Sliding Window Blockchain Architecture for Internet of Things

Prescilla K, *Associate Member, IEEE,* Sarath Babu, *Graduate Student Member, IEEE,* and B. S. Manoj, *Senior Member, IEEE.*

***Abstract*—Internet of Things (IoT) refers to the concept of enabling Internet connectivity and associated services to non-traditional computers formed by integrating essential computing and communication capability to physical things for everyday usage. Security and privacy are two of the major challenges in IoT. The essential security requirements of IoT cannot be ensured by the existing security frameworks due to the constraints in CPU, memory, and energy resources of the IoT devices. Also, the centralized security architectures are not suitable for IoT because they are subjected to single point of attacks. Defending against targeted attacks on centralized resources is expensive. Therefore, the security architecture for IoT needs to be decentralized and designed to meet the limitations in resources. Blockchain is a decentralized security framework suitable for a variety of applications. However, blockchain in its original form is not suitable for IoT, due to its high computational complexity and low scalability. In this paper, we propose a sliding window blockchain (SWBC) architecture that modifies the traditional blockchain architecture to suit IoT applications. The proposed sliding window blockchain uses previous $(n-1)$ blocks to form the next block hash with limited difficulty in Proof-of-Work. The performance of SWBC is analyzed on a real-time data stream generated from a smart home testbed. The results show that the proposed blockchain architecture increases security and minimizes memory overhead while consuming fewer resources.**

***Index Terms*—Blockchain, Internet of Things, smart home, security, sliding window**

I. Introduction

Blockchain is a distributed ledger used to record transactions between two or more parties. Unlike relational database systems, blockchain is a data structure where new entries get appended at the end of the ledger, and there exist no administrator permissions within a blockchain which allow modification of the data. Also, the addition of a new block to the chain needs to be verified by all other parties through a consensus algorithm. Since there exists a distributed control over the blockchain, it is difficult for attackers to modify the data compared to a relational database system. Relational databases are primarily designed for centralized data storage and blockchain are specifically designed for decentralized data storage. There exist two types of blockchains: (*i*) permissioned and (*ii*) permissionless. A permissioned blockchain is a private blockchain which requires pre-verification of the participants within the network who are assumed to know each other whereas, a permissionless blockchain is a public blockchain [1]. Traditional blockchain approach is not suitable for IoT with real-time data streams due to their computationally complex Proof-of-Work (PoW) [2]. As the computational time increases, blockchain security becomes infeasible to be used for IoT.

Prescilla K, Sarath Babu, and Manoj. B. S. are with the Department of Avionics, Indian Institute of Space Science and Technology, Thiruvananthapuram, India 695547.

The two major challenges involved in applying blockchain to IoT environments include: (*i*) computational complexity and (*ii*) scalability. The computational complexity depends on difficulty level and Merkle tree size. Merkle tree is a tree in which every leaf node is labeled with the hash of a transaction data and every non-leaf node is labeled with the cryptographic hash of the labels of its child nodes. Merkle tree grows with the number of transactions made and, thereby, increasing the time consumed for Proof-of-Work, which is less favorable for an IoT network.

Scalability refers to the limits on the number of transactions a blockchain can process within a specific time period. Bitcoin is a popular example of a blockchain. Bitcoin blockchain is a payment system that does not rely on a central authority to secure and control its money supply. Each block in a Bitcoin blockchain has limited block size. In Bitcoin, the block size is limited to 1 MB and a block is mined every ten minutes. Interestingly, the existing literature [3] suggests blockchain as one of the data security and privacy algorithms that can be implemented for IoT applications due to its distributed architecture.

In this paper, we propose a new blockchain architecture for IoT environments, especially in the context of smart home applications. A smart home monitors, analyzes, and reports the state of the home. Smart homes use devices connected to IoT to automate and monitor in-home systems [4]. Smart home can be considered as the smallest unit of a smart city. The security standardization of a smart home supports a smart city and vice versa.

In a smart home, the real-time data streams are generated by sensors which help us to monitor the current status of the home, analyze energy consumption, and investigate any accidents inside a smart home. The volume of data generated by a smart home depends on the number of sensors deployed and the frequency of data acquisition. Therefore, proper sampling of sensor data is required to produce meaningful information which can be later stored in the blockchain. The volume of data stored in a blockchain decides the packet overhead, memory overhead, and computational overhead. In this context, our proposed sliding window blockchain architecture tries to improve the security and reduce the memory overhead of IoT in a smart home environment.

A. Motivation and contributions

IoT is revolutionizing the living environments, therefore, it is necessary to provide security for the data generated from IoT. However, the limitations of CPU, memory, and energy resources of IoT devices make the traditional centralized security algorithms infeasible. Therefore, we propose a sliding window blockchain, which is a decentralized security architecture, that provides security while considering the limitations of IoT devices.

The contributions of our research work are as follows:

1) A novel Sliding Window Blockchain architecture for IoT is proposed to provide security while considering the limitations of IoT devices.
2) A smart home testbed is set up to implement and analyze the performance of the proposed architecture.
3) Analysis of the performance and security of the proposed sliding window blockchain architecture is carried out on the smart home testbed.

The rest of the paper is organized as follows: Section II briefly explains the work related to the blockchain approach in IoT. Section III is a preliminary section which gives an introduction to blockchain. Section IV defines the problem statement. Section V explains the proposed sliding window blockchain architecture. Sections VI and VII describe the experimental setup and performance analysis, respectively, of sliding window blockchain in a smart home environment. Finally, Section VIII concludes the paper along with the future scope.

II. Related Work

The concept of smart homes plays an important role in the planning of future housing-based models of health care [5]. Smart homes use IoT as their ambient networking environment. IoT is a network of things embedded with sensors and are connected to the Internet [6]. IoT helps to connect the resources of a smart home, both physical and virtual things, that are embedded with electronics, sensors, actuators, and software, to collect and exchange data [7], [8].

MavHome [9] is one of the earliest smart home projects, which created a home that acts as a rational agent. The agent seeks to maximize inhabitant comfort and minimize the operational cost. To achieve the goals, the agent predicts the mobility pattern and the device usage of the inhabitants.

As the smart home concept becomes important, it is essential to provide an adequate level of protection against cyber-attacks for residential customers. Traditional security solutions tend to be expensive for IoT in terms of processing and memory overhead due to the limited computational power and memory size of IoT devices [10]. Therefore, the resource-constrained nature of IoT devices involved in a smart home makes the standard security solutions infeasible. As a result, smart homes are prone to security vulnerabilities. The major challenges in adopting conventional security mechanisms in IoT include: (*i*) resource constraints, (*ii*) heterogeneous communication protocols, (*iii*) unreliable communications, and (*iv*) energy constraints [11].

Qu *et al.* [10] identified that the traditional security and privacy policies based on asymmetric encryption schemes are difficult to implement in an IoT environment due to its centralized key management system. In such a context, blockchain technologies help to track, coordinate, carry out transactions, and store information from a large number of devices and, thereby, enabling the creation of applications that do not require a centralized cloud. Blockchain forms a decentralized network which enables all parties to make transactions.

The blockchain approach has been widely applied in fields including finance [12], insurance [13], manufacturing [14], and health-care [15]. Kshetri demonstrated that blockchain based identity and access management systems have the ability to significantly strengthen IoT security [16]. Dorri *et al.* [17] proposed a hierarchical architecture that uses a centralized private immutable ledger operating at local IoT network level within a smart home to reduce the overhead and a decentralized public blockchain at higher-end devices for better trust. In [18], Shen *et al.* applied blockchain to a smart home system to ensure the security and privacy of information. Christidis *et al.* [19] used smart contracts in IoT to facilitate the sharing of service resources as well as to automate the process in a cryptographically verifiable manner. Table I provides a comparison of different blockchain architectures proposed for IoT applications.

From the literature, it is found that the difficulty level of Proof-of-Work that is applicable for IoT is not experimentally analyzed heretofore and the feasibility of a Merkle tree for IoT is not studied yet in the literature. Our paper brings more insights on the above-mentioned areas.

III. Preliminaries

A blockchain is a growing list of records, called blocks, that are linked using cryptographically generated hashes. Figure 1 shows the basic architecture of a simplified blockchain. Each block contains a cryptographic hash of the previous block, chaining the blocks together. Chaining blocks together makes it impossible to modify transactions included in any block without modifying all subsequent blocks. As a result, the cost of modifying a particular block increases with every new block added to the blockchain, magnifying the effect of the Proof-of-Work.

A block consists of timestamp, hash of the previous block, nonce (value representing the iteration for which the Proof-of-Work gets solved), and the transaction data (represented as a Merkle tree root). The first block of a blockchain is called genesis block and has no previous block hash. Each block in the blockchain is added through the process of mining (Proof-of-Work) which validates whether the transactions are legal. When the majority of the miners validate the block, a consensus is sent to miners to add the block to the blockchain.

IV. Problem Statement

Blockchain uses an immutable distributed ledger which replicates a copy of the ledger across the authorized parties and, thereby, preventing the single point of vulnerability

Table I: Comparison of existing blockchain architectures for IoT.

Blockchain Architectures for IoT	**Pros**	**Cons**
Guy Zyskind *et al.* [20]	Unlike blockchains, in Enigma [20], the computations, and data storage are not replicated by every node in the network. Only a small subset performs each computation over different parts of the data. The decreased redundancy in storage and computations enable more demanding computations.	The feature of Enigma which restricts the access to data in its entirety is not suitable for heterogeneous IoT data.
B. Liu *et al.* [21]	Addresses the data integrity concerns in the cloud storage service using blockchain.	The data integrity service based on blockchain is a public network and it requires gas (fee) to trade.
A. Bahga *et al.* [22]	Industrial machines are part of the blockchain network considered in this work. Here, the blockchain manages the ledger that keeps track of hardware, software, platform, and infrastructure services to the users.	Privacy concerns of the blockchain are not addressed.
A. Boudguiga *et al.* [23]	Blockchain infrastructure is used to secure the deployment updates for IoT devices.	The performance analysis of the proposed architecture was not carried out.
DiPietro *et al.* [24]	Designed a new blockchain named obligation chain, which is linked to another main blockchain to build a tamper-proof reputation system.	The obligation blockchain has high communication overhead to verify the obligations.
Pim Otte *et al.* [25]	TrustChain uses tamper-proof, temporal ordered, and cryptographically signed transaction records to create an irrefutable proof of past actions.	Full-scale deployment of TrustChain is yet to be done to make it into an operational 'bandwidth-as-a-currency'.
Feng Tian *et al.* [26]	Introduces BigchainDB in which each node stores data through the partial replication method, which results in high throughput, low latency, and high capacity blockchain.	The proposed blockchain architecture does not have the process of mining. Cryptographic keys are used to add data which is susceptible to hacking.
Bocek *et al.* [27]	A relational database is used to store the raw IoT data and smart contracts. The verification results of IoT data with the smart contracts are only stored in the blockchain.	Forking is allowed and to see the result of the fork takes a couple of days.
Dorri *et al.* [28]	Distributed trust strategy based on blockchain reduces the processing time by 50 percent.	Vulnerable to the addition of false blocks during 51% attack.
Samaniego *et al.* [29]	This work compared the performance of blockchain on fog and cloud.	Blockchain implemented on a Fog has lower network latency compared to cloud computing.

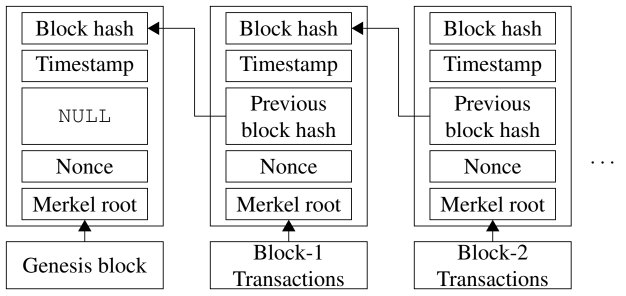

Figure 1: Blockchain architecture.

that is prone to be exploited. However, blockchain faces critical challenges for its application in an IoT environment: (*i*) The Proof-of-Work calculation is computationally intensive and time-consuming. Since the majority of IoT devices are resource-constrained and most IoT applications need low latency, application of traditional blockchain becomes infeasible. (*ii*) The Merkle tree implementation becomes a bottleneck for IoT due to the existence of numerous sensors in typical IoT deployment. (*iii*) The underlying blockchain protocols create significant network overhead, which is not suitable for communication among IoT devices.

The objective of this paper is to propose a blockchain architecture to increase the IoT security, reduce the memory overhead and network overhead, and also to evaluate the performance of the blockchain architecture on a smart home.

The prototype of a smart home environment has a heterogeneous network consisting of Wi-Fi, Zigbee, and Bluetooth technologies as shown in Figure 2. Different organizations such as hospital, health insurance, police, and NGO are connected to the smart home for smart management. The smart home owner chooses the organization that needs to be connected with the home. The organizations act as miners. In our prototype, sliding window blockchain is used to securely store the state of home as well as the transactions between the organizations through a secure channel. The blockchain used is private and permissioned. The data generated by the smart home are encrypted before it is being stored in the blockchain and the key for encryption is shared only with the concerned organization. The owner and organizations together form smart contracts and blocks are added to the chain only if all the group members of the blockchain validate the block.

V. Proposed Sliding Window Blockchain Architecture

The Sliding Window Blockchain (SWBC) utilizes a window that slides through the blockchain for every block addition. The window initially consists of one block and increases up to n blocks as defined by the window size. The blocks in the sliding window are used while creating a new block. In the proposed SWBC architecture, the *block hash* is generated by hashing the blocks in the window as shown in Figure 3. The size of the sliding window determines the number of recent past blocks used to perform the hash update function. The sliding window blockchain has a computational overhead of $\mathcal{O}(n)$ for a constant difficulty of mining, where n is the number of blocks in the window used for the hash update

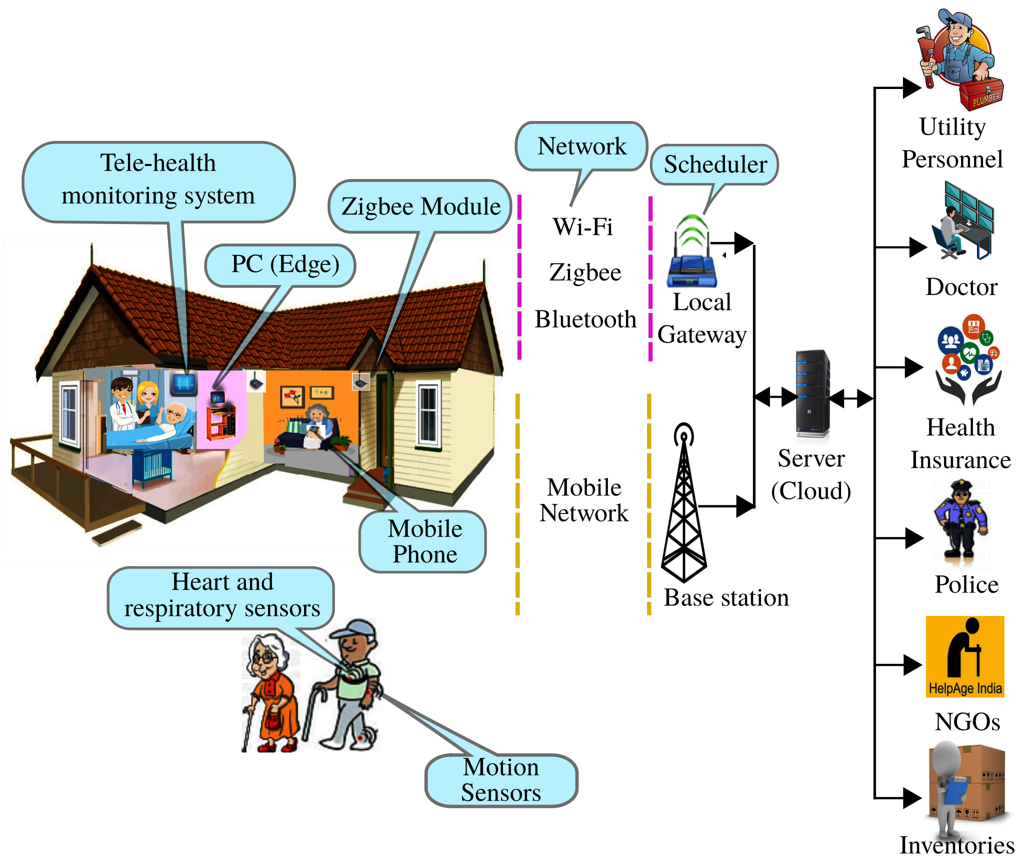

Figure 2: A typical smart home system for assisted living.

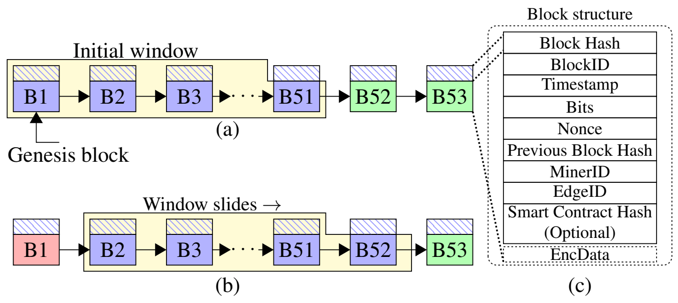

Figure 3: Sliding window blockchain.

function. Sliding window improves the immutability of the blockchain records. A false miner requires previous $(n-1)$ blocks and the window size n to mine a block. The window size is kept secret and sent only to the miners along with the genesis block. The limited part of the chain, i.e., the recent n blocks is stored in the memory of IoT device and the whole blockchain is stored in a private cloud. When the window slides, the older block comes out of the window (block $B1$ as shown in Figure 3(b)) and is deleted from the IoT device memory. Therefore, the memory overhead to store the blocks in IoT device is reduced. The SWBC structure and its comparison with a Bitcoin blockchain are discussed in the following sections.

A. Sliding window blockchain structure

Figure 3(c) shows the sliding window block structure. The SWBC block consists of *block hash, blockID, timestamp, bits, nonce, previous block hash, minerID, and edgeID*. *Block Hash* is generated by hashing current block and previous $(n-1)$ blocks. The *BlockID* represents a unique ID of a block. Only the members are allowed to access the block ID of the newly added block. The field *Timestamp* shows the time at which the block is created. The field *Bits* represents the difficulty level of mining. The difficulty level of mining is decided by the number of initial zeros of the hash value. Each zero is represented by four bits. The difficulty levels are represented as follows: Level 1 (4 bits), Level 2 (8 bits), Level 3 (12 bits), Level 4 (16 bits), and Level 5 (20 bits). As the number of zeros increases, the difficulty level of mining (i.e., computation time) increases rapidly. A high difficulty level for PoW leads to an increase in computing resources, which makes Bitcoin blockchain not suitable for IoT [17]. Also, to reduce the total computation time to mine the blocks, the difficulty level can be chosen at random between 1 and 5. The *Nonce* value represents the iteration for which the proof of work gets solved. The *Previous block hash* is the hash of the previous block which inherits the properties of previous n blocks, where n is the size of the window. *MinerID* represents the ID of the gateway and *EdgeID* represents the ID of the edge device. *Smart Contract Hash* represents the hash value of the smart contract accepted by all the miners. Smart contract hash field is optional and activating this field secures the smart contract from reentrancy attack. Smart contract hash field is not included in our experiment. The *EncData* consists of sensor data encrypted using the Advanced Encryption Standard algorithm with Password Based Key Derivation Function (PBKDF2) [30].

B. Sliding window blockchain parameters

In a Bitcoin blockchain, the block size is limited to 1 MB and a block is mined in every 10 minutes [31]. SWBC considers a variable block size with an upper limit of 1 MB. If the block size exceeds 1 MB, then the data is split to fit in more than one block. The block size is calculated using Equation (1).

$$\begin{aligned} Block\ size =& ((Data + Encryption\quad overhead) \times samples) \\ &+ Block\ overhead \end{aligned} \tag{1}$$

where *Encryption overhead* is the overhead due to encryption in bytes. *Samples* represents the number of data samples and *Block overhead* is the number of bytes used to represent a block. Here, the *Block size* is limited to $\leq$ 1 MB. The difficulty of finding a target value is given by the Equation (2).

$$difficulty = \frac{difficulty\_level\_target}{current\_target} \tag{2}$$

where *difficulty_level_target* for Level 1 starts with 4 zero bits and the rest are 1s, i.e., the hexadecimal equivalent is 0fff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff. Current target is the target to be achieved for the current block.

The approximate average time to generate a block is calculated using the Equation (3).

$$average\ time = \frac{(difficulty \times 2^{Bits})}{hash\ rate} \tag{3}$$

where *Bits* represents the difficulty level of mining and *hash rate* is the number of hashes miners compute per second. The difference in key parameters of the Bitcoin blockchain and the proposed SWBC architecture are given in Table II.

VI. Experimental Setup

In this section we describe the experimental studies using the following: (*i*) smart home testbed modules and their functions and (*ii*) blockchain implementation using Python and its communication protocol.

A. Smart home prototype

A prototype IoT system is implemented with SWBC in the context of a smart home environment. The IoT system testbed for the smart home is shown in Figure 4. The prototype consists of sensors, electrical devices (light and fan), an edge device (Arduino Uno), Wi-Fi module (ESP8266), and gateway (personal computer). The ambient parameters are sensed using an ambient light sensor, temperature sensor, pressure sensor, humidity sensor, fire sensor, hazardous gas sensor, proximity sensor, and sound sensor. The proximity sensors detect a

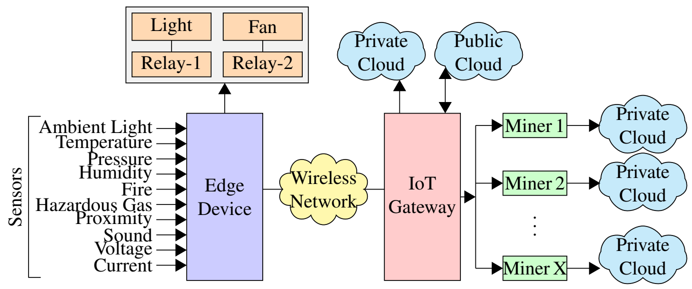

Figure 4: Smart home testbed used for studying sliding window blockchain.

person entering and exiting a room. Our smart home has the following functions: (*i*) Relay 1 is closed when people are present inside the room and the ambient light is less than the threshold value during the day time. (*ii*) Relay 2 is closed when people are present inside the room and the temperature is higher than the threshold value. (*iii*) Buzzer alarms when fire and gas leakage is sensed. (*iv*) The LED glows when sound is heard and people are not inside the home to detect theft. (*v*) The current sensor (ACS712) reads the current value and convert it into a relevant voltage value between (0V to 5V). The voltage sensor (ZMPT101B) measures the voltage between 0V to 1000V AC. The energy consumed by the smart home is calculated using the current sensor and voltage sensor values. (*vi*) The state of the room is time stamped with Unix time retrieved from the NTP server. (*vii*) The sensor data is transferred from the edge device to the gateway through ESP8266 (Wi-Fi module) using TCP/IP protocol. (*viii*) At the PC, the sensed data is encrypted using the Advanced Encryption Standard (AES) algorithm with PBKDF2 and securely stored using sliding window blockchain. The edge device is an Arduino Uno that has analog and digital pins to which the sensors are connected. Relays are connected to 5V and sensors are connected to the 3.3V pins of Arduino. The sensor terminals are connected to (*i*) digital/analog pins, (*ii*) Vcc (3.3V), and (*iii*) GND of the Arduino. The Tx and Rx of Arduino Uno are connected to the Rx and Tx of ESP8266. The ESP8266 (Wi-Fi module) is configured to communicate with the TCP/IP protocol using AT commands. ESP8266 is connected to a common access point through which it communicates with the server. It takes approximately 14 seconds for Arduino Uno to bootup, check readiness of ESP8266, and set up a new TCP/IP connection between ESP8266 and the gateway. The above experiment is also conducted on Arduino Due [32] as edge device and Raspberry Pi as the blockchain miner.

B. Realization of Sliding window blockchain

The sliding window blockchain is implemented on Intel Core-i5-3470 CPU 3.2 GHz processor with memory 4 GB, running Ubuntu 16.04 LTS operating system. The blockchain algorithm is implemented using Python.

As the initial step, the smart home owner selects the miners for the blockchain according to his/her preferences. The ID of each miner is known to the owner. A reliable communication

Table II: A Comparison of SWBC and Bitcoin blockchain parameters.

Sl.No.	Parameter	Sliding window blockchain	Bitcoin blockchain
1	Blockchain Visibility	Private	Public
2	Transaction mining	All transactions	All transactions
3	Mining requirement	Proof of Work	Proof of Work
4	Forking	Not allowed	Not allowed
5	Double Spending	Not applicable	Prohibited
6	Transaction verification	MinerID and EdgeID	Signature
7	Transaction parameters	Refer Figure 3	Hash of the block, nonce, timestamp, previous block hash, Merkle root
8	Transaction dissemination	Multicast	Broadcast
9	Deference in block hash	Puzzle	Puzzle
10	Blocks stored by miner	All blocks	All blocks
11	New block verification	Blocks and transactions in blocks	Blocks and transactions in blocks
12	BC control	Owner	None
13	Miner checks	IP Address, MinerID, and EdgeID	None
14	Miners	Miners are same and new miners can be added by the owner	Miners are different
15	Transactions per block	Depends on the sampling rate and volume of sensor data	Depends on the size of transactions
16	Miner joining overhead	Download all blocks in the blockchain	Download all blocks in the blockchain
17	Miner selection	Owner chooses the miner	Self-selection
18	Miner rewards	Nothing	Coins
19	Pool mining	Not applicable	Allowed
20	Malicious miner	Not allowed to join	Allowed to join
21	Effects of 51 percent attack	Not Possible	Double spending
22	Encryption method	Private key	Public/private keys
23	Number of blocks needed to add a new block	Sliding window size	Current block

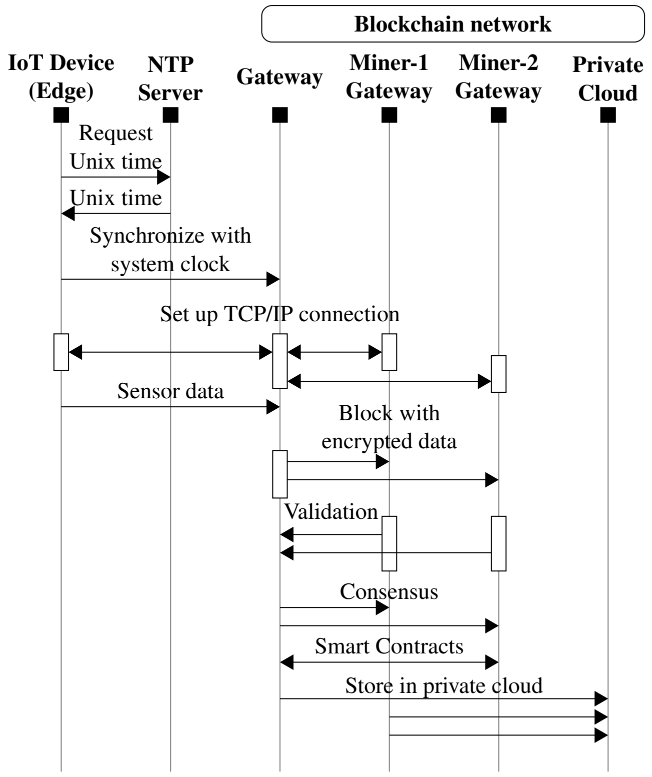

Figure 5: Communication between IoT devices and the Blockchain network.

session is established between the owner and the miners. Genesis block of the blockchain is created with the required fields as defined in Figure 3 by the owner and is broadcasted to all the miners. Miners add the genesis block as their first block. The genesis block carries the window size in the data field. The smart home sensor data is recorded from the second block onward. The miners of the group mine the block and send the validation back to the owner. When the validation is received from the miners, the owner sends the consensus message to add the block. All the miners have the privilege to create a block and send it to the group for validation (e.g., the biomedical data of a person is collected at the hospital). The access permissions and privileges of the miners are registered on the smart contracts formed by the group. The communication between IoT device and blockchain is shown in Figure 5.

As the blockchain builds up, the sliding window size increases from 1 to n. The entire blockchain is stored in the private cloud storage and the window containing n blocks is stored in the IoT devices. SWBC reduces the memory overhead and makes the blockchain feasible to be implemented on IoT devices. Table III shows the smart home prototype system parameters. From the smart home testbed, an average of 180 bytes of real-time data is generated. The data has an additional encryption overhead of 100 bytes and block overhead of 160 bytes.

Table III: The smart home prototype system parameters.

Average sensor data (bytes)	Encryption Overhead (bytes)	Block overhead (bytes)
180	100	160

VII. Performance Analysis

The performance of SWBC is analyzed by creating up to 30 blocks in the window for different difficulty levels, i.e., the levels between 1 and 5. The time taken for (*i*) solving the Proof-of-Work, (*ii*) creation and addition of a block, and (*iii*) validation of a block is analyzed. A computational complexity analysis of SWBC for different difficulty levels is also carried out.

A. Average time taken for block creation and addition

On receiving an encrypted sensor data, the block is first created by the owner and then it is sent to other miners in the group for validation. As the difficulty level increases, the time taken to create and add blocks also increases. Assuming constant network delay, the time taken for block creation and block addition primarily depends on the Proof-of-Work and validation time, respectively. For a difficulty of 20 bits, window size 30, and a single miner, the average time taken for different experimental setup are (*i*) SWBC on PC— 182.39 s, (*ii*) SWBC on Raspberry Pi— 2380.90 s, and (*iii*) Bitcoin blockchain on PC—18.37 s, as shown in Figure 6.

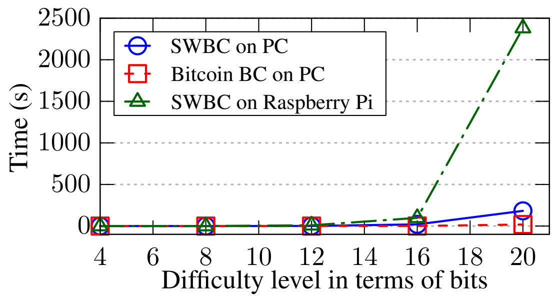

Figure 6: Time taken for block creation and addition.

Table IV: Average time taken to add 30 blocks.

Implementation	Difficulty Level	Max time taken to add a block (Minutes)	Average block creation and addition time for 30 blocks (Minutes)
SWBC on Raspberry Pi	Level 4	7.78	30.75
	Level 5	98.97	529.98
	Random (1–5)	69.07	9.10
SWBC on PC	Level 4	0.8	4.061
	Level 5	19.87	66.2
	Random (1–5)	6.43	0.022

The average time to add 30 blocks for SWBC on PC, and SWBC on Raspberry Pi for difficulty levels 4, 5, and random difficulty level between 1–5 are shown in Table IV. From Table IV it is inferred that a difficulty level lower than or equal to 4 is preferred for IoT devices similar to Raspberry Pi and a difficulty level lower than or equal to 5 is preferred for PC. The selection of random difficulty levels between 1–5 reduces the total computation time compared to constant high level of difficulty

The block addition time also increases as the number of miners increases. Figure 7 shows the total block creation and addition time for 30 blocks with a difficulty level 4 and different number of miners.

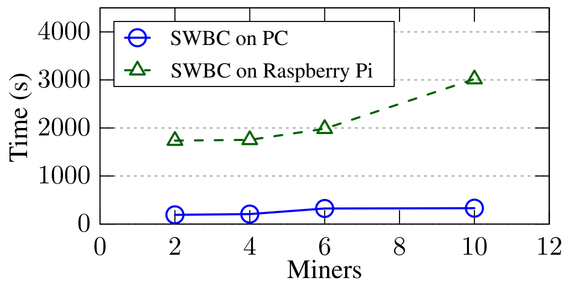

Figure 7: Total block addition time for 30 blocks.

B. Time taken to solve the blockchain puzzle

The Proof-of-Work target in our SWBC is a 256-bit number (i.e., 64 hexadecimal digits), e.g., 0000 0af3 1eab 34a7 92c5 eace 6e24 0148 0149 9e99 3f07 bdd0 e36a 396f 6e9b 6588. One hex value represents a nibble. Therefore, five zeros at the start represent a difficulty of 20 bits. The lower the target value, the more difficult is the generation of a block. The target value required is also represented as the difficulty, where a higher difficulty represents a lower target. Any change in the block data makes the block hash completely different due to the diffusion property of hashes. Since it is infeasible to predict the combination of bits that results in the specific hash, different *nonce* values are tried, and the hash is recomputed for each value until a hash equal to the current target of the network is found. The iteration for which the hash value meets the target is called the *nonce*. As this iterative calculation requires time and resources, the difficulty level is set in the range of 1 to 20 bits for the smart home application. The time taken to solve blockchain puzzle with a single miner and different levels of difficulty and window sizes (5, 10, 15, 20, 25, and 30) with a single miner are shown in Figure 8. The average time taken for different difficulty levels are observed as follows: Level 1 (0 to 0.03 s), Level 2 (0 to 0.4 s), Level 3 (0 to 10 s), Level 4 (0 to 150 s), and Level 5 (0 to 2000 s). The puzzle-solving time shown in Figure 8 does not always increase as the window size increases because puzzle-solving time is dependent on other factors such as (*i*) time to reach the target of current transaction block, (*ii*) difficulty level, (*iii*) window size, and (*iv*) sensor data size. For a particular difficulty level, the size of sensor data is assumed to be constant, whereas the window size and the time to reach the target vary for each block. The graphs for difficulty levels 2, 4, and 5 in Figure 8 do not vary linearly with the window size because the effect of *time to reach the target of current transaction block* is more pronounced compared to the computational complexity induced due to window size. Further, the process of puzzle solving can be explained as given below.

There are two main steps involved in puzzle solving

1) **To find the hash of the block:** The SHA-256 algorithm is used to find the hash of the block which is a 256 bit unique number. e.g., e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca 495991b7852b855. The sliding window has impact only

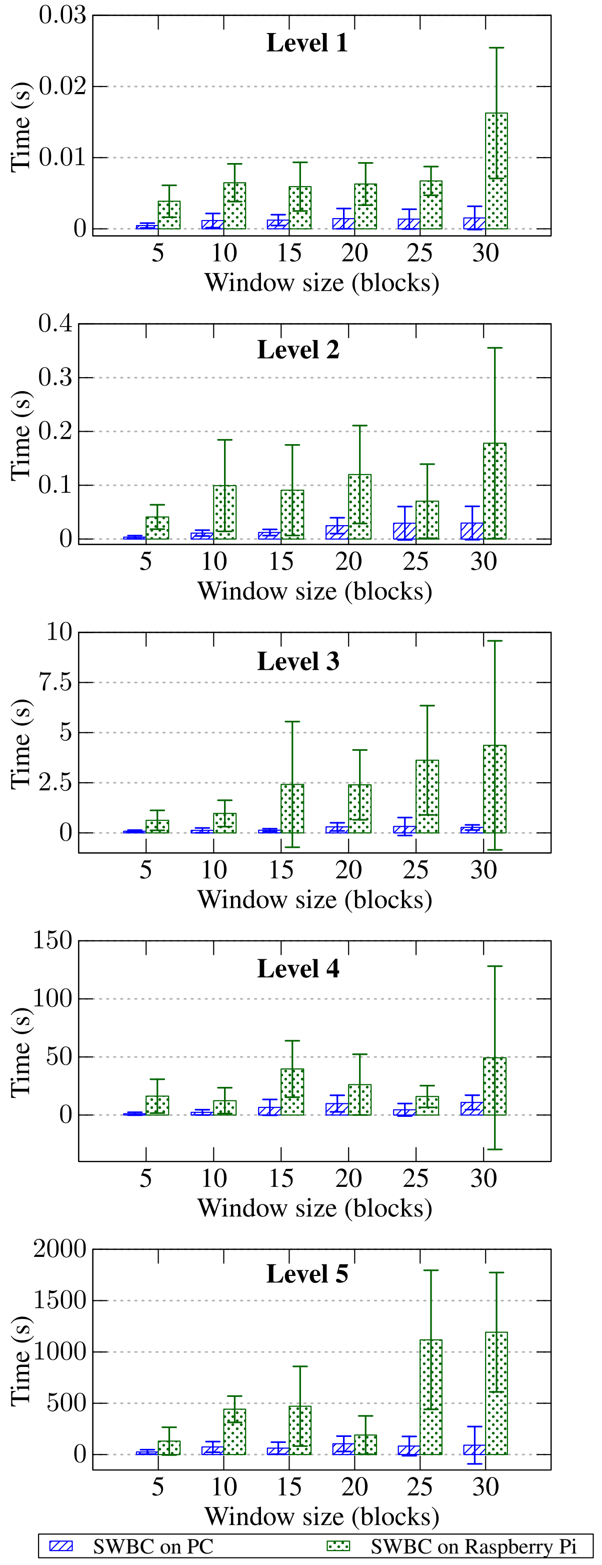

Figure 8: Comparison of time taken to solve the blockchain puzzle with different levels of difficulty and window sizes.

on the hash computation time with a computational complexity of $\mathcal{O}(n)$.

2) **To reach the target with a specified difficulty level:** The target is a 256-bit number. The target value consists of the difficulty. e.g., if the difficulty level is 3. The first three hexadecimal digits should be zeros. e.g., 000xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxx. Initially the hash of the block is compared with the target. If hash does not meet the target, nonce value (initial nonce value = 1) is incremented, and the hash of the block is recomputed. The process is repeated until the hash value meets the target. Since each block has different hash value to start with, the time t_t at which the block reaches the target varies. The worst-case time to reach the target t_t with a hash length of b bits is expressed as $t_t = \mathcal{O}(2^b)$. Therefore, puzzle solving time t_p can be expressed as $t_p = \mathcal{O}(n \times t_t)$, where $t_t >> n$. Such an uncertain puzzle solving time is the reason behind the non-linearity in the graphs for difficulty levels 2, 4, and 5 in Figure 8. The effect of window size alone on hash computation is shown in Figure 11.

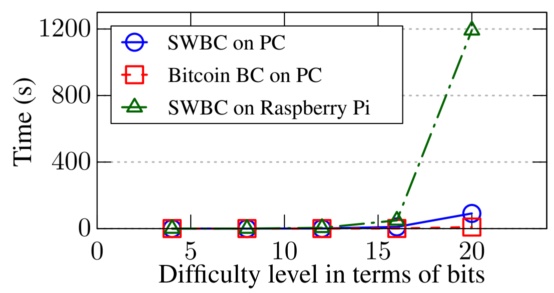

Figure 9: Time to solve a block chain puzzle.

The average time taken for proof of work with 20-bits difficulty level and window size 30 for the SWBC on PC, SWBC on Raspberry Pi, and Bitcoin blockchain on PC are 91.13 s, 1192.10 s, and 9.3 s, respectively, as shown in Figure 9.

C. Time taken for block validation

Validation of a block is carried out by the miner's gateway (organizations) of the group connected to the smart home. The time taken for validation is approximately equal to the time taken for generating the block hash without considering the network parameters, which may vary depending on the network traffic. The average time taken for the validation with 20-bits difficulty for the SWBC on PC, SWBC on Raspberry Pi, and Bitcoin blockchain on PC with a single miner are 91.27 s, 1188.80 s, and 9.06 s, respectively, as shown in Figure 10.

D. Time taken for hash computation for different window sizes

To study the effect of window size, files of different sizes are created. A 1 MB file represents one block and similarly a 10 MB file represents 10 blocks. The time taken for generating the hash value for a file is calculated. Figure 11 shows that the increase in window size has a linear impact on the hash computation time of a sliding window blockchain. The hash computation time is a linearly increasing function with respect to the window size. For our experiment, we fitted the

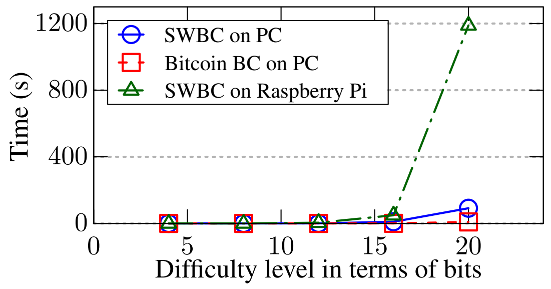

Figure 10: Time taken for validation.

hash computation points with a line equation as shown in Equation 4.

$$f(x) = mx + b. \tag{4}$$

where $f(x)$ is hash computation time, m = slope, x = window size, and b = constant. The slope obtained for the samples in Figure 11 is $m = 0.0052$ and $b = 0.01641$. Therefore, the hash computation time can be expressed as

$$\textit{hash computation time} = \textit{0.0052} \times \textit{window size} + \textit{0.01641}. \tag{5}$$

Therefore, SWBC increases blockchain security with a negligible increase in computational complexity.

The sliding window size can be fixed or variable. Variable window size makes the blockchain more secure and difficult to compromise. A variable sliding window size uses different window size for calculating the hash of each new block in the blockchain. Also, a variable sliding window blockchain needs a mechanism to store the sequence of window sizes used for each block generation. Our analysis of SWBC is carried out with fixed window size. The sliding window size can be chosen based on (*i*) our application, (*ii*) requirement of time taken to mine a block, and (*iii*) the required security level. SWBC takes less amount of time to mine a block with small window size.

The maximum data acquisition rate is about 10,000 times a second for Arduino ATmega based boards (UNO, Nano, Mini, and Mega). Therefore, a window size of 10 to 20 MB is more preferable for fast real-time IoT applications because they have an average hash computation time of 0.066 s, and 0.126 s, respectively.

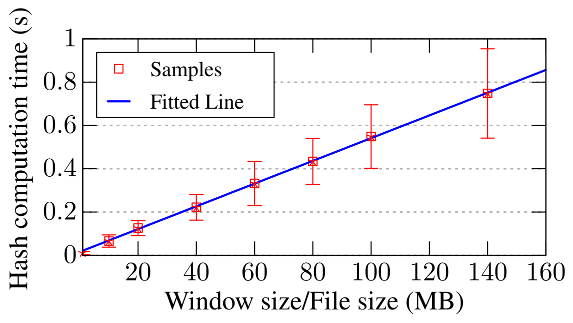

Figure 11: Hash computation time.

E. Security analysis

The security of the system is modeled as follows:

1) A classical symmetric encryption algorithm E takes as input security password P and randomly generates a key *Kpublic* at stage S_1.

2) At stage S_2 the encryption algorithm generates an encrypted data *EncData (Kpublic, D)* using the public key *Kpublic* and sensor data D. The public key *Kpublic* is stored in the smart contract with different access permission to the miners.

3) *Append-L* shown in *Algorithm 1* (Sliding Window Blockchain) which takes *BlockID, Timestamp, Bits, Nonce, Previous Block Hash, MinerID, EdgeID, Smart Contract Hash (Optional),* and *EncData* of the current block and *Block Hash, BlockID, Timestamp, Bits, Nonce, Previous Block Hash, MinerID, EdgeID, Smart Contract Hash (Optional),* and *EncData* of the previous $(n-1)$ blocks as input, where n is the size of the sliding window. The algorithm fails if the *BlockID* of current block already exists in any block in the ledger, otherwise, the m miners begin to solve a Proof-of-Work puzzle by incrementing nonce value by one for every iteration. The algorithm succeeds if the puzzle is solved and validation is received from the miners. Let B_i represents i^{th} block with data being the concatenation of block-fields such as *Block Hash, BlockID, Timestamp, Bits, Nonce, Previous Block Hash, MinerID, EdgeID, Smart Contract Hash (Optional), and EncData*. B_{curr} represents the current block to be mined with data being the concatenation of block fields such as *BlockID, Timestamp, Bits, Previous Block Hash, MinerID, EdgeID, Smart contract Hash (Optional), and EncData*. Let N_{curr} represents the nonce of the current block. Then the Block Hash of the current block H_{curr} is calculated using Equation 6.

$$H_{curr} = h\left(\sum_{i=1}^{n-1} B_{l-i} + [B_{curr} + N_{curr}]\right) \tag{6}$$

where $\sum_{1}^{n-1} B_{l-i}$ is the data from previous $n-1$ blocks used for mining and *l* represents the length of the blockchain including the current block, and h is the hash function.

Algorithm 1: Append-L

```
Data: $h(a)$— Function that computes hash of input $a$
1 $N_{curr} = 1$
2 while True do
3     $H_{curr} = h\left(\sum_{i=1}^{n-1} B_{l-i} + [B_{curr} + N_{curr}]\right)$
4     if $H_{curr}$ achieves target then
5         break
6     else
7         $N_{curr} = N_{curr} + 1$
```

When the window size n is known to the miner, the computational complexity for puzzle solving is t_p=$\mathcal{O}(nt_t)$, where t_t is the time to reach the target of current block in

Table V: Qualitative security analysis of SWBC as in the approach followed by [33].

Layer	Threats	Security Analysis
Application	a) Unauthorized access to an Exchange Server	The real-time data is encrypted and stored in the blockchain. Only the registered users with access permissions as mentioned in the smart contract can receive the key to decrypt the data.
	b) Exchange DDoS	Since we have a private blockchain network with limited users with their user ID registered in the blockchain, the DDoS attacker can be easily identified and eliminated.
	c) Employees Host Security	Each node in the blockchain is identified using the Miner ID and no personal details such as username, email address, and encrypted passwords are mentioned in the blockchain fields.
	d) Malicious Program Infection	A malicious program can be implanted into the exchange system which can result in the leakage of a large amount of sensitive information, including key and wallet files. The sliding window blockchain is vulnerable to this attack.
	e) Initial Coin Offering	Not applicable.
	f) Mining Pool Attack	Mining pool attack is not applicable for SWBC.
Smart contract	a) Reentrancy Attack	The hash value of the smart contract can be stored in the blockchain field so that any change in the smart contract will be notified to all the miners of the SWBC. Therefore, SWBC is resistant to reentrancy attack.
	b) Unauthorized Access Attack	Since each Miner ID is registered in the smart contract, unauthorized access attack is not possible.
	c) Solidity Development Security	The security against this attack depends on the smart contracts written in the smart contract layer.
Consensus	a) Bribe Attack	Not applicable for PoW.
	b) Long-Range Attack	SWBC is resistant to this attack because it uses a window size of n blocks to perform the proof of work which is confidential and known only to the miners.
	c) Coin Age Accumulation Attack	Not applicable for PoW.
	d) Precomputing Attack	Not applicable for PoW.
	e) Sybil Attack	SWBC is a private blockchain, therefore, it is less prone to Sybil Attack.
Network	a) Eclipse Attack	In SWBC, eclipse attack on a particular miner can be easily traced using the miner ID and its behavior in the network.
Data	a) Block Data	Illegal or inappropriate data inclusion must be carefully monitored by the miners.
	b) Signature and Encryption Method	SHA 256 algorithm, used at present, can be updated according to the requirement.

a search space of 2^b (b is the length of the hash in bits). Therefore, t_p can be expressed as $t_p = \mathcal{O}(n \times 2^b)$.

Since the window size is not known for an attacker, computation of hash for all possible window sizes is needed to successfully conduct an attack. Therefore, the computational complexity increases to $t_p = \mathcal{O}(n \times nt_t) = \mathcal{O}(n^2 2^b)$. Further, Table V shows how SWBC is resilient to different types of threats in each layer of the blockchain.

VIII. Conclusion

IoT devices face constraints on resources such as computational capability, energy sources, and memory. Therefore, the standard security algorithms are not feasible for IoT. We proposed a sliding window blockchain that meets the requirements of a resource constrained IoT network by reducing the memory overhead and limiting the computational overhead. The memory overhead is reduced by storing only a limited part of the blockchain, as defined by the sliding window size in the IoT device and maintaining the whole blockchain in the private cloud. Computational overhead is limited by using the difficulty level between 1 and 5 and by eliminating the Merkle tree. The security is increased by generating the block hash using the properties of n blocks in the sliding window. A false miner cannot mine a block unless he gets the previous $(n-1)$ blocks and the window size information.

From the experimental results, we observed the following: (*i*) The computational time of PoW for each level of difficulty increases exponentially. (*ii*) The total block addition time increases with the increase in the number of miners in the group. (*iii*) As the window size increases, the hash computation time increases linearly. (*iv*) A random selection of difficulty for each block in a blockchain reduces the total block addition time.

Future work can be carried out to analyze the impact of a variable size sliding window. New consensus algorithms can be developed to suit the IoT environment. Furthermore, energy consumption of the blockchain can also be analyzed to draw more insights on energy resources required for an IoT device.

Acknowledgment

The authors would like to thank Kerala State Council for Science Technology and Environment (KSCSTE) for funding the project.

References

[1] S. Kulkarni, "The beauty of the blockchain," *Open Source for You*, vol. 06, pp. 22–24, June 2018.

[2] T. M. F. Carames and P. F. Lamas, "A review on the use of blockchain for the Internet of Things," *IEEE Access*, vol. 6, pp. 32 979–33 001, May 2018.

[3] A. Dorri, S. S. Kanhere, and R. Jurdak, "Blockchain in Internet of Things: challenges and solutions," *arXiv preprint arXiv:1608.05187*, August 2016.

[4] IoT Agenda, "Smart home or building," April 2018. [Online]. Available: https://internetofthingsagenda.techtarget.com/definition/smart-home-or-building

[5] L. Jiang, D. Y. Liu, and B. Yang, "Smart home research," in *Proceedings of 2004 International Conference on Machine Learning and Cybernetics*, vol. 2, August 2004, pp. 659–663.

[6] theinstitute.ieee.org, "Towards a definition of the Internet of Things (IoT)," May 2015. [Online]. Available: https://iot.ieee.org/images/files/pdf/IEEE_IoT_Towards_Definition_Internet_of_Things_Revision1_27MAY15.pdf

[7] J. Wan, X. Gu, L. Chen, and J. Wang, "Internet of Things for ambient assisted living: Challenges and future opportunities," in *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, October 2017, pp. 354–357.

[8] D. Abbasinezhad-Mood, A. Ostad-Sharif, and M. Nikooghadam, "Novel anonymous key establishment protocol for isolated smart meters," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 4, pp. 2844–2851, April 2020.

[9] S. K. Das, D. J. Cook, A. Battacharya, E. O. Heierman, and T. Y. Lin, "The role of prediction algorithms in the MavHome smart home architecture," *IEEE Wireless Communications*, vol. 9, no. 6, pp. 77–84, December 2002.

[10] C. Qu, M. Tao, J. Zhang, X. Hong, and R. Yuan, "Blockchain based credibility verification method for IoT entities," *Security and Communication Networks*, vol. 2018, pp. 1–11, June 2018.

[11] C. Lee, L. Zappaterra, K. Choi, and H. A. Choi, "Securing smart home: Technologies, security challenges, and security requirements," in *IEEE Conference on Communications and Network Security*, October 2014, pp. 67–72.

[12] P. Treleaven, R. G. Brown, and D. Yang, "Blockchain technology in finance," *Computer*, vol. 50, no. 9, pp. 14–17, September 2017.

[13] V. Gatteschi, F. Lamberti, C. Demartini, C. Pranteda, and V. Santamaría, "To blockchain or not to blockchain: That is the question," *IT Professional*, vol. 20, no. 2, pp. 62–74, March 2018.

[14] P. A. Laplante and B. Amaba, "Introducing the Internet of Things department," *IT Professional*, vol. 20, no. 1, pp. 15–18, January 2018.

[15] C. Esposito, A. D. Santis, G. Tortora, H. Chang, and K. R. Choo, "Blockchain: A panacea for healthcare cloud-based data security and privacy," *IEEE Cloud Computing*, vol. 5, no. 1, pp. 31–37, January 2018.

[16] N. Kshetri, "Can blockchain strengthen the Internet of Things," *IT Professional*, vol. 19, no. 4, pp. 68–72, July/August 2017.

[17] A. Dorri, S. S. Kanhere, and R. Jurdak, "Towards an optimized blockchain for IoT," in *Proceedings of the Second International Conference on Internet of Things Design and Implementation*. ACM, August 2017, pp. 173–178.

[18] J. Shen, C. Wang, T. Li, X. Chen, X. Huang, and Z. H. Zhan, "Secure data uploading scheme for a smart home system," *Information Sciences*, vol. 453, pp. 186–197, July 2018.

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

[20] G. Zyskind, O. Nathan, and A. Pentland, "Enigma: Decentralized computation platform with guaranteed privacy," *arXiv preprint arXiv:1506.03471*, 2015.

[21] B. Liu, X. L. Yu, S. Chen, X. Xu, and L. Zhu, "Blockchain based data integrity service framework for IoT data," in *2017 IEEE International Conference on Web Services (ICWS)*, June 2017, pp. 468–475.

[22] A. Bahga and V. K. Madisetti, "Blockchain platform for industrial Internet of Things," *Journal of Software Engineering and Applications*, vol. 9, no. 10, p. 533, October 2016.

[23] A. Boudguiga, N. Bouzerna, L. Granboulan, A. Olivereau, F. Quesnel, A. Roger, and R. Sirdey, "Towards better availability and accountability for IoT updates by means of a blockchain," in *2017 IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*, April 2017, pp. 50–58.

[24] R. Di Pietro, X. Salleras, M. Signorini, and E. Waisbard, "A blockchain-based trust system for the Internet of Things," in *Proceedings of the 23nd ACM on Symposium on Access Control Models and Technologies*. ACM, June 2018, pp. 77–83.

[25] P. Otte, M. de Vos, and J. Pouwelse, "Trustchain: A sybil-resistant scalable blockchain," *Future Generation Computer Systems*, pp. 12–23, July 2017.

[26] Feng Tian, "A supply chain traceability system for food safety based on HACCP, blockchain & Internet of Things," in *2017 International Conference on Service Systems and Service Management*, June 2017, pp. 1–6.

[27] T. Bocek, B. B. Rodrigues, T. Strasser, and B. Stiller, "Blockchains everywhere - a use-case of blockchains in the pharma supply-chain," in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, May 2017, pp. 772–777.

[28] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, "Blockchain for IoT security and privacy: The case study of a smart home," in *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, March 2017, pp. 618–623.

[29] M. Samaniego and R. Deters, "Blockchain as a service for IoT," in *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, December 2016, pp. 433–436.

[30] B. Kaliski, "Password-based cryptography specification version 2.0," *Network Working Group, RSA Laboratories*, pp. 1–34, September 2000.

[31] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *2017 IEEE International Congress on Big Data (BigData Congress)*, June 2017, pp. 557–564.

[32] D. Abbasinezhad-Mood and M. Nikooghadam, "Design and extensive hardware performance analysis of an efficient pairwise key generation scheme for smart grid," *International Journal of Communication Systems*, vol. 31, no. 5, p. e3507, 2018.

[33] H. Wang, Y. Wang, Z. Cao, Z. Li, and G. Xiong, "An overview of blockchain security analysis," in *China Cyber Security Annual Conference*. Springer, February 2018, pp. 55–72.

Prescilla K received the B.E degree in Electrical Engineering from Coimbatore Institute of Technology, and M.E (Applied Electronics) and Ph.D. degree in Electronics and Communication from Karunya University, Coimbatore, India. She had worked as Assistant Professor in Karunya University, Mar Baselios college of Engineering and Technology, and Mohandas college of Engineering and Technology, India. She is currently a Post-Doctoral Fellow at Department of Avionics, Indian Institute of Space Science and Technology, Thiruvananthapuram, India. Her research interests include Internet of Things (IoT), blockchain architectures for IoT, real-time systems, and embedded systems.

Sarath Babu is currently a Senior Research Fellow at Department of Avionics, Indian Institute of Space Science and Technology, Thiruvananthapuram, India. He obtained his bachelors degree in Information Technology from Mahatma Gandhi University, Kottayam, India and completed his masters degree in Computer Science and Engineering (Information Security) from National Institute of Technology, Calicut, India. His research interests include delay tolerant networks, software defined networks, wireless mesh networks, and complex networks.

B. S. Manoj received his Ph.D. degree in Computer Science and Engineering from the Indian Institute of Technology, Madras, in July 2004. He is currently working as Professor in the Department of Avionics, Indian Institute of Space Science and Technology (IIST), Thiruvananthapuram, India. Before joining IIST, he worked as Assistant Research Scientist and Lecturer in the Electrical and Computer Engineering Department, University of California at San Diego (UCSD). His current research interests include complex networks, ad hoc wireless networks, delay tolerant networks, next generation wireless architectures, wireless sensor networks, wireless mesh networks, and Internet of Things.