BUAV: A Blockchain Based Secure UAV-Assisted Data Acquisition Scheme in Internet of Things

Anik Islam and Soo Young Shin

Abstract: Internet of things (IoT), mobile edge computing (MEC), and unmanned aerial vehicle (UAV) have attracted significant attention in both industry and academic research. By consolidating these technologies, IoT can be facilitated with improved connectivity, better data transmission, energy saving, and other advantages. However, the communication between these entities is subject to potential cyber threats. In addition, the integrity of the data must be maintained after storing into local storage. Blockchain is a data structure that supports features like pseudonymity, data integrity, etc. This paper presented a blockchain-based data acquisition process in which information is gathered from IoTs using UAV as a relay and is securely kept in blockchain at MEC server. In the proposed scheme, data are encrypted prior to transfer to MEC server with the assistance of a UAV. Upon receiving the data, MEC server validates the data and the identity of the sender. Successful validation is followed by stocking of the data into blockchain, subsequent to obtaining consent from the validators. Security analysis is conducted in order to show the feasibility of the proposed secure scheme. Finally, the performance of the proposed scheme is analyzed by simulation and implementation.

Index Terms: Blockchain, Internet of things, MEC, security, unmanned aerial vehicle.

I. INTRODUCTION

INTERNET of things (IoT) has generated considerable attention not only in industrial fields but also in academic research. IoT is a network of objects, equipped with sensors, actuators, software, etc., that are connected via Internet. Moreover, these objects can collect information from their surroundings and can interchange these data among each other [1]. Based on the promising possibilities attributed to IoT, an estimation of appending 30 billion devices in the network by 2020 has been made [2]. However, IoT is not only confined to sensing and information interchange. Currently, IoT can make spontaneous decisions based on the sensed data. IoT has improved the quality of our lives by contributing to the development of several different sectors including healthcare [3]–[8], agriculture [9]–[12], smart homes [13]–[15], smart grids [16]–[20], and others [21].

Unmanned aerial vehicles (UAVs) is an example of an emerging technology that achieved widespread popularity owing to its potential for diverse applications. Initially, UAVs were confined to applications within the military. However, the use of these devices has subsequently been extended to civil applications [22]. The advantages of UAVs is their ease of deployment, low maintenance and acquisition cost, and the ability to access almost any areas [23]. Moreover, some UAVs contain sensors, actuators, high computational power, etc., which not only can accumulate data along with spatial information but also can process it, reach decisions and act according to the decision [24]–[26]. UAVs can be posted to locations that are difficult to access in order to provide assistance to IoTs. Occasionally, UAVs can operate as a server while providing services to IoTs. In addition, UAVs can spread network connection to IoTs for the non-line-of-sight zones. Moreover, UAVs can save the energy of devices by assisting in forwarding the data and can quickly be replaced in the occurrence of a fault, while providing assistance to IoTs. UAVs can also cover a large area which can help to optimize the deployment cost.

Mobile edge computing (MEC) is another promising initiative that draws cloud computing facilities closer to user proximity at the edge of the mobile network and provides services by utilizing radio access networks (RANs) [27]. The purpose of MEC is to provide ultra-low latency, high bandwidth, and real-time services to mobile subscribers [28]. However, collecting data from IoTs via the cloud creates latency issues, along with challenges related to low throughput and single-node failure [29]. MEC is a potential candidate for alleviating the aforementioned issues by providing real-time data acquisition services from IoT devices via the utilization of UAVs. However, the data acquisition process is prone to threats like the man-in-the-middle, spoofing, etc. Moreover, the collected data may experience unauthorized modifications in the storage, which may lead to uncertainty regarding the integrity of the data. Therefore, a secure scheme for data acquisition via UAV is required.

Among the existing studies, in [30], a UAV based fog computing platform that aids IoT devices has been proposed. In addition, an architecture in which information is obtained from IoT using UAV (sometimes UAV to UAV) and stored in the cloud has been appended. In [31], UAV served as a fog node so that it could provide real-time assistance to IoT devices. However, this may consume a significant amount of energy and may reduce the flight time of UAV. In [31], an approach was proposed in which UAVs are deployed based on client positions so that the client can perform queries and acquire data from IoTs with less...
A blockchain based data acquisition scheme is proposed in which data are gathered from IoTs via UAV and kept securely in blockchain at MEC, to alleviate the aforementioned issues. The major contributions of this paper are summarized as follows.

- A scheme is proposed to perform data acquisition from IoT devices.
- A discussion on security based on different vulnerabilities is provided.
- The impact of \( \eta \)-hash bloom filter\(^1\) is simulated both in MATLAB for edge server and Python for UAV.
- We have implemented the proposed scheme and the performance is examined by throughput, processing time, energy consumption, and latency.
- We have implemented blockchain using Ethereum and the performance is investigated in terms of write, read, latency, and delay.

The rest of the sections are arranged as follows: Section II represents the data acquisition scheme. In Section III, the process of data acquisition is depicted. Section IV illustrates a security analysis of the proposed scheme. The simulation and experimental setup are discussed in Section V. Finally, Section VI summarizes the main conclusions and examines future research directions.

II. PROPOSED SCHEME

We devise a blockchain based data acquisition scheme (termed as “BUAV”) that facilitates the collection of data from IoTs using a UAV, as presented in Fig. 1.

A. Basic Idea

BUAV concentrates on data acquisition from IoT devices using UAVs as well as the deposition of the collected data in a secure way to avoid compromising data integrity and BUAV also ensures secure data transmission. It was assumed that all valid IoT devices and UAVs are registered in the scheme. In BUAV, IoT devices intend to transmit data to the MEC. Before transferring data, IoT device performs encryption. This data is then transmitted to UAVs along with their identity. Upon receiving the data, UAVs perform decryption and check the validity of the identity of the sender. For validating devices, \( \eta \)-hash bloom filter is adopted in BUAV. However, if the identity is not valid, then UAVs discard the data and append that invalid identity to a vulnerable list. If the invalid identity continues to transmit data, then UAVs block the communication channel for that identity for a specific period. In BUAV, UAV decrypts and validates the data in order to prevent malicious devices at an early stage. If a malicious device continuously sends invalid data and UAV just forwards it to the server without any validation then MECs gets a lot of traffic containing invalid data. But, MECs cannot block the sender because data are coming through the UAV (if MECs wants to block the sender then it has to block UAV) and only UAV can identify the sender. That is why UAV decrypts the data and prevents malicious devices from reaching the server. However, for a valid identity, UAVs forward the data to the MEC server. When MEC server receives the data, MEC server verifies the validity of the sender. In case of a valid identity, MEC stores the data in blockchain. When data is added to the network, block sealing process is initiated. Upon receiving the acknowledgement from other validators, data is added to the network including a private cloud.

\(^1\)Bloom filter is a data structure to check the existence of an item in a set consuming less memory [43].
B. Entities

BUAV consists of four entities: IoT device (IoTID), UAV, MEC server (MECS), and private cloud (PC). The entities of BUAV are depicted below:

- **IoTID**: IoT device collects data and transmits it to the MECS. This device consists of sensors and actuators and it can both sense the environment and perform tasks according to the instruction.
- **UAV**: UAV receives data from IoTID and relays these data to the server. UAV also authorizes requests prior to forwarding to the server. Every UAV maintains a list of IoTIDs and other UAVs, which it utilizes to perform the authentication process using $\eta$-hash bloom filters before forwarding data to MECS.
- **MECS**: MEC server collects data from IoTID via UAV that are subsequently stored in blockchain. MEC not only maintains basic information (i.e., device name, mac address, device type, etc.) but also spatial information (i.e., latitude, longitude, etc.) that assists MEC server in maintaining the data acquisition process. MEC server also validates UAV and IoTID before including data into blockchain.
- **PC**: The private cloud is composed of private servers; the PC is linked with MECS via backhaul networks. Since the PC is also serving as a client of blockchain, it also gets the update when new data is added to MECS.

C. Notations

A list of common notations including their description is compiled in Table 1.

III. COMMUNICATIONS IN THE PROPOSED SCHEME

A. Registration

We assumed that valid IoTIDs and UAVs are registered users. Prior to registration in the system, UAV or IoTID generates a unique hash based on the mac address of the device $\delta_{mac}$, timestamp $\tau_c$, and random salt hash $\tilde{S}_h$. Let $\tilde{h}$ be the hash,

$$h = \psi(\delta_{mac}, \tau_c, \tilde{S}_h) | \psi : \{0, 1\}^* \rightarrow \{0, 1\}^l,$$

where $h$ is used to generate private key $\varphi_n$ and subsequently, a public key $\varphi_n$ is created from $\varphi_n$. Let $E$ be a point on the elliptic curve containing $(x, y)$,

$$\varphi_n = \Gamma(h) | \Gamma : \{0, 1\}^* \rightarrow \{0, 1\}^{k_x},$$

$$\bar{\varphi}_n = \varphi_n \otimes (E_x, E_y).$$

After generating keys, IoTID transmits $\varphi_n$ to MECS along with device information (i.e., $\delta_{mac}$, latitude, longitude, etc.). $\bar{\varphi}_n$ is used as an identity of a IoTID. MECS then stores these information in a smart contract $\tilde{C}_d$ of blockchain.

B. Data Generation

BUAV utilizes $\eta$-hash bloom filters ($\eta = 1, 2, \cdots$) to validate IoTIDs before forwarding data to MECS. MECS fetches the device list from $\tilde{C}_d$. Let $\tilde{d}$ be the list,
\[ \bar{\beta} = \bigcup_{\ell \leq 1} \bar{\beta}(y^\ell), \] where the location containing (lat, lon) and \( \ell \) is the radius of the location of \( \ell \). Prior to validating IoT device, UAV requires filter table \( \bar{\beta} \mid \bar{\beta} \in \{0, 1\} \). When a UAV registers or connect with MECS, MECS generates \( \bar{\beta} \) and shares it with UAV. Let \( \bar{\beta} \) is the generated table, \[
\bar{\beta} = \bigcup_{i=1}^{n} j^{\beta F}(d_i)
\] for positive rate \( \beta F \), and \( \bar{\beta} \) is number of hashes that are employed in the process. Bloom filter has a false positive issue that can be resolved by controlling the number of hash functions and data size [43]. However, there is a limit to the use of hash functions in the bloom filter. Let \( n \) is the total number of devices, and \( p \) is the acceptance of false positive rate [44], 
\[ m = \left[ -\frac{n \times \ln(p)}{\ln(2)^2} \right] \mid p \in (0, 1), \eta = \left[ \frac{m}{n} \times \ln(2) \right], \] where \( \eta \) is the maximum number of hash function that is suitable for \( \beta F \).

**C. Data Transmission**

In BUAV, a IoT device \( \sigma \) starts transmitting a data \( \alpha \) to MECS \( \Upsilon \) via UAV \( \bar{\Upsilon} \). Prior to transmitting \( \theta \), \( \sigma \) creates a signature \( \varepsilon \) from \( \alpha \) by employing the private key \( \psi^\alpha \) of \( \sigma \),
\[
\varepsilon = \Theta_{\psi^\alpha}(\psi(\alpha)) \mid \psi : \{0, 1\}^* \rightarrow \{0, 1\}^l,
\]
\[
\Theta : \{0, 1\}^* \rightarrow \{0, 1\}^l.
\]

After generating \( \varepsilon \), \( \sigma \) encrypts \( \alpha \) with the public key \( \psi^\sigma \) of \( \Psi \). Let \( \beta \) be the encrypted data,
\[
\beta = \xi_{\bar{\psi}^\Upsilon}(\varepsilon, \alpha, \bar{\psi}^\Upsilon).
\]

Subsequently, \( \sigma \) transmits \( \beta \) to \( \bar{\Upsilon} \). Let \( \tau_p \) is the total processing time for preparing data at \( \bar{\Upsilon} \),
\[
\tau_p^\sigma = \tau_\varepsilon + \tau_\beta.
\]

Here, \( \tau_\varepsilon \) and \( \tau_\beta \) is the processing time for \( \varepsilon \) and \( \beta \), respectively. Let \( T(\sigma \rightarrow \bar{\Upsilon}) \) be the time for the transmission of \( \alpha \) from \( \sigma \) to \( \bar{\Upsilon} \),
\[
T(\sigma \rightarrow \bar{\Upsilon}) = \tau_p^\sigma + \tau_d^\sigma(\sigma \rightarrow \bar{\Upsilon}),
\]
where \( \tau_d^\sigma(\sigma \rightarrow \bar{\Upsilon}) \) is the transmission delay from \( \sigma \) to \( \bar{\Upsilon} \). Upon receiving \( \beta \), \( \bar{\Upsilon} \) decrypts \( \beta \) by employing the private key \( \psi^\bar{\Upsilon} \) of \( \bar{\Upsilon} \), as shown in Algorithm 1. Let \( \bar{d} \) be the decrypted data,
\[
\bar{d} = \xi_{\phi^\bar{\Upsilon}}(\beta).
\]

After obtaining \( \bar{d} \), \( \bar{\Upsilon} \) first checks \( B_L \) and subsequently, it checks the validity of \( \bar{d} \) by utilizing \( \beta F \). Let \( \bar{f} \) is the validity result,
requests then after a threshold number of requests $\tilde{R}_n^T$, $\Upsilon$ puts $\tilde{d}_n^T$ in $B_L$, and the channel is blocked for a threshold amount of time $R_n^T$. However, for the valid $\tilde{d}_n^T$, $\Upsilon$ continues to process $\tilde{d}$. Before adding to storage, $\Upsilon$ checks the integrity of $\alpha$ by verifying the signature of $\sigma$. Let $\tilde{v}$ is verification result.

$$\tilde{v} = \omega_{\tilde{d}_n^T}(\alpha, \tilde{v}) \text{ where } \tilde{v} \in \{true, false\}.$$  \hspace{1cm} (16)

If $\tilde{v}$ is true, then $\Upsilon$ transmits $\alpha$ for validating and adding in the blockchain. Let $\tau_p^T$ is the total processing time for validating data at $\Upsilon$.

$$\tau_p^T = \tau_d^T + \tau_f^T + \tau_v^T,$$ \hspace{1cm} (17)

where $\tau_d^T$, $\tau_f^T$, and $\tau_v^T$ are the processing time for $\tilde{d}$, $\tilde{f}$, and $\tilde{v}$, respectively. Let $T(\sigma \rightarrow (\Upsilon \rightarrow \Upsilon))$ is the total time to transmit and process $\alpha$ successfully from $\sigma$ to $\Upsilon$ via $\Omega$ including the validation process. By adding (8) and (13), it can be written that,

$$T(\sigma \rightarrow (\Upsilon \rightarrow \Upsilon)) = T(\sigma \rightarrow \Upsilon) + T(\Upsilon \rightarrow \Upsilon).$$ \hspace{1cm} (18)

When $\alpha$ is valid, $\Upsilon$ starts to add $\alpha$ to the blockchain $B$. We considered a private blockchain network in which the identity of validators $\tilde{v}$ (those that create blocks) is known to everyone. In BUAV, each block holds a $T^T$ number of transactions, as illustrated in Fig. 2. In Fig. 2, the first block is the genesis block that contains the list of validators, chain id (i.e., network id), etc. After that, each blockchain contains the data of IoT devices. However, a validator proposes a block $\alpha$ and other validators validate the block. This process is performed in a round-robin algorithm. Let $f$ contains the validation result,

$$f = \sum_{z=1}^{\tilde{m}} d_{\tilde{V}_z}(\alpha),$$ \hspace{1cm} (19)

where $\tilde{m}$ is the total number of validators. Once, every validator validates the block, the block is appended in the chain.

### UAV Requisition

If $\sigma$ disconnects with $\Omega$ and cannot reconnect with $\Omega$, $\sigma$ transmits request $\tilde{f}$ to $\Upsilon$ to send information regarding new $\Omega$ in order to facilitate the transfer of data. There exists a direct communication link between IoTD and MECs. However, IoTD utilizes that channel only when IoTD cannot locate any UAV nearby. Moreover, transmitting data through that channel drains more energy than the communication channel that exists between IoT and UAV. Therefore, continuous transmission can decrease the lifetime of the battery of IoTD. In order to increase the lifetime, BUAV utilizes UAV for relaying data. Prior to sending $\tilde{f}$, $\sigma$ creates a signature $\tilde{e}$ from $\tilde{f}$ by employing the private key $\psi_{\tilde{e}}$ of $\sigma$,

$$\tilde{e} = \Theta_{\psi_{\tilde{e}}}(\psi(\tilde{v})) \text{ where } \psi : \{0, 1\}^* \rightarrow \{0, 1\}^l,$$ \hspace{1cm} (20)

where $\psi_{\tilde{e}}$ is the encrypted data, $\tilde{e}$ is the encrypted data, $\psi$ is the decryption by using $\psi_{\tilde{e}}$, as shown in Algorithm 3. Let $d$ is the decrypted data,

$$d = \psi_{\tilde{e}}(\tilde{e}).$$ \hspace{1cm} (21)

After obtaining $d$, $\Upsilon$ checks the validity of $\tilde{d}_n^T$ by utilizing $\beta^T$. Let $f$ is the validity result,

$$f = \sum_{j=1}^{\eta} \beta F_j(\tilde{d}_n^T) \text{ where } f \in \{0, 1\}.$$ \hspace{1cm} (23)

If $\tilde{f}$ returns a value of 1, then $\Upsilon$ continues to process the request. Prior to searching $\Omega$ for $\sigma$, $\Upsilon$ verifies the signature which is transmitted within the request. Let $\tilde{v}$ is verification result,

$$\tilde{v} = \omega_{\tilde{d}_n^T}(\tilde{r}, \tilde{e}) \text{ where } \tilde{v} \in \{true, false\}.$$ \hspace{1cm} (24)

If $\tilde{v}$ contains true then $\Upsilon$ retrieves the spatial information of $\sigma$. Subsequently, $\Upsilon$ retrieves all UAVs from the list of devices $\tilde{d}$. Let $\tilde{\Omega}$ is the collection of UAVs,

$$\tilde{\Omega} = \bigcup_{k=1}^{\eta} \{\tilde{d}_k\}.$$ \hspace{1cm} (25)

After retrieving $\tilde{\Omega}$, $\Upsilon$ calculates the geo distance between $\sigma$ and $\tilde{\Omega}$. Subsequently, $\Upsilon$ sorts the list in the ascending order based on the geo distance and picks the top threshold number $\tilde{R}_n^T$ of $\tilde{\Omega}$. Before transmitting the results, $\Upsilon$ first creates a signature $\tilde{\varepsilon}_T$ by employing $\psi_{\tilde{e}}^T$,
Algorithm 3 UAV request processing in MECS.

Input: Encrypted received request.
Output: List of the nearest UAV.

1: $\delta \leftarrow \zeta_\varphi^\perp (\beta)$. 
2: if $d_{\varphi'} \notin B_j$ then 
3: $f \leftarrow \bigcap_{j=1}^{J} \beta f_j(d_{\varphi'}).$
4: if $f = 1 \land \omega_{d_{\varphi'}}(d_j, d_z) = \text{true}$ then 
5: $\Omega \leftarrow \bigcup_{k=1}^{J} (\delta_k), \delta \leftarrow \varphi \leftarrow \text{retrieve} d_{\varphi'}.$
6: while $\delta \in \Omega$ do 
7: $\tilde{a} \leftarrow 0.5 - \frac{\cos((\tilde{u}_{\text{lat}} - \sigma_{\text{lat}}) \times \frac{\pi}{180})}{2} + \frac{\cos(\tilde{u}_{\text{lon}} - \sigma_{\text{lon}}) \times \cos(u_{\text{lat}} \times \frac{\pi}{180})}{2} - \frac{\cos((\tilde{u}_{\text{lon}} - \sigma_{\text{lon}}) \times \frac{\pi}{180})}{2}$ 
8: $\tilde{u}_{\text{dist}} \leftarrow 2 \times 6371 \times \arcsin(\sqrt{\tilde{a}}).$
9: end while 
10: $\Omega \leftarrow \text{sort}(\Omega_{\text{prop}} \leftarrow \text{dist}), \Omega_{gd} \leftarrow \bigcup_{i=1}^{W}(\tilde{\Omega}_i).$
11: $\varepsilon_\theta \leftarrow \Theta_{\psi_{\varphi}} (\psi(\Omega_{gd})), \gamma \leftarrow \zeta_{\varphi'} (\Omega_{gd}, \varepsilon_\theta).$
12: end if 
13: end if

$$
\varepsilon_\theta = \Theta_{\psi_{\varphi}} (\psi(\Omega_{gd})) \mid \psi : \{0, 1\}^* \rightarrow \{0, 1\}^t, \Theta : \{0, 1\}^* \rightarrow \{0, 1\}^t. 
$$

Here, $\Omega_{gd}$ is the list of selected $\Omega$, where $\Omega_{gd} = \bigcup_{i=1}^{W}(\tilde{\Omega}_i)$.

After generating $\varepsilon_\theta$, $\Upsilon$ encrypts $\Omega_{gd}$ by employing $\varphi_{\varphi}$. Let $\gamma$ is the encrypted data,

$$
\gamma = \zeta_{\varphi'} (\Omega_{gd}, \varepsilon_\theta). 
$$

After completing $\gamma$, $\Upsilon$ returns $\gamma$ to $\sigma$. Upon receiving $\gamma$, $\sigma$ first performs a decryption by using $\varphi_{\varphi}$,

$$
\hat{\delta} = \zeta_{\varphi'} (\gamma). 
$$

Subsequently, $\sigma$ verifies the integrity from $\varepsilon_\theta$. Let $\hat{\nu}$ is the result of verification,

$$
\hat{\nu} = \nu_{\tilde{T}}(\Omega_{gd}, \varepsilon_\theta) \text{ where } \hat{\nu} \in \{\text{true}, \text{false}\}. 
$$

If $\hat{\nu}$ returns a value of true then $\sigma$ reads the list, connects with a suitable $\Omega$, and continues to transmit data. Let $T(\sigma \rightarrow \varphi \rightarrow \Upsilon)$ is the total time for requisitioning a UAV and get response from $\Upsilon$.

$$
T(\sigma \rightarrow \varphi \rightarrow \Upsilon) = e_{\delta} + e_{\beta} + e_{\varphi(\sigma \rightarrow \gamma)} + e_{\delta_{\Upsilon}} + e_{\gamma} + e_{\Upsilon}. 
$$

IV. SECURITY ANALYSIS

A. Protection Against Eavesdroppers

In BUAV, data is transferred from $\sigma$ to $\Upsilon$ via $\Omega$ and in addition, $\sigma$ sends the request for $\Omega$ to $\Upsilon$. The area between $\sigma$ and $\Upsilon$, $\varphi$, and $\Omega$ are vulnerable to an eavesdropper $\theta$. However, BUAV provides protection against eavesdroppers. In BUAV, every $\sigma$, $\Upsilon$, and $\varphi$ have their own $\varphi_{\varphi}$, and a corresponding $\varphi_{\varphi}$. When $\sigma$ transmits $a$ to $\varphi$, $\sigma$ encrypts the information by utilizing (6). Upon receiving $\beta$, it is decrypted by $\Upsilon$ using (9) and the identity is verified using (10). After verification, $\Upsilon$ re-encrypts information using (11) and forwards $a$ to $\Upsilon$. While requesting the location of a new $\Upsilon$, $\sigma$ first encrypts the request using (21) and sends the request to $\Upsilon$. Upon receiving $\beta$, it is decrypted by $\Upsilon$ by using (22) and the identity is verified by employing (23). After processing the request, $\Upsilon$ encrypts the response using (27) and returns to $\sigma$. In order to read data between $\sigma$ and $\Upsilon$, $\theta$ requires knowledge of $\varphi_{\varphi}'$, which is only known to $\Upsilon$. Without $\varphi_{\varphi}'$, $\theta$ cannot decrypt the data. If $\theta$ intends to read $\gamma$ then $\varphi_{\varphi}'$ is needed which is only known to $\Upsilon$. In order to read $\beta$ and $\gamma$, $\theta$ needs to know $\varphi_{\varphi}'$ and $\varphi_{\varphi}$, respectively, which are not publicly available. As such, $\theta$ cannot exercise eavesdropping in BUAV.

B. Key-Spoof Resistance

An attacker $\theta$ cannot directly perform eavesdropping because $\varphi_{\varphi}$ is unknown to $\theta$. To determine $\varphi_{\varphi}$, $\theta$ must guess $\varphi_{\varphi}$. According to (1), $\varphi_{\varphi}$ is generated using $\delta_{\text{mac}}$, $\tau$, and $S_i$. $\tau$ and $S_i$ are totally random. Suppose, $\varphi_{\varphi}$ is 32 bytes or 256 bits long. To predict the correct $\varphi_{\varphi}$, $\theta$ has to go through the sequence of 256. For 256 bits, there are $2^{256}$ possible sequences and among them, only one can be the right $\varphi_{\varphi}$. The probability of predicting $\varphi_{\varphi}$ is $1/2^{256} = 2^{-256}$ and that is not practically feasible. Let, $\delta_{\text{mac}}$ be $i$ bits, $\tau$ represents $j$ bits, and $S_i$ represents $k$. If $\theta$ wants to guess the properties of $\varphi_{\varphi}$, then the probability of guessing correctness is $1/2^i \times 1/2^j \times 1/2^k = 2^{-i} \times 2^{-j} \times 2^{-k}$, which is also not practically feasible.

C. Data Tampering Resistance

In BUAV, $\sigma$ transfers $a$ to $\Upsilon$, via $\Omega$. When $\Upsilon$ receives $\gamma$ from $\Omega$, $\Upsilon$ first decrypts $\gamma$ by employing (14) and subsequently, verifies the identity using (15). However, $\Upsilon$ requires verification that data originates from the original sender, $\sigma$. $\Upsilon$ decrypts $\beta$ by employing (9) and retrieves $\alpha$. In this case, there is a vulnerability in that the data may experience alteration. Before transmitting $\alpha$, $\sigma$ creates a signature of $\alpha$ using (5). When $\Upsilon$ receives $\alpha$, $\Upsilon$ verifies $\alpha$ by employing (16). If $\hat{\nu}$ is $\text{false}$, then $a$ is altered and $\Upsilon$ discards $a$. $\Upsilon$ only appends data to blockchain when $\hat{\nu}$ is $\text{true}$. During the requisition of $\Omega$, $\sigma$ creates a signature by utilizing (20), so that, $\Upsilon$ can verify the authenticity of $\sigma$ using (24). Based on this approach, BUAV prevents altered data from being appended to Blockchain. After storing data in traditional databases, there is the possibility that the data may be altered. Considering this issue, BUAV stores the data in Blockchain. When data arrives at $\Upsilon$, after verification, $\Upsilon$ transmits the data to the blockchain network. Once the data is received in the network, validators create blocks after utilizing (19). Given that data is stored in the chain, every validator holds the same copy of the data. In blockchain, every block has a unique hash which is the identity of that block. The next block keeps the hash of the previous block and they are chained together. However, this hash is generated from the data, timestamp, nonce, etc. To main-
tains integrity, blockchain uses a Merkle tree. This is a structure in which the leaf nodes hold the hash of the data. When an alteration is performed to the data, the hash of the tree is also changed. Subsequently, the hash of the block is also changed and thus, the chain of breaks. To restore the chain, consent from the majority of the validator is required, which makes these changes technically unfeasible.

V. PERFORMANCE EVALUATION

This section represents the performance evaluation of BUAV. The performance is analyzed by simulation and experiment and the results are discussed in two subsections.

A. Simulation Results

A simulation was performed using MATLAB for estimating the effects in MEC server and another simulation was performed for estimating the effects in UAV. In UAV, Python was utilized for simulating results. The simulation parameters are provided in Table 2. For simulation purposes, an Intel(R) Core(TM) i5-4670 CPU @ 3.40 GHz was used as MEC server with 32 GB RAM. For UAV, Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4 GHz was used with 1 GB RAM.

Fig. 3 demonstrates the results of the simulation that were performed in the MEC server. The time for processing the data of &-hash bloom filter that contains information on the different number of devices is shown in Fig. 3(a). Milliseconds was considered as the unit of the processing time. In Fig. 3(a), processing time increases with the increase in the number of devices. In addition, the value of & affects the processing time. Increasing the number of & results in an increase in the processing time. As a result, & = 3 requires more time than & = 2, and & = 2 requires more time-consuming than & = 1.

Fig. 3(b) illustrates the expected transmission of data containing information for the different devices that are generated from &-hash bloom filter. An increase in the number of devices also results in an increase in the data size. However, & = 1, 2, and 3 produces the same amount of data because the data in the list contains either 0 or 1 and the size of 0 and 1 is same. Therefore, differences in & have no impact on the data size generated from &-hash bloom filter.

Fig. 3(c) depicts the transmission of data generated from bloom filter and other data structure apart from bloom filter containing the information of the different number of devices. A hash table was considered as a non-bloom filter data structure.

---

Table 2. Simulation parameters.

| Parameters                  | Values                                      |
|-----------------------------|---------------------------------------------|
| Devices, &                 | [500, ⋯, 10000], [1, 2, 3]                  |
| MEC CPU                    | Intel(R) Core(TM) i5-4670 CPU @ 3.40 GHz    |
| MECS STORAGE, RAM, OS      | 1 TB, 32 GB, Ubuntu 18.04.1                 |
| UAV CPU                    | Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4 GHz |
| UAV STORAGE, RAM, OS       | 16 GB, 1 GB, Raspbian Stretch               |

Table 3. Experimental parameters.

| Parameters                  | Values                                      |
|-----------------------------|---------------------------------------------|
| &                           | 1                                           |
| Wi-Fi, Bluetooth            | 48 Mbps, Bluetooth 4.2                      |
| ψ(·), Encryption            | SHA-256, RSA (key = 1024)                   |
| IoT D CPU                  | 4 Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4 GHz |
| IoT D STORAGE, RAM, OS      | 4 GB, 1 GB, Raspbian Stretch                |
| UAV CPU                    | 1 Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4 GHz |
| UAV STORAGE, RAM, OS        | 16 GB, 1 GB, Raspbian Stretch               |

Given that, there is no influence of & on data size, & = 1 was used for this simulation. The amount of data in both the bloom filter and the hash table increases with an increase in the number of devices. However, the data size of the hash table increases dramatically in comparison with the bloom filter. In particular, the hash table is initially twice of the data size of the bloom filter. For more devices, this difference approaches thrice the data size of the bloom filter. Therefore, by using a bloom filter, BUAV compresses space in a UAV.

Fig. 4 demonstrates the simulation results that were obtained for the UAV. The time for validating the identity of the different devices utilizing &-hash bloom filter is shown in Fig. 4(a). Milliseconds was considered as the unit for the validation time. With the increase in the number of devices that require validation, the validation time also increases. The change in the value of & leads to a change in the validation time. Increasing the number of & requires more time for the validation process. This is the reason why & = 3 requires more time than & = 2 and & = 2 requires more time than & = 1.

Fig. 4(b) illustrates the energy consumption during the validation process. Joule was considered as the unit of energy consumption. With the increase in the number of devices for the validation process, energy consumption also increases. Given that, the increase in & requires more time for the validation process, & = 3 consumes more energy than & = 2 and & = 2 consumes more energy than & = 1.

Fig. 4(c) represents the time required to validate the devices in the presence of malicious devices. This simulation was performed for 10,000 devices. The increase in the percentage of malicious devices, even in the 100 percentage malicious devices scenario, does not affect the validation time. This is because, for both valid and malicious devices, &-hash bloom filter checks either 0 or 1. However, the validation time increases with an increase in &; but, it remains the same for different percentages of malicious devices.

Fig. 5 demonstrates the rate of false positive for the different percentage of malicious devices. As bloom filter surrounds with the false positive issue, it is necessary to check the false positive rate. 10,000 devices were considered for this simulation. In this simulation, the false positive rate is 0 for & = 1, 2, and 3, even
MEC server with 32 GB RAM. Node.js was used to create the MEC server via UAV. The middleware in IoTDS was built using Python. The communication between UAV and IoTDS was performed over Bluetooth and the communication between UAV and MEC server was performed over Wi-Fi. Parrot Bebop 2 was used as a UAV and raspberry pi 3 model b+ was attached for maintaining communication with IoTDS and MEC server, as shown in Fig. 6(a). The middleware for UAV was written using Python. IoTDS were deployed randomly and UAV was placed in the center of IoTDS, as shown in Figs. 6(b) and 6(c). The proposed scheme was built over ethereum. In ethereum, a private network was created, named UAV-B, containing 10 validators. Geth was used as an ethereum client and web.js for RPC call. In order to set up 10 validators, 10 computers were used to create a local area network (LAN). Each pc contained geth and connected with each other. Proof of authority (PoA) was used for performing the consensus mechanism.

B. Experimental Results

The experiment was performed indoor, as shown in Fig. 6. The parameters used in the experiment are described in Table 3. An Intel(R) Core(TM) i5-4670 CPU @ 3.40 GHz was used as MEC server with 32 GB RAM. Node.js was used to create the server. 4 raspberry pi 3 model b+ were utilized and sensors (i.e., flame, temperature, humidity, light) were attached in these devices. These devices were considered as IoTDS with the primary task of sensing the environment and transmitting the result to the

Fig. 3. Results of simulation that was performed in MEC server: (a) Time for processing devices, (b) expected transmission of processed data, and (c) expected transmission of processed data for q-hash bloom filter and for without q-hash bloom filter.

Fig. 4. Results of simulation that was performed in UAV: (a) Time for validating devices, (b) energy consumption during the validation, and (c) time for validating devices in the presence of malicious devices.

Fig. 5. Rate of false detection for validating devices.

in the 100 percentage malicious devices scenario.

Fig. 7 shows the energy consumption for transferring data from a IoTDS to other entities (e.g., UAV, MEC server, and cloud) directly. Intel(R) Xeon(R) Processor E5-2697A V4 @ 2.60 GHz with 32 GB RAM was considered for the cloud server and the cloud server is hosted in US server at 69.162.66.34. CentOS 7.5 was used as the OS in the cloud server. In order to connect to the cloud, the IoT device is connected with the ipTIME A2004NS-R router and the EFM ipTIME A2004NS-R is connected to the internet. To connect to the MEC server, an ipTime N100 mini was used to open an access point from the MEC server. To connect to the UAV, Bluetooth 4.2 was used. Fig. 7 shows that it requires
more energy to communicate with the cloud, and with the increase in data size, energy consumption also increases. On the contrary, communication with the MEC server consumes less energy although the energy consumption increases with an increase in the size of the data. Communication with UAV consumes less energy than the MEC server which is almost half of the MEC server because communicating using Bluetooth consumes less energy than Wi-Fi. With an increase in data size, the energy consumption increases very slowly in contrast with the MEC and cloud server. From the results, direct communication with the cloud server, IoTs spent more energy than with the MEC server and direct communication with the MEC, IoTs spent more energy than with UAV.

Fig. 8 demonstrates the result of the experiments performed in the network. In Fig. 8(a), the throughput of the network over different time is presented. The communication channel of BUAV is compared before and after the application of security. Over time, in both channel’s throughput increases. However, the throughput of the channel without security is higher than the channel with security. This is because, after applying security, data has to be validated before being forwarded to the next hop.

Fig. 8(b) illustrates the total amount of data received in MEC server. For this experiment, 1 and 3 were considered as valid IoTs and 2 and 4 were considered as malicious IoTs. This simulation was performed for 10 s. Data from IoT-1 and IoT-3 successfully reached MEC server because both of them are valid. As a result, both successfully passed the validation process in UAV and 10,370 and 10,889 bytes received at MEC server, respectively. However, data from IoT-2 and IoT-4 were discarded in UAV because they could not pass the validation because they are not in the device list. Therefore, IoT-2 and IoT-4 cannot reach MEC server.

Fig. 8(c) shows the time to process security actions in IoT device, UAV, and MEC server for different data sizes. Equation (7) was utilized for IoT device, (12) was utilized for UAV, and (17) was utilized for the MEC server in order to calculate the processing time. The processing time increased with the increase of the data size for IoT device, UAV, and MEC server. However, MEC server’s computation power is much higher than that of UAV and IoT. That is why the processing time for MEC server is negligible in contrast with IoT device and UAV. However, UAV performs a decryption process which is slower than encryption. As such, the processing time is higher in UAV than IoT device.

Fig. 8(d) represents the energy consumption during the processing of security actions in IoT device, UAV, and MEC server for different data sizes. Energy consumption in the CPU was considered while processing the security actions of the MEC server. Given that, the CPU has significant computational power, the processing time is much less in comparison with that of UAV and IoT device. However, MEC server requires a large amount of energy in order to maintain continuity. With the increase in the data size, the energy consumption of the IoT device, UAV, and MEC server also increases. Given that, UAV has a decryption mechanism in security actions, it consumes more power than IoT device.

Fig. 8(e) depicts the processing time for performing individual security mechanism for different data sizes. Given that, MEC has significant computational power, the difference in the processing time between MEC and IoT/UAV is very high. It is notable that the processing time for every action increases with an increase in the data size. However, decryption requires more time than encryption and signing the data is more time consuming than verifying the data. In RSA, the private key is larger than the public key and due to its size, more time is needed for processing. Both decryption and signing use a private key in order to decrypt and sign, respectively. As such, decryption and signing require more time in contrast to encryption and verifying. However, this phenomenon is also visible in the MEC server. Despite its high computation capability, decryption and signing take more time than encryption and verifying.

Fig. 8(f) represents the latency for searching UAV that varies.
for the different number of UAVs. Equation (30) was utilized in order to calculate latency. However, with the increase in the number of UAVs, latency also increases. For more UAVs, it is required more time to calculate distance and more time to sort...
the data set which increases the latency. Overall, that increase in latency is very small.

Fig. 9 illustrates the result of the experiments performed in blockchain in MEC server. Fig. 9(a) shows the amount of data that was added in blockchain over time. 2, 5, 8, and 10 validator scenarios were considered during these experiments. With the increase in time, the amount of data addition also increased. However, these increases vary depending on the number of validators. The speed of adding data is almost the same for 2 and 5, and the speed of data addition is almost the same for 5 and 10. However, with the increase of the validator, the rate of addition of data in the blockchain decreases. In PoA, from the onset, everyone has the list of validators and blocks are appended in a round robin manner. Prior to the addition of the block in the chain, it is necessary to get an agreement by validators which increases the delay in the network and as a result of this delay, the amount of data also decreases.

Fig. 9(b) represents the changes of the data added in blockchain over time when considering the different percentage of validators that are not available for the validation process. 10 validators were considered while performing this experiment. The experiment was performed for the full active node (everyone is active), 10% down (10% of the total validators), 20% down (20% of the total validators), 30% down (30% of the total validators), and 40% down (40% of the total validators). However, in PoA, one validator proposes a block and the other validators sign it if the proposal is valid. This process proceeds one at a time in a round robin way. However, if a validator is not available for the validation process, he misses the chance to propose a block, his chance moves to the next validator, and this process goes on. But, this creates a delay in the network which causes a decrease in the amount of data added to the network in contrast with the full active node. With the increase in the percentage of unavailability of validators, the rate of addition of data in the network decreases.

Fig. 9(c) outlines the number of queries that can be performed at different times. In blockchain, everyone has the same version of the data and during data reading from blockchain, data were retrieved from the local copy of the requester. This is why the validators do not play a role during the reading of data from blockchain. However, many queries can be performed at different times were considered. With the increase in time, the queries also increase.

Fig. 9(d) describes the latency associated with the addition of data in blockchain for the different validators. From 2 to 10 validators appearance were considered. In Fig. 9(d), latency increases with the increasing number of validators. It is notable that, to sign a block, a delay is created from the validators. Thus, an increase in the number of validators increases latency.

Fig. 9(e) presents the delay in transferring data from IoT to MEC server. In these experiments, data were transmitted in two ways: (1) Transmit 1 (IoT→UAV→MECS), and (2) Transmit 2 (IoT→UAV→MECS). Transmission by including blockchain (IoT→UAV→MECS→Blockchain and IoT→UAV→MECS→Blockchain) and excluding blockchain were considered. For the case without blockchain, the delay in Transmit 2 is higher than Transmit 1. This is because in Transmit 2, the data has to pass through one more hop than in the case of Transmit 1. In addition, this data also experiences more security mechanisms than Transmit 1, which increases the delay in Transmit 2 in compare to Transmit 1. In comparison with the exclusion of blockchain, the incorporation of blockchain increases the delay in both Transmit 1 and Transmit 2. This is because the inclusion of blockchain introduces additional latency during the addition of the block in blockchain along with the security validation delay in MEC server.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, a secure data collection scheme was introduced in which data are collected from IoT devices (IoTDS) with the assistance of unmanned aerial vehicle (UAV). After collection, the data are stored securely in blockchain at mobile edge computing (MEC) server. While transferring data from IoTDs, encryption is performed using UAV’s public key and a signature is created using IoTD’s private key. UAV receives that data and after the decryption, the identity of IoTD is validated using 𝑔-hash bloom filter. Subsequently, data are forwarded to MECs and after validating, it is stored by MEC in blockchain that is signed by the validators. Simulations were performed using MATLAB in order to examine the impact of 𝑔-hash bloom filter for both in MEC server and UAV. Implementation of BUAV was done and experiments were performed in that implementation in order to test the feasibility. Some future directions are listed below:

• BUAV supports single UAV in the data acquisition which can be extended with multi-UAV to increase the quality of service. But, providing security for multi-UAV is very challenging which can be a future research topic.

• The dynamic distribution of IoT devices and mobility of UAV during secure data acquisition needs to be investigated which can be subjected to future works.

• BUAV does not consider the incentivization of the validator which is kept for future research.

REFERENCES

[1] P. Asghari, A. M. Rahmani, and H. H. S. Javadi, “Internet of things applications: A systematic review,” Computer Netw., vol. 148, pp. 241–261, 2019.

[2] J. Portilla, G. Mujica, J. 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, 2019.

[3] 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.

[4] 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.

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

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

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

[8] P. Verma and S. K. Sood, “Fog-assisted IoT enabled patient health monitoring in smart homes,” IEEE Internet Things J., vol. 5, no. 3, pp. 1789–1796, June 2018.
