SecureIoD: A Secure Data Collection and Storage Mechanism for Internet of Drones

Cong Pu[†], Andrew Wall[†], Imtiaz Ahmed[§], and Kim-Kwang Raymond Choo[¶]
[†]Dept. of Computer Sciences and Electrical Engineering, Marshall University, Huntington, WV, United States
[§]Dept. of Electrical Engineering and Computer Science, Howard University, Washington, DC, United States
[¶]Dept. of Information Systems and Cyber Security, University of Texas at San Antonio, San Antonio, TX, United States
cong.pu@ieee.org wall48@marshall.edu imtiaz.ahmed@howard.edu raymond.choo@fulbrightmail.org

***Abstract*—Thanks to rapid advancements in microprocessors, battery technologies, and lightweight materials, unmanned aerial vehicles (UAVs), commonly known as drones, have received significant interest in the past few years. As drone-related commercial and civilian applications are flourishing, Internet-of-Drones (IoD) is moving into the fast lane and quickly becoming a highly anticipated network paradigm, where drones and Zone Service Providers (ZSPs) coordinate knowledge sharing in a reliable, accurate, and efficient way. However, for the sake of both strategic and financial value to business and mission critical applications, it is of vital importance to address both data security and privacy preservation issues brought by drones' inherent resource constraints and wide-open wireless medium. In this paper, we propose a secure data collection and storage mechanism, also called *SecureIoD*, for the IoD environment. In *SecureIoD*, drones and ZSPs first mutually authenticate each other and establish a secure session key before sharing any sensitive data via an insecure wireless channel. Then, ZSPs pack the collected data into blocks and compete to add their blocks into the blockchain. We also propose a joint Proof-of-Work (PoW) and Proof-of-Stake (PoS) consensus mechanism to select the miner ZSP, where the more transactions are in the block, the easier a ZSP can solve the cryptographic puzzle. We present security verification and analysis to show that *SecureIoD* can resist various security attacks. Finally, we develop a real-world testbed, implement *SecureIoD* and existing SDDM and BACSIoD schemes, and carry out extensive simulation experiments for performance evaluation and analysis. Experimental results reveal that not only does *SecureIoD* have lower computation cost, energy consumption, miner selection time, and communication overhead, but also offer better security features and capabilities.**

***Index Terms*—Drone, Internet of Drones, Data Security, Authentication and Key Agreement, Data Storage, Blockchain**

I. Introduction

Drones, officially known as unmanned aerial vehicles (UAVs), have attracted considerable attention for various applications such as disaster and emergency response, infrastructure inspection, and smart cities during the recent years [1]. For example, a 2.8 mile 10 minutes drone trip transported a kidney for transplant in May of 2019 in Baltimore (Maryland, U.S.) [2], where the drone helped to reduce delivery time significantly and avoid human contact during organ transportation. The size of global commercial drone market is estimated to be USD 35 billion by 2026. In the U.S., the economic outlook for drone technology is also escalating alongside the rise of drone industry, where the commercial drone market is expected to be valued at USD 5 billion by 2025 [3]. From a military innovation, to an exciting hobby, to a technology that is transforming commercial industries, future opportunities in the emerging technology field of drones are limitless [4].

In order to increase maneuverability while flying and reduce weight, drones are made of different light composite materials such as carbon fiber-reinforced composites and thermoplastics [5]. In addition, drones are usually equipped with a variety of additional equipment, including wireless communication device, global positioning system (GPS), navigation system, and various sensors. Thus, drones are viewed as an emerging form of new Internet of Things (IoT) devices, flying in the sky with full network connectivity capabilities [6]. Several attempts have been made to integrate drones into IoT to form an innovative communication paradigm called Internet of Drones (IoD), which aims to smoothly connect a plethora of drones with the Internet [7]. In the IoD, airspace is considered as shared resource made available to all drones, and is further partitioned into predetermined zones. Adjacent zones are reachable from each other through incoming and outgoing zone gates which belong to two adjacent zones. Each zone is under the administration of one or multiple Zone Service Providers (ZSPs) which act as Internet access points and provide/collect up-to-date information to/from drones. For example, surveillance drones can patrol target area, observe crowds, and deliver observational data to a nearby ZSP for predicting the spread of COVID-19 disease [8].

In a variety of IoD applications, starting from military setting (e.g., border security and surveillance [9]) to civilian scenario (e.g., enforcing stay-at-home order during the COVID-19 pandemic [10]), massive volume of highly critical data are collected and transmitted over physical networks to data centers for further processing and storage. This process might cause many data security and privacy challenges for both individuals and enterprises. Moreover, drones are resource-constrained devices and considered to be defenseless to security attacks. An adversary can deliberately target either onboard physical elements of drones or insecure wireless channel [11]. Unquestionably, investigating potential security and privacy challenges in the IoD environment and designing the state-of-the-art data collection and storage mechanisms are urgent affairs to ensure sustainable development of IoD industry.

Drones and ZSPs in the IoD environment communicate

over insecure wireless channel, thus they should mutually authenticate each other to verify the genuine identity before sharing any sensitive and critical information. The traditional cryptographic mechanisms (e.g., RSA and AES [12]) could be adopted to meet fundamental security requirements such as authentication, session key agreement, and privacy preservation. Unfortunately, those heavyweight cryptographic mechanisms demand long computation time and consume vast amount of energy [13]. Thus, lightweight security protocols are the only solution to protect the IoD environment. In addition, an adversary might physically capture a drone and attempt to extract credentials stored in the memory through memory disclosure attacks [14]. To defend against both software-based and physical memory disclosure attacks, drones should have tamper-resistant module to safeguard information stored in the electronic circuitry. Last but not least, in the traditional centralized storage management systems, the centralized server is responsible for storing and processing all collected (sensitive) data. As drones need to receive/send data and perform decision-making promptly without time delay, the strict quality-of-service (QoS) requirements cannot be guaranteed by the centralized system. Moreover, a centralized system can incur significant administrative costs and has other limitations such as single point of failure. Consequently, it is necessary to develop a decentralized reliable, consistent, and secure data storage mechanism in the IoD environment.

In this paper, we propose a security mechanism for the IoD environment, and analyze and measure their security resiliency and performance trade-off through security verification and analysis as well as experimental study. The major contribution of this paper is summarized in fourfold:

- We propose a secure data collection and storage mechanism (*SecureIoD*) for the IoD environment. The basic idea of *SecureIoD* is that drones and ZSPs first mutually authenticate each other and establish a secure session key based on physical unclonable function and Henon map before sharing any sensitive data via insecure wireless channel. Then, ZSPs pack the collected data into blocks and compete to add their blocks into the blockchain.
- We propose a joint Proof-of-Work (PoW) and Proof-of-Stake (PoS) consensus mechanism to select the miner ZSP in the IoD environment, where the more transactions are in the block, the easier a ZSP can solve the cryptographic puzzle. In this way, the block containing more transactions can be added in the blockchain earlier, as it affects the whole ledger significantly.
- We conduct security verification of *SecureIoD* using the security protocol verification tool such as AVISPA [15] and Scyther [16], and present a security analysis of *SecureIoD*. Our security verification and analysis results prove that *SecureIoD* is resilient and immune to various security attacks.
- We develop a real-world testbed which is composed of one HP ENVY Notebook laptop [17] and one Latte Panda development board [18] for performance evaluation. We also revisit prior approaches, SDDM [19] and BACSIoD [20], and modify them to work in the real-world testbed for performance comparison and analysis.

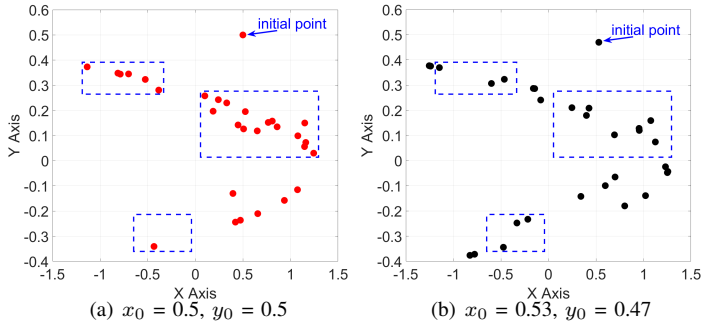
We conduct extensive simulation experiments and measure the performance of *SecureIoD*, SDDM, and BACSIoD in terms of running time, CPU time, the number of clock cycles, energy consumption, and communication cost. Simulation results demonstrate that *SecureIoD* can achieve better performance compared to prior approaches, indicating a viable and competitive approach for ensuring secure data collection and storage in the IoD environment. To drive creative advancement in the realm of data collection and storage within the IoD community, we open source[1] at the https://github.com/congpu/SecureIoD.

The rest of the paper is organized as follows. Existing literature and recent studies are provided and analyzed in Section II. Section III provides a brief introduction to physical unclonable function and Henon map. Section IV describes network and adversary models. The proposed secure data collection and storage mechanism is presented in Section V. The security verification followed by a security analysis are provided in Section VI. Section VII presents the experimental results. Finally, we conclude the paper in Section VIII.

II. Related Work

In [20], the authors propose a blockchain-based access control mechanism for the IoD environment, where the certificates of drones and ground station server are issued by the trusted control room. The proposed mechanism first achieves mutual authentication and key establishment between communication entities in the IoD environment based on elliptic curve cryptography, elliptic curve digital signature, and cryptographic one-way hash function. Then, the ground station server forms the accessed data into various transactions which will be placed in a block. Finally, the block is verified and added in the blockchain by the peer-to-peer cloud server network with the help of Ripple Protocol Consensus Algorithm (RPCA) [21]. The proposed mechanism provides better security and more functionality attributes compared to other approaches. However, the authors also observe a significant increase in both communication cost and computation cost because of the adoption of heavyweight cryptographic techniques. The authors in [22] propose a blockchain-based security mechanism to ensure secure transfer of information among drones in cyber-physical systems. The proposed security mechanism consists of three phases: registration, verification, and transaction. To select the miner node, a deep learning-based Boltzmann machine using features such as drones' computational resources, available battery power, and flight time is adopted. Even though the authors use security analysis and experimental evaluation (i.e., computation time and communication cost) to show that the proposed mechanism can provide data integrity and privacy for the IoD ecosystem, it is undeniable that a deep learning technique can result in excessive energy consumption in resource-constrained drones.

[1] *SecureIoD* source codes and its security verification programs are publicly available at the https://github.com/congpu/SecureIoD.



(a) $x_0 = 0.5$, $y_0 = 0.5$ (b) $x_0 = 0.53$, $y_0 = 0.47$

Fig. 1. Henon map with different initial conditions after 30 iterations, where the rectangles highlight different sequences of points in same regions.

The authors in [23] propose an authentication scheme based on blockchain for drone-enabled smart cities. A group of drones are divided into sub-groups based on their geographical locations, and drones from different sub-groups can communicate securely using a customized decentralized consensus algorithm without re-authenticating. In [24], a communication framework based on blockchain technique is designed for UAV systems. In order to reduce the overhead of computation and storage on UAVs, the authors modify the transaction and the structure of block. In addition, the authors also design a delegated Proof-of-Stake consensus protocol which can be used to achieve the agreement among a group of trustless UAVs. However, the common issue of the above two approaches [23], [24] is that drones do not have any tamper-resistant capabilities. Once the adversary physically captures a drone and extracts all stored credentials, the entire framework and its access control mechanism could be compromised. In [25], a data collection mechanism using blockchain is proposed to assist drones to collect data in the IoT environment. The primary responsibilities of drones are to fly over the IoT devices, provide wireless network access as well as collect data. As a return, drones will receive charging credits so that they can spend them for charging time. Moreover, drones will store the collected data in the blockchain to defend against malicious attacks. The authors in [26] present a survey on security issues in 5G-enabled UAV networks. In addition, they discuss blockchain-based security solutions and summarize potential research challenges in the integration of blockchain with 5G-enabled UAVs.

Quite recently, several blockchain-based security protocols and techniques have been proposed for the IoD environment. However, none of the above security protocols provide superior security and performance simultaneously. To the best of our knowledge, we design the first secure data collection and storage mechanism based on physical unclonable function, Henon map, and blockchain for the IoD environment.

III. Preliminary Background

A. Physical Unclonable Function

A physical unclonable function (PUF) is widely used as the electronic device's physical identity, and has emerged as a promising solution for defending against physical attacks [27]. The rationale behind the design of PUF is that the input-output mapping exploits the intrinsic fabrication and manufacturing variabilities in the process of making integrated circuit (IC). For every PUF, an input value, called ''challenge'', receives a corresponding output value, called ''response'', based on the physical microstructure of the device. A challenge together with the corresponding response is known as a challenge-response pair (CRP). Generally, a PUF can be represented as a secure one-way function F_{puf}, $RE = F_{puf}(CH)$, where *CH* and *RE* are the input challenge and output response of PUF, respectively. If the same challenge is fed into an authentic PUF multiple times, the same responses will be produced each time. On the other side, the same challenge provided to different PUFs will produce totally different responses.

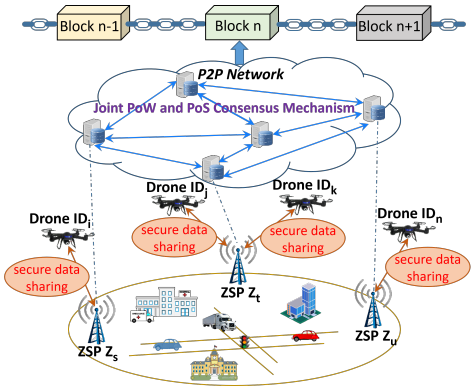


Fig. 2. Illustration of secure data collection and storage framework for the IoD environment.

B. Henon Map

A chaotic system is defined as one that significant difference in system's behavior can be caused by the smallest changes in the system. The word *chaos* is described as ''a state of utter confusion or disorder; a total lack of organization or order.'' The Henon map [28], sometimes called Henon-Pomeau attractor/map, is one of discrete-time and dynamical chaotic systems. The underlying patterns and deterministic laws, which are sensitive to initial conditions of chaotic system, determine the randomness and irregularities of Henon map. The Henon map is a two-dimensional chaotic system, accepting a coordinate point (x_n, y_n) and mapping it to a new point according to $x_{n+1} = 1 - ax_n^2 + y_n$ and $y_{n+1} = bx_n$, where a and b are defined as system parameters. In Henon map, the chaos can be shown with certain system parameter values (i.e., a and b) and initial condition (i.e., x_0 and y_0). The exact same chaos or sequence of points cannot be reproduced if there is no right initial condition (x_0, y_0). As shown in Fig. 1, a different sequence of points will be plotted when different initial conditions are provided to the Henon map.

IV. System and Adversary Models

A secure data collection and storage framework for the IoD environment is depicted in Fig. 2, where there are two major entities: drones and ZSPs. Drones are equipped with on-board sensors and communication devices, which are capable of gathering and sharing data. In addition, each drone is also armed with a PUF enabled integrated circuit. ZSPs form a peer-to-peer (P2P) network and complete the following two

tasks: i) generate a block: collect, validate, and pack data into a block; ii) add the block in the blockchain: compete to add the block in the blockchain using the consensus mechanism. We assume that a drone first chooses its real identity ID_i and the CRP challenge CH_i^{ts}. Then, the drone obtains its initial CRP (CH_i^{ts}, RE_i^{ts}) according to F_{puf} and computes its initial pseudonym PID_i^{ts}, PID_i^{ts} = $H(ID_i \parallel RE_i^{ts})$. Here, ts, $H(\cdot)$, and $\parallel$ are the current timestamp, the collision-resistant cryptographic one-way hash function, and the concatenation operation, respectively. The drone's real identity ID_i is not transmitted directly in plain text but in a masked form (or pseudonym) PID_i^{ts}. Due to the hardness of guessing a large sequence of random numbers (i.e., 256 bits for SHA-256), the adversary is infeasible to compute drone's real identity without knowing the CRP response. Therefore, the privacy of drones' identities or the drone anonymity can be guaranteed. In addition, the time-varying CRP response RE_i^{ts} is embedded in the hash function to calculate the pseudonym of drone, which can achieve un-traceability. Then, the drone shares its ID_i, (CH_i^{ts}, RE_i^{ts}), and PID_i^{ts} with ZSPs securely using the time-based OTP algorithm (TOTP) [29] during the system deployment phase. Once the system deployment phase is over, ZSPs store each drone's real identity, initial CRP, and initial pseudonym, while drones only store their real identities and challenges of initial CRP.

According to the widely adopted adversary model, any two communicating entities who communicate over an insecure wireless channel are assumed to be untrustworthy [12]. An adversary not only can overhear and intercept the transmitted messages, but also can duplicate, corrupt, alter, or replay the message contents. In addition, a drone with the collected sensitive information may move to an unattended hostile area, where an adversary can issue malicious commands (i.e., ''anti-drone-gun'' [30]), and capture and withhold the drone for malicious purposes. If the adversary tries to probe the IC of captured drone to retrieve critical cryptographic information, however, this attempt will irreversibly change the physical variations of IC, and finally destroy the PUF. Furthermore, ZSPs are considered as fully trusted entities and will not be compromised by the adversary. The goal of the adversary is to establish a communication with drones or ZSPs and inject malicious data/commands in the IoD environment without being detected. For example, if a drone is communicating with a ZSP for navigation information and the adversary plans to authenticate itself to the drone as a ''legitimate ZSP'', this scenario can pose a threat to government, national institutions, and assets (e.g., using drone as an improvised explosive device).

V. The Proposed Secure Data Collection and Storage Mechanism

The basic idea of *SecureIoD* is that drones and ZSPs first mutually authenticate each other and establish a secure session key based on physical unclonable function and Henon map before sharing any sensitive data via an insecure wireless channel. Then, ZSPs pack the collected data into blocks and compete to add their blocks into the blockchain based on the proposed joint Proof-of-Work (PoW) and Proof-of-Stake (PoS) consensus mechanism. *SecureIoD* is comprised of two parts: (i) mutual authentication and key establishment; and (ii) miner ZSP selection and block generation. Table I lists all notations used in this paper.

TABLE I
Notations and Their Descriptions

Notation	Description
Z_s	Identity of the *s*th ZSP
ID_i	Real identity of the *i*th drone
ts	Current timestamp
PID_i^{ts}	Pseudonym of the *i*th drone
N_i^{ts}	Random number generated by ID_i
N_s^{ts+1}	Random number generated by Z_s
(CH_i^{ts}, RE_i^{ts})	PUF CRP of ID_i
$S(\cdot)_{(CH_i^{ts}, RE_i^{ts})}$	Random shuffling with CRP
$S^{-1}(\cdot)_{(CH_i^{ts}, RE_i^{ts})}$	Reverse process of random shuffling with CRP
$C(\cdot)$	Message Authentication Code (MAC) function
$H(\cdot)$	Collision-resistant one-way hash function
$\oplus$	Bitwise XOR operation
$\parallel$	Concatenation operation
M_i	The *i*th message
MAC_i	MAC of the *i*th message
$SK_{i,s}$ or $SK_{s,i}$	Secure session key between ID_i and Z_s

A. Mutual Authentication and Key Establishment Phase

Considering the scenario that drones are deployed in a congested area where the chances of coronavirus disease (COVID-19) spreading are high. Drones help to collect and send an amount of health data (i.e., thermal images of people) to nearby ZSPs for further processing and taking appropriate action. Since maintaining confidentiality, privacy, and security of health data is of paramount importance, a drone ID_i who wishes to deliver the collected health data to a ZSP Z_s needs to first achieve mutual authentication and establish a secure session key with ZSP Z_s. The detailed steps are as follows.

1) Drone ID_i first computes its pseudonym as PID_i^{ts} = $H(ID_i \parallel RE_i^{ts})$ using its real identity ID_i and CRP response RE_i^{ts}. Then it generates a random number N_i^{ts} and calculates an encrypted message M_1, where the plain text, ($PID_i^{ts} \parallel Z_s \parallel N_i^{ts}$), will be arranged in a random shuffling using the Henon map with the CRP (CH_i^{ts}, RE_i^{ts}) as initial condition.

$$M_1 = S(PID_i^{ts} \| Z_s \| N_i^{ts})_{(CH_i^{ts}, RE_i^{ts})}.$$

It also calculates the message authentication code (MAC) MAC_1 as follows

$$MAC_1 = C(M_1 \| N_i^{ts}).$$

Finally, it sends authentication request message [M_1, MAC_1] to ZSP Z_s.

2) ZSP Z_s first tries to locate PID_i^{ts} in the database. If PID_i^{ts} is not found, the authentication request is rejected. Otherwise, it fetches the entry [ID_i, PID_i^{ts}, (CH_i^{ts}, RE_i^{ts})] for drone ID_i. Then, it retrieves $N_i^{ts'}$ from M_1 through $S^{-1}(M_1)$, which is the reverse process of shuffling with the CRP (CH_i^{ts}, RE_i^{ts}) as initial condition. With $N_i^{ts'}$, it can calculate $MAC_1^{'}$ = $C(M_1 \parallel$

$N_i^{ts'}$) and check it with the received MAC_1. If $MAC_1^{'}$ = MAC_1, the message verification succeeds. Otherwise, it discards the message. Next, it generates a random number N_s^{ts+1}, and calculates an encrypted message M_2 and MAC MAC_2 as follows

$$M_2 = S(PID_i^{ts} \| Z_s \| N_i^{ts'} \| N_s^{ts+1})_{(CH_i^{ts}, RE_i^{ts})},$$

$$MAC_2 = C(M_2 \| N_i^{ts'} \| N_s^{ts+1}).$$

Finally, it sends message [M_2, MAC_2] to drone ID_i.

3) Drone ID_i first retrieves $N_s^{ts+1'}$ from M_2 through $S^{-1}(M_2)$ and calculates $MAC_2^{'}$ as follows

$$MAC_2^{'} = C(M_2 \| N_i^{ts} \| N_s^{ts+1'}).$$

If $MAC_2^{'}$ = MAC_2, the message verification succeeds. Otherwise, it discards the message. Then, it generates a random number N_i^{ts+1} and computes its new CRP as follows

$$CH_i^{ts+1} = S(N_s^{ts+1'} \| N_i^{ts+1})_{(CH_i^{ts}, RE_i^{ts})},$$

$$RE_i^{ts+1} = F_{puf}(CH_i^{ts+1}).$$

After that, it calculates the following

$$M_3 = S(PID_i^{ts} \| Z_s \| N_s^{ts+1'} \| N_i^{ts+1})_{(CH_i^{ts}, RE_i^{ts})},$$

$$M_4 = S(PID_i^{ts} \| Z_s \| N_s^{ts+1'} \| N_i^{ts+1} \| RE_i^{ts+1})_{(CH_i^{ts}, RE_i^{ts})}$$

$$MAC_{34} = C(M_3 \| M_4 \| N_i^{ts+1} \| RE_i^{ts+1}).$$

Finally, it sends message [M_3, M_4, MAC_{34}] to ZSP Z_s, updates its CRP (CH_i^{ts+1}, RE_i^{ts+1}), and calculates the secret session key as follows

$$SK_{i,s} = H(N_i^{ts+1}) \oplus H(N_s^{ts+1'}).$$

4) ZSP Z_s first retrieves $N_i^{ts+1'}$ and $RE_i^{ts+1'}$ from M_3 and M_4 through $S^{-1}(M_3)$ and $S^{-1}(M_4)$, respectively. Then, it calculates $MAC_{34}^{'}$ as follows

$$MAC_{34}^{'} = C(M_3 \| M_4 \| N_i^{ts+1'} \| RE_i^{ts+1'}).$$

If $MAC_{34}^{'}$ = MAC_{34}, the message verification succeeds. Otherwise, it discards the message. After that, it computes drone ID_i's new CRP challenge CH_i^{ts+1} and new pseudonym PID_i^{ts+1}, and then updates the entry [ID_i, PID_i^{ts+1}, (CH_i^{ts+1}, $RE_i^{ts+1'}$)] in the database.

$$CH_i^{ts+1} = S(N_s^{ts+1} \| N_i^{ts+1'})_{(CH_i^{ts}, RE_i^{ts})},$$

$$PID_i^{ts+1} = H(ID_i \| RE_i^{ts+1'}).$$

Finally, it calculates the secret session key as follows

$$SK_{s,i} = H(N_s^{ts+1}) \oplus H(N_i^{ts+1'}).$$

At this moment, drone ID_i and ZSP Z_s have successfully verified each other's identities and established a secure session key $SK_{s,i}$ (or $SK_{i,s}$) for subsequent communications. The above process is shown in Fig. 3.

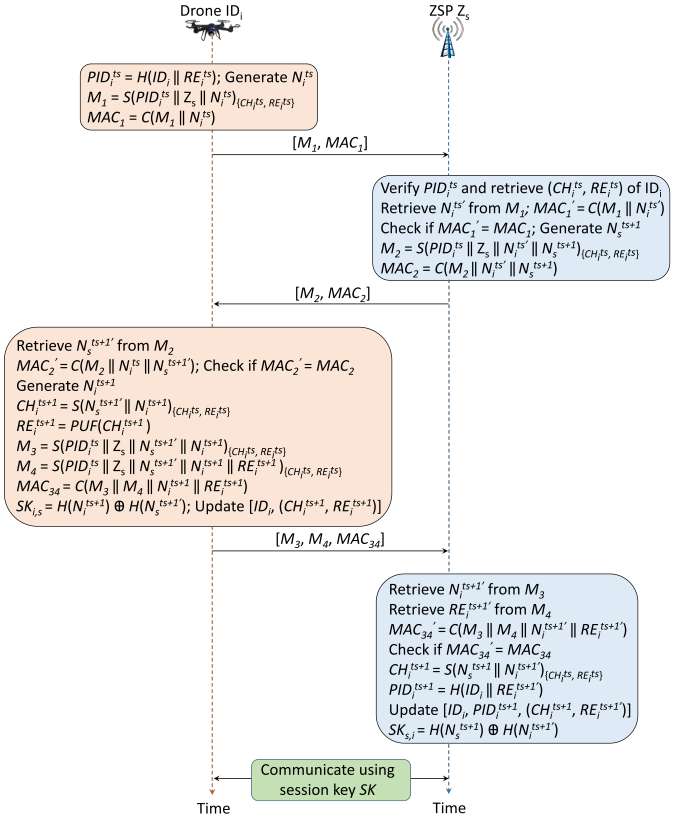


Fig. 3. Mutual authentication and secure session key establishment between drone ID_i and ZSP Z_s.

B. Miner ZSP Election and Block Generation

After collecting data from drones, ZSPs put these data into blocks and try to add them into the blockchain. However, the centralized ZSP does not exist in the decentralized network structure to manage the blockchain. In addition, many ZSPs may attempt to add a new block in the blockchain simultaneously, thus a miner ZSP should be elected from all ZSPs competitively. In this paper, we propose a joint Proof-of-Work (PoW) and Proof-of-Stake (PoS) consensus mechanism to select the miner ZSP. The basic idea is that ZSPs take the number of transactions in the block as the stake to determine the difficulty to complete the PoW, In other words, the more transactions are in the block, the easier a ZSP can solve the cryptographic puzzle. The rationale behind this design is that the block containing more transactions can be added in the blockchain earlier and quicker, as it affects the whole ledger significantly and timely. To be specific, a ZSP needs to find a hash value satisfying the following target criterion so that it can become the miner

$$H(ZSP_{ID}, ts, prevHash, nonce) \geq Hash_{ID}^{th}, \quad (1)$$

where the ZSP ID, ZSP_{ID}, the current timestamp, *ts*, the previous block's hash value, *prevHash*, as well as the *nonce* are used to calculate the block's hash value.

$Hash_{ID}^{th}$ is the hash threshold of ZSP ZSP_{ID} and can be adjusted to control the difficulty level of cryptographic puzzle or the block generation speed. To be specific, $Hash_{ID}^{th}$ is a

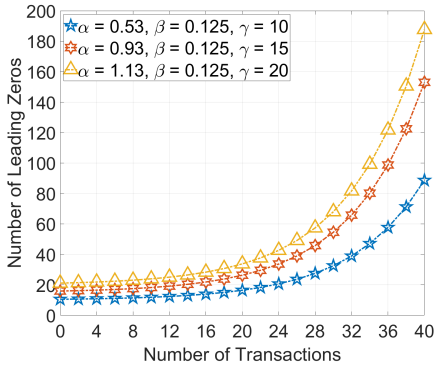


Fig. 4. Change of the number of leading zeros (N_{stake}) against the number of transactions (N_{trans}) in the block.

series of binary bits starting with a set of continuous zeros, which depends on the number of transactions N_{trans} in the block. Here, $Hash_{ID}^{th}$ is defined as follows

$$Hash_{ID}^{th} = concat(zeros(N_{stake}), Tgt^{th}), \tag{2}$$

where $zeros(N_{stake})$ and Tgt^{th} are the N_{stake} number of leading zeros in $Hash_{ID}^{th}$ and the random numbers following the N_{stake} number of leading zeros in $Hash_{ID}^{th}$, respectively. $concat(x, y)$ is the concatenation function that concatenates both binary numbers x and y as x + y. For example,

$$\begin{aligned} Hash_{ID}^{th} &= concat(zeros(3), 101) \\ &= concat(000, 101) \\ &= 000101. \end{aligned}$$

In addition,

$$N_{stake} = [\gamma + \alpha \cdot e^{N_{trans} \cdot \beta}], \tag{3}$$

$$Tgt^{th} = rand(2^{N_{hash} - N_{stake}} - 1), \tag{4}$$

where N_{hash} is the total number of bits of the hash value (i.e., N_{hash} is 256 for SHA-256) and $rand()$ is the random number generation function. [.] returns the integral part of the value, and α, β and γ are system parameters. Fig. 4 shows the change of the number of leading zeros (N_{stake}) against the number of transactions (N_{trans}) in the block. When the number of transactions in the block increases, the number of leading zeros also increases. In that case, a ZSP can find a nonce which meets the target criterion Eq. (1) quicker.

The ZSP will be selected as the miner ZSP to add its block into the blockchain if it is the first one who finds a *nonce* that satisfies the target criterion Eq. (1). The miner ZSP packs the transactions and other basic information into a block, and then distributes the block along with its digital signature to all other ZSPs through P2P network. In each transaction block, the number of transactions in the block body is used to verify the hash threshold according to Eq. (2) and (3). The ZSP ID, the timestamp of block generation, the hash of previous block, and the nonce in the block header are used to prove the validity of the miner ZSP. When other ZSPs receive the newly generated block, they will check whether the block satisfies the target criterion Eq. (1) and the piggybacked digital signature is valid. If both verification succeed, other ZSPs will add the

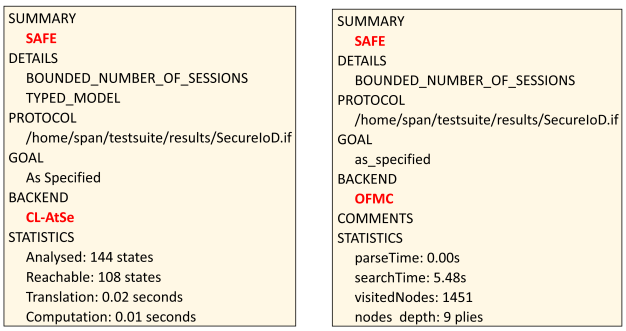


Fig. 5. Results of security verification using CL-AtSe and OFMC back-ends in AVISPA.

block into their blockchains. Otherwise, the newly generated block will be discarded and a different miner will be elected. When a ZSP starts receiving multiple blocks concurrently, the fork process will be initiated by the blockchain. In this paper, a distributed algorithm in [31] is being adopted. The basic idea is that each ZSP chooses one fork and continues to add new blocks after it. As time passed, the fork acknowledged by the largest number of ZSPs grows faster than others. Finally, the longest one becomes the distributed consensus of the network, while other forks are discarded.

VI. Security Analysis

A. Security Verification Using AVISPA and Scyther

We first evaluate *SecureIoD* using the Automated Validation of Internet Security Protocols and Applications (AVISPA) tool [15] to check whether it is secure against man-in-the-middle attacks and replay attacks. Security professionals can implement their security protocol along with security features in the High-Level Protocol Specification Language (HLPSL) and then verify its security performance in AVISPA. In addition, AVISPA integrates two major back-ends: On-the-fly Model-Checker (OFMC) and Constraint-Logic-based Attack Searcher (CL-AtSe). The OFMC can be used not only for the detection of potential attacks, but also for proving the correctness of protocol for a bounded number of sessions. The CL-AtSe is able to translate security protocol specification written in the intermediate format into a group of constraints which can be used to discover potential attacks on protocols.

We implement the mutual authentication and key establishment phase in HLPSL. In the implementation, there are two basic roles: drone and ZSP. In addition to these two basic roles, the other four mandatory roles, such as session, goal, environment, and intruder roles, are also implemented for the security analysis of *SecureIoD* in AVISPA. Finally, we set up a complete and fully functional SPAN+AVISPA [32] on Ubuntu 10.04 which is running in Virtual Box [33]. The results of security verification are shown in Fig. 5. As we can see that *SecureIoD* is secure against replay attacks and man-in-the-middle attacks. We also use the Semantics and Verification of Security Protocols (Scyther) tool [16] to further evaluate the security of *SecureIoD*. As shown in Fig. 6, *SecureIoD* can satisfy all security requirements. The HLPSL security

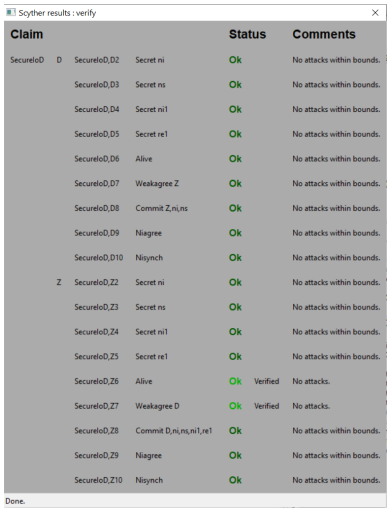


Fig. 6. Result of security verification using Scyther.

verification programs and the Scyther security verification program are available at *https://github.com/congpu/SecureIoD*.

B. Security Analysis

This subsection exhibits that *SecureIoD* is secure against drone capture attack, drone impersonation attack, message modification attack, and ZSP spoofing attack.

Drone Capture Attack: We assume that an adversary has successfully captured drone ID_i who is communicating with ZSP Z_s using session key $SK_{i,s}$. Through physical memory disclosure attacks, the adversary can retrieve stored information such as drone's real identity ID_i and session key $SK_{i,s}$. As a result, the current communication session between drone ID_i and ZSP Z_s would be compromised by the adversary with the obtained session key $SK_{i,s}$ In order to compromise future communication with ZSP Z_s, the adversary has to obtain valid CRP (CH_i^{ts+1}, RE_i^{ts+1}) from drone ID_i. This is because a new session key (i.e., requiring new random numbers and random shuffling with new CRP) will be generated for each communication session. Thus, the adversary may try to probe or alter the integrated circuit of drone ID_i to retrieve CRP (CH_i^{ts+1}, RE_i^{ts+1}). However, the adversary can only obtain the CRP challenge CH_i^{ts+1}, since the CRP response RE_i^{ts+1} is dynamically calculated via $F_{puf}(CH_i^{ts+1})$. In addition, this probing or alteration attempt will irreversibly modify the slight physical variations of integrated circuit, which in turn destroys the PUF. Thus, the adversary cannot obtain the valid CRP (CH_i^{ts+1}, RE_i^{ts+1}) to compromise future communication. Last but not least, since each drone will use a different secret session key to communicate with ZSP Z_s, the communication sessions between other non-captured drones and ZSP Z_s are still secure. In summary, *SecureIoD* is secure against drone capture attack.

Drone Impersonation Attack: We assume that an adversary tries to masquerade as legitimate drone ID_a to communicate with ZSP Z_s for malicious purposes. First, the adversary has to send an authentication request, [M_1, MAC_1], to ZSP Z_s. The adversary can easily generate a random number N_a^{ts}. However, it cannot shuffle and calculate valid M_1 which can be correctly decoded by ZSP Z_s. This is because the adversary does not have the valid CRP (CH_a^{ts}, RE_a^{ts}) which is stored in ZSP Z_s's database. Thus, the authentication request will be rejected by ZSP Z_s and the adversary cannot establish a valid communication with ZSP Z_s by impersonating legitimate drone ID_a. In summary, *SecureIoD* is secure against drone impersonation attack.

Message Modification Attack: We assume that an adversary captures and modifies the message, either M_1, M_2, M_3 or M_4, transmitting between drone ID_i and ZSP Z_s. Since both drone ID_i and ZSP Z_s will verify the received message by checking the piggybacked MAC, i.e., $MAC_2 \stackrel{?}{=} MAC_2^{'}$, they can easily detect any modification of messages. In summary, *SecureIoD* is secure against message modification attack.

ZSP Spoofing Attack: We assume that an adversary already captured the message [M_1, MAC_1] and attempts to imitate legitimate ZSP Z_a to communicate with drone ID_i. Since the adversary does not have the valid CRP (CH_i^{ts}, RE_i^{ts}), thus, it cannot retrieve the random number N_i^{ts} piggybacked in the M_1 correctly. The adversary can make up a random number to generate the message [M_2, MAC_2]. However, when drone ID_i receives the message [M_2, MAC_2], it can easily detect the misbehavior through checking whether MAC_2 equals to $MAC_2^{'}$ and reject the following communication. In summary, *SecureIoD* is secure against ZSP spoofing attack.

VII. Performance Evaluation

A. Experimental Testbed and Benchmarks

We build a real-world testbed which is composed of one HP ENVY Notebook laptop [17] and one Latte Panda development board [18]. The Latte Panda development board is attached to a power bank through a specific plastic holder which is made with 3D printer. The power bank is capable enough to support the Latte Panda development board for hours. In terms of testbed specifications, the HP ENVY Notebook laptop is running a 64-bit Windows 10 Pro operating system, and its central processing unit (CPU) is the 7th Generation Intel Core i7-7500U processor, 4M Cache, up to 3.5 GHz. The Latte Panda development board runs a full version of Windows 10, and has Intel Cherry Trail Z8350 Quad Core processor, 2M cache, up to 1.92 GHz, and 4GB random-access memory (RAM). The developed real-world testbed is shown in Fig. 7, where the laptop is used to simulate the ZSP while the Latte Panda development board is used to mimic the drone. We implement *SecureIoD* and benchmark schemes in the Eclipse environment [34] which is set up in the Latte Panda development board as well as the laptop.

According to [35], we implement the PUF as a 256-bit hash function [36]. In addition, the random shuffling function is implemented as follows. First, the to-be-shuffled message is represented as an array. Second, the CRP pair (CH_i^{ts}, RE_i^{ts}) is used as the initial condition of Henon map to generate a sequence of points. Third, starting from the first point in the sequence, the coordinates of a point are converted into an unique integer, indicating the new location where the first element of array is to be put in the output array. Now considering the

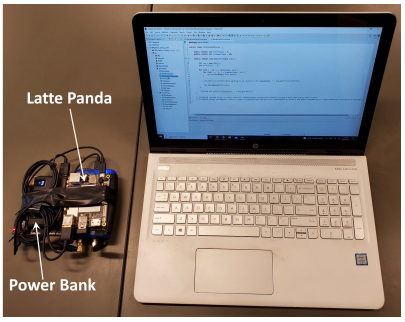


Fig. 7. Real-world testbed with one HP ENVY Notebook laptop and one Latte Panda development board.

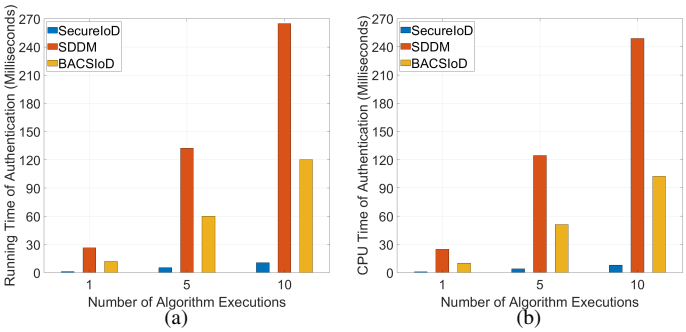


Fig. 8. The performance of running time and CPU time of authentication against the number of algorithm executions.

array element from the second to the last, the abovementioned process is repeated till the last array element is shuffled. Finally, the output array contains the shuffled message. Please note that the random shuffling function is executed differently in every communication session. This is because the drone will compute a new CRP pair during the process of mutual authentication and session key establishment. Since the CRP pair is used as the initial condition of Henon map (a = 1.4 and b = 0.3) and a minor change of initial condition in the Henon map will cause the generation of distinct sequence of points, the random shuffling operation is performed differently in every communication session.

We revisit prior approaches, SDDM [19] and BACSIoD [20], and implement them to work in the testbed for performance comparison and analysis. The original idea of these two benchmark schemes are briefly discussed in the following:

- SDDM [19]: In SDDM, the drone and the user first mutually authenticate each other through AKA scheme [37]. Then, a forger drone is selected based on an utility function using Game theory to create blocks which will be verified by other normal drones. Finally, the forger node computes the hash value of the block using the PoS mechanism, and then broadcasts the hash value to the distributed network. The other drones will verify and validate the hash value which was computed by the PoS mechanism with the hash value computed by Merkle tree.
- BACSIoD [20]: In BACSIoD, all system parameters such as the elliptic curve and its base point, private and public keys of the trusted control room, and so on are set up in the initialization phase. Then, the control room registers each drone and ground station server during the registration phase. Before sharing any information, the drone and the ground station server mutually authenticate each other and establish a session key through elliptic curve cryptography. Finally, the ground station server creates various transactions and forms the block which will be added in the blockchain through Ripple Protocol Consensus Algorithm (RPCA) [21].

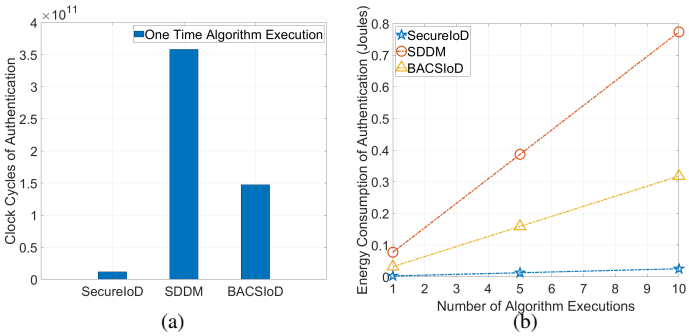
We measure the performance of *SecureIoD*, SDDM, and BACSIoD in terms of running time, CPU time, the number of clock cycles, energy consumption, and communication cost by changing the number of algorithm executions and the number of added blocks.

- Running Time: Running time is the elapsed time from when the algorithm starts running to when the algorithm finishes running.
- CPU Time: CPU time (or processing time) is the amount of time for which the CPU is used for processing instructions of the algorithm[2].
- The Number of Clock Cycles: The number of clock cycles is measured as the number of electronic pulses during running the algorithm.
- Energy Consumption: Energy consumption is measured as the amount of electronic power consumed during running the algorithm.

B. Experimental Results and Analysis

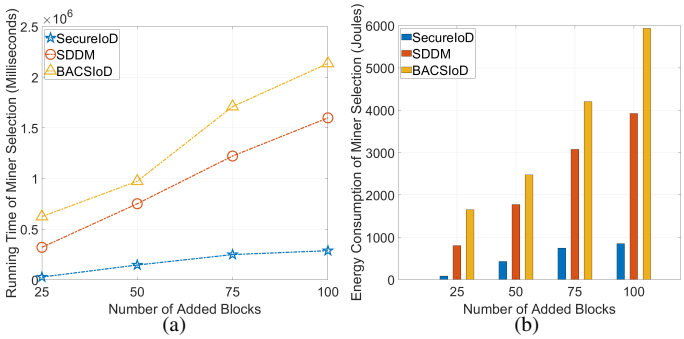
First, we measure the running time and CPU time of authentication by changing the number of algorithm executions in Fig. 8. As shown in Fig. 8(a), the overall running time of *SecureIoD*, SDDM, and BACSIoD increase as the number of algorithm executions increases. It is very straightforward that a larger number of algorithm executions takes a longer time to run and finish. *SecureIoD* shows the lowest running time with varying number of algorithm executions compared to SDDM and BACSIoD. This is because *SecureIoD* employs lightweight cryptographic operations such as one-way hash function and random shuffling. Most importantly, these cryptographic operations are executed less number of times. As a result, a less amount of running time is obtained by *SecureIoD*. SDDM adopts AKA scheme [37] as the authentication scheme. However, in AKA, the one-way hash function is used twenty-four times (i.e., the user, the drone, and the ground station execute ten, seven, and seven hash operations, respectively). In BACSIoD, elliptic curve cryptography is used to achieve mutual authentication between the drone and the ground station. Compared to AKA in SDDM, a lower running time is obtained by elliptic curve cryptography. However, *SecureIoD* still outperforms BACSIoD because less number of hash functions and lightweight shuffling operation are executed by *SecureIoD* . In Fig. 8(b), we measure the CPU time of authentication with varying number of algorithm executions. Overall, the best performance belongs to *SecureIoD*, where

[2]As opposed to running time, CPU time does not include waiting for input/output (I/O) operations or entering low-power (idle) mode.



(a) (b)

Fig. 9. The performance of the number of clock cycles and energy consumption of authentication against the number of algorithm executions.



(a) (b)

Fig. 10. The performance of running time and energy consumption of miner selection against the number of added blocks.

lightweight cryptographic operations are employed to implement the authentication scheme.

Second, we measure the number of clock cycles and energy consumption of authentication against the number of algorithm executions in Fig. 9. In Fig. 9(a), it is shown that the smallest number of clock cycles is obtained by *SecureIoD* when the authentication algorithm is executed one time. The rationale is that *SecureIoD* requires less number of operations during the authentication process, thus, a smaller number of clock cycles is measured by *SecureIoD* compared to SDDM and BACSIoD. SDDM delivers the largest number of clock cycles since the AKA scheme needs to execute more hash operations than *SecureIoD* and BACSIoD. In Fig. 9(b), the energy consumption of authentication are measured for all three schemes by changing the number of algorithm executions. When we increase the number of algorithm executions from one to ten, the energy consumption of *SecureIoD*, SDDM, and BACSIoD increases linearly. Since the algorithm is executed more times, a larger amount of energy resource is needed. However, the lowest energy consumption is still observed by *SecureIoD*. This is because *SecureIoD* employs lightweight cryptographic operations such as one-way hash function and random shuffling, which consumes less amount of energy.

Third, we measure the running time and energy consumption of miner selection with varying number of added blocks in Fig. 10. As shown in Fig. 10(a), the running time of miner selection for all three schemes increases as the number of added blocks increases. In *SecureIoD*, we propose a joint PoW and PoS consensus mechanism to select the miner ZSP. If there are more transactions in the block, the ZSP has more chances to be selected as the miner to add the block in the blockchain. Thus, a less amount of running time is observed for the selection of miner ZSP in *SecureIoD*. In SDDM, a forger node selection algorithm is proposed to select the miner drone. However, the miner drone selection algorithm is designed based on an utility function using Game theory, which takes more time for a drone to become the miner. In BACSIoD, Ripple Protocol Consensus Algorithm (RPCA) is adopted as the consensus mechanism in the blockchain. However, RPCA is more complex than the joint PoW and PoS consensus mechanism in *SecureIoD* and the forger node selection algorithm in SDDM. Thus, the largest running time of miner selection is obtained by BACSIoD. In Fig. 10(b), we measure the energy

TABLE II
COMPARISON OF COMMUNICATION COST

Metrics	*SecureIoD*	SDDM	BACSIoD
Number of Messages	3	7	3
Energy Consumption (joule)	3.38×10^{-4}	7.88×10^{-4}	3.38×10^{-4}

consumption of miner selection for *SecureIoD*, SDDM, and BACSIoD. Clearly, *SecureIoD* still outperforms SDDM and BACSIoD in terms of energy consumption of miner selection.

Finally, we measure the communication cost in terms of the number of messages and energy consumption of communication in Table II. To mutually authenticate drone and user and establish a session key, AKA scheme in SDDM requires seven messages to be exchanged among drone, control server, and user. To be specific, a drone and an user first exchange two messages with the control server during the registration phase, respectively. Then, the user sends an authentication request message to the control server through a public channel. After receiving the authentication request message from the user, the control server verifies the message and sends the message to drone via a public channel. Finally, the drone checks the validation of message and sends a message to user if the verification succeeds. In BACSIoD, the drone first sends a message piggybacked with a set of critical information (i.e., a hash value and certificate) to the ground station. After verifying the received certificate, the ground station generates a secret hash value and sends it back to the drone. Finally, the drone replies an acknowledge message back to the ground station. In *SecureIoD*, as shown in Fig. 3, only three messages are required to achieve mutual authentication and secure session key agreement. First, a drone sends an authentication request message piggybacked with its pseudonym, PUF response, and random number to ZSP. Then, ZSP verifies the received message and replies a message with one random number. Lastly, the drone forwards a message with the updated CRP to ZSP. By this time, the mutual authentication is completed and the secure session key has been established between drone and ZSP. In addition, the energy consumption of communication for *SecureIoD*, SDDM, and BACSIoD is 3.38×10^{-4} joules, 7.88×10^{-4} joules, and 3.38×10^{-4} joules, respectively. Here, the energy consumption of communication is measured based on the number of sent and received messages [38]. It is clear that *SecureIoD* has a lower communication overhead compared to other two schemes.

VIII. Conclusion

In this paper, a secure data collection and storage mechanism, called *SecureIoD*, was proposed for the IoD environment. The basic idea of *SecureIoD* is that drones and ZSPs first mutually authenticate each other and establish a secure session key before sharing any sensitive data via an insecure wireless channel. After that, ZSPs pack the collected data into blocks and compete to add their blocks into the blockchain. We also proposed a joint PoW and PoS consensus mechanism to select the miner ZSP, where a ZSP that has more transactions in the block can solve the cryptographic puzzle easier. We verified the security of *SecureIoD* through specific security protocol verification tools (i.e., AVISPA and Scyther) and security analysis. The verification and analysis results showed that *SecureIoD* is a secure protocol. Most importantly, it is immune against various types of security attacks. In addition, we developed a real-world testbed consisting of a Latte Panda development board and a laptop, conducted extensive simulation experiments, and compared *SecureIoD* with prior benchmark schemes for performance evaluation and analysis. The experimental results showed that *SecureIoD* provides better performance in terms of running time, CPU time, clock cycle, and energy consumption.

Acknowledgment

This research was supported by NASA West Virginia EPSCoR Grant # 80NSSC20M0055, West Virginia Research Higher Education Policy Commission's Division of Science and Research Grant # dsr.20.1698-001, and Marshall University Faculty Senate Research Committee Funding.

References

[1] C. Feng, K. Yu, A. Bashir, Y. Al-Otaibi, Y. Lu, S. Chen, and D. Zhang, "Efficient and Secure Data Sharing for 5G Flying Drones: A Blockchain-Enabled Approach," *IEEE Network*, vol. 35, no. 1, pp. 130–137, 2021.

[2] J. Pozner, *A Comprehensive List of Commercial Drone Use Cases*, 2020, https://www.dronegenuity.com/commercial-drone-use-cases-comprehensive-list.

[3] A. Lemert, M. Meyer, R. Mills, J. Paine, and A. Wong, "Drone Policy Overview," *Purdue Policy Research Institute Policy Briefs*, vol. 4, no. 1, p. 5, 2018.

[4] C. Pu, A. Wall, K. Choo, I. Ahmed, and S. Lim, "A Lightweight and Privacy-Preserving Mutual Authentication and Key Agreement Protocol for Internet of Drones Environment," *IEEE Internet of Things Journal*, pp. 1–1, 2022.

[5] A. Gameros, *The Use of Composite Materials in Unmanned Aerial Vehicles*, https://www.azom.com/article.aspx?ArticleID=12234.

[6] S. Zaidi, M. Atiquzzaman, and C. Calafate, "Internet of Flying Things (IoFT): A survey," *Computer Communications*, vol. 165, pp. 53–74, 2021.

[7] C. Pu and L. Carpenter, "Psched: A Priority-Based Service Scheduling Scheme for the Internet of Drones," *IEEE Systems Journal*, vol. 15, no. 3, pp. 4230–4239, 2021.

[8] A. Kumar, K. Sharma, H. Singh, S. Naugriya, S. Gill, and R. Buyya, "A drone-based networked system and methods for combating coronavirus disease (COVID-19) pandemic," *Future Generation Computer Systems*, vol. 115, pp. 1–19, 2021.

[9] J. Zhang and Y. Zhang, "A Method for UAV Reconnaissance and Surveillance in Complex Environments," in *Proc. IEEE ICCAR*, 2020, pp. 482–485.

[10] V. Chamola, V. Hassija, V. Gupta, and M. Guizani, "A Comprehensive Review of the COVID-19 Pandemic and the Role of IoT, Drones, AI, Blockchain, and 5G in Managing its Impact," *IEEE Access*, vol. 8, pp. 90 225–90 265, 2020.

[11] J. Yaacoub and O. Salman, "Security Analysis of Drones Systems: Attacks, Limitations, and Recommendations," *Internet of Things*, vol. 11, p. 100218, 2020.

[12] W. Stallings, *Cryptography and Network Security - Principles and Practices, 8th Edition*. Pearson, 2020.

[13] M. Ozmen and A. Yavuz, "Dronecrypt - An Efficient Cryptographic Framework for Small Aerial Drones," in *Proc. IEEE MILCOM*, 2018, pp. 1–6.

[14] C. Li, L. Guan, J. Lin, B. Luo, Q. Cai, J. Jing, and J. Wang, "Mimosa: Protecting Private Keys against Memory Disclosure Attacks using Hardware Transactional Memory," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 3, pp. 1196–1213, 2021.

[15] *Automated Validation of Internet Security Protocols and Applications*, http://www.avispa-project.org/.

[16] *The Scyther Tool*, https://people.cispa.io/cas.cremers/scyther/index.html.

[17] *Laptop Computers*, https://www.hp.com/us-en/shop/cat/laptops.

[18] *Latte Panda*, https://www.lattepanda.com/.

[19] S. Aggarwal, M. Shojafar, N. Kumar, and M. Conti, "A New Secure Data Dissemination Model in Internet of Drones," in *Proc. IEEE ICC*, 2019, pp. 1–6.

[20] B. Bera, D. Chattaraj, and A. Das, "Designing secure blockchain-based access control scheme in IoT-enabled Internet of Drones deployment," *Computer Communications*, vol. 153, pp. 229–249, 2020.

[21] D. Schwartz, N. Youngs, A. Britto *et al.*, "The Ripple Protocol Consensus Algorithm," *Ripple Labs Inc White Paper*, vol. 5, no. 8, p. 151, 2014.

[22] M. Singh, G. Aujla, and R. Bali, "A Deep Learning-Based Blockchain Mechanism for Secure Internet of Drones Environment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4404–4413, 2021.

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

[24] C. Ge, X. Ma, and Z. Liu, "A semi-autonomous distributed blockchain-based framework for UAVs system," *Journal of Systems Architecture*, vol. 107, p. 101728, 2020.

[25] X. Xu, H. Zhao, H. Yao, and S. Wang, "A Blockchain-enabled Energy Efficient Data Collection System for UAV-assisted IoT," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2431–2443, 2021.

[26] P. Mehta, R. Gupta, and S. Tanwar, "Blockchain envisioned UAV networks: Challenges, solutions, and comparisons," *Computer Communications*, vol. 151, pp. 518–538, 2020.

[27] G. Bansal, N. Naren, V. Chamola, B. Sikdar, N. Kumar, and M. Guizani, "Lightweight Mutual Authentication Protocol for V2G Using Physical Unclonable Function," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 7, pp. 7234–7246, 2020.

[28] M. Hénon, *A Two-Dimensional Mapping with A Strange Attractor*. Springer, 1976.

[29] B. Hammi, A. Fayad, R. Khatoun, S. Zeadally, and Y. Begriche, "A Lightweight ECC-Based Authentication Scheme for Internet of Things (IoT)," *IEEE Systems Journal*, vol. 14, no. 3, pp. 3440–3450, 2020.

[30] *Dronebuster*, Accessed on: June 07, 2020, http://flexforce.us/product/dronebuster/.

[31] Z. Yang, K. Yang, L. Lei, K. Zheng, and V. Leung, "Blockchain-Based Decentralized Trust Management in Vehicular Networks," *IEEE Internet Things Journal*, vol. 6, no. 2, pp. 1495–1505, 2018.

[32] *SPAN*, http://www.avispa-project.org/.

[33] *VirtualBox*, https://www.virtualbox.org/.

[34] *Eclipse*, https://www.eclipse.org/downloads/.

[35] J. Wallrabenstein, "Practical and Secure IoT Device Authentication Using Physical Unclonable Functions," in *Proc. IEEE FiCloud*, 2016, pp. 99–106.

[36] *Java Security Standard Algorithm Names*, https://docs.oracle.com/en/java/javase/12/docs/specs/security/standard-names.html.

[37] Y. Zhang, D. He, L. Li, and B. Chen, "A lightweight authentication and key agreement scheme for internet of drones," *Computer Communications*, vol. 154, pp. 455–464, 2020.

[38] C. Pu and S. Lim, "A Light-Weight Countermeasure to Forwarding Misbehavior in Wireless Sensor Networks: Design, Analysis, and Evaluation," *IEEE Systems Journal*, vol. 12, no. 1, pp. 834–842, 2018.