Secure Communication for Multiquadrotor Networks Using Ethereum Blockchain

Pramod Abichandani, *Member, IEEE*, Deepan Lobo, Smit Kabrawala, and William McIntyre

***Abstract*—Ethereum blockchain is a powerful, open-source technology for creating decentralized and secure information sharing systems. The main contribution of this article is the experimental validation of an Ethereum blockchain-based software and hardware architecture that enables secure communication for multiple small unmanned aerial vehicles (sUAVs). The experiments involved three DJI M100 quadrotors that shared images captured during flight based on smart contracts created using Ethereum's Turing complete programming language. The smart contract was designed so that only the intended recipient sUAV could access a specific image. The effect of image size, difficulty level, and consensus algorithms on image transfer times during flight are noted and point to the feasibility of this system in practical missions. The effects of wireless network disruptions on the Ethereum network are documented. The fully documented smart contract code is open sourced to assist readers in quick prototyping. As efforts for decentralization and security of multirobot systems continue to grow, the system architecture and implementation detailed here may serve as a guide for future research.**

***Index Terms*—Cyber–physical systems, secure communications, testbed and trials.**

Fig. 1. Three DJI M100 drones were fitted with the required hardware for a three-node Ethereum network.

I. Introduction

Multirotor small unmanned aerial vehicles (sUAVs) have emerged as an aerial platform of choice in commercial, research, and defense markets due to their distinct advantages. These advantages include their ability to perform vertical take-off and landing (VTOL), hover at a spot, and yaw at a zero-turn radius. Several market studies provide insight into the $50B+ (and growing) global sUAV market size [1]–[5]. This growth in market adoption of sUAVs is primarily due to their affordable cost and strict government regulations across most nations on the use of large UAVs for nonmilitary applications. Multirotor sUAVs are used for several tasks, such as agricultural yield monitoring, land surveying, photography, air quality assessment, search and rescue operations, formation control, target tracking, payload transportation, and military operations [6]–[10]. Additional uses of these sUAVs are found in meteorological and atmospheric studies [11]–[15] and inflow mapping [16]–[18]. This article reports on experiments involving a network of multiple quadrotors, as depicted in Fig. 1, that use Ethereum blockchain for securely sharing image data during outdoor flight missions.

Blockchain is an open source, distributed, immutable ledger that maintains a record of every information transaction among peers in a network. Every peer in the network has a copy of this ledger. It enables transactions between peers without an intermediate trusted central authority and verifies the transactions with the same amount of certainty as a central authority. The trustlessness provided is a key benefit of blockchain technology—it eliminates the need for an intermediary to govern and verify interactions on the network. Blockchain technology has been applied extensively in multiple industries, including finance, healthcare, energy, agriculture, smart cities, real estate, and vehicular networks [19]–[35]. A slew of recent applications illustrates the growing ubiquity of blockchain-based software systems for artificial intelligence, 5G, Internet-of-Things (IoT) security, identifying deepfake, and package delivery [36]–[40].

An important aspect of blockchain technology is the implementation of smart contracts. A smart contract is an executable software program that governs the transactions between peers in the network. Rules for the network can be programmed into smart contracts to automate sophisticated information-centric tasks. The introduction of Ethereum Blockchain in 2013 provided the ability to program smart contracts in a Turing complete language called Solidity. Carefully developed smart contracts enable autonomous systems comprised of multiple agents (peers) that can perform decentralized decision making

Manuscript received May 18, 2020; revised July 25, 2020; accepted August 4, 2020. Date of publication August 10, 2020; date of current version January 22, 2021. *(Corresponding author: Pramod Abichandani.)*

Pramod Abichandani, Deepan Lobo, and William McIntyre are with the Electrical and Computer Engineering Department, New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: pva23@njit.edu).

Smit Kabrawala is with the Department of Computer Science, New Jersey Institute of Technology, Newark, NJ 07102 USA.

Digital Object Identifier 10.1109/JIOT.2020.3015716

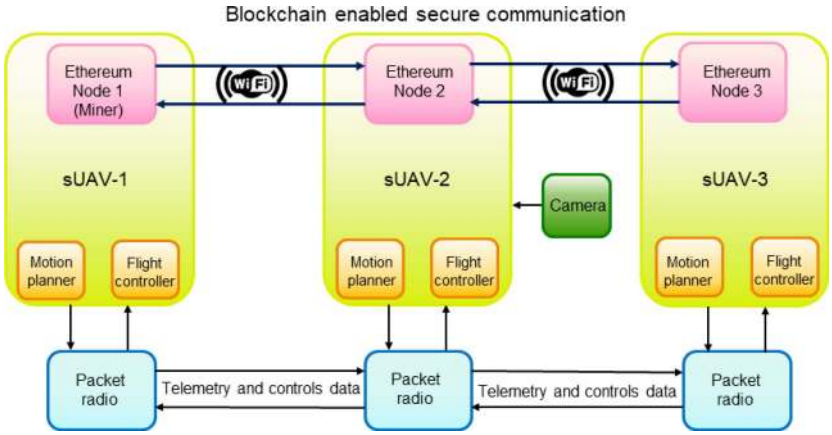

Fig. 2. Each node included the flight control hardware and a separate computer running an Ethereum blockchain node. Flight control data was communicated via a 915-MHz mesh network. Ethereum blockchain data was communicated via WiFi.

in applications, such as IOT, multirobot systems, smart cities, and artificial intelligence [36], [41].

In a multirobot system, each robot performs a specific task as the group works together toward completing a larger goal. Multirobot systems have been used in precision farming, creating interactive displays, IOT networks, and industrial robotics [42]–[45]. Blockchain can provide security for trust-sensitive multirobot systems in the form of data confidentiality, integrity, entity authentication, and nonrepudiation [46]. The distributed ledger of a blockchain ensures that decision making is distributed and collaborative missions can be easily programmed while eliminating a single point of failure for data storage.

This article elucidates the hardware and software architecture; the quadrotor, computing, and radio hardware used; and the software development tools used to implement an Ethereum-based secure communication system.

The main contributions of this work are given as follows.

1) Experimental validation of an Ethereum blockchain system for secure data transfer between three DJI Matrice M100 quadrotors during flight. Fig. 2 depicts the conceptual architecture of this system.
2) A software architecture that seamlessly integrates the Ethereum software stack with a quadrotor flight control stack.
3) Results of four key experimental studies:
   a) the effect of varying blockchain difficulty on inflight data transfer rates between three quadrotor sUAVs;
   b) the effect of varying data size on inflight data transfer rates between three quadrotor sUAVs;
   c) the effect of Ethereum network disruption on the communication between three quadrotor sUAVs while inflight;
   d) the effect of increased payload (Ethereum network hardware) on the flight-times of the quadrotor sUAV.
4) To further assist the development of such systems, the smart contract developed in this work has been open sourced and shared for use by the community [47].

The remainder of this article is as follows. Section II discusses existing literature with Ethereum blockchain applied to robotic systems. In Section III, the hardware and software architecture used in the experimentation is described. Section IV provides in-depth results of the experiments. Section V provides discussions about the results and future directions.

II. Related Work

The body of work that applies blockchain technology in robotics applications continues to grow [46], [48]–[58]. A majority of these studies have covered blockchain-based robotic system design and associated simulations. A common theme in these studies is improved security, behavior differentiation, and data integrity provided by using the blockchain technology to enhance robotic operations [46].

A. Simulation Studies

The intersection of blockchain technologies and robotics has been the focus of several simulation studies [48]–[51], [55]–[58]. Kapitonov *et al.* [48] simulated a communication protocol for multiagent systems to participate in business activities using Ethereum blockchain and smart contracts. The communication protocol consisted of smart contracts interacting with autonomous agents running the robot operating system (ROS). The verification of transactions took place using Air-token and Ether [59]. The authors used the proposed communication protocol for an sUAV employee management system that consisted of a dispatcher node and an air traffic control (ATC) node.

Strobel *et al.* [49] simulated an approach to establish secure swarm coordination mechanisms for a group of robots and exclude Byzantine members using smart contracts deployed on the Ethereum blockchain. The proposed approach was simulated in autonomous robots go swarming (ARGoS) robot swarm simulator interfacing with a Geth Ethereum client [60]. Smart contracts were used to register the robot members, define movement strategy, and select a decision making strategy using a voting system to create a consensus among

the members. The smart contract implemented strategies that included a member's time limit to cast a vote, a renewal of the decision strategy each time a new strategy was selected, and check on different blockchain versions by verifying the hash value of the blocks. The performance of the blockchain approach proved superior over classical swarming approaches [61].

Youssef *et al.* [55] proposed a cloud system architecture for an sUAV and sensors designed for surveillance of a dam site. The sUAV collected these data and delivered them to the dam monitoring center. The authors used the Bitcoin framework and the Proof-of-Work (PoW) consensus protocol to provide data integrity, traceability of the sensor data in the wireless sensor network. The authors simulated several scenarios to demonstrate the time delays for secure sensor data.

Aggarwal *et al.* [56] proposed a communication model based on the Ethereum blockchain for the Internet of Drones. Ethereum blockchain provided authorization, data integrity, and authentication for the data collected by the drones in the system model via Proof of Stake (PoS). A single drone was selected to be the forger node responsible for authenticating and validating the blocks in the blockchain. The computation cost for block creations was calculated as 384 b, of which 128 b represented the identity, and 256 b is the hash value. The computation time for block creation and validation was found to be 0.023 ms.

Barka *et al.* [57] developed a Bitcoin blockchain-based sUAS for monitoring and surveillance of critical infrastructures. Blockchain provided security against three types of breaches: 1) adversaries targeting software; 2) adversaries targeting hardware; and 3) adversaries targeting communications. The authors provided a performance evaluation of the proposed system using the NS-3 simulator over an area of 16 km^2 with four ground control stations. The authors programmed the occurrence of ten events around five randomly distributed critical infrastructure. Detection rates in the order of 95% were observed in simulation with lower false alarm rates as compared to classical approaches [62], [63]. The authors concluded that the high detection ratios and reduced overhead showed that a blockchain-based sUAS could provide accurate decisions which are crucial in monitoring critical infrastructure.

Kang *et al.* [51] proposed a reputation-based data sharing scheme for secure data sharing among vehicles to overcome the security vulnerabilities of vehicular edge computing servers. The scheme was based on a consortium blockchain where the nodes were preselected in a public blockchain to establish a shared and distributed database with smart contracts used to enable data management automation between the vehicles. Security analysis of the proposed scheme showed that the security of the data storage and high-quality data sharing over traditional reputation schemes.

Afanasev *et al.* [50] proposed an Ethereum blockchain-based decentralized network to monitor, control, and log workflow events for a cyber–physical production (CPP) system. The machines in the proposed CPP system communicated by exchanging cryptocurrency tokens. An essential feature in the system was that the cryptocurrency tokens would be distributed between the machines well in advance, according to the priority of services, thus creating a consensus where Nothing-is-at-Stake automatically. The authors proposed implementing their system for a PCB manufacturing plant by determining the rules of operation.

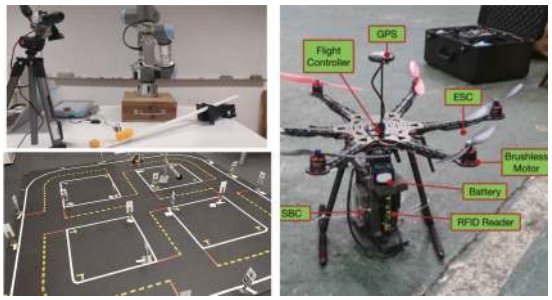

Fig. 3. Prior experimental studies with blockchain and robotics involve use of robotic arms (top-left) [54], ground robots [65] (bottom-left), and single sUAV with RFIDs (right) [66].

In [58], the application of a multirobot path-planning algorithm using the blockchain technology was investigated. A probabilistic road map (PRM), path-planning algorithm, was implemented alongside Hyperledger Fabric, an enterprise-grade blockchain platform [64]. The average latency to commit a transaction to the Hyperledger Fabric was 112.46 ms, thus validating that a blockchain platform has the potential for enabling secure and scalable distributed systems.

B. Experimental Studies

In contrast to the numerous simulation studies published in the literature, only a handful of experimental studies involving blockchain technology for robotics applications have been published thus far [54], [65], [66]. The key to these studies is the integration of a blockchain software stack with a robotic system, such as an industrial robotic manipulator arm, ground robot platforms, and quadrotors.

Network latency is a crucial metric to be tracked in experimental studies as it directly affects the data refresh rates of a robotic system. Fig. 3 depicts the experimental platforms used in representative experimental works.

Lopes *et al.* [54] proposed an experimental architecture to control a UR3 robotic arm where the robotic events are logged on the RobotChain blockchain and its movement controlled with a smart contract program [53]. Logic programmed in the smart contract sent instructions to a robotic arm to pick and place objects based on data processing of images obtained using an RGB camera by an external computer. The arm speed in picking up the objects was measured to validate that the proposed blockchain-based architecture was capable of controlling the robotic arm in real-time.

Danilov *et al.* [65] provided experimental validation procedures to identify flawed liability executions in order to suspend payments to questionable service providers using blockchain in a Duckietown environment [67]. The consensus protocol on blockchain technologies provides a method to detect malfunctioned agents in a network when agent behavior goes against the behavior consensus defined by the members in the network.

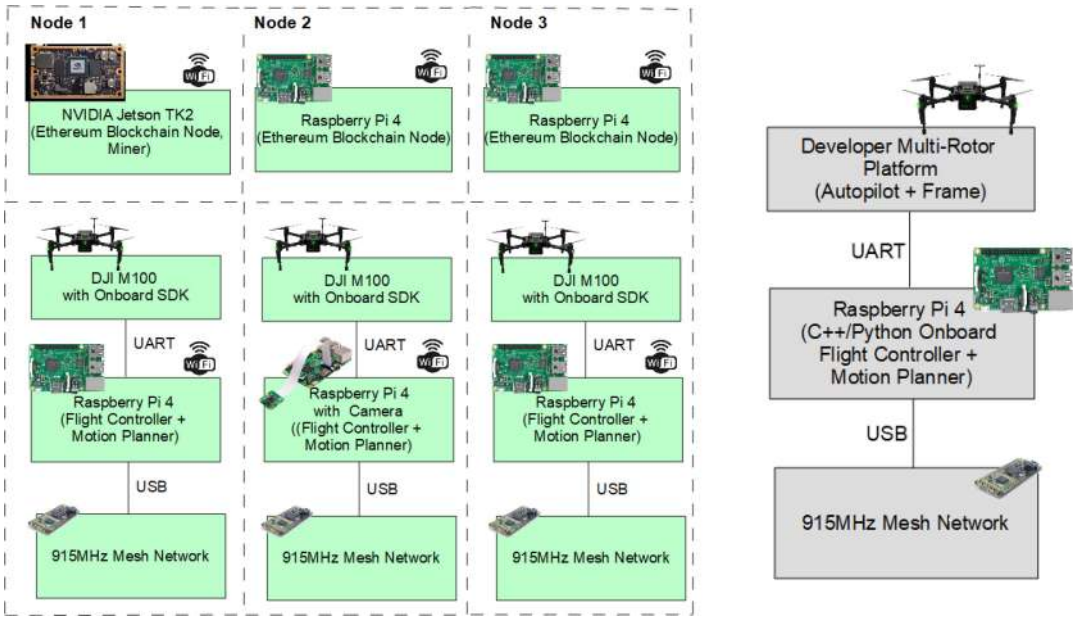

Fig. 4. Left: Hardware architecture for each node including the flight hardware and hardware running the Ethereum blockchain. Right: The DJI M100 quadrotor platform featured a Raspberry Pi 4 computer for controls and motion planning and a 915-MHz mesh radio for networking with other quadrotors. The mesh network was used to communicate position, velocity, acceleration, and other relevant controls data.

The blockchain platform used in this work was Ethereum. A prototype of the proposed system was implemented where smart contracts validated the liability of the autonomous agents and directed their movements.

Fernández-Caramés *et al.* [66] documented experiments for an automated inventory management system using RFID, a single sUAV, and Ethereum blockchain technology. The authors evaluated two consensus protocols for the Ethereum blockchain, Proof of Authority (PoA), and PoW. A quadrotor sUAV was manually controlled to fly around the warehouse and tag the inventory items using the RFID tags. The time taken to identify the inventory items using a quadrotor and store their identification value on the database was measured. It was shown that the time taken to validate blocks averaged at around 15 s for PoA while for PoW, the validation times varied significantly from 5 to 70 s.

These experimental works have paved the way for more sophisticated robotics studies involving blockchain technology. At the same time, there still exists a gap in the literature about the hardware and software architecture required to effectively implement such robotics systems as well as the evaluation of such systems in real-world field operations.

This article addresses the gap and expands on the above body of work by focusing on the physical implementation of popular blockchain technology—Ethereum—for multiple quadrotor sUAVs operating outdoors.

III. System Architecture

This section elucidates the experimental platform developed for the Ethereum-based secure exchange of image files.

A. Quadrotor Setup for Motion Planning

Three DJI Matrice M100 quadrotors, referred to as Nodes 1–3, were used [68]. Each quadrotor setup hosted a communication node in the Ethereum blockchain network. Fig. 4 depicts the quadrotor hardware setup.

The quadrotors came equipped with a proprietary DJI M100 N1 flight controller with inertial sensors and GPS. The N1 flight computer was responsible for position control using velocity and yaw-rate inputs. Each quadrotor provided a real-time telemetry stream of accelerometer, gyroscope, magnetometer, and GPS data. The quadrotor position was defined in a local frame using latitude, longitude, and height with a predefined origin point. For motion planning and control, each quadrotor Node was equipped with a Raspberry Pi 4 computer (Quad-core Cortex-A72 (ARM v8) 64-b SoC @ 1.5 GHz, 4-GB LPDDR4-3200 SDRAM). Motion commands and telemetry streams were communicated between the Raspberry Pi and DJI Flight stack using a UART connection using software function calls provided by the DJI's onboard C++ SDK. The flight processing Raspberry Pi 4 for Node 2 was equipped with a V2 camera module for image capture with 8 Megapixel still image resolution.

Each quadrotor was equipped with a 915-MHz transceiver radio module used for half duplex, bidirectional mesh networking between quadrotors for motion data telemetry using RF links at 300 kb/s. A round-robin scheduling technique for sequential wireless communication was implemented for the fair distribution of wireless bandwidth and fault tolerance. As part of this technique, only one quadrotor transmitted its information at a time. This information payload contained position and velocity data and was received by the remaining quadrotors either through one-hop direct communication or through indirect multihop relaying. The order in which the quadrotors communicated in the round-robin was randomly determined at the beginning of the mission.

The positioning of the quadrotor was defined in a frame with a predefined GPS origin coordinate O consisting of a latitude (radian), longitude (radian), and height from the

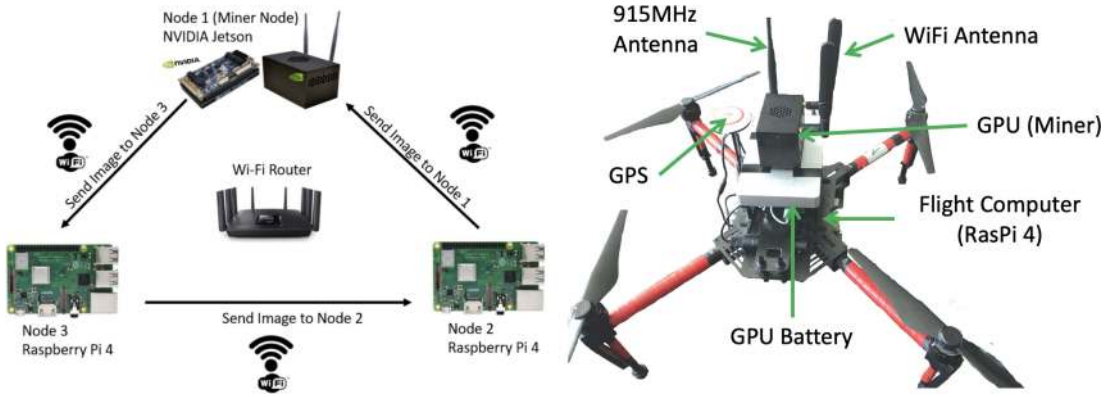

Fig. 5. Left: Three Ethereum Nodes were setup using 1 NVIDIA Jetson TX2 and two Raspberry Pi 4 computers. The nodes ran an EVM and communicated with each other using an 802.11 network setup using WiFi routers. Right: Ethereum computing hardware mounted on DJI Matrice 100. This picture depicts the miner node mounted on the quadrotor.

ground (meters). A quadrotor's motion during the experiments was broken into three components: 1) take-off; 2) offset alignment; and 3) waypoint following.

The first stage for each quadrotor was taking off to a height of 1 m. The quadrotor would then move to the predefined origin coordinate O and then to its defined offset position, which included a height of 5 m for all experiments. Once all three quadrotors were at their assigned offsets and communicated with each other through the transceiver radio modules, the waypoint following stage would begin in which all three quadrotors would proceed to the assigned waypoint, while maintaining their offsets. A set of custom spline generation functions, implemented in Python, were used to generate minimum snap 7th-order splines passing through N preselected waypoints [10]. A ground station computer hosted a Web-based user interface for mission configuration, clock synchronization, and mission upload.

B. Ethereum Hardware Setup

The Ethereum network was implemented using a separate set of computers that were mounted on the quadrotors and is depicted in Fig. 5. Keeping the blockchain computing hardware separate from the flight control hardware ensured modularity and avoided taxing any single computing system.

1) Miner Node: The transition of the Ethereum blockchain state takes place when a block is deemed valid. The validity of a block is determined through the process of mining. Node 1 was equipped with an NVIDIA Jetson TX2 computer (Quad-Core ARM Cortex-A57, 256-core NVIDIA Pascal GPU, 32-GB eMMC 5.1, 8 GB 128-b LPDDR4 Memory). The Jetson computer was selected as a miner due to its appropriate computational resources for validating and adding new blocks to the Ethereum. As depicted in Fig. 5, the Jetson was mounted on an Orbitty carrier board, and the assembly was mounted in a black colored OrbittyBox [69]. The miner node was also designated as the bootstrap node and was responsible for forming the overlay network with the nodes. The remaining nodes connected with each other over the TCP port of the bootstrap node.

2) Other Ethereum Nodes: Node 2 was equipped with a Raspberry Pi 4 computer (ARM Cortex-A72, 1.5 GHz, 4-GB LPDDR4 Memory) and acted as a node in the Ethereum network. A separate process on the flight processing Raspberry Pi 4 used openCV for image capture. The captured images were transferred to the corresponding Ethereum blockchain node 2 Raspberry Pi 4 using a Python-based server–client WebSocket connection. Nodes 3 was also equipped with a Raspberry Pi 4 computer and acted as a node in the Ethereum network.

3) Radio Hardware: The Ethereum network was established with the help of 2.4-GHz WiFi routers. Given that the Ethereum node hardware was hosted on a set of sUAVs, it was prudent to evaluate options that provided network connectivity with respect to their mobility. Accordingly, three different WiFi routers were evaluated. Each experiment used a specific type of WiFi router. The routers were: a Verizon 4G LTE WiFi mobile hotspot, an AC1750 wireless dual-band gigabit router from TP-Link, and an EA9500 Max-Stream AC5400 MU-MIMO Gigabit WiFi Router from Linksys [70]–[72]. Table I provides the ranges and bandwidth of these WiFi routers.

The Verizon hotspot hardware provided a most mobile option out of the three, as it could be mounted on the quadrotor itself. However, the bandwidth provided was relatively lower than the other two options. On the other hand, the Linksys router provided the strongest WiFi connection outdoors due to its long-range MIMO configuration.

C. Consensus Algorithm

Consensus algorithms provide blockchains with its characteristic features of decentralization—trustless security, immutability, privacy, and transparency. The PoW consensus algorithm is widely used in Bitcoin and Ethereum [73]. Other consensus algorithms include PoS [74], PoA [75], and practical Byzantine fault tolerance (PBFT) [76]. In our experiments, we evaluate the effect of PoW or PoA algorithms on time taken to transfer images through the Ethereum network. The PoW algorithm is based on the SHA-256 cryptographic hash function [77]. The PoA algorithm is implemented in Ethereum through a protocol called Clique [75]. An issue with PoW is

```
{
        "config": {
                "constantinopleBlock": 0,
                "petersburgBlock": 0
        },
        "difficulty": "0x10000",
        "gasLimit": "0x8000000",
        "alloc": {
                "0x8FA8c9E58BcF0201f927b175363365ee71F5d621": {"balance":"1000000000000000000"},
                "0xE168fa9d240c552257C0860AcC60684a3c347866": {"balance":"1000000000000000000"},
                "0xa081657c7a35399e5D24E08a90C1f631834b429C": {"balance":"1000000000000000000"}
                        }
}
```


Fig. 6. Example Genesis file with Ethereum blockchain parameters.

TABLE I
NOMINAL RANGE AND BANDWIDTH FOR RADIOS USED IN THE EXPERIMENTS

Radio Network	Nominal Range	Bandwidth
Verizon mobile hotspot	20meters	12 Mbps
Tp-Link AC1750 Wireless Dual Band Gigabit Router	33meters	1300 Mbps
Linksys EA9500 Max-Stream AC5400 MU-MIMO Gigabit WiFi Router	45meters	2166 Mbps

that it is impossible to control the mining frequency and control the block times. On the other hand, PoA block validation times are consistent and do not vary significantly.

D. Ethereum Software Setup

A private blockchain was set up between the three nodes (Nvidia Jetson and two Raspberry Pi computers) in the network with individual read and write permissions given to each node. Each node ran an Ethereum virtual machine (EVM) [78]. The EVM is the Turing complete virtual machine that processes and handles transactions being carried out in the Ethereum blockchain. Since this was a private blockchain, the identity of each Ethereum node was known and verifiable.

1) Geth Ethereum Client: The Geth Ethereum client was used in our software development. Geth is written in the Go language. There are two types of accounts in a blockchain: 1) externally owned accounts (EOAs) and 2) contract accounts (smart contracts). Using the Geth console, an EOA account was created for each of the Nodes 1–3. An Ethereum account is defined by a pair of keys, private key, and a public key, and each account is indexed by its address, which is derived from the first 20 B of the SHA3 hashed public key [79].

2) Truffle Development Environment: The Truffle development environment was used to develop, verify, and deploy our smart contracts. Truffle is a command-line interpreter (CLI) tool with a built-in compiler for smart contracts, automatic contract testing, scriptable, extensible deployment, and migrations framework [80]. The Truffle environment provided methods to interact with the deployed smart contracts. An automation script written in Javascript called instances of the functions in the smart contract. Each node featured its automation script to interact with the smart contract. This approach was adopted to maintain the secure communication protocol and ensure that each node calls only the functions in the smart contract meant for it. The smart contract was coded in Solidity, which is an object-oriented programming language [81]. After compiling the smart contract code, it is translated to bytecode, which is executable in the EVM.

E. Ethereum Smart Contract

1) Genesis Block: The first block in the Ethereum blockchain is called the Genesis block. The Genesis block contains parameters that define the blockchain. The Genesis file used in our experiments was written in JSON and is depicted in Fig. 6. The blockchain parameters initialized in the genesis file include the following.

1) `Difficulty`: This parameter is a measure of the computational complexity of mining a block to find the hash value. It determines the speed of mining of the blockchain network and can be used to calculate the deviation from the expected block time [82].
2) `Gas Limit`: The maximum amount of `gas` that the sender is willing to spend for a particular transaction. Gas is a unit used to measure the effort required for a particular computation in an EVM. Gas price is the value the transaction sender is willing to pay per `gas` unit and is measured in *GWei*. Ether is the token used to pay for `gas`.
3) `Alloc`: The alloc parameter is only used in the case of a private blockchain. This parameter is used to allocate funds (Ether) to the respective accounts in the network. The allocated fund `balance` should be significantly higher than the `gas` limit such that the accounts do not run out of `gas` when executing transactions. The allocated funds do not have any value outside of the private blockchain.

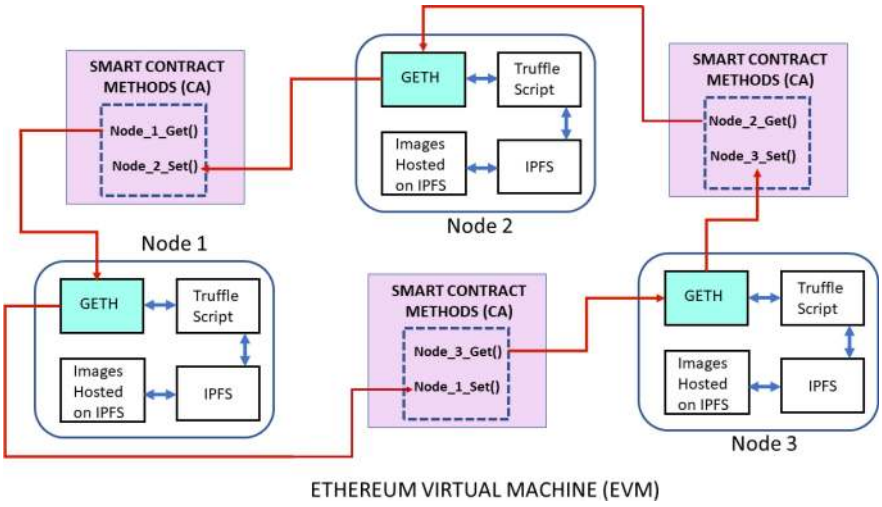

Fig. 7. Ethereum software setup used in our experiments. The smart contracts featured secure functions to ensure images can only be sent and received by predetermined nodes. In this case, the image transfer across the entire network happened from Node 2 to Node 1, from Node 1 to Node 3, and finally from Node 3 back to Node 2.

Nodes with the same Genesis file are connected to the same private blockchain network to maintain proper synchronization of the blocks. For these experiments, four genesis files were created with increasing orders of difficulty.

The block difficulty values were set to the following values.

1) *Difficulty 1:* 0x1.
2) *Difficulty 2:* 0x10000.
3) *Difficulty 3:* 0x100000.
4) *Difficulty 4:* 0x1000000.

The `alloc` and `gasLimit` parameters of the genesis file were left unchanged. Each run of the experiment was conducted using one of the genesis files copied in each of the nodes. Once the nodes were connected to the same private network and synced together, Node 1 then deployed the smart contracts to the private Ethereum blockchain.

In a private blockchain, it is up to the developer/administrator of the system to decide the gas price for each unit *Wei*. A private Ethereum network is not part of the public blockchain, so the gas prices are not affected by the publicly traded value of Ether. In the system documented here, the gas price was set to 0 when setting up the `Geth` environment, which means that the nodes were able to post transactions while the miner received 0 as payment [83].

Additionally, in a private network, it is assumed that the participant node identities are known, and therefore, the Ethereum addresses are added to an access control list at the beginning of the smart contract code. The nodes in the private Ethereum network access the read and write functions of the smart contract based on their Ethereum address definition in this access control list. The smart contract generates warning events in the case that an unlisted Ethereum address made an attempt to interact with the smart contract. A similar approach to providing trust management between the nodes in the network using an access control list to ensure trust between the nodes in the network can be found in [84]. A limitation in the approach, however, is the inability to dynamically update the access control list due to the permanent nature of the smart contract.

2) Smart Contract Functions: The smart contracts used in our experiments were written in Solidity Version 0.6.7. The contracts allowed for the secure transmission of images between different nodes in a manner that ensured that only the intended recipient received a specific image.

Fig. 7 depicts the smart contract and decentralized storage functionalities that were encapsulated in `Get()` and `Set()` functions. The decentralized storage was implemented using interplanetary file system (IPFS). The `Get()` functions were used to read an image hash, while the `Set()` functions were used to store the image hash on the blockchain network. The Truffle scripts compiled and deployed the smart contract into the blockchain environment. Interactions with the smart contract on the Ethereum took place through transactions verified by the miner.

Fig. 7 depicts the flow of data through the Ethereum network. Information about each node's user account was stored in the smart contract. A node could only access functions designated to it depending on its node number in the network.

Node 2 was the first node to receive a new image captured by the camera. This image was stored in a local folder on Node 2 and would be uploaded to the IPFS using a bash script. Once an image was uploaded to the IPFS setup, an image hash would be generated. Node 2 would use the `Node_2_Set()` function to store the IPFS hash in the smart contract. Node 1 can then use the `Node_1_Get()` functions to read the IPFS hash stored in the smart contract. This process was repeated for all three nodes. The hash obtained could then be used to access the image on the IPFS server. Important debugging messages were printed to the console for the developers and users to track the transfer of the image hash from one node to another.

A JavaScript program was used to automate the task of interacting with the smart contract using the Truffle environment. The JavaScript on each Node was executed when the bash script to upload images to the IPFS was executed. The

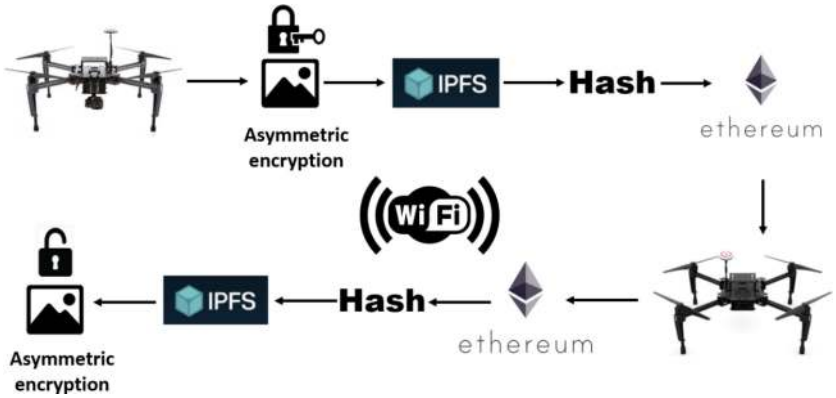

Fig. 8. Data flow in our experiments involved image capture and encryption, IPFS image upload, transfer of the image hash over the Ethereum blockchain followed by decryption of the image.

smart contract code developed for the system can be found on the RADLab Github repository [47].

F. File Storage

A limitation of blockchain technology is that it is computationally expensive to store data on it [77], [85]. The IPFS, a peer-to-peer distributed file-sharing system, was used for storing the images [86]. IPFS by itself provides a tamper-proof method of storing and sharing data between nodes in a network. However, it does not provide any method to timestamp when the data is added to the network, which is vital information when performing missions using multirobot swarms. Another limitation of IPFS is that anyone with a copy of the root content identifier (CID) has access to the data, which is problematic when sharing sensitive files between sUAVs.

By integrating Ethereum blockchain with IPFS along with a layer of encryption to the data, it is possible to build a secure information sharing system that overcomes these drawbacks. Fig. 8 depicts the detailed steps of the data flow involving encryption, IPFS, and Ethereum. After an image is captured, the node applies asymmetric encryption protocol OpenPGP, which allows encrypting a file with a public key that can be decrypted by only those members of the private blockchain that hold the decryption key [87]. After encrypting the image file, it is added to the IPFS, which generates a CID hash, which is stored on the blockchain via the smart contract. On the receiver end, the smart contract ensures that the quadrotor node requesting the CID hash is part of the private blockchain. This is ensured by verifying the private key of the EOA requesting the data. After verification, the hash obtained is used to access the data from the IPFS node via a local HTTP gateway. The data downloaded from the IPFS is the encrypted image file. All members of the private blockchain hold the decryption key, which is used to decrypt the downloaded image.

IV. Experimental Results

The experiments focus on studying the effects of image size, consensus algorithm type, and blockchain difficulty values on the time τ_{image} taken to transfer images across the multi-sUAV network successfully. In the following, we document image transfer flight tests that were conducted for 3 min each. The quadrotor 2 captures images using its camera and transfers it to the Ethereum Node 2 that is physically mounted on it. The smart contracts are then deployed to transfer these images across the entire network (Node 2 to Node 1, Node 1 to Node 3, and finally from Node 3 back to Node 2). τ_{image} is the total time taken for this image transfer across the entire Ethereum network.

As mentioned in Section III-B3, three wireless network routers were used to create three different sets of experiments. For all image transfer experiments, two consensus algorithms were tested—Ethash (PoW) and Clique (PoA).

Figs. 9 and 10 depict the results of our experiments. In each of these graphs, τ_{image} is averaged across multiple runs of the experiments and expressed in seconds.

A. Effect of Varying Image Size on τ_{Image}

The images from the Node 2 quadrotor camera were resampled to different sizes (100, 500, 750, 1000, 2000, and 3000 kB) before being transferred over the Ethereum network. The images were sent in increasing order of their resampled sizes. Fig. 9 depicts the τ_{image} values in seconds averaged across all experimental runs and difficulty values for the resampled images for the three different router types. As observed in Fig. 9(a), (c), and (e), the average τ_{image} varied between 1.66 and 3.01 s when PoA was used for consensus. In contrast, the average τ_{image} values varied significantly between 5.69 and 28.57 s when PoW was used for consensus as depicted in Fig. 9(b), (d), and (f).

B. Effect of Varying Block Difficulty on τ_{Image}

Fig. 10 depicts the average τ_{image} values in seconds as a function of difficulty values for the three different router types. As mentioned in Section III-E, four different difficulty levels were used in our experiments. The τ_{image} is averaged across all experimental runs and image sizes for the resampled images. As observed in Fig. 10(a), (c), and (e), the average τ_{image} varied between 1.89 and 2.87 s when PoA was used for consensus. In contrast when PoW was used for consensus, the average τ_{image} values varied significantly between 2.8 and 142.92 s as depicted in Fig. 10(b), (d), and (f).

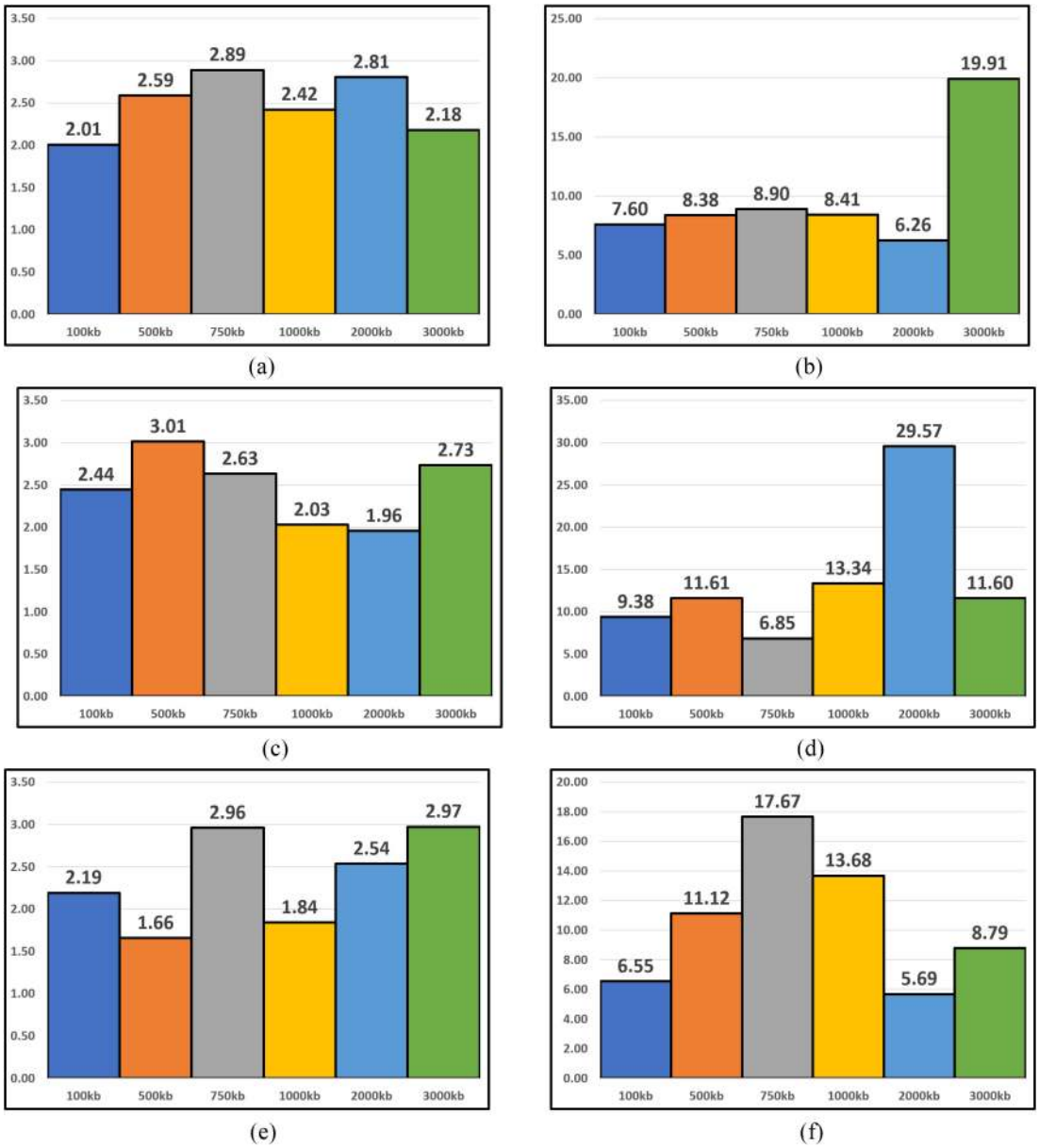

Fig. 9. Average τ_{image} in seconds for PoA and PoW tests for different image sizes for three different network routers. (a) PoA—Linksys Router. (b) PoW—Linksys Router. (c) PoA—TP-Link Router. (d) PoW—TP-Link Router. (e) PoA—Verizon Hotspot. (f) PoW—Verizon Hotspot.

C. Effects of WiFi Communications Disruption

In case an Ethereum node mounted on a quadrotor loses WiFi communication with its neighboring nodes, it is important that communication between the remaining nodes continues uninterrupted. This section documents the observed behavior of the Ethereum network in the face of such WiFi communication disruption. The flight mission aspect of the experiments was performed using a hardware-in-the-loop (HWIL) setup. The HWIL approach provided the ability to conduct safe and controlled experiments as per the guidelines of the U.S Federal Aviation Administration (FAA) [88]. All HWIL experiments were performed using the DJI Assistant 2 software. The software provided a real-time emulation of the DJI M100 rigid body dynamics to provide telemetry outputs and velocity and yaw rate inputs into the N1 flight controller.

The three quadrotors were commanded to fly to an altitude of 20 m above the ground and maintain a triangular formation. The secure communication commenced once the quadrotors reached the desired altitude of the quadrotors. After a period of 1 min, one of the nodes was commanded to break the flight formation and fly out of range of the remaining nodes. When a time period of 30-s elapsed, the quadrotor was commanded to fly back in range of the other nodes.

The following two scenarios were tested.

1) Miner Node Disruption: In the first scenario, the miner node (Node 1) was commanded to break flight formation and go out of WiFi range. Node 1 was also the bootstrap node. The transactions posted by the remaining nodes could not be validated by the miner. As such, no new blocks were added to the blockchain. The communication links between the nodes were disrupted, and the file transfer between the

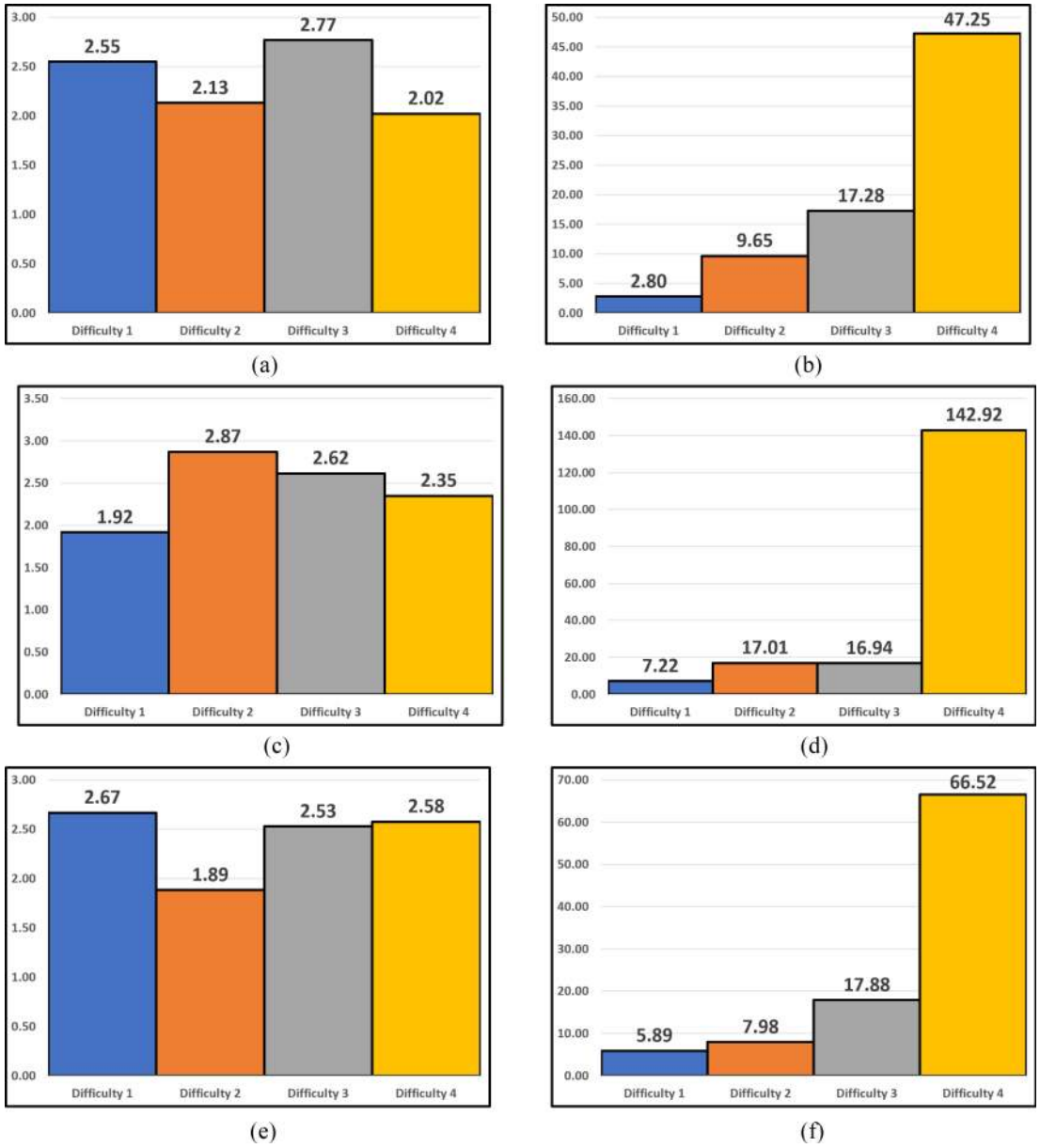

Fig. 10. Average τ_{image} in seconds for PoA and PoW tests for different difficulty levels for three different network routers. (a) PoA—Linksys Router. (b) PoW—Linksys Router. (c) PoA—TP-Link Router. (d) PoW—TP-Link Router. (e) PoA—Verizon Hotspot. (f) PoW—Verizon Hotspot.

nodes terminated. On Node 1, the Geth console and the Truffle program terminated, thereby disrupting the entire blockchain network between the nodes.

2) Nonminer Node Disruption: In the second scenario, a nonminer node (Node 3) was commanded to break formation and go out of WiFi range. The rest of the nodes remain unaffected and continued to interact with the smart contract. Due to the round-robin arrangement, Node 1 and Node 2 continued to exchange information over the blockchain. When Node 3 came back in range of the blockchain network, it was able to rejoin the Ethereum network. This is due to the fact that the Ethereum nodes connect with each other over the TCP port of the bootstrap node. Thus, when Node 3 was within the WiFi network range, it was able to rejoin the Ethereum network by connecting to the TCP port of Node 1.

D. Flight Duration With and Without Ethereum Payload

The flight time of multirotor sUAVs is critical for tasks, such as surveillance, transportation, and search and rescue. As such, it is vital that the flight times of blockchain-enabled multirotor sUAVs are not significantly hampered due to the increased payload. Outdoor flight tests were conducted to observe the difference in flight times of the multirotor sUAV with and without the Ethereum payload. A total of five outdoor flights were conducted where the DJI M100 quadrotor was flown to an altitude of 20 m and commanded to hover until the battery ran down to 20% charge. The quadrotors were equipped with the TB48D battery that has a capacity of 5700 mAh [89]. It was found that without the Ethereum payload, the average hovering flight time was approximately 21 min and 27 s. The nonminer node payload with the Raspberry Pi and associated

battery weighed 147.41 g with an average flight time of 19 min and 48 s. The miner node payload with the NVIDIA Jetson and associated battery weighed 997 g with an average flight time of 12 min and 23 s.

V. Discussion and Future Work

Several observations are made from the experiments.

1) For PoA, the difficulty level does not have any significant impact on the average τ_{image}. However, as expected, the average τ_{image} increases significantly with an increase in the difficulty levels for PoW.
2) At Difficulty 4, it was observed that the time taken to validate the transaction increased significantly for PoW. The time to validate transactions could not keep up with the rate at which images were being uploaded to the IPFS. Only 50% images were transmitted across the network when the TP-Link router was used. 33% of the images were successfully transferred when the Linksys router was used, and 66% images were successfully transferred when the Verizon mobile hotspot was used.
3) Overall, it was observed that PoA provided a lower average τ_{image} compared to PoW. This behavior is in line with the fact that PoA does not rely on the mining process to verify transactions. On the other hand, PoW relies on the nodes using their computational resources to solve the mining problem to validate transactions in a block, often taking up significant time. As such, PoA can validate transactions quicker than PoW as time goes on and proves to be the faster option for data transfer in multirobot systems.
4) *WiFi Disruptions:* When an Ethereum node experiences WiFi issues or disruptions, it is important that the remaining nodes continue the data transfer without getting affected. A miner node losing WiFi connection can be catastrophic to the network, as is observed by our experiments. Having more than one miner node in the network can add redundancy and provide protection against such failures. On the other hand, a nonminer node has the ability to drop out and rejoin the Ethereum network. All the logic that is required to ensure uninterrupted communication in the case of a nonminer node dropping out and rejoining can be coded in the smart contract.
5) *Security Analysis:* The security of smart contracts can be evaluated along the dimensions of data confidentiality, integrity, and nonrepudiation [90], [91].
   a) *Confidentiality:* Given the fact that this was a private Ethereum network, the Ethereum account addresses of the participating quadrotor nodes were known beforehand and were added to the smart contract code. Thus, only nodes whose address matched those defined in the smart contract could access the `get()` and `set()` functions of the smart contract. This mechanism provided confidentiality for secure information exchange. The tradeoff here was the inability to add more quadrotor nodes dynamically to the network. Almadhoun *et al.* [90] provided a novel procedure to automate the task of authenticating nodes requesting to join the network. Such automation approaches will be explored in future iterations of this work.
   b) *Data Integrity:* Each image was encrypted using an asymmetric encryption scheme OpenPGP, and then uploaded to the IPFS. When uploading the encrypted data to the IPFS, the second layer of encryption is added by passing the data through an SHA-256 algorithm and encoding it to base 58. Cryptographic hashes possess important characteristics, such as being deterministic, uncorrelated, unique, and one-way. The hash generated when uploading the encrypted image to the IPFS was then shared among the nodes using Ethereum transactions. The images exchanged over the network were protected by this double layer of encryption, ensuring that the data was not tampered with.
   c) *Nonrepudiation:* Transactions on the Ethereum blockchain are signed with a digital signature using the private key of the account issuing the transaction. The use of this approach provides three security advantages: i) a method to validate the Ethereum accounts in the network; ii) a signature that is unique to the account such that it cannot be forged; and iii) a guarantee of terms of service that ensures that the transaction data cannot be modified. As soon as the miner validated a transaction, it was recorded with its unique timestamp. Hence, nodes could not deny their actions as the actions were recorded in the tamper-proof logs.
6) *Extension to Other Blockchain Technologies:* In future work, the smart contract presented in this work will be implemented using other blockchain technologies to compare and contrast the performance of multiple blockchain platforms for multi-sUAV communications. The smart contract developed in this work can be extended to other blockchain technologies, such as Hyperledger and Rootstock blockchain for Bitcoin or RSK RBTC [92], [93]. Hyperledger Fabric provides permissioned blockchain and features the necessary framework to allow Ethereum smart contracts to run on its blockchain [93]. The smart contract presented here can be extended by creating an EVM wrapper around the Hyperledger burrow to run the smart contract bytecode written in Solidity. A key component included in Hyperledger fabric is the JSON RPC API Fab3 to mimic the web3.js library used by Ethereum DApps—this simplifies the extension process from Ethereum to Hyperledger. Similarly, RSK RBTC provides a framework to extend Ethereum smart contracts [92]. The RVM (RSK Virtual Machine) is compatible with EVM at the op-code level. This compatibility allows Ethereum smart contracts to run easily on the RSK-RBTC. RVM is also compatible with the tools used to deploy and interact with EVM smart contracts. Detailed instructions to

use Ethereum smart contracts on the RSK-RBTC can be found in [94]. Teams developing other blockchain technologies and frameworks, such as EOS are working toward creating functionality to extend Ethereum smart contracts to their platforms [95].

7) *Integration With Multi-sUAV Motion Planning Algorithms:* Integrating a blockchain-enabled multi-sUAV system with a decentralized motion planner is a natural extension to this work. A receding horizon, mixed-integer nonlinear programming (RH-MINLP)-based motion planner that allows multiple sUAVs to navigate toward multiple waypoints while securely transferring data over the blockchain will be the next objective of this study [10].

VI. Conclusion

This article documented the hardware and software setup, and experimental results for an Ethereum-based secure information exchange system mounted aboard multiple quadrotors sUAVs. The hardware architecture leveraged a low-cost NVIDIA GPU for mining during flight and Raspberry Pis as nonminer Ethereum nodes for data collection and storage. The software architecture leveraged Ethereum's ability to program smart contracts for secure, confidential, and tamper-proof data access, and a decentralized file system IPFS for data storage. The system was mounted on three DJI M100 quadrotors, and flight tests were performed to collect and securely share images across this 3-quadrotor network. The implementation of a private Ethereum network provided security features, such as confidentiality, data integrity, and nonrepudiation. The combined use of Ethereum blockchain and IPFS provided a distributed data storage system while avoiding a single point of database failure for the network. Experimental results focused on studying the average time taken to transfer an image across the network as a function of image size and consensus algorithm (PoA versus PoW) and the Ethereum difficulty level. The experiments also evaluated the resilience of the Ethereum network in the face of WiFi communication disruptions. Finally, the increased sUAV energy consumption due to additional Ethereum hardware payload as inferred by battery life and flight time was documented. The use of the PoA consensus algorithm provided faster image transfer compared to PoW. The image transfer times, ability to continue data transfer in the face of communication disruption and reasonable multi-sUAV flight times point to the feasibility of a blockchain-enabled communication system for multiple sUAVs.

References

[1] C. Pamela, G. Alastair, L. Meredith, and R. Melanie. (2017). *Commercial Drones Are Here: The Future of Unmanned Aerial Systems*. Accessed: Oct. 24, 2019. [Online]. Available: https://www.mckinsey.com/industries/capital-projects-and-infrastructure/our-insights/commercial-drones-are-here-the-future-of-unmanned-aerial-systems

[2] L. Richard. (2018). *Drone Industry Just Beginning to Take Off*. Accessed: Jan. 26. 2020. [Online]. Available: https://www.forbes.com/sites/richardlevick/2018/05/15/drone-industry-just-beginning-to-take-off/#7323272472bc

[3] G. S. Research. (2020). *Drones: Reporting for Work*. Accessed: Oct. 24, 2019. [Online]. Available: https://www.goldmansachs.com/insights/technology-driving-innovation/drones/

[4] (2019). *Unmanned Aircraft Systems: Forecast*. Accessed: Jan. 24, 2020. [Online]. Available: https://www.faa.gov/data_research/aviation/aerospace_forecasts/media/unmanned_aircraft_systems.pdf

[5] S. Hasan, "Urban air mobility (UAM) market study," NASA, Washington, DC, USA, Rep. 52, 2019.

[6] H. A. Almurib, P. T. Nathan, and T. N. Kumar, "Control and path planning of quadrotor aerial vehicles for search and rescue," in *Proc. IEEE SICE Annu. Conf.*, 2011, pp. 700–705.

[7] M. A. Ma'sum *et al.*, "Simulation of intelligent unmanned aerial vehicle UAV for military surveillance," in *Proc. IEEE Int. Conf. Adv. Comput. Sci. Inf. Syst. (ICACSIS)*, 2013, pp. 161–166.

[8] P. Y. Haas, C. Balistreri, P. Pontelandolfo, G. Triscone, H. Pekoz, and A. Pignatiello, "Development of an unmanned aerial vehicle UAV for air quality measurement in urban areas," in *Proc. 32nd AIAA Appl. Aerodyn. Conf.*, 2014, p. 2272.

[9] R. G. Valenti, Y. Jian, K. Ni, and J. Xiao, "An autonomous flyer photographer," in *Proc. IEEE Int. Conf. Cyber Technol. Autom. Control Intell. Syst. (CYBER)*, Jun. 2016, pp. 273–278.

[10] P. Abichandani, K. Levin, and D. Bucci, "Decentralized formation coordination of multiple quadcopters under communication constraints," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 3326–3332.

[11] I. de Boisblanc *et al.*, "Designing a hexacopter for the collection of atmospheric flow data," in *Proc. Syst. Inf. Eng. Design Symp. (SIEDS)*, Apr. 2014, pp. 147–152.

[12] R. T. Palomaki, N. T. Rose, M. van den Bossche, T. J. Sherman, and S. F. De Wekker, "Wind estimation in the lower atmosphere using multirotor aircraft," *J. Atmosp. Ocean. Technol.*, vol. 34, no. 5, pp. 1183–1191, 2017.

[13] C. A. Wolf *et al.*, "Wind data collection techniques on a multi-rotor platform," in *Proc. Syst. Inf. Eng. Design Symp. (SIEDS)*, Apr. 2017, pp. 32–37.

[14] G. W. Donnell, J. A. Feight, N. Lannan, and J. D. Jacob, "Wind characterization using onboard IMU of SUAS," in *Proc. Atmosp. Flight Mech. Conf.*, 2018, p. 2986.

[15] T. Shimura, M. Inoue, H. Tsujimoto, K. Sasaki, and M. Iguchi, "Estimation of wind vector profile using a hexarotor unmanned aerial vehicle and its application to meteorological observation up to 1000m above surface," *J. Atmosp. Ocean. Technol.*, vol. 35, no. 8, pp. 1621–1631, 2018.

[16] S. Prudden, A. Fisher, M. Marino, A. Mohamed, S. Watkins, and G. Wild, "Measuring wind with small unmanned aircraft systems," *J. Wind Eng. Ind. Aerodyn.*, vol. 176, pp. 197–210, May 2018.

[17] S. Waslander and C. Wang, "Wind disturbance estimation and rejection for quadrotor position control," in *Proc. AIAA Infotech Aerosp. Conf.*, 2009, p. 1983.

[18] M. Marino, A. Fisher, R. Clothier, S. Watkins, S. Prudden, and C. S. Leung, "An evaluation of multi-rotor unmanned aircraft as flying wind sensors," *Int. J. Micro Air Veh.*, vol. 7, no. 3, pp. 285–299, 2015.

[19] J. Kelly and A. Williams. *Forty Big Banks Test Blockchain-Based Bond Trading System*. Accessed: Oct. 24, 2019. [Online]. Available: https://www.reuters.com/article/banking-blockchain-bonds/forty-big-banks-test-blockchain-based-bond-trading-system-idUSL8N16A30H

[20] I. Kar. *Estonian Citizens Will Soon Have the World's Most Hack-Proof Health-Care Records*. Accessed: Oct. 24, 2019. [Online]. Available: https://qz.com/628889/this-eastern-european-country-is-moving-its-health-records-to-the-blockchain/

[21] W. Suberg. *Factom's Latest Partnership Takes on U.S. Healthcare*. Accessed: Oct. 24, 2019. [Online]. Available: https://cointelegraph.com/news/factoms-latest-partnership-takes-on-us-healthcare

[22] S. Lacey. *The Energy Blockchain: How Bitcoin Could Be a Catalyst for the Distributed Grid*. [Online]. Available: https://www.greentechmedia.com/articles/read/the-energy-blockchain-could-bitcoin-be-a-catalyst-for-the-distributed-grid

[23] D. Oparah. *Ways That the Blockchain Will Change the Real Estate Market*. [Online]. Available: https://techcrunch.com/2016/02/06/3-ways-that-blockchain-will-change-the-real-estate-market/

[24] A. Mizrahi. (2015). *A Blockchain-Based Property Ownership Recording System*. Accessed: Jul. 7, 2020. [Online]. Available: https://www.paperdue.com/essay/blockchain-based-property-ownership-recording-2166424

[25] J. Wan, J. Li, M. Imran, and D. Li, "A blockchain-based solution for enhancing security and privacy in smart factory," *IEEE Trans. Ind. Informat.*, vol. 15, no. 6, pp. 3652–3660, Jun. 2019.

[26] H. M. Kim and M. Laskowski, "Toward an ontology-driven blockchain design for supply-chain provenance," *Intell. Syst. Account. Finance Manag.*, vol. 25, no. 1, pp. 18–27, 2018.

[27] S. He, W. Ren, T. Zhu, and K.-K. R. Choo, "BoSMoS: A blockchain-based status monitoring system for defending against unauthorized software updating in industrial Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 2, pp. 948–959, Feb. 2020.

[28] K. Liu, W. Chen, Z. Zheng, Z. Li, and W. Liang, "A novel debt-credit mechanism for blockchain-based data-trading in Internet of Vehicles," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 9098–9111, Oct. 2019.

[29] Z. Yang, K. Yang, L. Lei, K. Zheng, and V. C. M. Leung, "Blockchain-based decentralized trust management in vehicular networks," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1495–1505, Apr. 2019.

[30] K. Gai, Y. Wu, L. Zhu, L. Xu, and Y. Zhang, "Permissioned blockchain and edge computing empowered privacy-preserving smart grid networks," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7992–8004, Oct. 2019.

[31] S. Mondal, K. P. Wijewardena, S. Karuppuswami, N. Kriti, D. Kumar, and P. Chahal, "Blockchain inspired RFID-based information architecture for food supply chain," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 5803–5813, Jun. 2019.

[32] M. Shen, X. Tang, L. Zhu, X. Du, and M. Guizani, "Privacy-preserving support vector machine training over blockchain-based encrypted iot data in smart cities," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7702–7712, Oct. 2019.

[33] C. Lin, D. He, N. Kumar, X. Huang, P. Vijaykumar, and K.-K. R. Choo, "Homechain: A blockchain-based secure mutual authentication system for smart homes," *IEEE Internet Things J.*, vol. 7, no. 2, pp. 818–829, Feb. 2020.

[34] J. Xu *et al.*, "Healthchain: A blockchain-based privacy preserving scheme for large-scale health data," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8770–8781, Jun. 2019.

[35] J. Luo, Q. Chen, F. R. Yu, and L. Tang, "Blockchain-enabled software-defined industrial Internet of Things with deep reinforcement learning," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5466–5480, Jun. 2020.

[36] K. Salah, M. H. U. Rehman, N. Nizamuddin, and A. Al-Fuqaha, "Blockchain for AI: Review and open research challenges," *IEEE Access*, vol. 7, pp. 10127–10149, 2019.

[37] H. R. Hasan and K. Salah, "Combating deepfake videos using blockchain and smart contracts," *IEEE Access*, vol. 7, pp. 41596–41606, 2019.

[38] A. Chaer, K. Salah, C. Lima, P. P. Ray, and T. Sheltami, "Blockchain for 5G: Opportunities and challenges," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, 2019, pp. 1–6.

[39] A. Suliman, Z. Husain, M. Abououf, M. Alblooshi, and K. Salah, "Monetization of IoT data using smart contracts," *IET Netw.*, vol. 8, no. 1, pp. 32–37, 2018.

[40] M. A. Khan and K. Salah, "IoT security: Review, blockchain solutions, and open challenges," *Future Gener. Comput. Syst.*, vol. 82, pp. 395–411, May 2018.

[41] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the Internet of Things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[42] N. Noguchi, J. Will, J. Reid, and Q. Zhang, "Development of a master-slave robot system for farm operations," *Comput. Electron. Agricult.*, vol. 44, no. 1, pp. 1–19, 2004.

[43] M. Le Goc, L. H. Kim, A. Parsaei, J.-D. Fekete, P. Dragicevic, and S. Follmer, "Zooids: Building blocks for swarm user interfaces," in *Proc. ACM 29th Annu. Symp. User Interface Softw. Technol.*, 2016, pp. 97–109.

[44] L. A. Grieco *et al.*, "IoT-aided robotics applications: Technological implications, target domains and open issues," *Comput. Commun.*, vol. 54, pp. 32–47, Dec. 2014.

[45] V. Lippiello, B. Siciliano, and L. Villani, "Position-based visual servoing in industrial multi-robot cells using a hybrid camera configuration," *IEEE Trans. Robot.*, vol. 23, no. 1, pp. 73–86, Feb. 2007.

[46] E. C. Ferrer, "The blockchain: A new framework for robotic swarm systems," in *Proc. Future Technol. Conf.*, 2018, pp. 1037–1058.

[47] L. Deepan. *Smart Contract Code*. [Online]. Available: https://github.com/radlab-sketch/Ethereum

[48] A. Kapitonov, S. Lonshakov, A. Krupenkin, and I. Berman, "Blockchain-based protocol of autonomous business activity for multi-agent systems consisting of UAVs," in *Proc. Workshop Res. Educ. Develop. Unmanned Aerial Syst. (RED-UAS)*, Oct. 2017, pp. 84–89.

[49] V. Strobel, E. C. Ferrer, and M. Dorigo, "Managing Byzantine robots via blockchain technology in a swarm robotics collective decision making scenario," in *Proc. 17th Int. Conf. Auton. Agents Multiagent Syst.*, 2018, pp. 541–549.

[50] M. Y. Afanasev, Y. V. Fedosov, A. A. Krylova, and S. A. Shorokhov, "An application of blockchain and smart contracts for machine-to-machine communications in cyber-physical production systems," in *Proc. IEEE Ind. Cyber Phys. Syst. (ICPS)*, May 2018, pp. 13–19.

[51] J. Kang *et al.*, "Blockchain for secure and efficient data sharing in vehicular edge computing and networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4660–4670, Jun. 2019.

[52] D. Calvaresi, A. Dubovitskaya, J. P. Calbimonte, K. Taveter, and M. Schumacher, "Multi-agent systems and blockchain: Results from a systematic literature review," in *Proc. Int. Conf. Practical Appl. Agents Multiagent Syst.*, 2018, pp. 110–126.

[53] M. Fernandes and L. A. Alexandre, "Robotchain: Using Tezos technology for robot event management," *Ledger*, vol. 4, no. 1, p. 5, 2019.

[54] V. Lopes, L. A. Alexandre, and N. Pereira, "Controlling robots using artificial intelligence and a consortium blockchain," 2019. [Online]. Available: arXiv:1903.00660.

[55] S. B. H. Youssef, S. Rekhis, and N. Boudriga, "A blockchain based secure IoT solution for the dam surveillance," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, 2019, pp. 1–6.

[56] S. Aggarwal, M. Shojafar, N. Kumar, and M. Conti, "A new secure data dissemination model in Internet of Drones," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2019, pp. 1–6.

[57] E. Barka, C. A. Kerrache, H. Benkraouda, K. Shuaib, F. Ahmad, and F. Kurugollu, "Towards a trusted unmanned aerial system using blockchain for the protection of critical infrastructure," *Trans. Emerg. Telecommun. Technol.*, Jul. 2019, Art. no. e3706. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.3706

[58] A. Mokhtar, N. Murphy, and J. Bruton, "Blockchain-based multi-robot path planning," in *Proc. IEEE 5th World Forum Internet Things (WF-IoT)*, 2019, pp. 584–589.

[59] *Aira*. Accessed: Dec. 5, 2020. [Online]. Available: https://aira.life/

[60] C. Pinciroli *et al.*, "ARGoS: A modular, parallel, multi-engine simulator for multi-robot systems," *Swarm Intell.*, vol. 6, no. 4, pp. 271–295, 2012.

[61] G. Valentini, D. Brambilla, H. Hamann, and M. Dorigo, "Collective perception of environmental features in a robot swarm," in *Proc. Int. Conf. Swarm Intell.*, 2016, pp. 65–76.

[62] E. Barka, C. A. Kerrache, N. Lagraa, A. Lakas, C. T. Calafate, and J.-C. Cano, "UNION: A trust model distinguishing intentional and unintentional misbehavior in inter-UAV communication," *J. Adv. Transp.*, vol. 2018, Apr. 2018, Art. no. 7475357.

[63] C. A. Kerrache, E. Barka, N. Lagraa, and A. Lakas, "Reputation-aware energy-efficient solution for FANET monitoring," in *Proc. 10th IFIP Wireless Mobile Netw. Conf. (WMNC)*, 2017, pp. 1–6.

[64] *Hyperledger Fabric Documentaion*. Accessed: Oct. 24, 2019. [Online]. Available: https://wiki.hyperledger.org/display/fabric/Hyperledger+Fabric

[65] K. Danilov, R. Rezin, I. Afanasyev, and A. Kolotov, "Towards blockchain-based Robonomics: Autonomous agents behavior validation," in *Proc. IEEE Int. Conf. Intell. Syst. (IS)*, 2018, pp. 222–227.

[66] T. M. FernÁndez-Caramés, O. Blanco-Novoa, I. Froiz-Míguez, and P. Fraga-Lamas, "Towards an autonomous industry 4.0 warehouse: A UAV and blockchain-based system for inventory and traceability applications in big data-driven supply chain management," *Sensors*, vol. 19, no. 10, p. 2394, 2019.

[67] L. Paull *et al.*, "Duckietown: An open, inexpensive and flexible platform for autonomy education and research," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2017, pp. 1497–1504.

[68] *DJI Matrice M100: Quadcopter for Developers*. Accessed: Oct. 10, 2019. [Online]. Available: https://www.dji.com/matrice100

[69] *Orbitty carrier for NVIDIA Jetson TX2*. Accessed: May 12, 2020. [Online]. Available: http://connecttech.com/product/orbitty-carrier-for-nvidia-jetson-tx2-tx1/

[70] *Linksys EA*9500 *Max-Stream—AC5400 MU-MIMO Gigabit WiFi Router*. Accessed: May 12, 2020. [Online]. Available: https://www.linksys.com/us/p/$P-EA$9500/

[71] *AC1750 Wireless Dual Band Gigabit Router*. Accessed: May 12, 2020. [Online]. Available: https://www.tp-link.com/us/home-networking/wifi-router/archer-c7/

[72] *Franklin Wireless Corp Ellipsis*. Accessed: May 12, 2020. [Online]. Available: https://www.verizon.com/internet-devices/verizon-ellipsis-jetpack-*mhs*900*l*/

[73] *Understanding Blockchain Fundamentals*. Accessed: May 12, 2020. [Online]. Available: https://medium.com/loom-network/understanding-blockchain-fundamentals-part-2-proof-of-work-proof-of-stake-*b*6*ae*907*c*7*edb*

[74] *Proof of Stake*. Accessed: May 12, 2020. [Online]. Available: https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ

[75] *Proof of Authority Chains*. Accessed: May 12, 2020. [Online]. Available: https://openethereum.github.io/wiki/Proof-of-Authority-Chains

[76] *Implementing PBFT in blockchain*. Accessed: May 12, 2020. [Online]. Available: https://medium.com/coinmonks/implementing-pbft-in-blockchain-12368*c*6*c*9548

[77] G. Wood. (2014). *Ethereum Yellow Paper*. Accessed: Oct. 24, 2019. [Online]. Available: https://github. com/ethereum/yellowpaper

[78] *Ethereum Virtual Machine*. Accessed: May 12, 2020. [Online]. Available: https://github.com/ethereum/wiki/wiki/Ethereum-Virtual-Machine-(EVM)-Awesome-List

[79] *Ethereum Homestead Documentation*. Accessed: Jul. 7, 2020. [Online]. Available: https://ethdocs.org/

[80] *Truffle Documentation*. Accessed: Jul. 23, 2020. [Online]. Available: https://www.trufflesuite.com/

[81] *Solidity v0.6.11 Documentation*. Accessed: Jul. 23, 2020. [Online]. Available: https://solidity.readthedocs.io/

[82] *Ethereum Difficulty Bomb Explained*. Accessed: May 12, 2020. [Online]. Available: https://www.mangoresearch.co/ethereum-difficulty-bomb-explained/

[83] *Free Gas Networks*. [Online]. Available: https://besu.hyperledger.org/en/stable/HowTo/Configure/FreeGas

[84] S. Malik, V. Dedeoglu, S. S. Kanhere, and R. Jurdak, "TrustChain: Trust management in blockchain and IoT supported supply chains," in *Proc. IEEE Int. Conf. Blockchain (Blockchain)*, 2019, pp. 184–193.

[85] P. Albert. (2018). *Storing on Ethereum. Analyzing the Costs*. Accessed: Oct. 10, 2019. [Online]. Available: https://medium.com/coinmonks/storing-on-ethereum-analyzing-the-costs-922d41d6b316

[86] P. Labs. (2018). *IPFS Documentation*. Accessed: Oct. 24, 2019. [Online]. Available: https://docs.ipfs.io/

[87] *OpenPGP*. Accessed: May 12, 2020. [Online]. Available: https://www.openpgp.org/

[88] *Educational Users*. Accessed: Jul. 7, 2020. [Online]. Available: https://www.faa.gov/uas/educational_users/

[89] *Matrice 100 TB48D Battery*. Accessed: Jul. 7, 2020. [Online]. Available: https://store.dji.com/product/matrice-100-tb48d-battery

[90] R. Almadhoun, M. Kadadha, M. Alhemeiri, M. Alshehhi, and K. Salah, "A user authentication scheme of IoT devices using blockchain-enabled fog nodes," in *Proc. IEEE/ACS 15th Int. Conf. Comput. Syst. Appl. (AICCSA)*, 2018, pp. 1–8.

[91] H. R. Hasan and K. Salah, "Proof of delivery of digital assets using blockchain and smart contracts," *IEEE Access*, vol. 6, pp. 65439–65448, 2018.

[92] *RSK Documentation*. Accessed: Jul. 7, 2020. [Online]. Available: https://developers.rsk.co/rsk/

[93] *Using Hyperledger Fabric to Set Up Ethereum Smart Contracts*. Accessed: Jul. 7, 2020. [Online]. Available: https://www.devteam.space/blog/how-to-deploy-smart-contract-on-ethereum/

[94] *RSK Documentation: Geth*. Accessed: Jul. 7, 2020. [Online]. Available: https://developers.rsk.co/tutorials/ethereum-devs/geth-attach-local-node/

[95] M. Dalton. *EOS Pursuing Compatibility With Ethereum Smart Contracts*. Accessed: Jul. 7, 2020. [Online]. Available: https://cryptobriefing.com/eos-pursuing-compatibility-ethereum-smart-contract-code/