

ENHANCING CYBERSECURITY DEFENSE OF IOT ECOSYSTEM USING BLOCKCHAIN

Prashant Setia^{1,*} and S. Sandosh²

Received: April 21, 2023; Revised: May 18, 2023; Accepted: June 19, 2023

Abstract

The expansion of IoT devices over the past years has led to the development of smart ecosystems susceptible to cyberattacks. These ecosystems include smart industries and smart home security systems. However, when these devices are widely deployed and generate significant amounts of data, this data must be securely stored on the cloud to prevent it from being compromised by serious cyber attacks. Although traditional security methods partly meet the need to protect the privacy and security of smart IoT ecosystems, their centralized structure and limited computing power make them inefficient. However, recent advancements in blockchain technology have made it possible to employ decentralized architectures to enhance the cybersecurity defense mechanisms of IoT ecosystems. The utilization of blockchain-enabled methodologies is considered one of the most auspicious advancements in the domain of the IoT. These techniques offer a secure and decentralized mode of communication among internet-connected devices. This study presents a blockchain-based novel framework to enhance the cybersecurity measures of smart IoT ecosystem installations. The proposed framework prioritizes the immutability of devices and users present in such environments and incorporates smart contracts to ensure their overall security. Furthermore, the system includes a module for regularly verifying the firmware of connected devices by comparing them to the valid firmware hashes present in the smart contract's storage. The proposed approach has also come up with the idea of blacklisting the malicious IP addresses trying to connect with IoT ecosystems network.

Keywords: Blockchain; Cybersecurity; Firmware; IoT; Smart contract; Smart ecosystem

Introduction

The Internet of Things (IoT) pertains to a system of interconnected computing devices that possess the capacity to communicate and share data with one another. IoT has the potential to enhance the capabilities of these devices and change the way we interact with them. In order to establish a connection between these devices, data is kept within

centralized cloud servers (as shown in Figure 1. (a)). However, these servers are susceptible to security threats as a result of their reliance on trust.

In modern computing, the majority of logic, whether computational or operational, is commonly located on centralized servers. These servers are typically cloud platforms like AWS, Azure, and

¹ Vellore Institute of Technology, Chennai, India. E-mail: prashantsetia888@gmail.com

² SCOPE Department, Vellore Institute of Technology, Chennai, India. E-mail: sandosh.s@vit.ac.in

* Corresponding Author

DOI: <https://doi.org/10.55766/sujst-2023-04-e01071>

Suranaree J. Sci. Technol. 30(4):010238(1-14)

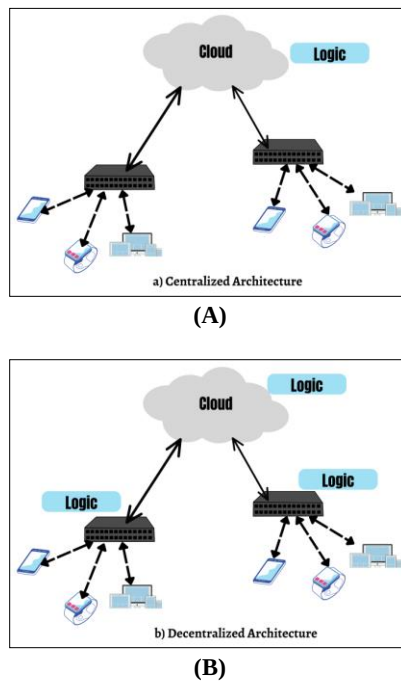


Figure 1. IoT Architecture Types

others. By consolidating computing resources in one location, centralized servers can provide increased processing power, storage, and accessibility. While centralization simplifies the process by bringing all data to a single centralized location, it comes at the cost of increased traffic utilization and resource load, as all data must be sent to a central controller before any decision-making can occur. In practice, multiple forwarding devices are involved in communication between end devices and the central controller, which further exacerbates this issue. As a result, centralization often results in unnecessary resource consumption and higher traffic utilization.

In contrast, edge computing-based IoT platforms offer an innovative approach that shifts certain decision-making functions from the cloud to the network edge, namely intermediate devices, and gateways (as shown in Figure 1. (b)). This decentralized architecture distributes the computational load throughout the network, resulting in reduced data transfer, lower communication delays, and less reliance on cloud services. Additionally, IoT platforms based on decentralized architecture demonstrate superior performance in environments with numerous connected devices, unreliable networks, and significant data volumes. The distributed logic between devices in such platforms provides efficient utilization of resources, reducing cloud service usage and decreasing communication delay (Mocnej *et al.*, 2018).

The incorporation of blockchain technology is imperative in decentralizing data and minimizing potential vulnerabilities to enhance the reliability, security, and feasibility of IoT systems. The employment of blockchain technology has become more prevalent across different domains as a means of augmenting security in a decentralized fashion in recent years. Given its intrinsic characteristics, this technology is particularly suitable for improving security-related use cases in IoT's smart environments.

Currently, IoT finds its primary application in smart devices across various domains. When these devices are interconnected, they form a network of smart devices that comprises a computing infrastructure. Najmi *et al.* (2021) investigated diverse forms of attacks targeting IoT ecosystems and their corresponding effects on the security of the system. The research revealed that IoT ecosystems exhibit significant susceptibility to a diverse array of attacks, primarily attributable to the extensive quantity of interconnected devices and insufficient security protocols. It is imperative to maintain and secure this infrastructure against potential breaches and critical attacks.

This study puts forth a proposition for an IoT framework that utilizes blockchain's decentralized architecture to augment the privacy and security of the IoT ecosystem. The framework under consideration incorporates smart contracts that are intended to uphold the authenticity and integrity of IoT devices and their data. These smart contracts ensure that solely authorized devices are capable of accessing and altering the data while also protecting it against tampering and unauthorized access. Additionally, the framework includes a module that maintains the firmware integrity of the devices. It regularly verifies the firmware integrity of an IoT device by comparing its firmware hash against valid firmware hashes stored in other smart contracts and updates them as needed. Furthermore, the proposed framework also includes a module that protects the IoT ecosystem from malicious IP addresses by maintaining a list of blacklisted IPs and preventing them from participating in the network, thus increasing the overall security of the IoT ecosystem.

IoT and Blockchain Working Fundamentals

In today's digital age, the utilization of blockchain technology has emerged as a promising solution for enhancing security across various domains, including the IoT. However, to fully comprehend how blockchain technology can be utilized to

enhance IoT security, it is vital to have a fundamental understanding of both technologies.

Blockchain Fundamentals

Blockchain technology comprises two fundamental components - Transactions and Blocks. The utilization of blockchain technology enables secure and transparent transactions between entities without the need for intermediaries, owing to its digital ledger nature. Every transaction conducted in a blockchain network is recorded in a block and subsequently added to a chain of other blocks, creating a decentralized and immutable ledger. The distributed nature of a blockchain network enables every participant to obtain access to an identical version of the ledger, thereby promoting transparency and accountability.

A transaction in blockchain refers to the transfer of digital assets, such as cryptocurrencies or other digital tokens, between two or more parties. Transactions are initiated by a sender who initiates a transfer of assets to a receiver, and the transaction is validated and confirmed by a network of nodes on the blockchain. Upon validation, a transaction is documented within a block.

In the context of blockchain technology, a block denotes a compilation of authenticated transactions that are grouped and appended to the existing blockchain. Each block in the chain is associated with a distinct digital signature, commonly referred to as a hash, which serves to establish a connection with the preceding block. The linking mechanism employed in this context guarantees that every block is connected to its preceding block, thereby preserving the overall integrity of the blockchain.

The correlation between transactions and blocks in blockchain technology is easily comprehensible. The process of validating and confirming transactions on the blockchain network is executed by a cluster of nodes. Upon successful validation, the transaction is then recorded in a block. The process of appending a new block to the blockchain necessitates a consensus mechanism that guarantees unanimity among all nodes within the network regarding the legitimacy of the transactions encompassed within the block. This consensus mechanism is what makes blockchain secure and tamper-proof, as any attempt to alter a transaction or a block would require the consensus of the entire network, which is practically impossible to achieve.

The classification of blockchain is based on its usage, with three main types being public, private or permissioned, and consortium. Public blockchains allow all users to read and add blocks, while private blockchains restrict access to trusted members or participants within a single organization. In some

public blockchains, users may have limited read and write permissions depending on their designated role. Consortium blockchains are regulated by an entire group of organizations and are particularly useful for government entities and their affiliates.

Blockchain technology is not limited to verifying transactions alone. Its numerous applications have made it an ideal example of the "Trust Machine" concept. Instead of relying on trust in each other or an intermediary, the participants in a blockchain network put their trust in the system itself. This decentralized network system is characterized by its ability to guarantee security and transparency in various applications beyond transactions (Miraz and Ali, 2018).

IoT Fundamentals

The Internet of Things is a technology that facilitates the interconnection of diverse devices, sensors, and machines with the internet, enabling them to exchange data and engage in mutual communication. This interconnectedness provides us with many benefits such as automation, optimization, and efficiency in various fields, from home automation to industrial manufacturing. The proliferation of IoT devices is continuously expanding beyond basic machine-to-machine communication and encompasses a diverse range of networks, protocols, and application domains (Benattia and Ali, 2008). The IoT employs a range of wireless technologies, including Bluetooth, WiFi, and Zigbee to establish connectivity between tangible entities and the internet. These wireless technologies allow devices to communicate with each other, collect and transmit data, and receive instructions from a centralized system or cloud-based platform (Feki *et al.*, 2013).

The benefits of the IoT encompass enhanced operational effectiveness, decreased expenses, heightened output, and superior real-time data-driven decision-making capabilities. One instance of this is the use of smart devices in households, which can be operated from a distance, resulting in energy conservation and heightened safety. Similarly, in the manufacturing industry, IoT can streamline production processes, minimize operational downtime, and ensure worker safety. Nevertheless, IoT presents potential hazards, including but not limited to issues related to safeguarding data privacy and protecting against cyber attacks.

IoT establishes an ecosystem of devices that collaborate to accomplish a common objective. Each device collects and shares data, and the system uses this data to make decisions and take action. The system can be controlled remotely, making it convenient and easy to manage. Overall, IoT has

transformed the way we interact with technology, making our lives easier and more convenient.

Overview of an IoT Ecosystem

An IoT ecosystem typically consists of five key components. The first building block of the system is the sensors, which gather and transform data. The second component is a computing node, which contains a central processing unit (CPU) that processes the sensor data. The third constituent is a receiver, which refers to a transceiver that gathers data transmitted by both local and remote computing nodes, along with related devices as well. The fourth component is an actuator, which may be electro-mechanical in nature and functions based on the decision made by the server node. The system receives data from sensors or the internet and subsequently executes a designated action by activating the corresponding device. The final constituent is the device itself, which executes the intended function upon activation by the actuator. The integration of these constituent elements results in a comprehensive IoT ecosystem that has versatile applications across various domains.

Literature Survey

As IoT networks have developed, they have become increasingly centralized. The novel decentralized blockchain system has built-in solutions to many of the fundamental security problems associated with manipulating data from devices. Christidis and Devetsikiotis (2016) investigated the utilization of blockchain technology in the context of IoT. This is done with the goal of improving service and resource sharing and the automation of multiple tedious processes in a safe and efficient manner. It was concluded that blockchain and IoT together are formidable and can open the door to new kinds of distributed applications and business models. The integration of blockchain and the Industrial Internet of Things (IIoT) is the focus of (Zhao *et al.*, 2019). The authors addressed the industrial perspective of this integration and introduced a framework that combines blockchain with IIoT. They presented the fundamental techniques involved in enabling blockchain in the IIoT context. Additionally, the paper addressed the main applications and key challenges associated with this integration. The authors aimed to provide insights into the potential benefits, limitations, and areas of improvement in the integration of blockchain and IIoT in industrial settings.

Kim *et al.* (2017) carried out a formal analysis of popular IoT protocols and identified significant challenges that arise in real-world IoT scenarios.

The study employed formal symbolic modeling techniques to analyze the protocols under different security goals and adversarial scenarios to tackle these challenges. The paper also delves into the realm of cryptographic DoS attacks and highlights the vulnerabilities of most IoT protocols to such resource exhaustion attacks. The researchers proposed a countermeasure for cryptographic DoS attacks that can be universally applied across many IoT protocols. However, the implementation of encryption may not be feasible for mobile devices that have limited storage and computing resources. Therefore, there is a need for lightweight security measures to address this issue.

Zheng *et al.* (2018) conducted a study, where they classified blockchain applications into various domains, including finance systems, reputation systems, public and social services, and security and privacy applications. Additionally, Reyna *et al.* (2018) proposed three distinct communication approaches for integrating blockchain with IoT devices. The first approach enables off-chain interactions, resulting in faster communication as only a fraction of the IoT data is stored on the blockchain. The second approach, known as IoT-Blockchain, ensures that all interactions occur exclusively through the blockchain, guaranteeing complete data storage and traceability. The third approach, referred to as the hybrid approach, combines blockchain and direct interactions between IoT devices, striking a balance between performance and security.

Aman *et al.* (2017) conducted a study that presents a lightweight authentication mechanism that eliminates the need for a centralized server for secret key storage. The authors have identified an issue related to current authentication mechanisms that necessitate the retention of authorization data on a particular authentication server. The present issue is approached through the implementation of physical unclonable functions, which leverage the inherent physical attributes of the devices and exhibit a high degree of resistance to impersonation. The process entails no less than five instances of authentication message exchanges, which could potentially result in additional overhead. Furthermore, it is important to acknowledge that the authentication data relying on PUFs is retained on the hosting node, making it vulnerable to attacks that exploit a single point of failure.

Boudguiga *et al.* (2017) introduced a methodology for updating firmware on IoT devices. The procedure entails the selection of devices to be updated by manufacturers and the subsequent uploading of software to their respective web portals. The IoT devices then receive a notification about the update. Two firmware update methods are

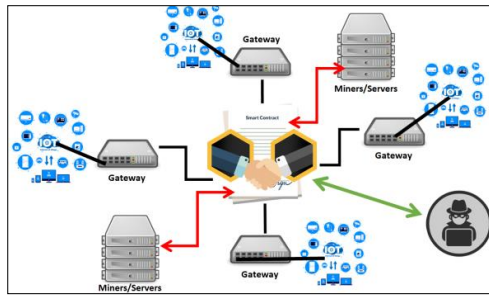


Figure 2. Proposed System Architecture

suggested. The first method involves manufacturers uploading the binary format of the firmware within the transaction and issuing it to the blockchain network. This method preserves the firmware binaries permanently, but it also makes the size of blocks larger. Another option is to create secure and trustworthy nodes. The approach employed by manufacturers involves the exclusion of firmware binaries from the transaction. The secure nodes acquire the firmware in binary format straight from producers for the purpose of verifying the integrity and identifying any potential bugs within the firmware file. The aforementioned methodology could potentially lead to an increase in block sizes.

Yohan and Lo (2018), proposed a solution that outlines a framework to update the firmware of IoT devices. The framework employs blockchain technology to guarantee the secure verification of firmware that is released by manufacturers and to uphold the authenticity of the decentralized firmware to IoT devices. Manufacturers have the ability to disseminate updated firmware versions by releasing the relevant firmware data onto the blockchain network. The architecture of the system incorporates a verification mechanism that utilizes a consensus protocol to authenticate all firmware and smart contracts utilized within the network. Overall, the framework that has been proposed offers a promising and dependable resolution for ensuring secure firmware updates in IoT devices. Nevertheless, there exist apprehensions concerning the possibility of an extravagant influx of traffic and expenses linked with this particular framework. The reason for this is that manufacturers are obligated to implement smart contracts whenever they disseminate novel firmware files or replicate firmware files to intermediary nodes. Moreover, the possibility of update failure increases in the event of gateway compromise.

The utilization of blockchain technology has garnered considerable interest within the realm of cybersecurity, particularly with the inclusion of smart contracts in a DDoS defense mechanism, as outlined in a recent study (Rodrigues *et al.*, 2017).

The study exploits the existing Ethereum public infrastructure by advertising blacklisted IPs suspected of participating in ongoing DDoS attacks. While this work has the advantage of utilizing existing infrastructure and introducing a new security mechanism for simplified data sharing, it still relies on a centralized component for proper system functioning.

Proposed Solution

The IoT has witnessed rapid expansion in recent years, presenting innumerable opportunities for innovation and convenience. However, the escalation in the number of interconnected devices and their susceptibility to cyber-attacks necessitates the need for secure and reliable IoT ecosystems. To address this challenge, this paper presents a blockchain-based framework that can be integrated into an IoT ecosystem's gateway to enhance cyber-security measures for both the installation administrator and the IoT ecosystem's owner.

The architectural overview of the proposed framework in Figure 2, depicts that the smart contract is responsible for authentication, registration, firmware updates, and maintaining a blacklist of malicious IP addresses will be present on a private or permissioned blockchain network. In this architecture, every gateway of the IoT ecosystem will be in contact with the deployed smart contract. Through consensus protocols, any malicious activity can be detected and avoided. This ensures the privacy and security of an IoT ecosystem, as well as the safety of the users and their data. The use of a private or permissioned blockchain network enhances security by limiting access to authorized users only. This architecture offers a reliable and secure solution for IoT ecosystems that are vulnerable to cyber threats.

Moreover, the framework is composed of three main modules. The first module utilizes smart contract technology to register users, obtain their consent, and gather information about administrators and devices present in the ecosystem's network. The second module focuses on addressing the issue of the firmware integrity of IoT devices and offers an appropriate solution. Finally, the third module is responsible for blacklisting malicious IP addresses attempting to enter the network to prevent cyber-attacks.

Methodology

The proposed IoT system, based on a permissioned blockchain, ensures security through controlled

access, identity verification, and auditing capabilities. By restricting participation to a trusted group of entities or nodes who undergo identity verification, the system mitigates the risk of malicious actors, even within the administrative role, infiltrating the network and exploiting users. Additionally, the permissioned blockchain enables auditing and traceability, leveraging the transparency of the technology to record and securely store transactions. This fosters accountability and discourages malicious activities, allowing for the easy tracing of actions taken by participants, including administrators. Hence, the system provides safeguards against potential malicious intentions from administrators, ensuring the security and integrity of user interactions within the IoT ecosystem.

This section of the paper outlines the various modules comprising the blockchain-based IoT architecture. Each module plays a crucial role in ensuring the system's functionality, security, and overall performance. This section provides a concise description of each module, highlighting their purpose and how they contribute to the seamless operation of the system.

1. Device Management Module

This module presents an innovative solution that tightly couples the processes of user and device registration with consent management. The module establishes a unified structure for the administration of said functions through the implementation of a single smart contract. The data stored within the contract is protected by keeping it in the form of anonymized hashes, which can be accessed by the distributed ledger and cross-referenced with local information. This streamlined approach simplifies the registration process for both gateways and devices while also offering a straightforward method for handling user permissions. By bringing together these critical functions, the module provides a secure and efficient system for managing user and device registration and consent management, which can enhance the overall user experience.

This module has been designed to cater to a diverse range of IoT ecosystems, providing a comprehensive solution for managing user and device registration alongside consent management. Within this module, there are three distinct types of users, each performing specific functions to ensure seamless operations. The first type of user is the "Owner," who manages the registration of administrators and maintenance of IoT devices. The second type is the "Administrator," who oversees the entire system, managing access permissions and other administrative tasks. The third type is the

"User," who interacts with the IoT devices and services, providing feedback and managing their own permissions. All three types of users are essential components of any IoT ecosystem; their roles are carefully defined and outlined below:

1.1 Owner

In the context of a smart contract, the individual or organization that deploys the smart contract on the blockchain or obtains the role through a transfer of ownership from a previous owner is known as the contract Owner. This entity holds a significant amount of power within the ecosystem, being able to manage the addition or removal of administrators, Gateways, devices, and certified software hashes. They may also be responsible for creating and deploying the smart contracts that define the rules and logic governing the use of the devices. Owners may include individuals, businesses, or organizations. However, there is one limitation to their authority, as the Contract Owner is not permitted to own a Gateway. The contract owner plays a crucial role in the smooth functioning of the smart contract, responsible for configuring and managing the devices and setting the terms and conditions for the use of the devices.

1.2 Administrator

The Administrator is a crucial entity within the IoT ecosystem, responsible for managing access permissions and enforcing the rules established in the smart contracts. They possess the same privileges as the Owner, but they are unable to add or remove other Administrators. Additionally, Administrators are prohibited from being Gateway owners to avoid conflicts of interest. Their responsibilities include verifying transactions, managing access controls, and ensuring compliance with the rules defined in the smart contracts. The Administrator role may be fulfilled by individuals, businesses, or organizations, appointed either by the owners or as independent entities.

1.3 User

Users are a fundamental entity within the IoT ecosystem, playing an essential role in accessing and utilizing the IoT devices integrated with the blockchain smart contracts. Users can be individuals, businesses, or organizations, and their access to the devices is granted by the owners or administrators. The User, as an individual, has the ability to grant or revoke consent concerning the ownership of a Gateway. However, apart from this specific function, Users do not possess any additional privileges within the system. Users interact with the IoT devices, providing feedback

and managing their own permissions in accordance with the rules and guidelines set by the system. The User's ability to grant or revoke consent provides a crucial mechanism for ensuring transparency and accountability within the IoT ecosystem.

The smart contract will store and manage the information of all the gateways within the IoT ecosystem. The information is stored in the form of an array of objects or structures, with each structure containing specific details about the Gateway and its associated IoT devices. These details include the Gateway ID, status, and address of the Gateway owner (or user). The structure also holds critical information about user consent, ensuring that users have control over their data and how it is used within the system. The Gateway structure comprises the following members:

GW_ID (Gateway ID)

Within the Gateway structure, each Gateway is assigned a unique identifier, referred to as the "Gw_ID." To uniquely identify the Gateway, an identifier can be generated in the form of a distinct string or a hashed string obtained by using the Gateway's MAC address or a hashed representation of it. To ensure security, the derivation process utilizes the secure SHA3 (Keccak256) hashing method. When a new Gateway is added, its unique identifier is checked to make sure there are no other Gateways with the same identifier. To register a Gateway with the smart contract, the identifier must be provided as a required field.

GW_Status (Gateway Status)

The "Gateway status" is an essential component of the Gateway structure, providing valuable information regarding its operational status. This status is typically represented by a Boolean variable that can take on the values of either true or false, which respectively represent active or deleted states. When the gateway is active, it is capable of communicating with the IoT devices, transmitting and receiving data, and performing various functions required for the IoT ecosystem's smooth operation. On the other hand, when the gateway status is deleted, it is no longer operational and is removed from the system.

Last Modification

The "last modification" is an essential member of the Gateway structure, which captures the latest time at which any of the Gateway's properties or members were modified. This timestamp provides a crucial reference point for understanding the current state of the Gateway and its history of changes. Typically, the last modified time corresponds to the timestamp of the block containing the transaction

that modified the gateway. By tracking this information, users can better understand the changes made to the gateway over time, enabling them to make informed decisions about its usage and management.

userAddress (Gateway user's address)

A key member of the Gateway structure is the "userAddress", which holds the critical information needed to associate a gateway with its owner. This information is necessary to restrict consent signing or unsigned to the owner, ensuring that only the authorized owner can access the gateway's functionalities. When registering a Gateway with the smart contract, the owner's address is a mandatory input, and this information is stored securely within the userAddress member.

Consent

Within the Gateway structure of IoT ecosystems, a critical member is the concept of "consent," referring to the explicit permission granted by users for the collection, use, and sharing of their personal data by IoT devices and services. Consent is represented by a Boolean variable that can be set to either true or false, indicating whether or not the user has agreed to the data collection and processing. The concept of consent is a fundamental aspect of data privacy and security in IoT services, particularly given the vast amount of personal data that IoT devices collect, including sensitive information such as location data, health data, and financial data. Obtaining explicit user consent is critical to ensuring that user data is collected, used, and shared responsibly and that users have control over how their personal data is used.

IoTdeviceList (string array of IoT devices)

The "IoTdeviceList" is a fundamental member of the Gateway structure within an IoT ecosystem, responsible for maintaining a list of IoT devices that are connected to the Gateway. The MAC addresses of the IoT devices are stored in an array within the "IoTdeviceList". This ensures that the Gateway can quickly and easily reference each IoT device connected to it.

NumOfIoTdevices (Total Number of IoT devices)

The member variable "NumOfIoTdevices" is an integral part of the gateway structure, playing a critical role in tracking the number of IoT devices registered with the system. This variable is an unsigned integer and holds a record of both active and inactive IoT devices, providing a comprehensive overview of the gateway's current status. It serves as a useful metric for monitoring the

performance of the system, enabling administrators to gauge usage levels and identify any issues that may arise.

The smart contract utilizes a similar methodology for IoT devices as it followed for Gateways, maintaining an array of IoT structures that contain critical information such as the IoT device ID, status, last modification, IoT type, and UUID. By containing an array of objects for IoT structures, the smart contract provides a streamlined approach to managing IoT devices, allowing administrators to monitor device activity, track modifications, and ensure compliance with established rules and regulations. With this approach, the module provides a comprehensive solution for managing both IoT devices and Gateways within a single smart contract.

Here are details of the IoT structure members that are part of the smart contract's list of objects:

Device ID

The "device Id" in the IoT structure serves as a unique identifier for each device. Similar to the gateway ID, the IoT device ID can be a plain string or a hashed version of its MAC address. This ID is a mandatory requirement during the IoT device registration process.

IoT Status

The member variable "device status" is a crucial component of the IoT structure, providing critical information about the status of the device. Similar to the gateway status, the device status can be stored in a Boolean variable, where the values of true or false indicate whether the device is active or deleted.

IoTlastModification (Last Modification)

The "IoTlastModification" member of an IoT structure plays a critical role in tracking any changes made to an IoT Device's properties or members. Typically, the "IoTlastModification" timestamp is set to the same value as that of the block containing the transaction that altered the device properties.

IoT Type

The member variable "IoT Type" is a crucial component of the IoT structure, providing information about the type of IoT device being monitored. It is expressed in the form of a string. This string value denotes the category of the IoT device being monitored, providing valuable information about the device's intended use and capabilities.

IoT uuid

The member variable "IoT UUID" is a crucial component of the IoT structure within the smart

contract's array of objects. The value of this variable is generated through the hashing of a universally unique identifier, which enables administrators to easily add IoT devices to the local gateway database. The use of hashed values ensures the security of sensitive information stored in the database and helps to prevent unauthorized access.

While both a MAC address and a UUID are unique identifiers, they serve different purposes and are used in different contexts. MAC addresses are used for the local identification of devices on a physical network, while UUIDs are used for the globally unique identification of objects in distributed systems.

Algorithm 1 presents the deep execution flow of this module where a smart contract has been created by the name "AuthModule." The purpose of this contract is to enable the authentication and authorization of IoT devices and gateways, which can be added or removed by the contract owner or an administrator.

The logic for the "AuthModule" contract starts by declaring the SPDX license identifier and the Solidity version used for the contract using the "pragma" keyword. The contract is then defined with a private variable "owner" to keep track of the contract deployer. A constructor is defined to set the "owner" variable to the address of the contract deployer. A modifier "onlyBy" is defined to restrict access to certain functions to only specific authorized addresses. A function "changeOwner" is defined to allow changing of the "owner" variable only if the message sender is the current owner.

A mapping variable "adminToBool" (address => bool) is defined to keep track of registered administrators, and a function "registerAdmin()" is defined to allow the contract owner to register a new administrator. Before mapping an admin's address with the boolean value "true," it will be verified that the admin with that particular address is currently mapped with the value "false," indicating that it is unregistered. Once verified, the address will be mapped with the value "true," signifying it is now a registered admin. Any address not registered will have the default value of "false." The "Gateway" structure using the "struct" keyword is defined to represent the gateway devices, which has several properties, including the gateway ID (string), gateway status (bool), last modification time (uint), user address (address), consent status (bool), list of IoT devices (string array) connected to it, and the number of IoT devices (uint). An array "gatewaysArr[]" of the "Gateway" type is defined to store all registered gateway devices.

A mapping variable "GidToJ" is defined to map gateway IDs to their index in the "gatewaysArr[]"

array like this mapping(string => uint256), and a function “addGateway()” is defined to allow the contract owner or administrators to add a new gateway device, and also update the “GidToJ” mapping and the “numOfgateways” variable. This function maps Gateway IDs to unsigned integers, allowing for easy access to the array of Gateways. The integer value associated with a registered gateway falls within the range of {1... numOFgateways}, where numOFgateways represents the overall count of Gateways. The mapping allows for easy addition and removal of gateways in an array. If a gateway is not yet registered, the mapping sets its corresponding integer value to the default value of 0. To randomly access a gateway in the ‘gatewaysArr[]’ array with zero-based indexing, you can use the formula gatewaysArr[GidToJ(ID)-1]. As the GidToJ(ID) mapping function returns a mapped integer value to the specific ID, it can also be written as gatewaysArr[J-1]. To add a new gateway, both GidToJ(ID) or J must be zero, and the new gateway is added to the end of the ‘gatewaysArr[]’ array at the position of the “numOFgateways” variable’s value. To add a new gateway, we simply set its mapping to numOFgateways + 1, where numOFgateways is the current value of the numOFgateways variable’s integer value. This ensures that the new gateway is assigned a unique integer value that is one larger than any existing value.

The algorithm continues by defining a function “removeGW()” to allow the contract owner or administrators to remove a gateway device and also update the “GidToJ” mapping and the “numOFgateways” variable. To remove an existing gateway with an ID equal to (gwID), we set its mapping to GidToJ(gwID) = 0 and its “gw_status” or gateway status to ‘false,’ indicating that it is no longer in use. The process of removal described here does not require any movement or copying of structures. This is advantageous because it reduces the computational complexity and cost associated with blockchain deployments. The algorithm also keeps track of the total number of registered gateways through a global variable, “numOFgateways” to improve performance. Whenever a new gateway is added, this variable increases by 1, and it decreases by 1 when a gateway is removed. By retaining this number in the contract, searching operations necessary for get-type functions are avoided, which further reduces the cost of deploying the contract. It’s important to note that removing a Gateway also removes all the IoT devices associated with it.

Following the completion of the entire logic for changing the owner, registering admins, and adding/removing gateways, the struct “IoT” is

defined to represent the IoT devices. A mapping “IoTidToJ” is defined to map device IDs to their index in the “iotDevicesArray” array, and a function “addIoTDevice” is defined to allow the contract owner or administrators to add a new IoT device and also update the “IoTidToJ” mapping and the “numOFiotDevices” variable. IoT devices are registered using a similar process, which involves registering them through a mapping process, where their “device_id” is mapped to an unsigned integer J using the mapping method “mapping (string => uint256) public IoTidToJ” to enable random access to an array of IoT structures. To access elements of the IoT array, the formula iotDevicesArray[IoTidToJ(device_id) - 1] = iotDevicesArray[J-1] is utilized, with addition only allowed when IoTidToJ(device_id) equals zero. In order to delete a device, its status in a variable “IoT_status” is set to “false” and IoTidToJ(device_id) or J is set to zero. Additionally, each gateway keeps track of the count of registered IoT devices through the “numOfIoTdevices” variable to avoid search operations during the call of get type functions, which helps reduce the cost of contract deployment and improves overall performance. Finally, it’s important to note that only administrators are authorized to register or delete gateways and IoT devices.

The smart contract provides users with several “get” functions to retrieve information about gateways and IoT devices. These functions can be used to obtain information about all gateways, a specific gateway with a gateway id “gwID,” or all IoT devices associated with a specific “gwID.” The SC has implemented the following primary functions to support this functionality:

1. Change of ownership

The ‘changeOwner()’ function allows the contract owner to change ownership of the contract to a new address. It checks whether the caller is the owner and, if so, updates the owner's address with the new owner’s address provided as a parameter.

2. Registration of administrators by the owner

The ‘registerAdmin()’ function allows the owner to register a new administrator by mapping the address of the new admin to a Boolean value. The function checks whether the caller is the owner and whether the new admin has not already been registered. If the conditions are met, the admin's address is added to the mapping, and a message is returned indicating success.

3. Gateway registration

The “addGateway()” function allows the owner or an admin to add a new gateway to the

“gatewaysArr[]” array. The function checks whether the caller is authorized and whether the gateway ID provided as a parameter is a 12-byte string and has not already been registered. If the conditions are met, a new gateway object is created and added to the array, and a success message is returned.

4. Get the device list associated with gateway

The “getDvListByGWid()” function allows the owner or an admin to retrieve the list of IoT devices associated with a particular gateway. The function checks whether the caller is authorized and whether the gateway ID provided as a parameter is valid. If the conditions are met, the list of IoT devices associated with the gateway is returned.

5. Removal of gateway

The “removeGW()” function allows the owner or an admin to remove a gateway from the ‘gatewaysArr[]’ array. The function checks whether the caller is authorized and whether the gateway ID provided as a parameter is valid. If the conditions are met, the gateway’s status is updated to “false” and its “gwID” is mapped to 0, and the gateway is removed from the array.

Addition of an IoT device

The “addIoTDevice” function allows the owner or an admin to add a new IoT device to the “iotDevicesArray[]” array. The function checks whether the caller is authorized and whether the device ID and gateway ID provided as parameters are 12-byte strings and whether the device has not already been registered. If the conditions are met, a new IoT device object is created and added to the array, and a success message is returned.

Get device details

The function “getDvcDtlsByID()” takes a device ID as input and returns the device details from an array of registered IoT devices named “iotDevicesArray[].” The “require” statement at the beginning of the function ensures that the caller of this function is either the owner or an admin. Additionally, it verifies that the length of the device ID is exactly 12 characters and that the device ID is not empty. If these conditions are not met, an error message will be displayed. If the caller is authorized and the device ID exists in the “iotDevicesArray[],” the function returns the details of the device.

Get last used index

The “getLastUsedIndex()” function is used internally to get the last used index of the IoT device in a gateway’s list of IoT devices.

Delete IoT device

The function “dltioTDvcById()” deletes a registered IoT device by its device ID from both the “iotDevicesArray[]” array and the gateway’s list of IoT devices. The function first checks if the caller of this function is either the owner or an admin and verifies that the device ID is not empty. Then it removes the device from the “iotDevicesArray[]” array by changing the “IoT_status” to “false” and setting the “IoTidToJ” mapping to 0. It also decrements the number of IoT devices by 1. Next, the function removes the device from the gateway’s list of IoT devices by first finding the index of the device and then shifting the elements up to fill the gap. Finally, it deletes the last element in the list and decrements the number of IoT devices in the gateway.

Algorithm 1: Proposed SmartHubNet algorithm for decentralized management of the IoT ecosystems

Input: gateway id, status, consent, IoT device id, IoT status, IoT type

Output: Ownership changed, Gateway registered or Deleted, IoT device Registered or Removed

```

1: Begin
2: pragma solidity 0.8.8;
3: address private owner;
4: contract AuthModule{
    // assigning contract deployer as the owner
    constructor(){ owner=msg.sender; }
5:     modifier onlyBy(address _account) {
// creating a modifier
        require(
// message sender should be contract owner
            msg.sender == _account,
            "Sender not authorized.");
        }
6:     // function for change of ownership
        function changeOwner(address
        _newOwner) public onlyBy(owner) {
            owner = _newOwner;
        }
7:     mapping(address=>bool) adminToBool;
8:     function registerAdmin(address adrs)
public returns(address,string memory){
        if((msg.sender==owner) &&
        adminToBool[adrs]==false) then
            // map the address to true
            return (adrs,"Address registered as admin");
        }
9:     uint256 public numOFgateways;
10:    struct Gateway{ // define structure
        members and their data types here
    }

```

```

11:     Gateway[] public gatewaysArr;
12:     mapping(string => uint256) public GidToJ;
13:     function addGateway(gwID,status,consent)
public returns(string memory, Gateway memory){
    // logic for adding gateway to the gateway
    array here}
14:     function getDvListByGWid(gw_id) public
view returns(string[] memory){
    // get list of devices associated to gateway id }
15:     function removeGW(string memory gwID)
public returns(string memory){
    // logic for removing a gateway }
16:     uint256 public numOFiotDevices;
17:     struct IoT{ // define structure members
and their data types }
18:     IoT[] public iotDevicesArray;
19:     mapping(string => uint256) public IoTidToJ;
20:     function addIoTDevice(gw_id, device_id,
IoT_status, IoTtype, IoTuuid) public returns(string
memory){
    // logic for adding an IoT device to a gateway
    return "Device Added Successfully";
    }
21:     function getDvcDtlsByID(device_id)
public view returns(IoT memory){ }
22:     function dlIoTDeviceByGWid(gw_id, device_id)
public returns(string memory){
    // logic for deleting an IoT device
    }
23: } // contract end
24: END

```

Firmware Integrity Module

The firmware integrity module of the proposed system is a critical component that ensures the security and protection of gateways and IoT devices from malicious firmware. This module utilizes a smart contract acting as a storage directory to store valid firmware versions and their corresponding hashed values. The smart contract is administered and owned by an authorized party who maintains records of the hashes of all authorized firmware and software versions.

A background service or program runs on the gateway of the IoT ecosystem, which periodically generates the hash of the firmware installed on the system. This hash, along with the firmware version, is then sent to the deployed smart contract for verification. The smart contract compares the received firmware version and its hashed value with the already stored valid firmware information. If the comparison indicates that the firmware is valid, the smart contract returns a confirmation to the gateway service, and the service continues to work normally. However, if the comparison indicates that the firmware is not valid, which indicates that it may be malicious, the gateway service raises an alert to the

administrator. The administrator is then notified, and the smart contract is commanded to send a valid firmware update to the gateway, as the current firmware is deemed malicious. The service installs the new and valid firmware, which has been previously tested and found successful in research (Anastasiou *et al.*, 2020). In the context of firmware updates for IoT devices, Anastasiou *et al.* (2020) introduced a blockchain-based framework that utilizes the LoRa communication protocol. The framework ensures secure firmware updates, addressing the challenges of network size and firmware size through an evaluation process. The evaluation highlights the necessity of leveraging multiple gateways for improved reliability and performance during the firmware update process.

Implementing a blockchain-based system for updating IoT devices brings numerous benefits when compared to existing solutions. The decentralized and immutable nature of blockchain ensures the integrity and security of updates, mitigating the risks associated with centralized systems. By distributing verification tasks across multiple nodes, blockchain-based updates enhance resilience and reduce the potential for tampering. The use of smart contracts streamlines the update process, automating tasks and eliminating intermediaries, resulting in improved efficiency, cost reduction, and faster deployment of critical updates. Overall, this approach enhances the functionality and security of IoT devices.

This firmware integrity module ensures that only authorized and valid firmware versions are installed on gateways and IoT devices, preventing the installation of malicious firmware that could potentially compromise the security and integrity of the system. By utilizing a smart contract as a storage directory and conducting periodic verification of firmware, this module adds an additional layer of security to the proposed system, enhancing its overall resilience against potential cyber threats. Figure 3, depicts the sequence of actions involved in the firmware integrity module of the proposed system. It outlines the various interactions between the gateway service, smart contract, and administrator, demonstrating how the module ensures the integrity of firmware installed on IoT devices. The diagram provides a clear and concise visual representation of the firmware verification process, making it easier to understand and implement the module in practice.

IP Filtering Module

The proposed IP filtering module adds an additional layer of security to the IoT ecosystem. With the increasing number of IoT devices being deployed, it is essential to prevent malicious actors

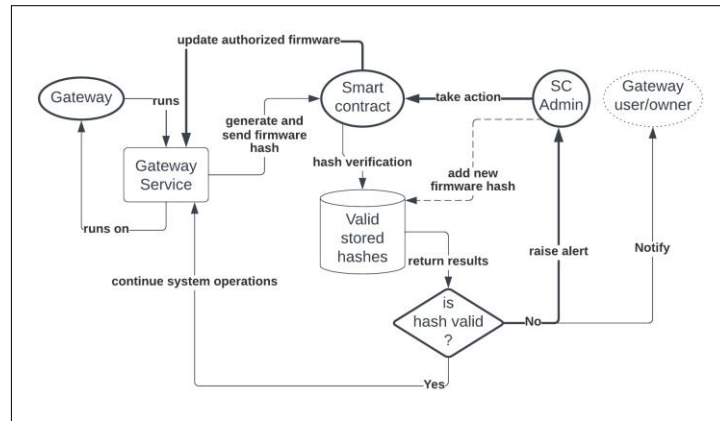


Figure 3. Firmware Integrity Module's Flow Diagram

from compromising the system. One common attack vector is the exploitation of vulnerabilities in IoT devices by using malicious IP addresses.

Whenever an IP address attempts to act maliciously, the IoT gateway will run a service that sends the IP address to a smart contract to check the reputation score. The smart contract will retrieve the score by calling a reputation score API, which provides a score that indicates the reputation of the IP address. The gateway will then use the score to determine whether to allow or block traffic from that IP address. If the reputation score is low, the gateway may block traffic from the IP address or subject it to further inspection before allowing it through.

Moreover, the malicious IP address and its mapped score will be stored in a blacklist directory maintained in the smart contract, making it visible to all gateways. When a gateway requests again for the reputation score of an IP, the smart contract searches in its directory first. In case of a repeated request for the already stored malicious IP, the mapped score will be reduced by a certain extent, and it will increase if no requests are received for a specific time period. This approach ensures that the system adapts to changes in malicious activity and can automatically update the reputation scores.

This proposed approach has been previously implemented and found to be effective (Giannoutakis *et al.*, 2020). By adding this additional layer of security, the IoT ecosystem can better protect itself against malicious actors and ensure the safe and secure operation of IoT devices.

Experiment Results

experimental results obtained from testing the implemented smart contract. Our experiments specifically focused on evaluating the performance

of algorithms utilized in the proposed framework. By conducting experiments using these algorithms, we aimed to assess their effectiveness and suitability within the context of the developed smart contract. The obtained results will be discussed in detail, shedding light on the performance and capabilities of the implemented framework.

Environment setup

The smart contract implementation was conducted using the remix-Ethereum IDE, a platform that facilitates the deployment and testing of smart contracts. This open-source interface allows developers to write and test smart contract code in the widely recognized solidity language. The environment setup for the Authmodule. sol smart contract can be seen in Figure 4, showcasing the tools and configurations used during the implementation process. Figure 5, demonstrates the successful deployment of the smart contract, initiated from the account address 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 and deployed to 0xd9145CCE52D386f254917e481eB44e9943F39138.

Cost Evaluation

The gas serves as a cryptocurrency unit within the Ethereum network, and it is deducted from users' accounts when they engage in transactions. The gas consumption of a smart contract's operation is depicted in Figure 6. In our proposed work, the smart contract incorporates four distinct functions including deployment of contract: (1) add gateway, (2) add IoT device, (3) change owner, and (4) remove gateway. The gas consumed by these functions varies based on the complexity of the smart contract. Notably, deploying the smart contract is a resource-intensive operation in Ethereum, requiring 3009349 gas in our case.

Additionally, specific transactions within the contract have associated gas costs: adding a gateway consumes 251598 gas, adding an IoT device utilizes 264875 gas, changing ownership incurs 27149 gas, and removing a gateway necessitates 35747 gas. These gas values demonstrate the gas usage associated with each function and underscore the cost considerations involved in deploying and executing transactions within the Ethereum network.

Discussion and Conclusions

The considerable interest that blockchain technology continues to attract from both research and industrial circles suggests its potential as a viable means of enabling the forthcoming foundational infrastructure for the Internet of Things. Incorporating blockchain-based systems in various industries and regions introduces regulatory compliance challenges due to the decentralized and immutable nature of the technology. Additional measures must be implemented to ensure compliance with data protection, privacy, and other regulations. When comparing blockchain technology with IoT and AI/ML technology in IoT ecosystems, each approach offers unique benefits and considerations. Blockchain enhances security and trust by providing immutable and auditable transaction records, mitigating the risk of unauthorized data tampering. However, its distributed nature can pose scalability and latency challenges for processing large volumes of IoT data. On the other hand, AI/ML technology enables intelligent data analysis, real-time insights, anomaly detection, and advanced decision-making capabilities. However, it requires significant computing power and continuous training with large datasets, potentially leading to resource constraints and operational complexities. In this paper, we proposed an innovative solution to enhance the security of IoT ecosystems using blockchain technology. The decentralized approach offers a strong and dependable mechanism for user and device registration, software and firmware integrity verification, and malicious IP blacklisting. The smart contracts developed through the proposed approach were successfully deployed on a test network, demonstrating the practicality and effectiveness of this solution. With the rapid expansion of the IoT ecosystem and its increasing vulnerability to security breaches, this approach has the potential to offer a robust solution to enhance the security of IoT systems.

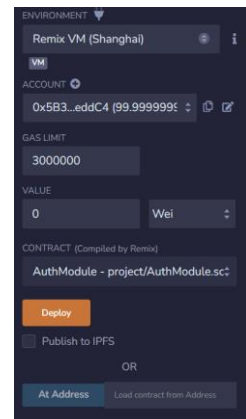


Figure 4. Environment Setup

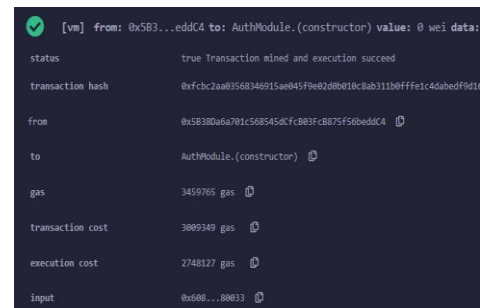


Figure 5. Smart Contract Deployment Cost Evaluation Environment Setup

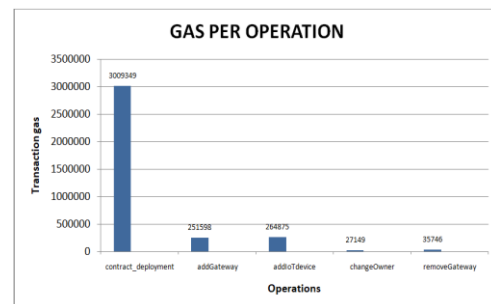


Figure 6. Transactional gas associated per operation

Future Scope

The potential for future advancements in this approach lies in the exploration of integrating emerging technologies, such as artificial intelligence and machine learning, to optimize the system's security and efficiency. Furthermore, it is possible to conduct research aimed at optimizing the consensus mechanism employed in the private blockchain network with the objective of enhancing its scalability and transaction speed. The suggested

methodology has the potential to be applied to various fields, including industrial IoT and healthcare, where safeguarding data confidentiality and privacy is of paramount significance. The investigation of user-friendly interfaces and APIs can be undertaken to facilitate the acceptance of IoT security systems based on blockchain by end-users and developers.

References

- Aman, M.N., Chua, K.C., and Sikdar, B. (2017). Mutual authentication in IoT systems using physical unclonable functions. *IEEE Internet of Things Journal*, 4(5):1,327-1,340. <https://doi.org/10.1109/JIOT.2017.2703088>
- Anastasiou, A., Christodoulou, P., Christodoulou, K., Vassiliou, V., and Zinonos, Z. (2020). IoT Device Firmware Update over LoRa: The Blockchain Solution. 2020 16th International Conference on Distributed Computing in Sensor Systems (DCOSS), Marina del Rey, CA, USA, p. 404-411. <https://doi.org/10.1109/DCOSS49796.2020.00070>
- Benattia, A., and Ali, M. (2008). Convergence of technologies in the machine-to-machine (M2M) space. In *Proceedings of the IEEE International Conference on Applied Electronics*. p. 9-12.
- Boudguiga, A., Bouzerna, N., Granboulan, L., Olivereau, A., Quesnel, F., Roger, A., and Sirdey, R. (2017). Towards Better Availability and Accountability for IoT Updates by Means of a Blockchain. 2017 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), Paris, France, p. 50-58. <https://doi.org/10.1109/EuroSPW.2017.50>
- Christidis, K., and Devetsikiotis, M. (2016). Blockchains and Smart Contracts for the Internet of Things. in *IEEE Access*, 4:2,292-2,303. <https://doi.org/10.1109/ACCESS.2016.2566339>
- Feki, M.A., Kawsar, F., Boussard, M., and Trappeniers, L. (2013). The Internet of Things: the next technological revolution. *Computer*, 46(2):24-25. <https://doi.org/10.1109/MC.2013.63>
- Giannoutakis, K.M., Spathoulas, G., Filelis-Papadopoulos, C.K., Collen, A., Anagnostopoulos, M., Votis, K., and Nijdam, N.A. (2020). A Blockchain Solution for Enhancing Cybersecurity Defence of IoT. 2020 IEEE International Conference on Blockchain (Blockchain), Rhodes, Greece, p. 490-495. <https://doi.org/10.1109/Blockchain50366.2020.00071>
- Jozef, Mocnej., Winston, K.G. Seah., Adrian, Pekar., and Iveta, Zolotova. (2018). Decentralised IoT Architecture for Efficient Resources Utilisation, *IFAC-PapersOnLine*, 51(6):168-173. <https://doi.org/10.1016/j.ifacol.2018.07.148>
- Kholoud, Y. Najmi., Mohammed, A. AlZain., Mehedi, Masud., N.Z. Jhanjhi., Jihad, Al-Amri., and Mohammed, Baz. (2021) A survey on security threats and countermeasures in IoT to achieve users confidentiality and reliability, *Materials Today: Proceedings*, 8(2):377-382. <https://doi.org/10.1016/j.matpr.2021.03.417>.
- Kim, J.Y., Holz, R., Hu, W., and Jha, S. (2017). Automated analysis of secure Internet of Things protocols. In *Proceedings of the 33rd Annual Computer Security Applications Conference*, p. 238-249. <https://doi.org/10.1145/3134600.3134624>
- Miraz, M.H., and Ali, M. (2018). Blockchain Enabled Enhanced IoT Ecosystem Security. In: Miraz, M., Excell, P., Ware, A., Soomro, S., Ali, M. (eds) *Emerging Technologies in Computing. iCETiC 2018. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, vol 200. Springer, Cham, p. 38-46. https://doi.org/10.1007/978-3-319-95450-9_3
- Reyna, A., Martín, C., Chen, J., Soler, E., and Díaz, M. (2018). On blockchain and its integration with IoT. *Challenges and opportunities*. *Future generation computer systems*, 8:173-190. <https://doi.org/10.1016/j.future.2018.05.046>
- Rodrigues, B., Bocek, T., Lareida, A., Hausheer, D., Rafati, S., and Stiller, B. (2017). A Blockchain-Based Architecture for Collaborative DDoS Mitigation with Smart Contracts. In: Tuncer, D., Koch, R., Badonnel, R., Stiller, B. (eds) *Security of Networks and Services in an All-Connected World. AIMS 2017. Lecture Notes in Computer Science()*, vol 10356. Springer, Cham. https://doi.org/10.1007/978-3-319-60774-0_2
- Yohan, A., and Lo, N.W. (2018). An Over-the-Blockchain Firmware Update Framework for IoT Devices. 2018 IEEE Conference on Dependable and Secure Computing (DSC), Kaohsiung, Taiwan, p. 1-8. <https://doi.org/10.1109/DESEC.2018.8625164>
- Zhao, S., Li, S., and Yao, Y. (2019). Blockchain-enabled industrial Internet of Things technology. *IEEE Transactions on Computational Social Systems*, 6(6):1,442-1,453. <https://doi.org/10.1109/TCSS.2019.2924054>
- Zheng, Z., Xie, S., Dai, H.N., Chen, X. and Wang, H. (2018). Blockchain challenges and opportunities: A survey. *International journal of Web and Grid Services*, 14(4):352-375. <https://doi.org/10.1504/IJWGS.2018.095647>