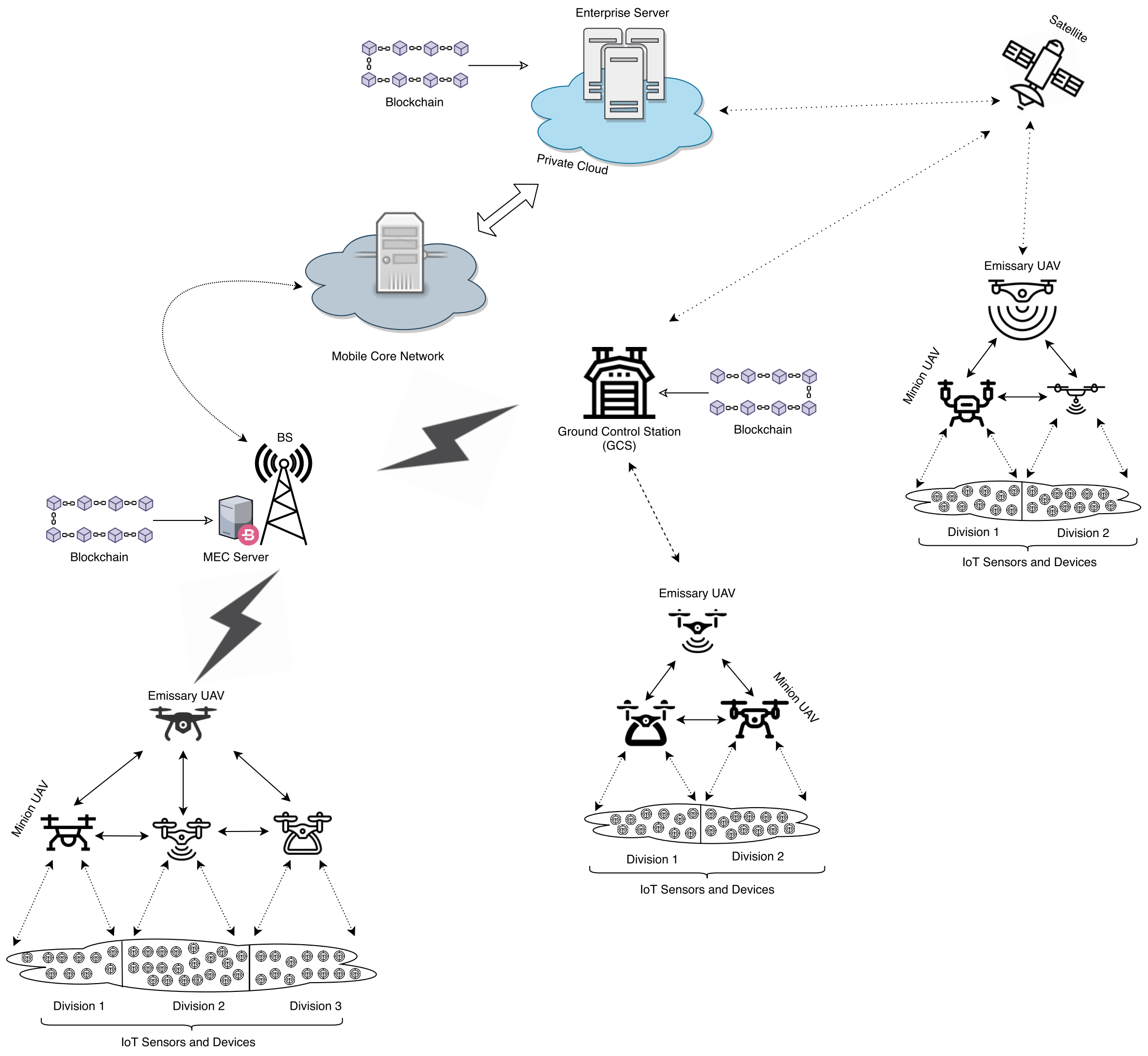

FIGURE 1. **Blockchain based secure data acquisition utilizing UAV swarm.**

BUS. However, BUS utilizes a swarm of UAVs to collect data from IoT devices in order to extend connectivity and also to assist IoT devices to preserve battery power by allowing low powered data transmission. In BUS, before deploying the UAV swarm, the server creates a mission and stores this mission information in blockchain utilizing a smart contract (SC). Then, all UAV swarms have to register their public keys in their missions. Spatial information (i.e., latitude and longitude) of IoT devices along with their public keys are mentioned in the mission. When a UAV swarm is deployed in the field, the UAV swarm first determines its role and then, each UAV in the UAV swarm is allocated a region of IoT devices to assist based on the assigned role. Each UAV in the UAV swarm creates a shared key and shares it with the IoT devices. IoT devices send data encrypted using the shared key. UAV decrypts that data using the shared key and validates the sender's identity using the π-hash bloom filter. Upon successful validation, the UAV forwards the data to the nearest server. The server also validates the sender, and after successful validation, the server prepares data to add to blockchain. After receiving consent from validators (i.e., servers), the data are added to the blockchain network. BUS utilizes different links among entities (i.e., IoT-UAV, UAV-UAV, UAV-server) and indeed, each link has different channel states. Considering the situation that various links are used together, BUS carefully design wireless technologies for each link, e.g., IoT-UAV, UAV-UAV and UAV-server use BT, Wi-Fi, and LTE, respectively, considering bandwidth (bottleneck) and interference. Moreover, BUS is composed of asynchronous processing inside of the entities. For example, when a data comes to UAV, UAV process it and after processing, UAV adds it in the queue before forwarding it to the next entity. These are asynchronous process and each entity follows a similar way of processing and storing data before transmitting. Thus, BUS keep the consistency in the data acquisition process although having different links.

C. SYMBOLS

A list of relevant symbols, that is going to be used in this paper, along with their description is presented in Table 2.

TABLE 2. **Symbols and their description.**

Symbol	Description
ϱ	Private key.
ρ	Public key.
$\wp$	Shared key.
$\Xi(.)$	Hash function.
$\daleth(.)$	Key generator.
κ	Key size.
$\beta\mathcal{F}(.)$	Bloom filter.
$\vartheta_e, \vartheta_m, \Psi$	Emissary UAV, minion UAV, server.
$\xi(.)$	Encryption function for asymmetric encryption.
$\zeta(.)$	Decryption function for asymmetric encryption.
$\Theta(.)$	Signature generator.
$\omega(.)$	Signature verification function.
$\xi^{'}(.)$	Encryption function for symmetric encryption.
$\zeta'(.)$	Decryption function for symmetric encryption.
$\partial\mathcal{I}(.)$	Vote function for adding and removing a validator.
$\partial\mathcal{V}(.)$	Validate blocks proposed by a validator.

III. BLOCKCHAIN-ENABLED UAV SWARM ASSISTED DATA ACQUISITION MECHANISM

A. REGISTRATION

To participate in BUS, every device (i.e., IoT and UAV) has to register in BUS. Prior to registering in BUS, each device generates a private key based on device MAC address $\mathfrak{d}_{\mathfrak{m}}$, current timestamp τ_c, and random salt hash $\mathcal{S}_{\hbar}$. Let $\hbar$ be the generated hash,

$$\hbar = \Xi(\mathfrak{d}_{\mathfrak{m}}, \tau_c, \mathcal{S}_{\hbar}) \mid \Xi : \{0, 1\}^* \mapsto \{0, 1\}^{\ell}.$$

A private key ϱ_κ is generated from $\hbar$ and a corresponding public key ρ_κ is generated from ϱ_κ. Let $\mathcal{G}$ be a set of (x, y) coordinates on the elliptic curve,

$$\varrho_\kappa = \daleth(\hbar) \mid \daleth : \{0, 1\}^* \mapsto \{0, 1\}^s, \rho_\kappa = \varrho_\kappa \otimes (\mathcal{G}_x, \mathcal{G}_y). \quad (1)$$

Before deploying an IoT device, $\mathfrak{d}_{\mathfrak{m}}$, the spatial information (i.e., the latitude and longitude) of the deployed area, and ρ_κ are stored in the server, which generates a trust token for that area that is used for communicating with IoTs. Let δ be the trust token,

$$\delta = \Xi(\mathcal{S}_{\hbar}, \tau_c, \mathfrak{d}_{\mathfrak{m}}, \Psi_{\langle lat,lon\rangle}) \mid \Xi : \{0, 1\}^* \mapsto \{0, 1\}^{\ell}.$$

Here, $\Psi_{\langle lat,lon\rangle}$ is the location of the server Ψ. However, the server stores this information in a smart contract $\mathfrak{C}_{\mathfrak{d}}$ of blockchain. In BUS, ρ_κ is used as an identifier of the device.

B. UAV SWARM DEPLOY

Before deploying UAV swarm $\overline{\vartheta}$, the server Ψ fetches the device list (i.e., IoT devices and UAVs) from $\mathfrak{C}_{\mathfrak{d}}$. Let $\overline{\sigma}$ be the device list,

$$\overline{\sigma} = \bigcup \mathfrak{C}_{\mathfrak{d}}(\sigma_i^{\mathcal{K}}), \quad \forall i, \sigma_i \in \mathfrak{C}_{\mathfrak{d}} \cap \mathcal{K} \leq \mathcal{L}_{\mathfrak{r}}.$$

Here, $\mathfrak{r}$ is the radius of the location selected $\mathcal{L}$ and $\mathcal{K}$ is the location of the device σ where $\mathcal{K}$ contains $\langle lat, lon\rangle$. BUS employs π-hash bloom filter for validating the authenticity of the devices. However, to validate user authenticity, π-hash bloom filter ($\pi = 1, 2, 3, ...$) requires a pre-generated dataset based on the $\overline{\sigma}$. Let $\overline{\Im}$ be the pre-generated table,

$$\overline{\Im} = \bigcup_{\forall i \in \overline{\sigma}} \bigcup_{\forall j \in \pi} \beta\mathcal{F}_j(\sigma_i^k)$$
$$|\beta\mathcal{F} : \{0, 1\}^* \to \iota \mid \iota \in \mathbb{Z}^* \cap \Im_\iota \in \{0, 1\} \cap k \in \{0, 1, 2\}.$$

Here, π is the total number of filters used in π-hash bloom filter and k is the device type ($0 =$ IoT device, $1 =$ UAV, $2 =$ Server). However, the bloom filter may lead to false positive issues which can be mitigated by increasing the number of filters and the size of the data table. Let η be the total number of σ [51],

$$\ddot{m} = \lceil \frac{-\eta \times \ln(\ddot{p})}{\ln(2)^2} \rceil \mid \ddot{p} \in (0, 1], \quad \pi = \lceil \frac{\ddot{m}}{\eta} \times \ln(2) \rceil.$$

Here, $\ddot{p}$ is the rate of false positive and π is the maximum number of hash functions. After generating $\overline{\Im}$, Ψ creates the mission data. Let $\overline{\mathfrak{y}}$ be the mission data,

$$\overline{\mathfrak{y}} = \langle \mathfrak{y}_{id}, \mathfrak{y}_{name}, \overline{\sigma}^k, \mathcal{L}_{\mathfrak{r}}, \tau_{\mathfrak{y}} \rangle.$$

Upon creating $\overline{\mathfrak{y}}$, Ψ stores these data in a smart contract $\mathfrak{C}_{\mathfrak{y}}$ of blockchain. After that, Ψ shares $\overline{\mathfrak{y}}$, $\overline{\Im}$, and δ with $\overline{\vartheta}$.

C. ROLE SELECTION

Before starting data acquisition, $\overline{\vartheta}$ selects the role of each ϑ utilizing UAV resources (termed as proof of UAV resources (PoUR)), as described in Algorithm 1. First, each ϑ calculates a score based on the resource (e.g., CPU, battery, ram, etc.). Let $\mathfrak{s}$ be the score,

$$\mathfrak{s} = \sum_{\forall i \in m} r_i \times w_i.$$

Here, m is the total number of properties that are considered for PoUR, r is the resource, and w is the weight used to provide the priority in the properties of the resource. Prior to sending the score to each ϑ, the sender ϑ encrypts $\mathfrak{s}$ using each ϑ's public key. Let $\mathfrak{s}_e$ be the encrypted score,

$$\mathfrak{s}_{e_i} = \xi_{\rho_{\kappa_i}^{\vartheta}}(\mathfrak{s}_i, \rho_\kappa^{\vartheta}, \theta_{\varrho_{\kappa_i}^{\vartheta}}(\mathfrak{s}_i)), \quad \forall i \in (\hat{n} - 1).$$

Here, $\hat{n}$ is the total number of ϑ. Then, each ϑ shares $\mathfrak{s}_e$ among themselves. Upon receiving $\mathfrak{s}_e$ from another ϑ, a ϑ decrypts each of the $\mathfrak{s}_e$ employing ϱ_κ. Let $\mathfrak{s}_d$ be the decrypted data,

$$\mathfrak{s}_{d_i} = \zeta_{\varrho_\kappa^{\vartheta}}(\mathfrak{s}_{e_i}), \quad \forall i \in (\hat{n} - 1).$$

ϑ then checks the validity of the sender using π-hash bloom filter. Let $\mathfrak{f}$ be the filter data,

$$\mathfrak{f}_i = \bigwedge_{\forall j \in \pi} \beta\mathcal{F}_j(\rho_{\kappa_i}^{\vartheta}), \quad \forall i \in (\hat{n} - 1) \cap \mathfrak{f} \in \{0, 1\}.$$

Algorithm 1 Role selection in UAV swarm.

```
1: $\mathfrak{s} \rightarrow 0$.
2: while $i \in m$ do
3:     $\mathfrak{s} \leftarrow \mathfrak{s} + (r_i \times w_i)$.
4: end while
5: $\overline{\mathfrak{s}_e} \rightarrow \emptyset$.
6: while $i \in (n-1)$ do
7:     $\overline{\mathfrak{s}_e} \leftarrow \overline{\mathfrak{s}_e} \cup \xi_{\rho^{\vartheta}_{\kappa_i}}(\mathfrak{s}_i, \rho^{\vartheta}_{\kappa}, \theta_{\varrho^{\vartheta}_{\kappa_i}}(\mathfrak{s}_i))$.
8: end while
9: while $i \in (n-1)$ do
10:    $\vartheta \xrightarrow[transmits]{\mathfrak{s}_{e_i}} \vartheta_i$.
11: end while
12: $wait()$.
13: $\overline{\mathfrak{s}_d} \rightarrow \emptyset$.
14: while $i \in (n-1)$ do
15:    $\overline{\mathfrak{s}_d} \leftarrow \overline{\mathfrak{s}_d} \cup \zeta_{\varrho^{\vartheta}_{\kappa}}(\mathfrak{s}_{e_i})$.
16: end while
17: $\overline{s} \rightarrow \emptyset$
18: while $i \in (n-1)$ do
19:    $\mathfrak{f}_i \leftarrow \bigwedge_{\forall j \in \pi} \beta F_j(\rho^{\vartheta}_{\kappa_i})$.
20:    if $\mathfrak{f}_i == 1$ then
21:        $\mathfrak{s}_{v_i} \leftarrow \omega_{\rho^{\sigma}_{\kappa_i}}(\mathfrak{s}_i, \theta_{\varrho^{\vartheta}_{\kappa_i}}(\mathfrak{s}_i))$.
22:        if $\mathfrak{s}_{v_i} == TRUE$ then
23:            $\overline{s} \leftarrow \overline{s} \cup \overline{s}_i$.
24:        end if
25:    end if
26: end while
27: $\mathfrak{s}_{max} \leftarrow \max_{\forall i \in \overline{\vartheta}}(\mathfrak{s}_i)$.
28: $\vartheta \leftarrow (\mathfrak{s} == \mathfrak{s}_{max})$ ? $emissary$ : $minion$.
```

Upon successful validation, ϑ checks the authenticity of the sender by verifying the signature utilizing the sender's ρ_κ. Let $\mathfrak{s}_v$ be the verified result,

$$\mathfrak{s}_{v_i} = \omega_{\rho^{\sigma}_{\kappa_i}}(\mathfrak{s}_i, \theta_{\varrho^{\vartheta}_{\kappa_i}}(\mathfrak{s}_i)), \quad \forall i \in (\hat{n}-1) \cap \mathfrak{s}_{v_i} \in \{true, false\}.$$

After obtaining all the scores, each ϑ identifies the highest among them. Let $\mathfrak{s}_{max}$ be this maximum score,

$$\mathfrak{s}_{max} = \max_{\forall i \in \overline{\vartheta}}(\mathfrak{s}_i).$$

When a ϑ determines that his $\mathfrak{s}$ is the maximum, that ϑ elects to be an emissary UAV ϑ_e and the others become minion UAVs ϑ_m. Let $\mathcal{T}_{role}$ be the total time needed to select the role and dividing the area,

$$\mathcal{T}_{role_i} = \mathcal{T}_{\mathfrak{s}} + \mathcal{T}_{\mathfrak{s}_{max}} + (\hat{n}-1) \times (\mathcal{T}_{tt} + \mathcal{T}_{tr} + \mathcal{T}_{\mathfrak{s}_e} + \mathcal{T}_{\mathfrak{s}_d} + \mathcal{T}_{\mathfrak{f}_i} + \mathcal{T}_{\mathfrak{s}_v}), \quad \forall i \in \pi. \quad (2)$$

Here, $\mathcal{T}_{\mathfrak{s}}$ is the time neede to calculate the score, $\mathcal{T}_{\mathfrak{s}_{max}}$ is the time needed to find the maximum score, n is the total number of ϑ, $\mathcal{T}_{tt}$ is the time needed to transmit the score, $\mathcal{T}_{tr}$ is the waiting time before receiving the score, $\mathcal{T}_{\mathfrak{s}_e}$ is the time required to encrypt the score, $\mathcal{T}_{\mathfrak{s}_d}$ is the time needed to decrypt the scores from others, $\mathcal{T}_{\mathfrak{f}}$ is the time needed to validate using π-hash bloom filter, and $\mathcal{T}_{\mathfrak{s}_v}$ is the total time required to verify the sender. Then, ϑ_e divides the area $\mathfrak{a}$ based on the score of ϑ_m. Before dividing $\mathfrak{a}$, ϑ_e sorts the IoT devices based on spatial information. Let $\overline{\sigma}^k_{sort}$ be the sorted IoT device list,

$$\overline{\sigma}^k_{sort} = sort(\sigma^k_i)^{ord=asc}_{prop=\{lat,long\}}, \quad \forall i \in \hat{n} \cap k = 0.$$

Here, n is the total number of IoT devices. Then, ϑ_e generates the percentage of the capability for each ϑ_m and divides the IoT devices accordingly. Let $\overline{\sigma}^k_s$ be the number of IoT devices selected,

$$\overline{\sigma}^k_{s_j} = \frac{\mathfrak{s}_j}{\sum_{\forall i \in \ddot{n}} \mathfrak{s}_i} \times n, \quad \forall j \in \ddot{n}.$$

Here, $\ddot{n}$ is the total number of ϑ_m. ϑ_e generates a new trust token for IoT devices to replace the previous one. Let δ_ϑ be the new trust token,

$$\delta_\vartheta = \Xi(\mathcal{S}_\hbar, \tau_c, \eth_\mathfrak{m}, \vartheta_{e_{\langle lat,lon \rangle}}) \mid \Xi : \{0,1\}^* \mapsto \{0,1\}^\ell.$$

ϑ_e then generates a shared key used among $\overline{\vartheta}$ to maintain communication. Let $\wp^{\{\vartheta \leftrightarrow \vartheta\}}_\kappa$ be the shared key,

$$\wp^{\{\vartheta \leftrightarrow \vartheta\}}_\kappa = \daleth(\Xi(\eth_\mathfrak{m}, \tau_c, \mathcal{S}_\hbar)) \mid \daleth : \{0,1\}^* \mapsto \{0,1\}^s \cap \Xi : \{0,1\}^* \mapsto \{0,1\}^\ell.$$

ϑ_e then selects the area of IoT based on $\overline{\sigma}^k_s$ and broadcasts to ϑ_m. Prior to broadcasting the information, ϑ_e encrypts the information employing each ϑ_m's ρ_κ. Let d be the encrypted information,

$$d_i = \xi_{\rho^{\vartheta_m}_{\kappa_i}}(\delta_\vartheta, \wp^{\{\vartheta \leftrightarrow \vartheta\}}_\kappa, \rho^{\vartheta_e}_\kappa, \overline{\sigma}^k_{s_i}, \theta_{\varrho^{\vartheta_e}_\kappa}(\delta_\vartheta)), \quad \forall i \in \ddot{n}.$$

ϑ_e then broadcasts d to all ϑ_m. Upon receiving the IoT device information, each ϑ_m takes their position accordingly. However, ϑ_e encrypts δ_ϑ, and the role selection information $\mathcal{I}_r$ along with the flight status $\mathcal{I}_f$ employing Ψ's ρ^Ψ_κ. Let e be the encrypted information,

$$e = \xi_{\rho^\Psi_\kappa}(\delta_\vartheta, \mathcal{I}_r, \mathcal{I}_f, \rho^{\vartheta_e}_\kappa, \theta_{\varrho^{\vartheta_e}_\kappa}(\delta_\vartheta)).$$

ϑ_e then transmits e to Ψ. Upon receiving e, Ψ decrypts e employing ϱ^Ψ_κ. Let d_Ψ be the decrypted information,

$$d_\Psi = \zeta_{\varrho^\Psi_\kappa}(e).$$

Following the decryption, Ψ first validates the sender using π-hash bloom filter. Let $\mathfrak{f}_\Psi$ be the filtered data,

$$\mathfrak{f}_\Psi = \bigwedge_{\forall j \in \pi} \beta F_j(\rho^{\vartheta_e}_\kappa), \quad \mathfrak{f}_\Psi \in \{0,1\}.$$

Upon successful validation, Ψ verifies the sender using $\rho^{\vartheta_e}_\kappa$. Let v_Ψ be the verified data,

$$v_\Psi = \omega_{\rho^{\vartheta_e}_\kappa}(\delta_\vartheta, \theta_{\varrho^{\vartheta_e}_\kappa}(\delta_\vartheta)), \quad v_\Psi \in \{true, false\}.$$

Upon successful verification, Ψ updates the trust token in $\mathfrak{C}_\eth$ and updates $\mathcal{I}_r$ and $\mathcal{I}_f$ in $\mathfrak{C}_\mathfrak{y}$.

D. DATA SYNCHRONIZATION

When ϑ_m gets its allocated IoT devices, ϑ_m generates a shared key that is used to communicate between ϑ_m and IoT devices α. Let $\wp_\kappa^{\{\vartheta_m \leftrightarrow \alpha\}}$ be the shared key,

$$\begin{aligned}\wp_\kappa^{\{\vartheta_m \leftrightarrow \alpha\}} &= \daleth(\Xi(\eth_{\vartheta_m}, \tau_c, \mathcal{S}_\hbar, n)) \\ &\quad |\daleth : \{0,1\}^* \mapsto \{0,1\}^s \cap \Xi : \{0,1\}^* \mapsto \{0,1\}^\ell.\end{aligned} \tag{3}$$

Here, n is the total number of α. ϑ_m then sends the request to $\overline{\alpha}$ (termed as a "Seheri Call") to register themselves with ϑ_m. Before transmitting, ϑ_m encrypts information employing each of the $\overline{\alpha}$. Let e^{ϑ_m} be the encrypted data,

$$e_i^{\vartheta_m} = \xi_{\rho_{\kappa_i}^\alpha}(\rho_\kappa^{\vartheta_m}), \quad \forall i \in n.$$

Upon receiving the request, α decrypts e^{ϑ_m} employing ϱ_κ^α. Let d_α be the decrypted data,

$$d_\alpha = \zeta_{\varrho_\kappa^\alpha}(e^{\vartheta_m}).$$

Following this, α encrypts its public key $\rho_{\kappa_i}^\alpha$ using a trust token δ. Then, α encrypts again using ϑ_m's $\rho_\kappa^{\vartheta_m}$ and transmits to ϑ_m. Let e^α be the encrypted data,

$$e^\alpha = \xi_{\rho_\kappa^{\vartheta_m}}(\rho_\kappa^\alpha, \xi'_\delta(\rho_\kappa^\alpha)).$$

When ϑ_m receives e_α, ϑ_m decrypts e_α employing $\varrho_\kappa^{\vartheta_m}$. Let d^{ϑ_m} be the decrypted data,

$$d_i^{\vartheta_m} = \zeta_{\varrho_\kappa^{\vartheta_m}}(e_i^\alpha), \quad \forall i \in n.$$

Following the decryption, ϑ_m first validates the sender using π-hash bloom filter. Let $\mathfrak{f}^{\vartheta_m}$ be the filtered data,

$$\mathfrak{f}_i^{\vartheta_m} = \bigwedge_{\forall j \in \pi} \beta\mathcal{F}_j(\rho_{\kappa_i}^\alpha), \quad \forall i \in n \cap \mathfrak{f}^{\vartheta_m} \in \{0,1\}.$$

ϑ_m decrypts again using δ. Let $d^{\vartheta_m^\delta}$ be the decrypted data,

$$d_i^{\vartheta_m^\delta} = \zeta'_\delta(\xi'_{\delta_i}(\rho_\kappa^\alpha)), \quad \forall i \in n.$$

ϑ_m verifies that $d^{\vartheta_m^\delta}$ is identical to ρ_κ^α. Upon successful validation, it encrypts a new trust token δ_ϑ using the previous δ and creates a signature based on δ_ϑ employing $\varrho_\kappa^{\vartheta_m}$. Following this, $d^{\vartheta_m^\delta}$ encrypts all information employing ρ_κ^α. Let e_i^f be the encrypted data,

$$\begin{aligned}e_i^f &= \xi_{\rho_{\kappa_i}^\alpha}(\wp_\kappa^{\{\vartheta_m \leftrightarrow \alpha\}}, \rho_\kappa^{\vartheta_m}, \theta_{\varrho_\kappa^{\vartheta_m}}(\delta_\vartheta), \xi'_\delta(\delta_\vartheta)), \\ \forall i &\in n \cap d_i^{\vartheta_m^\delta} = \rho_{\kappa_i}^\alpha.\end{aligned}$$

Let $\mathcal{T}_{sync}$ be the total time to sync new information from ϑ_m to α.

$$\begin{aligned}\mathcal{T}_{sync_i} &= n \times (\mathcal{T}_{e^{\vartheta_m}} + \mathcal{T}_{tr} + \mathcal{T}_{d^{\vartheta_m}} \\ &\quad + \mathcal{T}_{\mathfrak{f}_i^{\vartheta_m}} + \mathcal{T}_{d^{\vartheta_m^\delta}} + \mathcal{T}_{e^f}), \forall i \in \pi.\end{aligned} \tag{4}$$

Upon receiving e^f, α decrypts e^f employing ϱ_κ^α. Let d_α be the decrypted data,

$$d_\alpha = \zeta_{\varrho_\kappa^\alpha}(e^f).$$

Algorithm 2 Create time slot γ_τ in ϑ_m.

```
1:  $d_{np} \leftarrow \zeta'_{\wp_\kappa^{\{\vartheta_m \leftrightarrow \alpha\}}}(e_{np})$.
2:  $\tau_{diff} \leftarrow \tau_c - \hat{n}$.
3:  if $\tau_{diff} \leq \tau_{th}$ then
4:      $\mathfrak{f}^{\vartheta_m} \leftarrow \bigwedge_{\forall j \in \pi} \beta\mathcal{F}_j(\rho_\kappa^\sigma)$.
5:      if $\mathfrak{f}^{\vartheta_m} == 1$ then
6:          $\mathfrak{v}^{\vartheta_m} \leftarrow \omega_{\rho_\kappa^\sigma}(\hat{n}, \theta_{\varrho_\kappa^\sigma}(\hat{n}))$.
7:          if $\mathfrak{v}^{\vartheta_m} == TRUE$ then
8:              $\gamma_{id} \leftarrow \Xi(\tau_c, \mathcal{S}_\hbar)$.
9:              $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{\rho_\kappa^\sigma, \hat{n}, \gamma_{id}, \gamma_\tau\}$.
10:         else
11:             $\rho_\kappa^\sigma \rightarrow V_L$.
12:         end if
13:     else
14:         $\rho_\kappa^\sigma \rightarrow V_L$.
15:     end if
16: else
17:     $\rho_\kappa^\sigma \rightarrow V_L$.
18: end if
```

Then, α decrypts δ_ϑ utilizing δ. Let d_δ^α be the decrypted data.

$$d_\delta^\alpha = \zeta'_\delta(\xi'_\delta(\delta_\vartheta)).$$

Following this, α verifies the sender using $\rho_\kappa^{\vartheta_m}$. Let v_α be the verified data,

$$v_\alpha = \omega_{\rho_\kappa^{\vartheta_m}}(d_\delta^\alpha, \theta_{\varrho_\kappa^{\vartheta_m}}(\delta_\vartheta)), v_\Psi \in \{true, false\}.$$

Upon successful verification, α updates δ_ϑ and $\wp_\kappa^{\{\vartheta_m \leftrightarrow \alpha\}}$.

E. DATA ACQUISITION

An IoT device σ wants to transmit data $\hat{\mathfrak{a}}$ to the server. First, σ sends a request packet (termed as "nonce packet") for the transfer of $\hat{\mathfrak{a}}$ to the associated ϑ_m. Before sending the request, σ encrypts the nonce packet utilizing $\wp_\kappa^{\{\vartheta_m \leftrightarrow \alpha\}}$. Let e_{np} be the encrypted nonce packet,

$$e_{np} = \xi'_{\wp_\kappa^{\{\vartheta_m \leftrightarrow \alpha\}}}(\hat{n}, \theta_{\varrho_\kappa^\sigma}(\hat{n}), \rho_\kappa^\sigma, \sigma_{sp}, \hat{\mathfrak{a}}_s, \overline{\tau}_{dve}). \tag{5}$$

Here, $\overline{\tau}_{dve}$ is the time required to decrypt, verify and encrypt data. σ_{sp} is the spatial information on σ. $\hat{\mathfrak{a}}_s$ is the size of the data, and $\hat{n}$ is the nonce derived from the current timestamp. Upon receiving e_{np}, ϑ_m decrypts e_{np} utilizing $\wp_\kappa^{\{\vartheta_m \leftrightarrow \alpha\}}$, as described in Algorithm 2. Let d_{np} be the decrypted nonce packet,

$$d_{np} = \zeta'_{\wp_\kappa^{\{\vartheta_m \leftrightarrow \alpha\}}}(e_{np}).$$

Subsequently, ϑ_m picks the current timestamp τ_c. Let τ_{diff} be the difference between $\hat{n}$ and τ_c.

$$\tau_{diff} = \tau_c - \hat{n}. \tag{6}$$

ϑ_m verifies whether τ_{diff} is within τ_{th} or not. Here, τ_{th} is the threshold time range for accepting a packet. If τ_{diff} is not within τ_{th}, ϑ_m discards the request by considering that the packet comes from an attacker who stored that packet to attack later. However, ϑ_m checks the authenticity of the sender utilizing π-hash bloom filter. Let $\mathfrak{f}^{\vartheta_m}$ be the filtered data,

$$\mathfrak{f}^{\vartheta_m} = \bigwedge_{\forall j \in \pi} \beta F_j(\rho_\kappa^\sigma), \quad \mathfrak{f}^{\vartheta_m} \in \{0, 1\}.$$

Upon successful authentication, ϑ_m verifies the data of the sender by verifying the signature. Let $\mathfrak{v}^{\vartheta_m}$ be the verification result,

$$\mathfrak{v}^{\vartheta_m} = \omega_{\rho_\kappa^\sigma}(\hat{n}, \theta_{\varrho_\kappa^\sigma}(\hat{n})), \quad \mathfrak{v}^{\vartheta_m} \in \{true, false\}.$$

BUS utilizes time-division multiple access (TDMA) protocol in the communication. Following this, ϑ_m creates a slot number based on the current timestamp τ_c, and a random salt hash $\mathcal{S}_\hbar$. Let γ_{id} be the slot number,

$$\gamma_{id} = \Xi(\tau_c, \mathcal{S}_\hbar) \mid \Xi : \{0, 1\}^* \mapsto \{0, 1\}^\ell. \tag{7}$$

ϑ_m then creates the time slot γ_τ by estimating the channel condition, $\hat{\mathfrak{a}}_s$, and $\bar{\tau}_{dve}$. σ has to send data within γ_τ. However, ϑ_m picks γ_τ^{tr} and γ_τ^{tcl} to estimate the response time from ϑ_m to σ and to enforce a gap before starting the next slot, respectively. Prior to returning γ_τ, ϑ_m encrypts the response using $\wp_\kappa^{\{\vartheta_m \leftrightarrow \alpha\}}$. Let e_{ts} be the encrypted time slot,

$$e_{ts} = \xi'_{\wp_\kappa^{\{\vartheta_m \leftrightarrow \alpha\}}}(\hat{n} + 1, \theta_{\varrho_\kappa^{\vartheta_m}}(\hat{n} + 1), \rho_\kappa^{\vartheta_m}, \gamma_\tau).$$

Let $\mathcal{T}^{\vartheta_m}$ be the total time for processing the data in ϑ_m,

$$\mathcal{T}_i^{\vartheta_m} = \mathcal{T}_{d_{np}} + \mathcal{T}_{\tau_{diff}} + \mathcal{T}_{\mathfrak{f}_i^{\vartheta_m}} + \mathcal{T}_{\mathfrak{v}^{\vartheta_m}} + \mathcal{T}_{\gamma_{id}} + \mathcal{T}_{e_{ts}}, \quad \forall i \in \pi. \tag{8}$$

Here, $\mathcal{T}_{d_{np}}$ is the total time required for d_{np}, $\mathcal{T}_{\tau_{diff}}$ is the total time required for validating nonce, $\mathcal{T}_{\mathfrak{f}^{\vartheta_m}}$ is the total time needed to validate the sender, $\mathcal{T}_{\mathfrak{v}^{\vartheta_m}}$ is the total time required for verifying the sender, $\mathcal{T}_{\gamma_{id}}$ is the total time needed to create and store the time slot, and $\mathcal{T}_{e_{ts}}$ is the total time for e_{ts}. However, when σ receives e_{ts}, σ decrypts e_{ts} employing $\wp_\kappa^{\{\vartheta_m \leftrightarrow \alpha\}}$. Let d_{ts} be the decrypted time slot,

$$d_{ts} = \zeta'_{\wp_\kappa^{\{\vartheta_m \leftrightarrow \alpha\}}}(e_{ts}).$$

Following this, σ checks whether the $\hat{n} + 1$ returned is actually the addition of the transmitted $\hat{n}$ with 1 or not. If the $\hat{n}+1$ received is invalid, then σ discards the data. Afterwards, σ verifies the data of the sender by verifying the signature. Let $\mathfrak{v}_{ts}$ be the verification result,

$$\mathfrak{v}_{ts} = \omega_{\rho_\kappa^{\vartheta_m}}(\hat{n} + 1, \theta_{\varrho_\kappa^{\vartheta_m}}(\hat{n} + 1)),$$
$$\mathfrak{v}_{ts} \in \{true, false\}.$$

Upon successful verification, σ prepares and encrypts the data using $\wp_\kappa^{\{\vartheta_m \leftrightarrow \alpha\}}$. Let e_d be the encrypted data,

$$e_d = \xi'_{\wp_\kappa^{\{\vartheta_m \leftrightarrow \alpha\}}}(\theta_{\varrho_\kappa^\sigma}(\varpi), \rho_\kappa^\sigma, \varpi).$$

Here, ϖ is the transmitted data. Let $\mathcal{T}_\sigma$ be the total time needed for processing the data in σ,

$$\mathcal{T}_\sigma = \mathcal{T}_{d_{ts}} + \mathcal{T}_{(\hat{n}+1)} + \mathcal{T}_{\mathfrak{v}'_{\vartheta_e}} + \mathcal{T}_{e_d}. \tag{9}$$

Here, $\mathcal{T}_{d_{ts}}$ is the total time required for d_{ts}, $\mathcal{T}_{(\hat{n}+1)}$ is the total time needed to validate nonce, and $\mathfrak{v}'_{\vartheta_e}$ is the total time taken to verify sender, and $\mathcal{T}_{e_d}$ is the total time required for $\mathcal{T}_{e_d}$. However, when ϑ_m receives e_d, ϑ_m decrypts e_d employing $\wp_\kappa^{\{\vartheta_m \leftrightarrow \alpha\}}$. Let d_d be the decrypted data,

$$d_d = \zeta'_{\wp_\kappa^{\{\vartheta_m \leftrightarrow \alpha\}}}(e_d).$$

ϑ_m then picks the current timestamp τ_c and checks that σ has been allocated to any γ_τ and whether this slot belongs to that σ or not. If σ has not been assigned to any slot, ϑ_m discards the packet immediately and puts σ into the vulnerable list V_L. However, upon successful time slot verification, ϑ_m checks the authenticity of the sender utilizing π-hash bloom filter. Let $\mathfrak{f}^{\vartheta_m}$ be the filtered data,

$$\mathfrak{f}^{\vartheta_m} = \bigwedge_{\forall j \in \pi} \beta F_j(\rho_\kappa^\sigma), \quad \mathfrak{f}^{\vartheta_m} \in \{0, 1\}.$$

Upon successful authentication, ϑ_m verifies the data of the sender by verifying the signature. Let $\mathfrak{v}^{\vartheta_m}$ be the verification result,

$$\mathfrak{v}^{\vartheta_m} = \omega_{\rho_\kappa^\sigma}(\varpi, \theta_{\varrho_\kappa^\sigma}(\varpi)), \quad \mathfrak{v}^{\vartheta_m} \in \{true, false\}.$$

ϑ_m then adds d_d in the queue $\mathcal{Q}$ for forwarding to ϑ_e. When ϑ_m wants to forward data to ϑ_e, ϑ_m sends the request to ϑ_e to provide a time slot γ'_τ. Prior to sending the request, ϑ_m encrypts data utilizing $\wp_\kappa^{\{\vartheta \leftrightarrow \vartheta\}}$. Let e_{ϑ_m} be the encrypted data,

$$e_{\vartheta_m} = \xi'_{\wp_\kappa^{\{\vartheta \leftrightarrow \vartheta\}}}(\hat{n}, \theta_{\varrho_\kappa^{\vartheta_m}}(\hat{n}), \rho_\kappa^{\vartheta_m}, \hat{\mathfrak{a}}'_s, \bar{\tau}'_{dve}).$$

Here, $\bar{\tau}'_{dve}$ is the time needed to decrypt, verify, and encrypt data in ϑ_m, $\hat{\mathfrak{a}}'_s$ is the size of the data, and $\hat{n}$ is the nonce derived from the current timestamp. When ϑ_e receives a request from ϑ_m, ϑ_e decrypts e_{ϑ_m} utilizing $\wp_\kappa^{\{\vartheta \leftrightarrow \vartheta\}}$. Let d_{ϑ_m} be the decrypted data,

$$d_d = \zeta'_{\wp_\kappa^{\{\vartheta \leftrightarrow \vartheta\}}}(e_{\vartheta_m}).$$

ϑ_e then picks the current timestamp τ_c. Let τ_{diff} be the difference between $\hat{n}$ and τ_c.

$$\tau_{diff} = \tau_c - \hat{n}.$$

ϑ_e checks whether τ_{diff} is within τ_{th} or not. Here, τ_{th} is the threshold time range for accepting the packet. If τ_{diff} is not within τ_{th}, ϑ_m discards the request by considering that this packet comes from an attacker who stored it to attack later. Subsequently, ϑ_e first validates ϑ_m using π-hash bloom filter. Let $\mathfrak{f}_{\vartheta_e}$ be the filtered data,

$$\mathfrak{f}_{\vartheta_e} = \bigwedge_{\forall j \in \pi} \beta F_j(\rho_\kappa^{\vartheta_m}), \quad \mathfrak{f}_{\vartheta_e} \in \{0, 1\}.$$

ϑ_e then checks the data of the sender by verifying the signature. Let $\mathfrak{v}_{\vartheta_e}$ be the verification result,

$$\mathfrak{v}_{\vartheta_e} = \omega_{\rho_\kappa^{\vartheta_m}}(\hat{n}, \theta_{\varrho_\kappa^{\vartheta_m}}(\hat{n})), \quad \mathfrak{v}_{\vartheta_e} \in \{true, false\}.$$

Upon successful verification, ϑ_e creates a slot number utilizing Equation 7 and then, creates a time slot γ'_τ by estimating the channel condition, $\hat{\mathfrak{a}}'_s$, and τ'_{dve}. ϑ_m has to send the data within γ'_τ. However, ϑ_e picks $\gamma_\tau^{tr'}$ and $\gamma_\tau^{tcl'}$ to estimate the response time from ϑ_e to ϑ_m and to enforce a gap before starting the next slot, respectively. Prior to returning γ'_τ, ϑ_e encrypts the response using $\wp_\kappa^{\{\vartheta\leftrightarrow\vartheta\}}$. Let e'_{ts} be the encrypted time slot,

$$e'_{ts} = \xi'_{\wp_\kappa^{\{\vartheta\leftrightarrow\vartheta\}}}(\hat{n}+1, \theta_{\varrho_\kappa^{\vartheta_e}}(\hat{n}+1), \rho_\kappa^{\vartheta_e}, \gamma'_\tau).$$

When ϑ_m receives e'_{ts}, ϑ_m decrypts e'_{ts} employing $\wp_\kappa^{\{\vartheta\leftrightarrow\vartheta\}}$. Let d'_{ts} be the decrypted time slot,

$$d'_{ts} = \zeta'_{\wp_\kappa^{\{\vartheta\leftrightarrow\vartheta\}}}(e'_{ts}).$$

Following this, ϑ_m checks whether the returned $\hat{n}+1$ is actually the addition of the transmitted $\hat{n}$ and 1 or not. If the received $\hat{n}+1$ is invalid, ϑ_m discards the data. ϑ_m then verifies the data of the sender by verifying the signature. Let $\mathfrak{v}'_{ts}$ be the verification result,

$$\mathfrak{v}'_{ts} = \omega_{\rho_\kappa^{\vartheta_e}}(\hat{n}+1, \theta_{\varrho_\kappa^{\vartheta_e}}(\hat{n}+1)), \quad \mathfrak{v}'_{ts} \in \{true, false\}.$$

Upon successful verification, ϑ_m prepares and encrypts the data using $\wp_\kappa^{\{\vartheta\leftrightarrow\vartheta\}}$. Let e'_d be the encrypted data,

$$e'_d = \xi'_{\wp_\kappa^{\{\vartheta\leftrightarrow\vartheta\}}}(\hat{n}+2, \theta_{\varrho_\kappa^\sigma}(\varpi), \rho_\kappa^\sigma, \theta_{\varrho_\kappa^{\vartheta_m}}(\hat{n}+2), \rho_\kappa^{\vartheta_m}, \varpi).$$

Here, ϖ is the transmitted data.When ϑ_m receives e_d, ϑ_e decrypts e_d employing $\wp_\kappa^{\{\vartheta\leftrightarrow\vartheta\}}$. Let d'_d be the decrypted data,

$$d'_d = \zeta'_{\wp_\kappa^{\{\vartheta\leftrightarrow\vartheta\}}}(e'_d).$$

ϑ_e then picks the current timestamp τ_c and checks whether ϑ_m has been allocated to any γ'_τ and whether that slot belongs to that ϑ_m or not. If ϑ_m has not been assigned to any slot, ϑ_e discards the packet immediately and puts ϑ_m into the vulnerable list V_L. However, upon successful time slot verification, ϑ_e checks the authenticity of the sender utilizing π-hash bloom filter. Let $\mathfrak{f}'_{\vartheta_m}$ be the filtered data,

$$\mathfrak{f}'_{\vartheta_m} = \bigwedge_{\forall j\in\pi} \beta F_j(\rho_\kappa^\sigma) \wedge \beta F_j(\rho_\kappa^{\vartheta_m}), \quad \mathfrak{f}'_{\vartheta_m} \in \{0, 1\}.$$

Upon successful authentication, ϑ_e verifies the data of the sender by checking the signature. Let $\mathfrak{v}'_{\vartheta_m}$ be the verification result,

$$\mathfrak{v}'_{\vartheta_m} = \omega_{\rho_\kappa^\sigma}(\varpi, \theta_{\varrho_\kappa^\sigma}(\varpi)) \wedge \omega_{\rho_\kappa^{\vartheta_m}}(\hat{n}+2, \theta_{\varrho_\kappa^{\vartheta_m}}(\hat{n}+2)), \qquad \mathfrak{v}'_{\vartheta_m} \in \{true, false\}.$$

After that, ϑ_e adds d'_d in the queue $\mathcal{Q}'$ for forwarding to Ψ. Prior to sending the data, ϑ_e encrypts it using ρ_κ^Ψ. Let e'_{ϑ_e} be the encrypted data,

$$e'_{\vartheta_e} = \xi_{\rho_\kappa^\Psi}(\hat{n}, \theta_{\varrho_\kappa^\sigma}(\varpi), \rho_\kappa^\sigma, \theta_{\varrho_\kappa^{\vartheta_e}}(\varpi), \rho_\kappa^{\vartheta_m}, \varpi).$$

Let $\mathcal{T}^{\vartheta_e}$ be the total time for processing data in ϑ_e,

$$\mathcal{T}_i^{\vartheta_e} = \mathcal{T}_{d'_d} + \mathcal{T}_{\gamma'_\tau} + \mathcal{T}_{\mathfrak{f}_i^{\vartheta_m}} + \mathcal{T}_{\mathfrak{v}'_{\vartheta_m}} + \mathcal{T}_{e'_{\vartheta_e}}, \quad \forall i \in \pi. \tag{10}$$

Here, $\mathcal{T}_{d'_d}$ is the total time for d'_d, $\mathcal{T}_{\gamma'_\tau}$ is the total time needed for validating a time slot, $\mathcal{T}_{\mathfrak{f}^{\vartheta_m}}$ is the total time required to validate the sender, $\mathcal{T}_{\mathfrak{v}'_{\vartheta_m}}$ is the total time needed to verify the sender, and e'_{ϑ_e} is the total time for e_{ts}. However, when Ψ receives e'_{ϑ_e}, Ψ decrypts the data using ϱ_κ^Ψ. Let d_{ϑ_e} be the decrypted data,

$$d_{\vartheta_e} = \zeta_{\varrho_\kappa^\Psi}(e'_{\vartheta_e}).$$

Following this, Ψ picks the current timestamp τ_c. Let τ_{diff} be the difference between $\hat{n}$ and τ_c.

$$\tau_{diff} = \tau_c - \hat{n}. \tag{11}$$

Ψ then checks whether τ_{diff} is within τ_{th} or not. Here, τ_{th} is the threshold time range for accepting a packet. If τ_{diff} is not within τ_{th}, Ψ discards the request by considering that the packet comes from an attacker who stored that packet for a subsequent attack. Afterwards, Ψ checks the authenticity of the sender utilizing π-hash bloom filter. Let $\mathfrak{f}'_{\vartheta_e}$ be the filtered data,

$$\mathfrak{f}'_{\vartheta_e} = \bigwedge_{\forall j\in\pi} \beta F_j(\rho_\kappa^\sigma) \wedge \beta F_j(\rho_\kappa^{\vartheta_e}), \quad \mathfrak{f}'_{\vartheta_e} \in \{0, 1\}.$$

Upon successful authentication, Ψ verifies the data of the sender by checking the signature. Let $\mathfrak{v}'_{\vartheta_e}$ be the verification result,

$$\mathfrak{v}'_{\vartheta_e} = \omega_{\rho_\kappa^\sigma}(\varpi, \theta_{\varrho_\kappa^\sigma}(\varpi)) \wedge \omega_{\rho_\kappa^{\vartheta_m}}(\varpi, \theta_{\varrho_\kappa^{\vartheta_m}}(\varpi)), \qquad \mathfrak{v}'_{\vartheta_e} \in \{true, false\}.$$

Let $\mathcal{T}^\Psi$ be the total time required to process the data in Ψ,

$$\mathcal{T}_i^\Psi = \mathcal{T}_{d_{\vartheta_e}} + \mathcal{T}_{\tau_{diff}} + \mathcal{T}_{\mathfrak{f}_i^{\vartheta_e}} + \mathcal{T}_{\mathfrak{v}'_{\vartheta_e}}, \quad \forall i \in \pi. \tag{12}$$

Here, $\mathcal{T}_{d_{\vartheta_e}}$ is the total time for d_{ϑ_e}, $\mathcal{T}_{\tau_{diff}}$ is the total time needed to validate nonce, $\mathcal{T}_{\mathfrak{f}^{\vartheta_e}}$ is the total time needed to validate the sender, and $\mathfrak{v}'_{\vartheta_e}$ is the total time required for verifying the sender. However, following this, Ψ adds d_{ϑ_e} in the pool $\mathcal{P}'$ to add the data in blockchain $\mathbb{B}$.

F. DATA MANAGEMENT

BUS considers a consortium blockchain in the proposed scheme. When data ϖ arrives at Ψ, Ψ adds ϖ to the transaction pool $\mathcal{P}'$. In BUS, every validator ν holds the list of total validator $\overline{\nu}$. In BUS, in order to add or remove a ν, one ν raises an issue in the network. When the majority gives a vote in support of the issue, a new ν is added or removed. Let vr be the vote result,

$$vr = \bigcup \partial \mathcal{I}_\Phi(\mathfrak{o}_i), \quad \forall i \in (n_\nu - 1) \cap \Phi \in \{0, 1\}.$$

Here, $\mathfrak{o}$ is the opinion shared by other ν, Φ is the issue type (i.e., $0 =$ remove a ν and $1 =$ add a ν), and n_ν is the total ν number. Every ν gets a chance to make blocks $\mathfrak{b}'$ in a round-robin algorithm. When a ν proposes a block, other ν checks that it is the proposer's turn to propose a block and whether the block contains proper data or not. After successful verification, each ν provides their vote. Let vb be the vote result,

$$vb = \bigcup \partial \mathcal{V}_{\mathfrak{b}'}(\mathfrak{o}_i), \quad \forall i \in (n_\nu - 1).$$

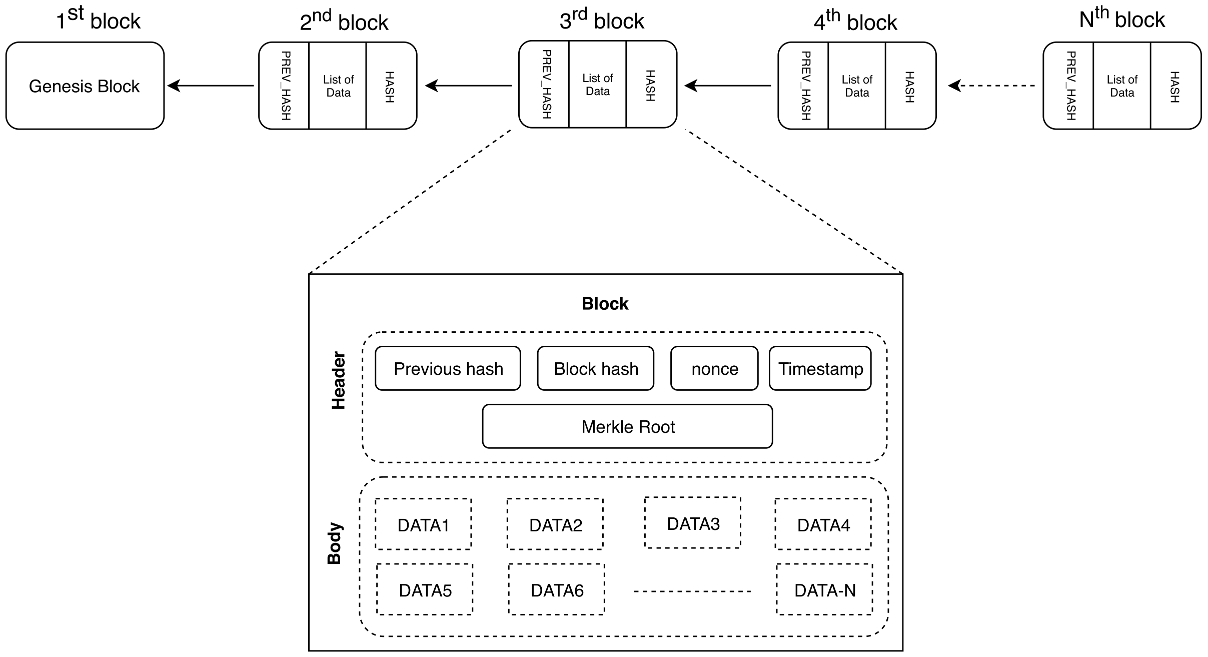

FIGURE 2. **Data management inside blockchain.**

If a ν tries to make a block before its turn, other validators cast a vote against ν and remove ν from the network. However, each block contains a t number of transactions, as shown in FIGURE 2. In BUS, each block contains multiple data in the same block. The block is divided intro part parts, such as (1) Header and (2) Body, as shown in FIGURE 2. The header contains the hash of the block, nonce, timestamp, Merkle root and hash of the previous block. The body contains data that is collected from IoT Devices. There may store one or more data in the body. Let $\mathfrak{b}'$ be the proposed block,

$$\mathfrak{b}' = \langle \mathfrak{b}_{p\hbar}, \mathfrak{b}_{\hbar}, \mathfrak{n}, \tau_{\mathfrak{b}}, \Re, \cup(\varpi_i)\rangle,$$
$$\forall i \in t \cap \mathfrak{b}_{\hbar} = \Xi(\mathfrak{n}, \tau_{\mathfrak{b}}, \Re) \mid \Xi : \{0, 1\}^* \mapsto \{0, 1\}^{\ell}.$$

Here, $\mathfrak{b}_{p\hbar}$ is the hash of the previous block, $\mathfrak{b}_{\hbar}$ is the hash of $\mathfrak{b}'$, $\mathfrak{n}$ is the nonce, $\tau_{\mathfrak{b}}$ is the timestamp when $\mathfrak{b}'$ is created, $\Re$ is the Merkle Root, and ϖ is the data inside the block.

IV. SECURITY ANALYSIS

A. PROTECTION AGAINST MAN-IN-THE-MIDDLE ATTACK

There are two types of Man-In-The-Middle (MITM) attacks, namely (1) eavesdropping and (2) manipulation [52]. Regarding eavesdropping, the attacker resides between the two parties and stores the data that passes between them. Then, the attacker analyzes and later leaks this data. In a manipulation attack, the attacker also resides between two parties. However, here the attacker catches packets from the sender, and then manipulates the packets and sends them to the receiver. The receiver thinks that these packets are coming directly from the sender. In BUS, the vulnerabilities are in the following connections (1) from the IoT device σ to minion UAV ϑ_m, (2) from ϑ_m to the emissary UAV ϑ_e, and (3) from ϑ_e to the server Ψ. Initially, when a ϑ_m starts communicating with σ, ϑ_m encrypts data utilizing σ's ρ_κ and, in order to give a response, σ encrypts data utilizing ϑ_m's ρ_κ. To decrypt the data, an attacker needs to know the sender and receiver's ϱ_κ. However, this is only known to the owner of the keys (i.e. the sender and the receiver). After the initial setup, ϑ_m and σ communicate utilizing $\wp_\kappa$ shared during the initial setup and is only known to ϑ_m and σ. The same process occurs when ϑ_m and ϑ_e, and ϑ_e and Ψ start to communicate with each other. As attackers do not have either the private key or shared key, they can not decrypt the data and conduct a MITM attack in BUS.

B. KEY-SPOOF RESISTANCE

Attackers can not directly perform a MITM attack because ϱ_κ and $\wp_\kappa$ are not known to attackers. In order to figure out ϱ_κ and $\wp_\kappa$, an attacker Ω must guess ϱ_κ and $\wp_\kappa$. In BUS, ϱ_κ and $\wp_\kappa$ are generated utilizing Equation 1 and Equation 3, respectively. Suppose that, ϱ_κ is 32 bytes or 256 bits long and $\wp_\kappa$ is 16 bytes or 128 bits long. In order to predict the correct ϱ_κ and $\wp_\kappa$, Ω needs to figure out the sequence of 256 and 128, respectively. For 256 bits and 128 bits, there are 2^{256} and 2^{128} possible sequences, respectively. The probability of guessing ϱ_κ, which is 256-bit long, is $\frac{1}{2^{256}} = 2^{-256}$, which is not practically feasible. The probability of guessing $\wp_\kappa$, which is 128-bit long, is $\frac{1}{2^{128}} = 2^{-128}$, which will take a very long time to guess. Moreover, if the key length is unknown to the attacker, the difficulties increase more. Besides, as $\wp_\kappa$ is a temporary key, it might get changed while one is attempting a guess.

C. RESISTANCE AGAINST INTRUSION

BUS utilizes a two-phase verification process. To get accepted in ϑ and Ψ, first, a sender has to pass π-hash bloom filter and then, the sender has to pass the signature verification process. As a σ doesn't hold the list of ϑ, σ only utilizes the signature verification process. However, if an attacker wants to take control over the UAV or IoT device, an attacker has to manage a valid identity in the network, otherwise, the attacker cannot infiltrate the system.

D. DATA TAMPERING RESISTANCE

In BUS, data acquisition is performed in σ and then, data travels through ϑ_m, ϑ_e, and finally, reaches Ψ. As data is decrypted in ϑ_m and ϑ_e, there might be remaining vulnerabilities where data experiences illegal alteration. In BUS, before transmitting data, σ creates a digital signature based on the transferred data. A single altered bit causes a failure in the signature verification process. If Ψ finds that the data has tampered when verifying the digital signature, Ψ discards the data immediately. In this way, BUS prevents data from being altered before arriving at the Ψ. However, after data are stored in BUS, it may be vulnerable to illegal tampering. To prevent that, BUS utilizes blockchain. In BUS, there is a list of validators and each validator identity is known. When a validator creates a block, other validators check that block and validate the block based on whether it is the requester's turn to create a block and the content of the block. Upon successful validation, the block is appended in the network. If anyone tries to make any illegal alteration, the chain of the block is broken and validator discards this data and synchronizes the chain from other validators. In blockchain, every block has its own unique hash generated from the content, nonce, timestamp, etc. Each block retains the hash of the previous block, except for the genesis block, and this way, blocks are chained together. Blockchain utilizes a Merkle tree in order to maintain the integrity of the data. A Merkle tree contains the hash of the data in all of its leaf nodes. If any change is made to the data, the hash of the Merkle tree also changes which in turn changes the hash of the block. This way, the chain breaks from this block onward. However, to make the changes acceptable, every validator has to agree to this change which is not technically feasible.

E. PROTECTION AGAINST REPLY ATTACK

A replay attack is an attack in which attacker stores transmitted data packets and utilizes these packets to gain access in the system or to overflow the system with malicious data [53]. Between σ and ϑ_m, between ϑ_m and ϑ_e, and between ϑ_e and Ψ, there are vulnerabilities where an attacker can launch a reply attack. During data requisition, there is a two-phase communication between σ and ϑ_m. First, σ requests a time slot. After getting the slot, σ transmits the data. During the communication, σ utilizes a nonce $\hat{n}$ derived from the timestamp. Before providing any slot, ϑ_m verifies the $\hat{n}$ utilizing Equation 6. After getting a slot, σ verifies the received nonce based on the addition of the transmitted $\hat{n}$. However, when ϑ_m gets the data, ϑ_m verifies it based on the assigned slot. If the received data packet does not have a valid time slot, ϑ_m discards the data. ϑ_m and ϑ_e also utilize nonce and time slot verification before forwarding the data. When ϑ_e transmits data to Ψ, ϑ_e provides $\hat{n}$ in the data. Upon receiving the data, Ψ verifies $\hat{n}$ using Equation 11. As the data in BUS must undergo a time verification process, a replay attack is technically very difficult in BUS.

TABLE 3. Simulation parameters.

Parameters	Values
Devices	[400,..., 9800]
π	[1, 2, 3]
MECS_{CPU}	Intel(R) Core(TM) i5-4670 CPU @ 3.40GHz
$\text{MECS}_{\text{Memory}}$	32GB
MECS_{OS}	Ubuntu 18.04.2 LTS (Bionic Beaver)
UAV_{CPU}	Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
$\text{UAV}_{\text{Memory}}$	1GB
UAV_{OS}	Raspbian Stretch

V. RESULT AND DISCUSSION

This section represents the simulation and the experimental results of BUS. These results are explained in two different subsections.

A. SIMULATION RESULTS

Simulations are performed utilizing MATLAB for estimating the effects of π-hash bloom filter in the server and the effects of applying π-hash bloom filter along with role selection and data synchronization in UAV. Parameters for the simulation are provided in Table 2.

FIGURE 3(a) illustrates the time needed to process π-hash bloom filter data required for validating the user. With an increase in the number of devices, processing time also increases. With the increase of π, processing time also increases because more filters are used in data generation.

FIGURE 3(b) shows the expected transmission of data generated using π-hash bloom filter. The data size is measured in bytes. As the number of devices increases, the generated π-hash bloom filter data also increases. Indeed, to integrate all the devices, the data table in π-hash bloom filter also increases. For the same number of devices, $\pi = \{1, 2, 3\}$ generates the same data size. Indeed, π-hash bloom filter requires 1 bit, either 0 or 1 in the data table.

FIGURE 3(c) depicts the expected transmission of data with respect to π-hash bloom filter and non π-hash bloom filter (NHBF). In this simulation, $\pi = 1$ was used, a hash table was considered as NHBF, and data size was measured in bytes. However, with an increase in the number of devices, data size in both π-hash bloom filter and NHBF increases. But, in contrast with π-hash bloom filter, NHBF generates three times more data because, to validate the data, NHBF stores hash string of the identity in the table which causes the

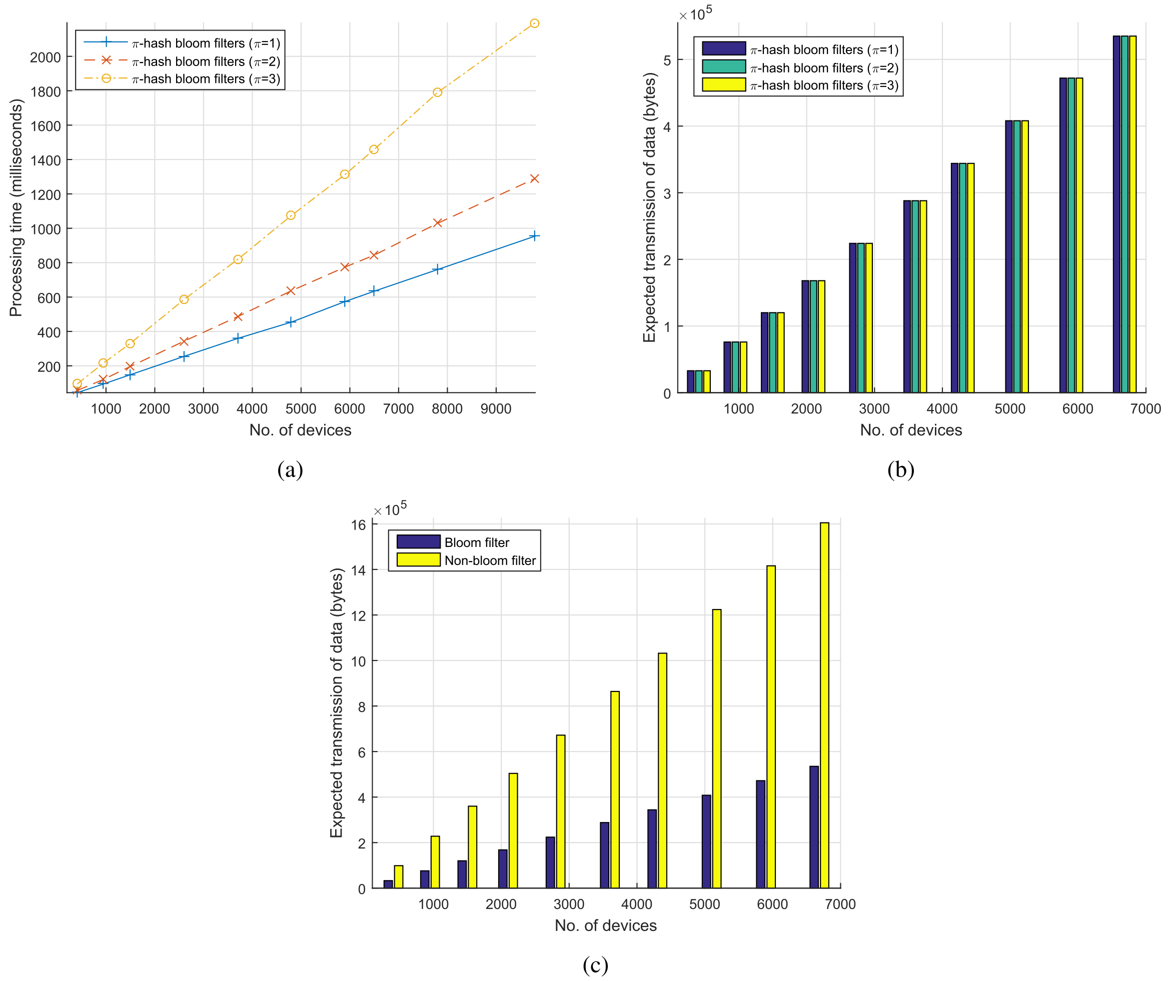

FIGURE 3. **π-hash bloom filter simulation result performed in the server: (a) processing time vs. number of devices for generating π-hash bloom filter data, (b) expected transmission of π-hash bloom filter data for different number of devices, and (c) expected transmission of π-hash bloom filter data w.r.t π-hash bloom filter and without π-hash bloom filter for different number of devices.**

increase in the size of the data. Overall, by utilizing π-hash bloom filter, BUS saves a lot of space in UAV for validating IoT devices.

FIGURE 4 presents the result of the simulation performed on the UAV to check the feasibility of π-hash bloom filter as well as the effectiveness of role selection and data synchronization algorithm. FIGURE 4(a) depicts the time needed to validate devices utilizing π-hash bloom filter in the UAV. As the number of devices increases, the time required for the validation also increases. Indeed, every device has to experience hash generation before checking the existence in the data set and the validation time increases when generating the hash. Moreover, with an increase in π, the validation time also increases because an increase in π means more filters are added in the π-hash bloom filter.

FIGURE 4(b) represents the energy consumption in joule (J) while performing the validation in the UAV. With an increase in the number of devices, the energy consumption for validating devices also increases. Indeed, every device has to experience hash generation before checking the existence in the data set and the validation time increases for generating the hash. With the increase in validation time, energy consumption also increases. Moreover, with an increase of π, the energy consumption for validating also increases because an increase in π means more filters are added in the π-hash bloom filter.

FIGURE 4(c) illustrates the time needed for validating a device in the presence of malicious devices. 10000 devices were considered for this simulation. With an increase in the number of π, validation time also increases. Because more filters were added when π was increased. Moreover, an increase in the malicious device ratio doesn't impact validation time. Before checking a device, π-hash bloom filter generates an index of the device utilizing hash functions. π-hash bloom filter then checks whether the index hash is 0 or 1 in the dataset. For either a valid or a malicious device, π-hash bloom filter follows this same procedure. This explains why the validation time is not impacted.

FIGURE 4(d) describes the false positive rate of π-hash bloom filter in the presence of malicious devices. However, the false positive rate is observed as 0 even with a higher percentage of malicious devices. This result shows that BUS is effectively validating the devices and provides protection against malicious devices. An increase in π does not affect the false positive rate in π-hash bloom filter. That's mean, π-hash bloom filter successfully validates the devices.

FIGURE 4(e) shows the time needed for selecting the role among UAVs. A UAV's internal Wi-Fi module was utilized to create an ad-hoc network. Equation 2 was considered for this simulation. With an increase in the number of drones, the role selection time also increases. With more UAVs, more time is required to perform steps such as encryption, decryption, signature creation, and signature verification. Moreover, with the increase in π, the role selection time also increases because an increase in π means more filters are added in the π-hash bloom filter. However, this increase in selection time is too small to be considered.

FIGURE 4(f) demonstrates the time required to synchronize control data (i.e. trust token and shared key) with IoT devices from a UAV. Equation 4 was considered for this simulation. However, with an increase in the number of IoT devices, the synchronization time also increases. With more IoT devices, more time is required to perform steps such as encryption, decryption, signature creation, and signature verification. Moreover, an increase in π, the synchronization time also increases. However, this increase in time is too small to be considered.

TABLE 4. Experimental parameters.

Parameters	Values
π	[1,2,3]
$\Xi(.)$	SHA-256
Encryption_{sym}	AES (key = 128 bit)
Encryption_{asym}	ECC (key = 168 bit)
Sign & verify	ECC (key = 168 bit)
IoTD_{n}	4
IoTD_{CPU}	Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
IoTD_{Memory}	1GB
IoTD_{OS}	Raspbian Stretch
$\text{IoTD}_{Bluetooth}$	Bluetooth 4.2
IoTD_{Sensor}	Flame, Temperature, Humidity, Light
MECS_{CPU}	Intel(R) Core(TM) i5-4670 CPU @ 3.40GHz
MECS_{Memory}	32GB
MECS_{OS}	Ubuntu 18.04.1 LTS (Bionic Beaver)
$\text{MECS}_{Wi\text{-}Fi}$	ipTime N100 mini
UAV_{n}	3
UAV_{CPU}	Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
UAV_{Memory}	1GB
$\text{UAV}_{Wi\text{-}Fi}$	2.4GHz IEEE 802.11.b/g/n/ac wireless LAN
UAV_{OS}	Raspbian Stretch
Blockchain	Ethereum
Blockchain client	Geth
Communication	Web3.py (JSON-RPC)
Consensus	POA
Validators	[1,...,10]

B. EXPERIMENTAL RESULTS

The parameters utilized in the experiment are presented in Table 4. In the experiment, a MEC server (MECS) was considered. The setup contains 4 IoT devices, 3 UAVs, and 1 MECS. Raspberry Pi 3 model b+ was used as an IoT device and various types of sensors (e.g. temperature, humidity, light, and flame) were attached to these devices. The middleware in the IoT devices was written in python. The communication between the IoT device and the UAV was performed over Bluetooth. 1 DJI Mavic 2 Pro and 2 Parrot Bebop 2 were used for UAV, and raspberry pi 3 model b+ was attached to each of the UAV to maintain communication with the IoT device and the MECS. Random waypoint model was considered during the data acquisition via UAV. The physical layer parameters are not considered in our experiment as our motivation was to observe the impact of the security module in real hardware during secure data acquisition. UAV created an ad-hoc network utilizing internal Wi-Fi module in order to communicate among UAVs. Intel(R) Core(TM) i5-4670 CPU @ 3.40GHz with 32GB memory was considered as the MECS. An ipTime N100 mini was utilized to create an access point from the MECS that was utilized to communicate with UAVs. The middleware in the MECS was also written in Python. Blockchain in BUS was implemented on top of Ethereum. There, a consortium network was constructed, named BUS-BC, comprising 10 validators. Geth was utilized as an Ethereum client and web.py was used for JSON-RPC. For consensus, proof of authority (PoA) was exercised for proposing and validating the blocks.

FIGURE 5 represents the results of the experiments performed in BUS. The throughput of the implemented BUS is provided in FIGURE 5(a). With an increase in π, the throughput decreases. Indeed, as π increased, more filters were added, requiring additional time to filter data. Moreover, the throughput decreases over time as the channel is not constant and also depends on the responsiveness of the devices that validate and forward the data. The result shows that it's needed to be careful while choosing the π as it affects the network.

FIGURE 5(b) shows energy consumption in an IoT device during data transmission. BUS is compared with [27], [32], and cloud computing. In FIGURE 5(b), cloud computing consumes more energy because in order to connect with the cloud IoT devices have to spend more energy than edge server. Both [32] and [27] provide a better result than the cloud in terms of energy consumption during data transmission because, in these schemes, they considered edge server. However, [27] consumes more energy than [32] because [27] transmits data over Wi-Fi and [32] uses IEEE 802.15.6 protocol during the transmission and IEEE 802.15.6 is low powered than Wi-Fi. BUS shows a better result than [32] because of BUS considered Bluetooth low energy (BLE) in the transmission. The results show that

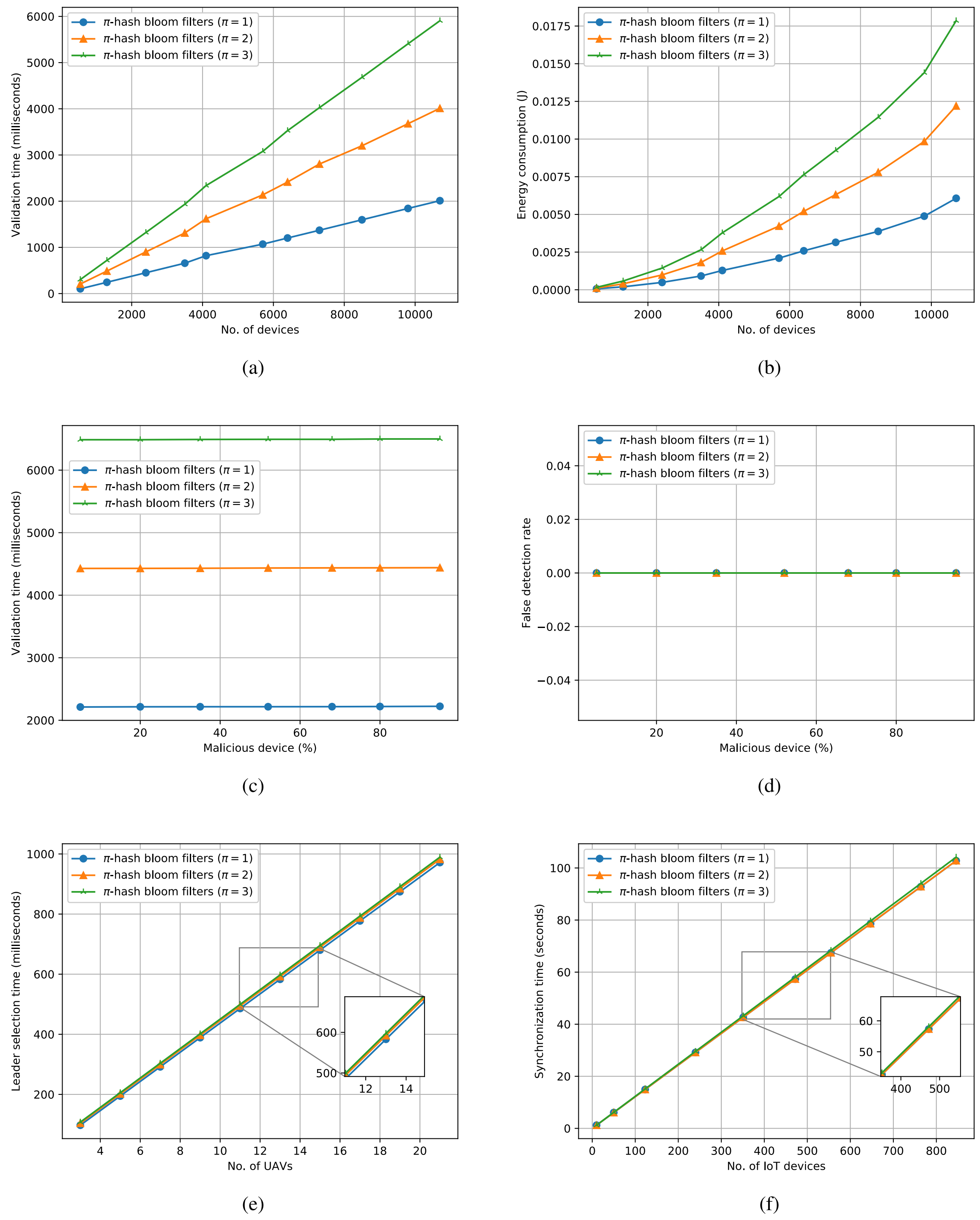

FIGURE 4. Simulation result performed in UAV: (a) validation time vs. number of devices for validating using π-hash bloom filter, (b) energy consumption vs. number of devices for validating using π-hash bloom filter, (c) validation time vs. percentage of malicious devices for validating using π-hash bloom, (d) false detection rate in π-hash bloom filter for validating different percentage of malicious devices, (e) Role selection time vs. number of UAVs for selecting role among UAVs before starting data acquisition, and (f) synchronization time vs. number of IoT devices for synchronizing control data before starting data acquisition.

BUS provides better support to IoT devices in terms of data transmission than existing data acquisition schemes.

FIGURE 5(c) illustrates the time required for processing data during data acquisition. It includes the results from the IoT device, Emissary UAV (UAV$_e$), Minion UAV (UAV$_m$), and MEC server. For the IoT device (phase-1), Equation 5 was utilized. The processing time increases as the size of the data increases in the IoT device (phase-1). For the IoT device

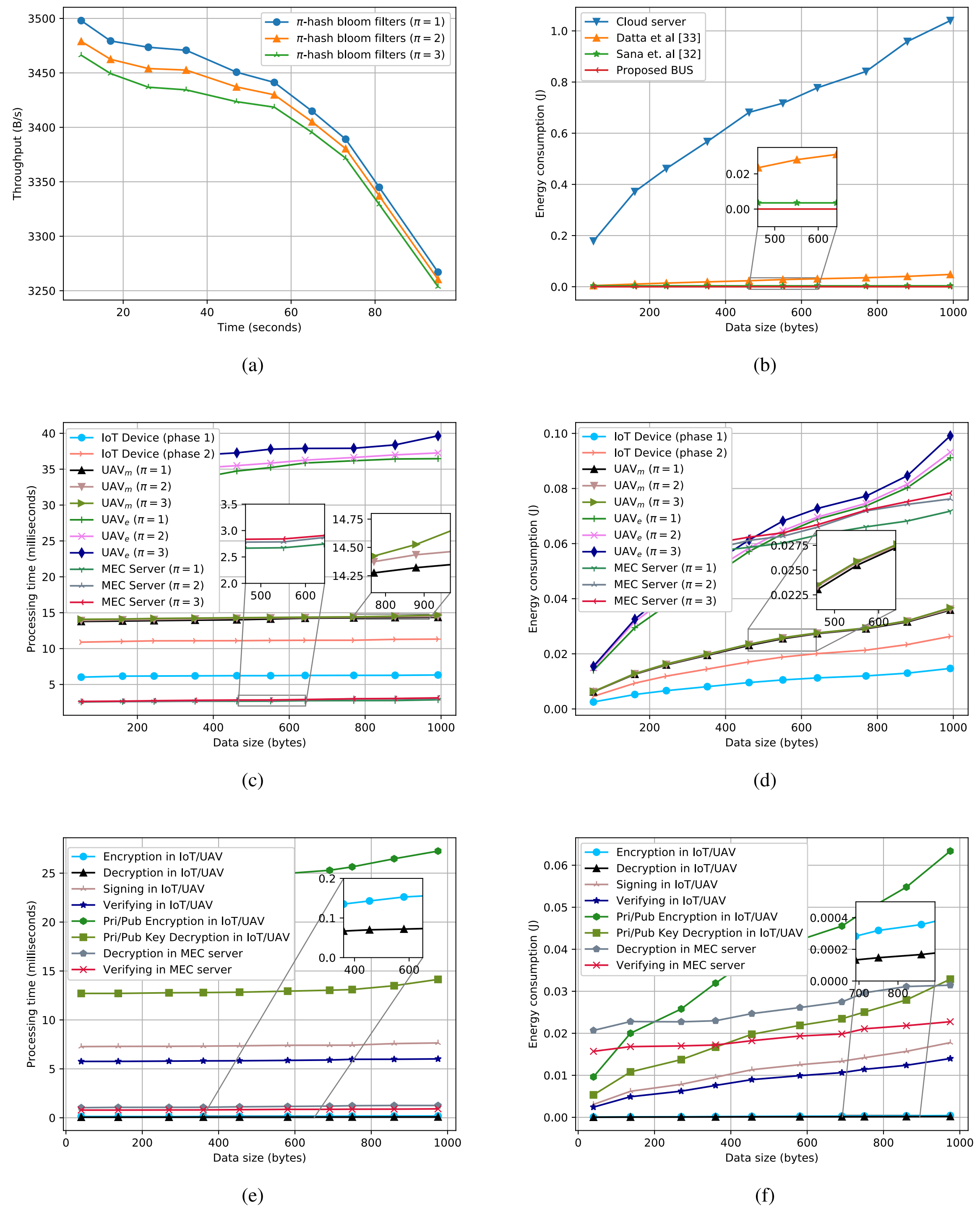

FIGURE 5. Experimental result performed in the BUS network: (a) throughput vs. time for different number of π, (b) energy consumption vs. data size for communicating with different entity from IoT devices, (c) processing time vs. data size of performing task in different entities, (d) energy consumption vs. data size during performing task in different entities, (e) processing time vs. data size for individual security actions, and (f) energy consumption vs. data size for performing individual security actions.

(phase-2), Equation 9 was utilized. Similarly, here, with an increase in data size, the processing time also increases. However, it is too small to consider. In UAV$_m$, Equation 8 was utilized in this experiment. With an increase in data size, the processing time also increases and for an increase in π, the processing time also increases for UAV$_m$. In UAV$_e$, Equation 10 was utilized in this experiment. With an increase in data size, the processing time also increases and for an increase in π, the processing time also increases for UAV$_e$. However, UAV$_e$ utilizes asymmetric encryption while com-

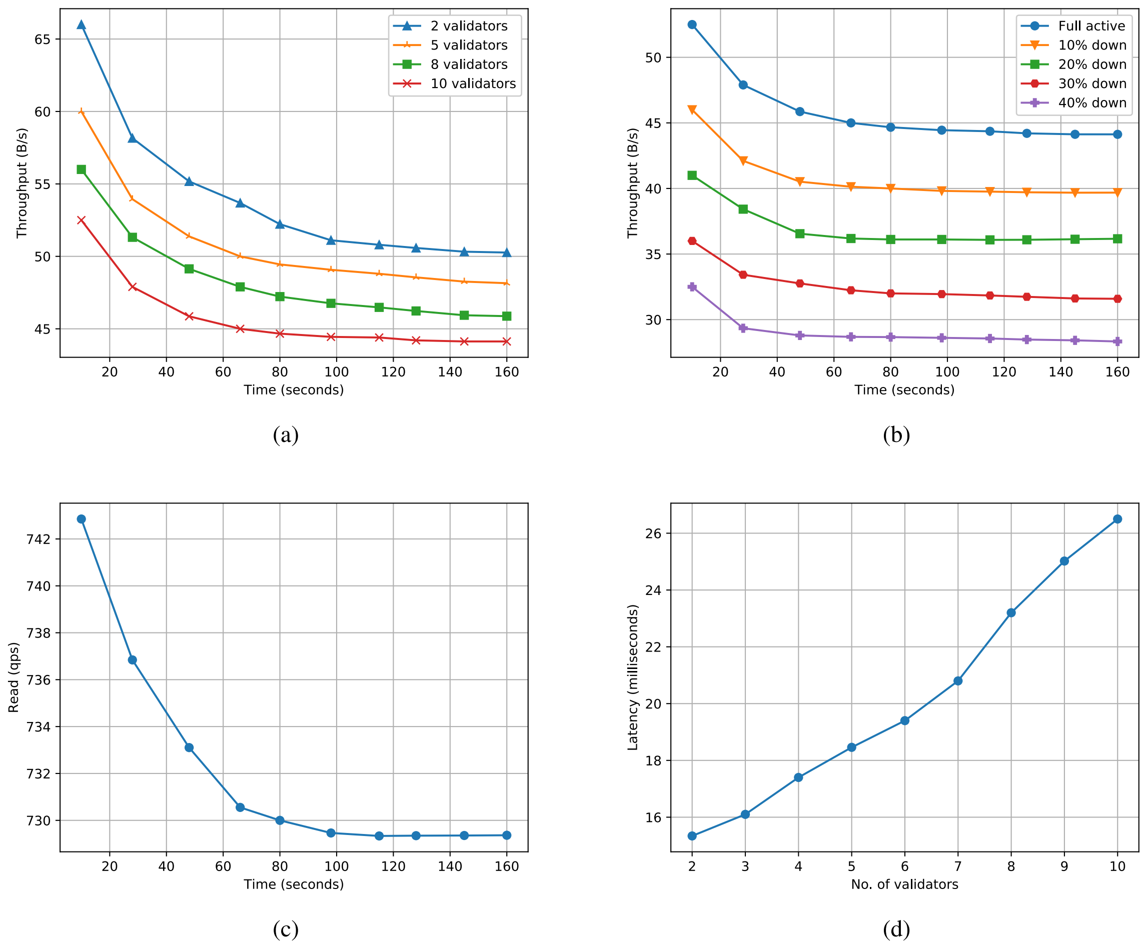

FIGURE 6. Experimental result performed in blockchain: (a) throughput vs. time for different number of validators, (b) throughput vs. time while different number of validators are not present in the network, (c) query performed in blockchain over time, and (d) latency for adding a data in the presence of different validators.

municating with the MEC server that leads to an increase in processing time for data. In the MEC server, $\pi = \{1, 2, 3\}$ was considered and Equation 12 was utilized in this experiment. However, the MEC server has significant computational power which causes less time for processing data. The result shows that, if IoT devices connect directly with the MEC server then IoT devices have to spend more time in preparing data than time for preparing data for UAV.

FIGURE 5(d) depicts the energy consumption for processing data during data acquisition. It includes the results from the IoT device, Emissary UAV (UAV_e), Minion UAV (UAV_m), and MEC server. For IoT device (phase-1), Equation 5 was utilized. The energy consumption increases with the increase in data size in the IoT device (phase-1). Because of an increase in data size, more processing time is required which in turn causes more energy consumption. For the IoT device (phase-2), Equation 9 was utilized. Here, with the increase in data size, processing time also increases leading to higher energy consumption. In UAV_m, Equation 8 was utilized in this experiment. With an increase in data size, energy consumption also increases and for an increase in π, the energy consumption also increases for UAV_m. In UAV_e, Equation 10 was utilized in this experiment. With an increase in data size, the energy consumption also increases and for an increase in π, the energy consumption also increases for UAV_e. However, UAV_e utilizes asymmetric encryption while communicating with the MEC server, which increases time for processing data leading to higher energy consumption. In the MEC server, $\pi = \{1, 2, 3\}$ was considered and Equation 12 was utilized in this experiment. However, the MEC server requires a lot of power to operate but only CPU power consumption while processing data is considered. The result shows that, if IoT devices connect directly with the MEC server then IoT devices have to spend more energy in preparing data than energy for preparing data for UAV.

FIGURE 5(e) demonstrates the time required for processing a data security task (e.g., sign, verify, encryption, and decryption) during data acquisition. As IoT device and UAV are utilizing the same device for communicating, they are both mentioned in the same label. In the IoT device, encryption (symmetric), decryption (symmetric), signature, verification, pri/pub encryption (asymmetric), and pri/pub decryption (asymmetric) were considered during the experiment. With the increase in data size, the processing time for each of the task also increases. In the MEC server, the verification, and decryption (asymmetric) were considered during the experiment. With the increase in data size, the processing time for each task also increases. As the MEC server has significant computational power, it requires less processing time.

FIGURE 5(f) outlines the energy consumption to process a data security task (e.g., signature, verification, encryption, and decryption) in the data acquisition phase. As the IoT device and the UAV utilize the same device to communicate, they are both mentioned in the same label. In the IoT device, encryption (symmetric), decryption (symmetric), signature, verification, pri/pub encryption (asymmetric), and pri/pub decryption (asymmetric) were considered during the experiment. With the increase in data size, the processing time for each task also increases which leads to higher energy consumption. In the MEC server, the verification, and decryption (asymmetric) were considered during the experiment. With the increase in data size, the processing time for each task also increases which leads to higher energy consumption. However, the MEC server requires a lot of power to operate but only CPU power consumption while processing data is considered.

FIGURE 6 provides information on the experimental results that were performed in blockchain. FIGURE 6(a) illustrates the throughput of blockchain over time. In BUS, the throughput measures the quantity of data that are being added to blockchain over time. The throughput was measured in the presence of 2, 5, 8, and 10 validators. As time elapses, the throughput gradually decreases. Indeed, the network status changes over time and the responsiveness of the validator is also relevant to the decrease in throughput. With an increase in the number of validators, the throughput also decreases. Indeed, more validators mean more time required for the data to be accepted.

FIGURE 6(b) points to the changes in throughput over time when the validator is missing. 10 validators were considered during the experiment. With an increase in the number of missing validators, the throughput also decreases. In the PoA, every validator has an opportunity to create blocks and this process follows a round robin algorithm. When one validator does not respond, his turn is skipped to the benefit of the next validator. This, in turn, creates a delay in the network and the dynamic network status may also cause a decrease in the throughput.

FIGURE 6(c) portrays the number of queries that can be executed over time. As, in blockchain, everyone holds an identical copy of the data and while querying data from blockchain, data is read from the local copy of the reader. Hence, the number of validators has no effect when querying data from blockchain. The query per second (qps) was used in the experiment. As time elapses, the number of queries decreases. Query processing is dependent on the responsiveness of the blockchain and the system on which blockchain is set up. As time elapses, the responsiveness of the blockchain, as well as the system, decreases leading to a reduction in the number of queries processed.

FIGURE 6(d) outlines the latency when adding data in the presence of a different number of validators in blockchain. Validators from 2 to 10 were considered in the experiment. With the increase in the number of validators, the latency also increases. Based on this, the addition of a validator causes a delay in the network when signing blocks, and the latency, in turn, increases due to rising network delays.

VI. CONCLUSION AND FUTURE WORKS

In the paper, a UAV swarm assisted data acquisition scheme (termed as ''BUS'') is introduced in which data is collected from IoT devices via a UAV swarm and then stored in the nearest server with the assistance of blockchain. A smart contract is employed in order to handle the IoT devices and missions in BUS. Prior to starting the data acquisition mission, UAVs define roles by themselves utilizing the resources (e.g., CPU, battery, ram, etc.) available to them. Upon deciding roles, UAVs create a shared key to maintain communications. After completing data acquisition, the server creates a block and requests permission from other validators. Upon receiving consent from all validators, this data is added in blockchain. A security analysis is conducted to show the feasibility of BUS in terms of providing security. Simulations were conducted using MATLAB and Python to investigate the effect of π-hash bloom filter in the server and the UAV, respectively. The result of the simulation shows that BUS is successfully utilized π-hash bloom filter and can filter malicious devices completely. BUS was implemented and experiments were conducted in that implementation to test the feasibility. The result and security analysis demonstrates that utilizing UAV in the assistance of IoT devices not only extends the connectivity but also helps to reduce the energy consumption in IoT. Currently, BUS adopts consortium blockchain in the proposed scheme. BUS has planned to use a public blockchain in the future extension of this paper. However, using public blockchain raises the privacy issue of the data that requires more investigation. This issue is kept for the future extension of this paper. Moreover, BUS does not support the incentivization of the validator; this may instead be considered in a future extension of this paper. Furthermore, BUS considers satellite communication in remote areas, which requires more research in terms of secure data acquisition, is going to be explored in details in the future extension of this paper.

REFERENCES

[1] P. Asghari, A. M. Rahmani, and H. H. S. Javadi, ''Internet of Things applications: A systematic review,'' *Comput. Netw.*, vol. 148, pp. 241–261, Jan. 2019.

[2] V. Tsiatsis, S. Karnouskos, J. Holler, S. Avesand, D. Boyle, and C. Mulligan, *Internet of Things*, 2nd ed. New York, NY, USA: Academic, 2019, pp. 3–7.

[3] R. Dagar, S. Som, and S. K. Khatri, ''Smart farming—IoT in agriculture,'' in *Proc. Int. Conf. Inventive Res. Comput. Appl. (ICIRCA)*, Jul. 2018, pp. 1052–1056.

[4] S. J. Balakrishna, H. Marellapudi, and N. A. Manga, "IoT based status tracking and controlling of motor in agricultural farms," in *Proc. 5th IEEE Uttar Pradesh Sect. Int. Conf. Elect., Electron. Comput. Eng. (UPCON)*, Nov. 2018, pp. 1–5.

[5] I. Zyrianoff, A. Heideker, D. Silva, and C. Kamienski, "Scalability of an Internet of Things platform for smart water management for agriculture," in *Proc. 23rd Conf. Open Innov. Assoc. (FRUCT)*, Nov. 2018, pp. 432–439.

[6] V. Ramachandran, R. Ramalakshmi, and S. Srinivasan, "An automated irrigation system for smart agriculture using the Internet of Things," in *Proc. 15th Int. Conf. Control, Automat., Robot. Vis. (ICARCV)*, Nov. 2018, pp. 210–215.

[7] M. M. Alam, H. Malik, M. I. Khan, T. Pardy, A. Kuusik, and Y. L. Moullec, "A survey on the roles of communication technologies in IoT-based personalized healthcare applications," *IEEE Access*, vol. 6, pp. 36611–36631, 2018.

[8] D. He, R. Ye, S. Chan, M. Guizani, and Y. Xu, "Privacy in the Internet of Things for smart healthcare," *IEEE Commun. Mag.*, vol. 56, no. 4, pp. 38–44, Apr. 2018.

[9] A. A. Abdellatif, M. G. Khafagy, A. Mohamed, and C.-F. Chiasserini, "EEG-based transceiver design with data decomposition for healthcare IoT applications," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 3569–3579, Oct. 2018.

[10] M. Elhoseny, G. Ramírez-González, O. M. Abu-Elnasr, S. A. Shawkat, N. Arunkumar, and A. Farouk, "Secure medical data transmission model for IoT-based healthcare systems," *IEEE Access*, vol. 6, pp. 20596–20608, 2018.

[11] E. Luo, M. Z. A. Bhuiyan, G. Wang, M. A. Rahman, J. Wu, and M. Atiquzzaman, "Privacyprotector: Privacy-protected patient data collection in IoT-based healthcare systems," *IEEE Commun. Mag.*, vol. 56, no. 2, pp. 163–168, Feb. 2018.

[12] C. K. Lee, H. Liu, D. Fuhs, A. Kores, and E. Waffenschmidt, "Smart lighting systems as a demand response solution for future smart grids," *IEEE J. Emerg. Sel. Topics Power Electron.*, to be published.

[13] Y. Li, X. Cheng, Y. Cao, D. Wang, and L. Yang, "Smart choice for the smart grid: Narrowband Internet of Things (NB-IoT)," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1505–1515, Jun. 2018.

[14] L. De M. B. A. Dib, V. Fernandes, M. de L. Filomeno, and M. V. Ribeiro, "Hybrid PLC/wireless communication for smart grids and Internet of Things applications," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 655–667, Apr. 2018.

[15] Y. Sun, L. Lampe, and V. W. S. Wong, "Smart meter privacy: Exploiting the potential of household energy storage units," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 69–78, Feb. 2018.

[16] H. A. H. Hassan, D. Renga, M. Meo, and L. Nuaymi, "A novel energy model for renewable energy-enabled cellular networks providing ancillary services to the smart grid," *IEEE Trans. Green Commun. Netw.*, vol. 3, no. 2, pp. 381–396, Jun. 2019.

[17] Y. Meng, W. Zhang, H. Zhu, and X. S. Shen, "Securing consumer IoT in the smart home: Architecture, challenges, and countermeasures," *IEEE Wireless Commun.*, vol. 25, no. 6, pp. 53–59, Dec. 2018.

[18] A. Elsts, X. Fafoutis, P. Woznowski, E. Tonkin, G. Oikonomou, R. Piechocki, and I. Craddock, "Enabling healthcare in smart homes: The SPHERE IoT network infrastructure," *IEEE Commun. Mag.*, vol. 56, no. 12, pp. 164–170, Dec. 2018.

[19] P. Wang, F. Ye, and X. Chen, "A smart home gateway platform for data collection and awareness," *IEEE Commun. Mag.*, vol. 56, no. 9, pp. 87–93, Sep. 2018.

[20] D. E. Kouicem, A. Bouabdallah, and H. Lakhlef, "Internet of Things security: A top-down survey," *Comput. Netw.*, vol. 141, pp. 199–221, Aug. 2018.

[21] J. Portilla, G. Mujica, J.-S. Lee, and T. Riesgo, "The extreme edge at the bottom of the Internet of Things: A review," *IEEE Sensors J.*, vol. 19, no. 9, pp. 3179–3190, May 2019.

[22] H. Shakhatreh, A. H. Sawalmeh, A. Al-Fuqaha, Z. Dou, E. Almaita, I. Khalil, N. S. Othman, A. Khreishah, and M. Guizani, "Unmanned aerial vehicles (UAVs): A survey on civil applications and key research challenges," *IEEE Access*, vol. 7, pp. 48572–48634, 2019.

[23] L. Gupta, R. Jain, and G. Vaszkun, "Survey of important issues in UAV communication networks," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1123–1152, 2nd Quart., 2016.

[24] S. Hayat, E. Yanmaz, and R. Muzaffar, "Survey on unmanned aerial vehicle networks for civil applications: A communications viewpoint," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 4, pp. 2624–2661, 4th Quart., 2016.

[25] Q. Zhang, M. Jiang, Z. Feng, W. Li, W. Zhang, and M. Pan, "IoT enabled UAV: Network architecture and routing algorithm," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3727–3742, Apr. 2019.

[26] T. Lagkas, V. Argyriou, S. Bibi, and P. Sarigiannidis, "UAV IoT framework views and challenges: Towards protecting drones as 'Things,'" *Sensors*, vol. 18, no. 11, p. 4015, Nov. 2018.

[27] S. K. Datta, J.-L. Dugelay, and C. Bonnet, "IoT based UAV platform for emergency services," in *Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Oct. 2018, pp. 144–147.

[28] W. Chen, B. Liu, H. Huang, S. Guo, and Z. Zheng, "When UAV swarm meets edge-cloud computing: The QoS perspective," *IEEE Netw.*, vol. 33, no. 2, pp. 36–43, Mar./Apr. 2019.

[29] M. Rosalie, G. Danoy, S. Chaumette, and P. Bouvry, "Chaos-enhanced mobility models for multilevel swarms of UAVs," *Swarm Evol. Comput.*, vol. 41, pp. 36–48, Aug. 2018.

[30] Z. Lihua, D. Jianfeng, W. Yu, and W. Zhiqiang, "An online priority configuration algorithm for the UAV swarm in complex context," *Procedia Comput. Sci.*, vol. 150, pp. 567–578, Jan. 2019.

[31] Z. Yuan, J. Jin, L. Sun, K.-W. Chin, and G.-M. Muntean, "Ultra-reliable IoT communications with UAVs: A swarm use case," *IEEE Commun. Mag.*, vol. 56, no. 12, pp. 90–96, Dec. 2018.

[32] S. Ullah, K.-I. Kim, K. H. Kim, M. Imran, P. Khan, E. Tovar, and F. Ali, "UAV-enabled healthcare architecture: Issues and challenges," *Future Gener. Comput. Syst.*, vol. 97, pp. 425–432, Aug. 2019.

[33] J.-J. Wang, C.-X. Jiang, Z. Han, Y. Ren, R. G. Maunder, and L. Hanzo, "Taking drones to the next level: Cooperative distributed unmanned-aerial-vehicular networks for small and mini drones," *IEEE Veh. Technol. Mag.*, vol. 12, no. 3, pp. 73–82, Sep. 2017.

[34] W. Feng, J. Wang, Y. Chen, X. Wang, N. Ge, and J. Lu, "UAV-aided MIMO communications for 5G Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1731–1740, Apr. 2019.

[35] F. Qi, X. Zhu, G. Mang, M. Kadoch, and W. Li, "UAV network and IoT in the sky for future smart cities," *IEEE Netw.*, vol. 33, no. 2, pp. 96–101, Mar./Apr. 2019.

[36] J. Wang, C. Jiang, Z. Wei, C. Pan, H. Zhang, and Y. Ren, "Joint UAV hovering altitude and power control for space-air-ground IoT networks," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1741–1753, Apr. 2019.

[37] V. Sharma, R. Kumar, and R. Kaur, "UAV-assisted content-based sensor search in IoTs," *Electron. Lett.*, vol. 53, no. 11, pp. 724–726, May 2017.

[38] X. He, J. Bito, and M. M. Tentzeris, "A drone-based wireless power transfer anc communications platform," in *Proc. IEEE Wireless Power Transf. Conf. (WPTC)*, May 2017, pp. 1–4.

[39] N. Mohamed, J. Al-Jaroodi, I. Jawhar, H. Noura, and S. Mahmoud, "UAVFog: A UAV-based fog computing for Internet of Things," in *Proc. IEEE SmartWorld, Ubiquitous Intell. Comput., Adv. Trusted Computed, Scalable Comput. Commun., Cloud Big Data Comput., Internet People Smart City Innov.*, Aug. 2017, pp. 1–8.

[40] S. Liu, Z. Wei, Z. Guo, X. Yuan, and Z. Feng, "Performance analysis of UAVs assisted data collection in wireless sensor network," in *Proc. IEEE 87th Veh. Technol. Conf. (VTC Spring)*, Jun. 2018, pp. 1–5.

[41] X. Wang, X. Zha, W. Ni, R. P. Liu, Y. J. Guo, X. Niu, and K. Zheng, "Survey on blockchain for Internet of Things," *Comput. Commun.*, vol. 136, pp. 10–29, Feb. 2019.

[42] I. Makhdoom, M. Abolhasan, H. Abbas, and W. Ni, "Blockchain's adoption in IoT: The challenges, and a way forward," *J. Netw. Comput. Appl.*, vol. 125, pp. 251–279, Jan. 2019.

[43] L. W. Cong and Z. He, "Blockchain disruption and smart contracts," *Rev. Financial Stud.*, vol. 32, no. 5, pp. 1754–1797, Apr. 2019.

[44] A. Islam, M. B. Uddin, M. F. Kader, and S. Y. Shin, "Blockchain based secure data handover scheme in non-orthogonal multiple access," in *Proc. 4th Int. Conf. Wireless Telematics (ICWT)*, Jul. 2018, pp. 1–5.

[45] J. Xu, L. W. Wei, Y. Zhang, A. D. Wang, F. C. Zhou, and C.-Z. Gao, "Dynamic fully homomorphic encryption-based Merkle tree for lightweight streaming authenticated data structures," *J. Netw. Comput. Appl.*, vol. 107, pp. 113–124, Apr. 2018.

[46] T. T. A. Dinh, R. Liu, M. Zhang, G. Chen, B. C. Ooi, and J. Wang, "Untangling blockchain: A data processing view of blockchain systems," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 7, pp. 1366–1385, Jul. 2018.

[47] F. Grandi, "On the analysis of Bloom filters," *Inf. Process. Lett.*, vol. 129, pp. 35–39, Jan. 2018.

[48] M. A. Mughal, X. Luo, A. Ullah, S. Ullah, and Z. Mahmood, "A lightweight digital signature based security scheme for human-centered Internet of Things," *IEEE Access*, vol. 6, pp. 31630–31643, 2018.

[49] W. Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, and A. Ahmed, "Edge computing: A survey," *Future Gener. Comput. Syst.*, vol. 97, pp. 219–235, Aug. 2019.
[50] H. H. Elazhary, "Internet of Things (IoT), mobile cloud, cloudlet, mobile IoT, IoT cloud, fog, mobile edge, and edge emerging computing paradigms: Disambiguation and research directions," *J. Netw. Comput. Appl.*, vol. 128, pp. 105–140, Feb. 2019.
[51] L. Luo, D. Guo, R. T. B. Ma, O. Rottenstreich, and X. Luo, "Optimizing Bloom filter: Challenges, solutions, and comparisons," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1912–1949, 2nd Quart., 2018.
[52] M. Conti, N. Dragoni, and V. Lesyk, "A survey of man in the middle attacks," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 2027–2051, 3rd Quart., 2016.
[53] H. S. Sánchez, D. Rotondo, T. Escobet, V. Puig, J. Saludes, and J. Quevedo, "Detection of replay attacks in cyber-physical systems using a frequency-based signature," *J. Franklin Inst.*, vol. 356, no. 5, pp. 2798–2824, Mar. 2019.

ANIK ISLAM was born in 1992. He received the B.Sc. degree in software engineering and the M.Sc. degree in computer science from American International University-Bangladesh (AIUB), Dhaka, Bangladesh, in 2014 and 2017, respectively. He is currently pursuing the Ph.D. degree with the WENS Laboratory, Kumoh National Institute of Technology, Gumi, South Korea. He has more than five years of working experience in the software development field. He has participated in various software competitions with good achievements. His current research interests include blockchain, the Internet of Things, unmanned aerial vehicle, the social Internet of Things, edge computing, and distributed systems.

SOO YOUNG SHIN (M'07–SM'17) received the Ph.D. degrees in electrical engineering and computer science from Seoul National University, in 2006. He was with WiMAX Design Lab, Samsung Electronics, Suwon, South Korea, from 2007 to 2010. He joined as a full-time Professor with the School of Electronics, Kumoh National Institute of Technology, Gumi, South Korea, where he is currently an Associate Professor. He was a Postdoctoral Researcher with the University of Washington, Seattle, WA, USA, from 2006 to 2007. He was a Visiting Scholar with The University of the British Columbia, in 2017. His current research interests include wireless communications, next generation mobile wireless broadband networks, signal processing, and the Internet of Things.