Noname manuscript No.
(will be inserted by the editor)

A Decentralized Lightweight Blockchain-based Authentication Mechanism for IoT Systems

Umair Khalid · Muhammad Asim · Thar Baker · Patrick C. K. Hung · Muhammad Adnan Tariq · Laura Rafferty

Received: date / Accepted: date

Abstract The Internet of Things (IoT) is an emerging paradigm branded by heterogeneous technologies composed of smart ubiquitous objects that are seamlessly connected to the Internet. These objects are often deployed in open environments to provide innovative services in various application domains such as smart cities, smart health, and smart communities. These IoT devices produce a massive amount of confidentiality and security-sensitive data. Thus, security of these devices is very important in order to ensure the safety and effectiveness of the system. In this paper, a decentralized authentication and access control mechanism is proposed for lightweight IoT devices and is applicable to a large number of scenarios. The mechanism is based on the technology of the fog computing and the concept of the public blockchain. The results gained from the experiments demonstrate a superior performance of the proposed mechanism when compared to a state-of-the-art blockchain-based authentication technique.

Keywords Internet of Things · Blockchain · Fog computing · Authentication · Smart city

Umair Khalid
FAST-NUCES, Islamabad, Pakistan (E-mail: i171016@nu.edu.pk)

*Corresponding author: Muhammad Asim
FAST-NUCES, Islamabad, Pakistan (E-mail: Muhammad.Asim@nu.edu.pk)

Thar Baker
Liverpool John Moores University, Liverpool, UK (E-mail: t.baker@ljmu.ac.uk)

Patrick C. K. Hung
Faculty of Business and Information Technology, Ontario Tech University, Canada
(E-mail: patrick.hung@uoit.ca)

Muhammad Adnan Tariq
FAST-NUCES, Islamabad, Pakistan (E-mail: muhammad.adnan@nu.edu.pk)

Laura Rafferty
Faculty of Business and Information Technology, Ontario Tech University, Canada
(E-mail: laura.rafferty@uoit.ca)

1 Introduction

The Internet of things (IoT) is an emerging technology, which acts as an enabler to interconnect intelligent and self-configurable devices "things" to establish an efficient and dynamic platform for communication and collaboration [1, 6]. These devices are heterogeneous and resource-constrained in term of storage, power and computation [3]. According to the recent study published by Gartner, the number of connected devices will raise up to 20 billion by the year 2020 [2]. Currently, IoT participates in almost every field of life (e.g., healthcare and transportation), while there are several IoT applications that automate daily processes (i.e., home automation). These applications include intelligent waste management systems, intelligent transportation systems, smart grid, smart parking, environmental monitoring, traffic management and multiple other applications [3, 31, 56].

IoT devices generate a huge amount of data, some of which may be confidential. For example, in smart health-care systems, the devices attached to patients generate sensitive data, such as patient personal health status. This data is then, sent to the hospital and is constantly monitored to trigger alarms in emergency situations. Thus, the security of these devices along with the sensed data is essential to ensure the normal behaviour of the IoT system because all-important decisions made by the IoT system are based on this sensed data [32]. If any malicious device gets into the IoT network, it may disrupt the normal functionality of the system and result in disastrous situations. There are several dimensions of IoT security including data aggregation [44], confidentiality [45], integrity [46], availability [47] and non-repudiation [48]. However, authentication and access control are the first line of defence, which limits data access to only those with the correct permissions. Mutual authentication between IoT devices and other systems is an integral part of secure IoT systems to ensure data confidentiality and integrity, Otherwise, these systems will be vulnerable to a variety of potential security threats, such as unauthorized access, information theft, and data alteration [4].

IoT systems are expected to operate in a distributed manner with a stringent minimum-delay requirement so, that, devices from various IoT systems can interact with each other to provide value-added services. Therefore, distributed security measurements are required to secure such systems. Existing security approaches (such as authentication mechanisms) are not suitable for IoT systems because of their centralized nature and scalability issues. Moreover, the need for delay-sensitive authentication and access control mechanisms has become even more imperative for systems where IoT devices are often resource-constrained and heterogeneous and must operate and perform in safety-critical settings. For example, a battery-powered aerial drone may need to authenticate with multiple command stations in a short period for the exchange time-critical data [11]. Latency issues can be addressed by employing fog nodes that put a substantial amount of communication, control, storage and management at the edge of a network as opposed to establishing dedicated channels to a more centralized remote infrastructure [33, 34].

The recent emerging blockchain technology can be a suitable solution to provide authentication and access control services in IoT [3,35], mainly due to its decentralized nature and cryptographic properties. Blockchain was initially utilized in the money exchange protocol known as Bitcoin. However, security experts around the world are concentrating on the blockchain to fortify privacy and security issues in IoT. The characteristics of blockchain such as improved reliability, unforgeability and fault tolerance make blockchain a good solution for authentication problems. Blockchain also allows the integration of smart contracts that offer a fine-grained access control mechanism for IoT devices. Moreover, the blockchain technology and fog computing offer good grounds to build and manage distributed and decentralized trust and security solutions for time-sensitive fog-enabled IoT systems [34]. For example, the work of [54] proposed a blockchain-based framework for securing connected and autonomous vehicles. Similarly, the authors in [55] presented a profitable and energy-efficient cooperative fog-based solution for IoT services.

Benefiting from the characteristics of fog computing and the distributed nature of blockchain, we propose a delay-sensitive blockchain-enabled authentication and access control mechanism for IoT systems. The main contributions of this paper are given below:

- A novel decentralized mechanism that provides authentication and access control to create a controlled and secured environment for IoT systems.
- A proof-of-concept of the proposed mechanism with results that show clearly its ability to meet the IoT security requirements.
- Comparison with a state-of-the-art IoT authentication technique to evaluate the performance of the proposed mechanism.

1.1 Use Case Study - Smart Hospital

We use smart hospital as a case study due to its significance in IoT-based smart city scenarios. Moreover, it helps in better understanding the research problem and analyzing the importance of intra-system communication between IoT devices from different systems.

IoT systems are now transforming from centralized to distributed systems in which two or more IoT systems need to communicate with each other to provide valued services to users. Such a scenario is presented [36], in which the example of a smart healthcare system is presented. In the given scenario, a patient is discharged from a hospital, but he/she is still required to remain under observation. The doctors attach health monitoring devices such as heartbeat sensors and blood pressure monitoring devices to the body of the patient before he/she leaves the hospital. These devices sense the heart-beat and blood pressure of the patient and send it to the hospital system through a secure channel. However, these devices will also communicate with the devices of the patient's smart home. For instance, an emergency alarm that will be triggered automatically if the condition of the patient becomes critical. To check the

availability of hospital beds in a smart city, the devices belonging to the hospitals must communicate with each other so that the correct count of hospital beds can be provided. The interaction between devices from multiple systems led to the formation of a distributed system. Therefore, to ensure the existence of legitimate devices in these systems, a distributed authentication mechanism is required. In such a mechanism, if a device is authenticated with one system, it does not need to be re-authenticate with other systems. Such as in our example, if the medical sensor device is authenticated with the hospital system, it need not to be authenticated again with the smart home.

Furthermore, the use of access control is also critical in distributed systems where devices belong to different systems [7]. If access control policies are not defined in distributed systems, any device can have access to the data of any system without any restriction that could cause security or privacy issues. An access control mechanism that supports the distributed nature of IoT systems is of imperative need. Moreover, in centralized access control mechanisms, every system defines its own access control policies. Thus, the scope of the policies is limited to individual systems and is not suitable for distributed environments.

1.2 Problem Statement

Authentication and authorization are indispensable means to achieve the identification of legitimate entities and secure the communication in a network. Due to the heterogeneous and resource-constrained nature of IoT devices, conventional mechanisms are not appropriate and thus, a lightweight authentication mechanism is needed that requires low computation power, less storage capacity and have low latency and communication overhead. Current state-of-the-art authentication and authorization mechanisms proposed for IoT are centralized and offer high communication overhead, which results in higher energy consumption. In centralized mechanisms, such as in [30,8], either Trusted Third Party (TTP) or Centralized Authority (CA) is usually required for authentication and authorization of IoT devices. Such mechanisms are highly prone to security threats, scalability and single point of failure issues. Moreover, in case of a Third Trusted Party (TTP), privacy concerns rise, which causes less reliability and confidence in data sharing. To overcome these issues, a decentralized authentication and authorization mechanism is proposed in [11], but it has limited scope in communication. The principal issue in this mechanism is that the devices can only communicate with the devices of its own group. The devices are not allowed to communicate with the devices belonging to other groups or systems. Additionally, communication overhead is also an issue where the number of messages required for the completion of the authentication process is significantly high. For example, the authentication mechanisms proposed in [9] require at least 5 and 8 exchanges of association messages for authentication. This may not only add an additional delay, but also uses extra resources of the resource-constrained IoT devices.

1.3 Paper Structure

This paper is organized as follows: Section 2 introduces the blockchain technology and presents the related work. Section 3 explains our decentralized blockchain based mechanism. The experimental set up and results are articulated in Section 4. Finally, Section 5 concludes the paper and identifies some future work.

2 Background

This section first provides a brief overview of the blockchain technology, followed by a discussion on state-of-art authentication and access control mechanisms for IoT.

2.1 Blockchain

Blockchain can be defined as a distributed ledger that maintains permanent records of transactions executed and processed in the network. The blockchain is decentralized in nature and applied on a peer-to-peer network. Each node of the network on which blockchain is implemented maintains a full copy of the ledger. On the validation of each transaction, these ledgers update continuously [10].

The blockchain was initially conceived as the financial transaction protocol and was first used in Bitcoin. However, its characteristics such as unforgeability, decentralized nature and fault tolerance make it suitable for the cybersecurity ecosystem. Currently, there are several security mechanisms [11,26,27] including authentication and access control that use the blockchain technology to provide the essential security parameters to secure a system. The ledger includes the number of blocks that are chained together with a hash mechanism. Each block consists of two parts in which the first part contains the number of transactions that are executed and validated. These transactions could be a health record, a financial transaction, or a network communications message. These mechanisms are organized using different data structures. For example, in the Merkle tree data structure, the reverse hash mechanism is used, and the central root hash saved as the block hash [19]. The second part of the block is called block header in which the header information like transaction time-stamp, hash of that block and hash of the previous block is stored. Thus, a set of existing blocks are joined together forming a hash-supported chain. As the length of the chain increases, more resilient it becomes against falsification. Moreover, if a malicious user wants to change or alter the transactions of a block, he/she must make the corresponding changes in all the subsequent blocks as they are linked through hashes. Figure 1 shows the basic structure of blockchain that is implemented on the network.

There are mainly two types of nodes in the blockchain. The first type of nodes is known as a passive node or validating node and are responsible for the

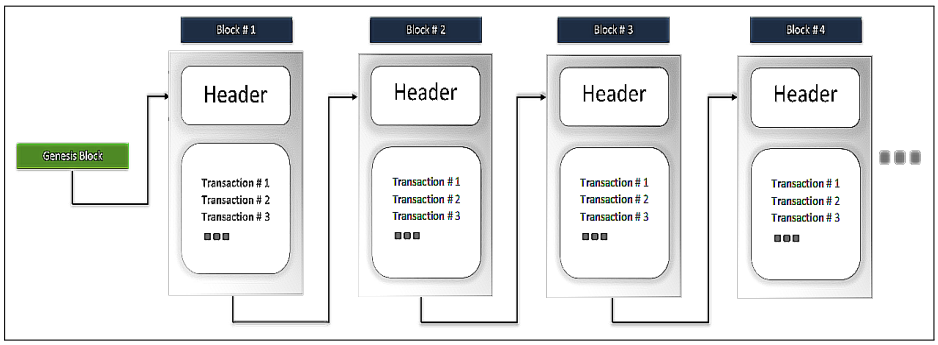


Fig. 1: Basic structure of blockchain

storage and reading of the block data, but they cannot create a new block or trigger transaction. The second type of node is known as miner node and can also generate a block and validate transactions. In order to validate new blocks and attach them to the genesis blockchain, a number of consensus algorithms are used. The consensus process enable blockchain network nodes to agree upon adding a new block to the blockchain. One such consensus algorithm in the Bitcoin network is called Proof-of-Work (PoW). In PoW algorithm, a mathematical puzzle is added that must be solved by miner nodes to validate a block. The difficulty of the puzzle can be changed according to the computation power of the miner nodes, and the time needed to validate new blocks. The PoW algorithm is used in environments where there is no restriction on computation power [50]. Miners are rewarded for adding new blocks, which is one of main motivation for nodes to participate as minors. Apart from PoW, there are other alternative consensus algorithms including the Proof-of-stack (PoS) algorithm [22], Byzantine Fault Tolerance algorithm [21], and the Ripple algorithm [20]. To address the limitations of PoW algorithm, the PoS is proposed. In PoS algorithm, there are nodes called forgers that validate the new blocks. The forgers are selected by the size of their account balance, and the amount they are willing to lock up as a stack. The node having a higher stake has a greater chance to validate a block and attach it to the genesis blockchain. Ripple utilizes XRP Ledger Consensus Protocol to achieves consensus in the network. This protocol makes use of a Unique Node List (UNL) maintained by each node in the ripple network. The transactions are batched in a data structure named Candidate Set. Each transaction in the candidate set requires 80% votes to become part of the ledger [51]. The protocol is energy efficient. However, it is generally considered centralized as it requires vote from selected nodes.

The selection of these algorithms depends on the application area and features like minimum resource requirements, ease of access, latency, and privacy [23].

2.2 Related Work

This section discusses the related work of the authentication and access control mechanisms proposed for IoT.

Fan et al. [13] proposed an authentication mechanism that is based on wearable devices, smart-phones and a cloud server. The manufacturers of the smart-phones and wearable devices add details of the devices to the cloud. To save space in the wearable device and smart-phone, the sensed data is transferred to the cloud. Symmetric key cryptography is employed in the proposed mechanism. The main problem with the mechanism is that, all symmetric keys are stored in a centralized server making the mechanism prone to a single point of failure attack.

In [18], authors proposed a proximity-based authentication mechanism between the smart-phone and the IoT device. The problem defined by the researcher is that the existing mechanisms don't provide security against physical attacks like impersonation attack and device capturing. In the proposed mechanism researchers used the RSS signal variation concept and to match the variations with the real one, RSS-trace concept is used. The issue with the proposed work is that it requires the storage of authentication data on a centralized local server. Moreover, the mechanism requires the devices to be close enough if they want to authenticate each other.

Aman et al. [8] proposed a lightweight authentication mechanism without the need for storing the secret keys in a central server. The problem identified by the researcher in this paper is that the existing mechanism require to store authentication credentials on a secure server. The proposed mechanism is based on Physical Unclonable Functions (PUFs), the physical functions that are based on the physical properties of the devices and are nearly impossible to impersonate. The issue with the proposed mechanism is that it requires at least 5 exchanges of authentication messages. Moreover, PUF based authentication data is stored on the server node that is prone to single point of failure attack.

In [12], the authors proposed an efficient authentication mechanism for end devices that require very less information about devices thus increasing its privacy. The mechanism is composed of short group signatures and Shamir's secret sharing scheme. The Shamir's secret sharing scheme is used for the establishment and distribution of encryption keys for secure communication while a short group signature provides anonymity to the signers' identity. The main problem with the proposed mechanism is that it needed to store the secret in every device to anonymize the identity and thus requires high storage. Moreover, the mechanism requires a higher number of message exchange for authentication.

Prosanta et al. [16] proposed a lightweight two-factor authentication mechanism for IoT devices that not only authenticate the IoT device but also incorporate the physical properties of the device to identify and eliminate the physical attacks on the device like impersonation attacks and side channel attacks. The main limitation in the proposed mechanism is that the authenti-

cation data is stored in a centralized server. Moreover, the mechanism requires 5 exchanges of messages between the device and the server.

In [5], authors focused on the problem of remote attestation mechanisms of the devices for either IoT or Cyber-Physical Systems. The devices involved in such systems are resource constraint and thus could not handle complex computation. The authors proposed a software-based remote authentication mechanism that is based on the physical behaviour of the devices. In the proposed mechanism the memory is filled, and the filling rate is monitored. The mechanism requires the storage of hardware properties on a local server. Every time a device wants to authenticate, the hardware signature is taken for the device and matched with the one stored on the local server. Thus, the mechanism is not efficient for the environments which involve resource constrained devices.

Georgios et al. [14] proposed a lightweight secure authentication mechanism for the industrial IoT devices so that they can securely communicate with each other and transfer the sensed data. The proposed mechanism is based on hash functions and XOR operations and requires less energy as compared to the asymmetric based mechanism. The main problem with the proposed mechanism is that, it requires the storage of the authentication data on a local server. Thus, the mechanism is vulnerable to a single point of failure.

Authors in [15] proposed a remote validation mechanism to identify the node trust and to monitor their behaviour in real time environments. The problem identified by the researchers is that, most of the remote authentication mechanisms deployed in the perception layer are not able to continuously validate the trust level of a node, and are unable to provide tracking of these sensing nodes. The authors proposed a trust measurement model in which the data transmission behaviour of the sensing node is taken under consideration. It then uses hash operations to generate a trust threshold value and stores it on a local server. The mechanism is not secure against single point of failure attacks. Moreover, the mechanism requires 4 stages of message exchange for the completion of the authentication process. This makes the mechanism not suitable for the resource constrained devices.

In [17] authors proposed a group authentication mechanism that allows a group of devices to authenticate before entering into the system. The proposed mechanism has three main entities: device, a group leader and a home subscriber server. The device must be registered and authenticated by the home subscriber server to enter the network. The main drawback of the proposed mechanism is that the home subscriber server stores all the authentication data for the various groups of IoT devices, and, thus, is prone to single point of failure.

The authors in [11] analysed the applicability of the blockchain concept to overcome various security issues in IoT. The work propose an authentication mechanism which is based on blockchain. It provides a decentralized authentication for IoT. The main issue with the proposed mechanism is that the devices of one system cannot communicate with the devices of a different system. Thus, it is not applicable to a number of distributed IoT applications

where it is critical to allow communication between IoT devices belonging to different systems.

To summarize, Table 1 provides technical comparison of the studied related work. The main challenge for providing security in IoT systems is that the nodes of IoT systems are mostly resource-constrained and are heterogeneous in nature that limits their ability to host complex cryptographic algorithms [36]. These devices are deployed in an open environment that readily increases the risk of being the victim of physical attacks. In addition, the existing mechanisms are mostly centralized and suffer from the scalability issue due to their centralized orientation [52]. They are mainly based on a single authorized entity to which all the IoT devices are connected. The devices communicate with the central entity for various purposes such as authentication and communication of time-sensitive data. These mechanisms perform well when the size of the network is small and devices are deployed in the close proximity of the central entity. However, centralized approaches are not suitable for those IoT systems where operations are time-sensitive and the devices are deployed in a distributed manner. For example, devices deployed at a large distance from the central entity may introduce high-latency when it comes to authenticating IoT devices and processing time-sensitive data [53].

Table 1: Technical comparison of existing mechanisms

Paper	**Mechanism Orientation**	**Encryption Type**	**Key Generation Mechanism**	**Mutual Authentication**	**Access Control**	**Data Integrity**	**Data Anonymity**
[13]	Centralized	Symmetric	Random number generator algorithm	**O**	✗	✗	**O**
[14]	Centralized	Asymmetric	Random number generator and Hash function	**O**	✗	✗	✗
[18]	Centralized	Symmetric	Hash Functions and XOR operation based	✗	✗	✗	✗
[8]	De-Centralized	Symmetric	XOR operation based	**O**	**O**	**O**	✗
[12]	Centralized	Symmetric	Shamir's secret sharing	✗	✗	✗	**O**
[5]	Centralized	Asymmetric	FE.Gen algorithm	**O**	**O**	**O**	✗
[15]	Centralized	Asymmetric	Random number generator and Hash function	✗	**O**	**O**	**O**
[11]	De-Centralized	Asymmetric	ECC	**O**	**O**	**O**	✗
[16]	Centralized	Symmetric	PUF and FE	**O**	**O**	**O**	**O**
[17]	Centralized	Asymmetric	ECC	**O**	**O**	**O**	✗

3 Proposed Mechanism

The main aim of this work is to provide a blockchain-based distributed authentication and authorization mechanism that allows communication among devices from various systems. The communication can either be made between the devices of the same system or between devices belonging to different ones.

3.1 System Architecture

Figure 2 illustrates the proposed mechanism and refers to 2 layers, device layer, and Fog Layer, with focus on the latter in this paper.

Device Layer This layer mainly contains IoT devices that are deployed in different environments for sensing, actuating and communication. These devices sense and generate data that may be further transmitted to other devices like actuators. IoT devices can be grouped together based on the system type. For example, in figure 2, there are different types of systems e.g., smart home, smart hospital and smart school. The devices belonging to these systems often want to communicate with each other to provide value-added services.

Fog Layer This layer contains a network of blockchain-enabled fog nodes that work together via the Internet. The fog nodes are generally owned by different parties and may not come from the same provider. Each system (such as a smart hospital) and all its corresponding smart devices are associated with a nearest blockchain-enabled fog node. These blockchain-enabled fog nodes communicate with each other for the synchronization of data associated with authentication and authorization. A smart contract, which contains a set of rules, can also be defined on top of the blockchain-enabled fog nodes. It provides an essential add-on in terms of executing transactions only if they comply with a pre-defined policy. Moreover, to validate the transactions, the consensus algorithm is executed, and blocks are created for these transactions [11]. The transaction blocks are shared among blockchain-enabled fog nodes to support authentication and authorization in a distributed fashion.

3.1.1 Types of Communications

In the proposed mechanism, there are primarily three types of communications that include device-to-fog communication, fog-to-fog communication and device-to-device communication.

Device-to-Fog Communication: The device-to-fog communication has two main objectives in the proposed mechanism. The first objective is to register the IoT system and devices with the blockchain-enabled fog node. And the second objective is to authenticate the devices when it tends to enter into the network.

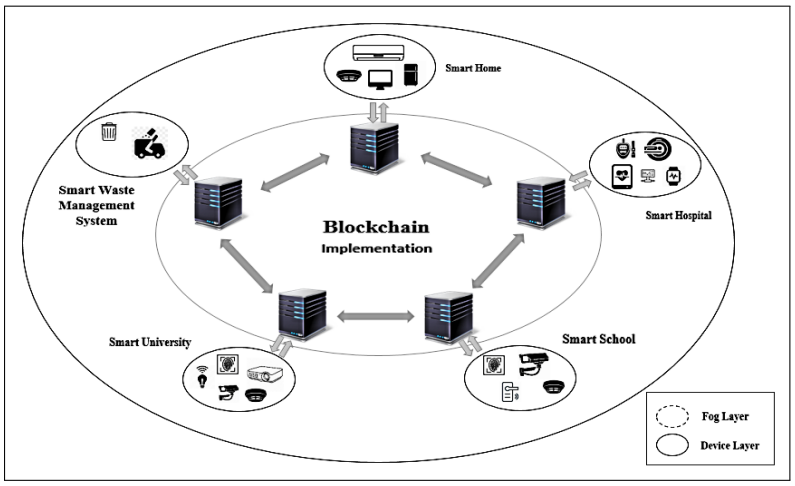


Fig. 2: Architecture of proposed mechanism

Fog-to-Fog Communication: The main objective of fog-to-fog communication is the synchronization of the authentication and authorization data among all the blockchain-enabled fog nodes.

Device-to-Device Communication: When two devices (either belong to the same system or a different one) once successfully authenticated, they are now able to initialize secure connection and transfer data among each other.

3.1.2 Authentication of IoT Devices

Each IoT system has a corresponding blockchain-enabled fog node that is closest to the system and is used for the registration and authentication of the IoT devices belonging to the same system. The devices will first register with their corresponding blockchain-enabled fog node. The identity of these devices is stored in the blockchain as transaction and blocks are created for them. These blocks are then distributed among all the other connected blockchain-enabled fog nodes. If a device belonging to a group wants to authenticate, it will provide its identification credentials to the corresponding blockchain-enabled fog node. The blockchain validates the provided credentials and if the credentials appear to be valid then the devices will be successfully authenticated. Otherwise, the device will be rejected and will not be allowed to enter into the network.

3.1.3 Access Control for IoT Devices

The proposed mechanism also provides access control for the devices in the IoT system. In the proposed mechanism the devices can only communicate with devices that are registered with blockchain-enabled fog nodes and are authenticated. The device that is not registered on the blockchain cannot authenticate

itself and thus is not allowed to communicate with the authenticated devices either it is in the same group or it belongs to another group. This will reduce the possibility of the interaction of the malicious device with the legitimate device. Moreover, the device can only register to one blockchain-enabled fog node.

3.1.4 Key Generation Algorithm

In the proposed mechanism Elliptic Curve Digital Signature Algorithm (ECDSA) algorithm is used for the generation of public and private keys for the devices and the fog nodes. The selection of this algorithm is made on the basis of the comparison among various encryption algorithm presented by [25]. The ECDSA algorithm requires less power as compared to other and has same security level as of Rivest–Shamir–Adleman (RSA).

3.2 System Functioning

In this section, the overall working of the proposed system is briefly explained. The proposed mechanism consists of three main phases that include, the initialization phase, device authentication phase and device-to-device communication phase. In the initialization phase, registration of the system and its corresponding devices are discussed. The system initialization phase allow new systems to register with the network and ensures that all systems are uniquely identifiable. Once a new system is registered with the network, the next phase is the device registration phase. In the device registration phase, smart devices are allowed to get registered with the network. These devices are registered against their corresponding systems (already registered) in order to be part of the network. The device registration phase is followed by the device authentication phase where the devices are authenticated with the blockchain-enabled fog nodes. The registration phase provides a certificate to a device so that it can be authenticated with the blockchain-enabled fog nodes. In this phase, the authentication certificate provided by the device is analyzed. If all the authentication conditions are met, then the device is allowed to become part of the network. This phase ensures that only authorised devices can join the network. Finally, in the device-to-device communication phase, the communications between the devices either belong to the same groups or different is discussed. In this phase, devices belonging to various systems are allowed to communicate with each other and share critical data between them.

3.2.1 Assumptions

Prior to proceeding with the proposed mechanism, we deem appropriate to clarify the assumptions made:

– Blockchain-enabled fog nodes are legitimate and trusted.

– The system can securely share the signature (generated by the fog nodes) with the legitimate devices of that system.
– The system admin is trusted, and all the actions carried out by the admin are legitimate.

This is further explained in section 3.2.2. The proposed mechanism requires an initialization phase in which the systems and the devices are registered by their respective admins on the blockchain-enabled fog nodes.

3.2.2 Initialization Phase

In the initialization phase, the system and its corresponding devices are registered with the blockchain-enabled fog nodes. The registration is required in order to allow the devices to be re-authenticated. It further consists of two phases: (i)registration phase, and (ii) the device registration phase. These phases are explained below:

System Registration Phase: In this phase, an admin registers a system on the nearest blockchain-enabled fog node using a unique System ID *(SID)*. After the successful registration, a block is created for that system and is distributed among all the blockchain-enabled fog nodes present on the Fog Layer. Furthermore, a new certificate is generated by the registering fog node for the newly registered system which contains *(SID)* of the registered system, that is used in the device registration phase. Figure 3 shows the complete process of the system registration phase. Moreover, the steps shown in figure 3 are explained below.

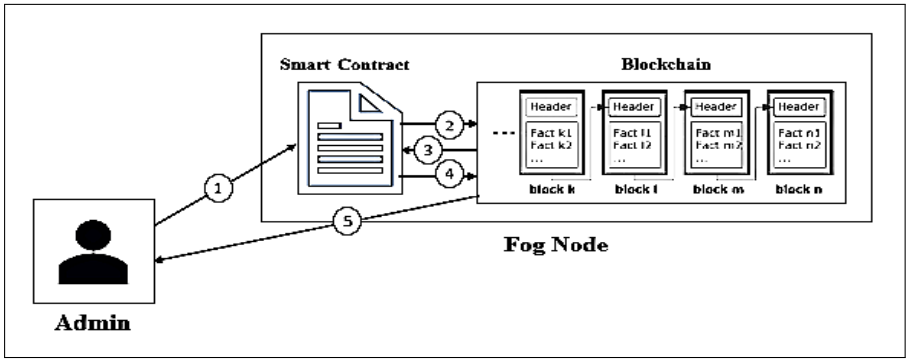


Fig. 3: Registration of system on blockchain-enabled fog node

1. The system admin first generates a unique *(SID)* for the new system that is made up of the system name and the last 5 hashed digits of the blockchain-enabled fog node Media Access Control address (MAC address), randomized to ensure the uniqueness of the *(SID)*. The system admin then shares

this *(SID)* information with the corresponding blockchain-enabled fog node by creating a transaction T_1.

$$SID = System\ Name + SHA\text{-}1(MAC\ of\ Blockchain\text{-}enabled\ Fog\ Node)$$
$$T_1 = F_{PK}(SID)$$

Here F_{PK} is the public key of the blockchain enabled fog node used to encrypt the message.

2. The blockchain-enabled fog node using the smart contract first verifies the transaction by checking if the given *(SID)* exists in the blockchain or not. Algorithm 1 describes this rule for the system registration phase.

Algorithm 1: System Registration Rule for Smart Contract

Parameters:
block_chain: Blockchain
sys: Object

```
1 begin
      // Checking if the provided system ID exists in the
       blockchain or not
2     if (SID_exists(sys.id,block_chain) = true) then
          // If the provided system ID exists in the blockchain
           than return error
3         return error()
4     else
5         register_SID(sys.id,block_chain)  // Register the provided
           system ID with blockchain
```

3. If the *(SID)* already exists in the blockchain, the transaction will not be allowed, and the registration process will be halted with an error notification.
4. If the *(SID)* does not exist in the blockchain, the smart contract will allow the transaction and a new block will be created for it. Furthermore, this block will also be distributed among other blockchain-enabled fog nodes.
5. If the system is successfully registered, the blockchain-enabled fog node will generate a certificate for the *(SID)* using its private key F_{IK} and send this certificate to the system admin by making another transaction T_2.

$$T_2 = UID_{PK}(F_{IK}(SID))$$

Here UID_{PK} is the public key of the admin. Admin extracts the certificate by using its private key UID_{IK} when transaction T_2 is received.

$$UID_{IK}(UID_{PK}(F_{IK}(SID)) = F_{IK}(SID)$$

This certificate $F_{IK}(SID)$ is distributed among all other devices of the corresponding system.

Device Registration Phase: Once the system is successfully registered on the blockchain-enabled fog node, the next phase is the device registration phase. In this phase, the system admin securely shares the certificate it received from the blockchain-enabled fog node, with the devices who wish to connect to this system as shown in figure 4. Furthermore, system admin also shares the list of public addresses of the devices that receive a certificate with the blockchain enabled-fog node. The addresses are saved in the blockchain in order to protect the addition of any malicious device even if the adversary is able to spoof the *(SID)* certificate.

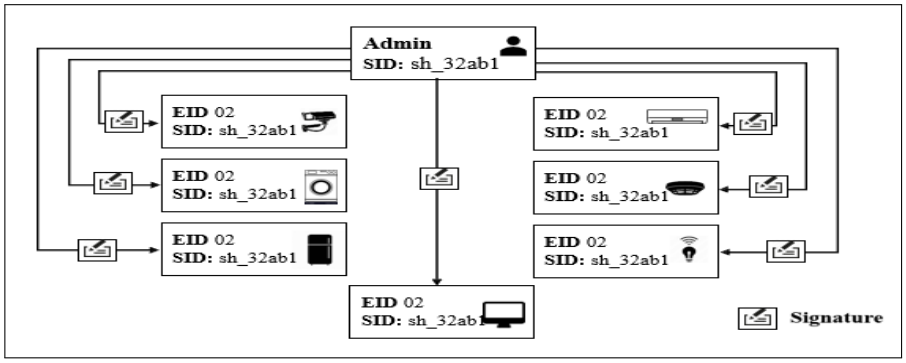


Fig. 4: Distribution of Signature Among Things of Smart Home

After the secure distribution of the certificate, each device generates another certificate called *registration_token* using its private key EID_{IK}. This token is unique for each device. This token contains the device ID *(EID)*, device public address *(EIP)* and system ID *(SID)* provided by the system admin.

$$registration\_token = EID_{IK}(EID, EIP, SID)$$

The device sends this token along with the *(SID)* certificate to the blockchain-enabled fog node. Here the smart contract checks the legitimacy of the *(SID)* certificate and the existence of the *(SID)* in the blockchain. If the provided certificate is legitimate and *(SID)* exists in the blockchain, the smart contract will verify whether the public key used for the verification of *registration_token* is present in the blockchain or not. This should be the same public address that was saved by the system admin earlier. After all the confirmations the smart contract allows the transaction and a block of mapping between the device ID *(EID)*, system ID *(SID)* and device IP *(EIP)* is created. This block is then distributed among all the blockchain-enabled fog nodes. Lastly, the fog node provides a new certificate named *auth_pass* to the newly registered device. This *auth_pass* contains the mapping which is created above. This *auth_pass* will be used by the device for authentication in future. Figure 5 shows the complete process of device registration with blockchain-enabled fog node.

Steps shown in figure 5 are explained below.

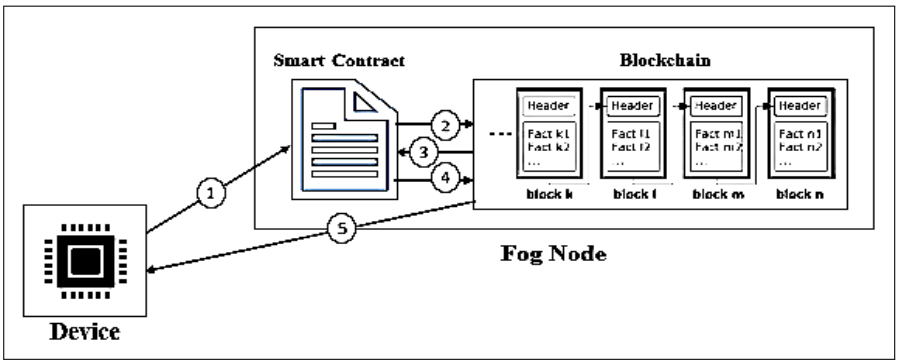


Fig. 5: Registration of Device on Blockchain-Enabled Fog Node

1. Device sends the *registration_token* along with the *(SID)* certificate to the blockchain-enabled fog node by making transaction T_3.

$$T_3 = F_{PK}(EID_{IK}(EID, EIP, SID), F_{IK}(SID))$$

F_{PK} is the public key of the fog node used for encryption of message. The fog node verifies the authenticity of the received packet by applying its private key F_{IK}. The certificate F_{IK}(SID) is also verified by the blockchain enabled fog node in order to complete the device registration phase.

$$F_{IK}(F_{PK}(EID_{IK}(EID, EIP, SID), F_{IK}(SID))) = EID_{IK}(EID, EIP, SID), F_{IK}(SID)$$
$$F_{PK}(F_{IK}(SID)) = SID$$
$$EID_{PK}(EID_{IK}(EID, EIP, SID)) = EID, EIP, SID$$

2. Here the smart contract deployed over the blockchain-enabled fog node checks the existence of provided *(SID)* and *(EID)* in the blockchain. Algorithm 2 describes this rule for the device registration phase.

Algorithm 2: Device Registration Rules for Smart Contract

Parameters:
block_chain: Blockchain
sys: Object
div: Object

```
1 begin
        // Checking if the provided system ID exists in the
          blockchain or not
2   if (SID_exists(sys.id,block_chain) = true) then
            // Checking if the provided device ID exists in the
              blockchain or not
3     if (EID_exists(sys.id,block_chain) = false) then
              // Checking if the provided device public address
                exists in the blockchain or not
4       if div.pk = EIP then
5         create_mapping(sys.id, div.id, div.pk, block_chain)
            // Saving the mapping of device ID, system ID and
              device public address
6   else
7     return error()     // If any of the above conditions is not
        met than system will return error
```

3. If *(SID)* does not exist or the device with ID *(EID)* is already registered in the blockchain then the transaction will not be allowed, and the registration process will be terminated.
4. If SID exists in the blockchain and *(EID)* is unique and new to blockchain, the smart contract verifies the existence of public key EID_{PK} used for token verification in the blockchain. If the correct match is found in the blockchain smart contract allow the transaction. A new block is created which contains the mapping of *(SID)*, *(EID)* and *(EIP)*. Furthermore, the block is distributed among other blockchain-enabled fog nodes deployed in Fog Layer.
5. After the successful registration of the device, the blockchain-enabled fog node will generate an *auth_pass* and send it to the device by making another transaction T_4.

$$T_4 = EID_{PK}(F_{IK}(SID, EID, EIP))$$

When the device receives this transaction, the device will extract the *auth_pass* from the received packet.

$$EID_{IK}(EID_{PK}(F_{IK}(SID, EID, EIP))) = F_{IK}(SID, EID, EIP)$$
$$auth\_pass = F_{IK}(SID, EID, EIP)$$

This *auth_pass* will be used by the device at the time of authentication.

3.2.3 Device Authentication Phase

After the initialization phase, there is an authentication phase in which the devices registered with the specific system are required to be authenticated on the blockchain-enabled fog nodes. A device sends the *auth_pass* to the blockchain-enabled fog node. This *auth_pass* contains the mapping of *(SID)*, *(EID)* and the device public address *(EIP)*. The *auth_pass* verifies the following conditions to authenticate and allow devices to communicate with other authenticated devices: (i) *SID* exists in the blockchain, (ii) *(EID)* exists in the blockchain, (iii) public address *(EIP)* used for the verification of transaction is associated with the saved public address *(EIP)*, and (iv) there exists a valid mapping between *(SID)*, *(EID)* and *(EIP)*. The complete process of device authentication is shown in figure 6.

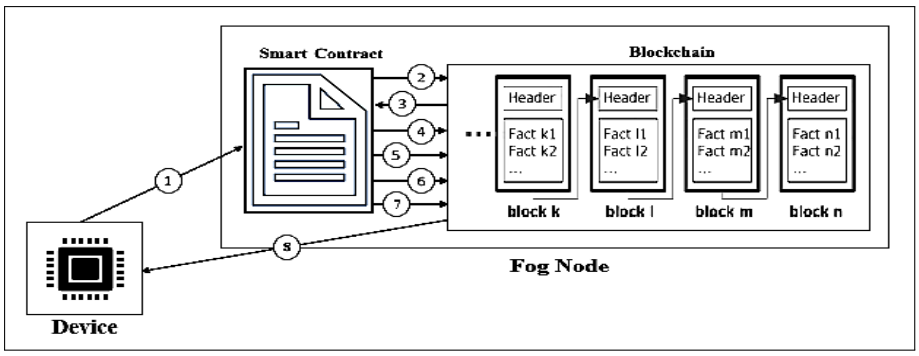


Fig. 6: Authentication of Device on Fog Node

Steps shown in figure 6 are explained below.

1. Device encrypts the *auth_pass* with its private key and sends it to the blockchain-enabled fog node by triggering a transaction T_5.

$$T_5 = EID_{IK}(F_{IK}(SID, EID, EIP))$$

The blockchain-enabled fog node verifies the legitimacy of the received packet by applying the public key of device EID_{PK}.

$$EID_{PK}(EID_{IK}(F_{IK}(SID, EID, EIP))) = F_{IK}(SID, EID, EIP)$$

To verify the extracted *auth_pass*, blockchain-enabled fog node applies its public key F_{PK}.

$$F_{PK}(F_{IK}(SID, EID, EIP)) = SID, EID, EIP$$

2. Here the smart contract checks the existence of given *(SID)* in the blockchain. Algorithm 3 describes this rule for the device authentication phase.

Algorithm 3: Device Authentication Rules for Smart Contract

Parameters:

block_chain: Blockchain
sys: Object
div: Object

```
1 begin
      // Checking if the provided system ID exists in the blockchain
     or not
2     if (SID_exists(sys.id,block_chain) = true) then
                // Checking if the provided device ID exists in the
       blockchain or not
3         if (EID_exists(sys.id,block_chain) = true) then
                  // Checking if the provided public address of device
           exists in the blockchain or not
4             if (div.pk = EIP) then
                    // Checking if the provided authentication details
               exists as mapping in the blockchain
5                 if (validate_mapping(sys.id,div.id,div.pk,block_chain)
                   = true) then
6                     //Device Authenticated Successfully
7     else
8         return error()    // If any of the above conditions is not met
           than system will return error
```

3. If *(SID)* does not exist in blockchain the authentication process is ended with error, otherwise the process continues to the next step.
4. The smart contract checks the existence of given *(EID)* in the blockchain.
5. If the given *(EID)* does not exist in the blockchain, the authentication process halts with an error. Otherwise, the process continues to the next step.
6. The smart contract now checks that the given mapping *(SID, EID, EIP)* is valid or not.
7. If the mapping is not valid then the device will not be authenticated. Otherwise, the process continues to the next step.
8. At the end, the public address of the device *(EIP)* is compared with the one stored for the same device at the time of registration (i.e., *(EIP)*). If the keys are same than the device will be successfully authenticated, otherwise the device will not be authenticated.

3.2.4 Device-to-Device Communication Phase

In device-to-device communication phase, two devices either from the same group or different, communicate with each other. Before they communicate both the devices are mutually authenticated on blockchain-enabled fog node. For example, device D_{v_1} from the System S_A wants to communicate with device D_{v_2} that belongs to System S_2. The device D_{v_1} sends a transaction to blockchain-enabled fog node in which the device provides its *auth_pass* and the *(EID)* of device D_{v_2}. The blockchain-enabled fog node will verify the *auth_pass*. If the certificate is verified successfully, the next step is to verify whether the *(EID)* exists in the blockchain or not. After the validations of the mapping and associations, a block is created in the blockchain by storing the mapping of the device IDs (EID_1, EID_2), and a secure communication channel is established between the devices. The main goal of creating a block is to avoid re-authentications, whenever they want to communicate with each other. Moreover, it also provides security against repudiation. Figure 7 shows the complete process of communication between the two devices.

Steps shown in figure 7 are explained below.

1. Device sends a transaction T_6 to the blockchain-enabled fog node, containing its *auth_pass* and the *(EID)* of the device to which it wants to communicate. For example, device A from system A wants to communicate with device B belonging to system B. Device A will send the transaction containing the *auth_pass* of device A and EID of device B, to blockchain-enabled fog node A.

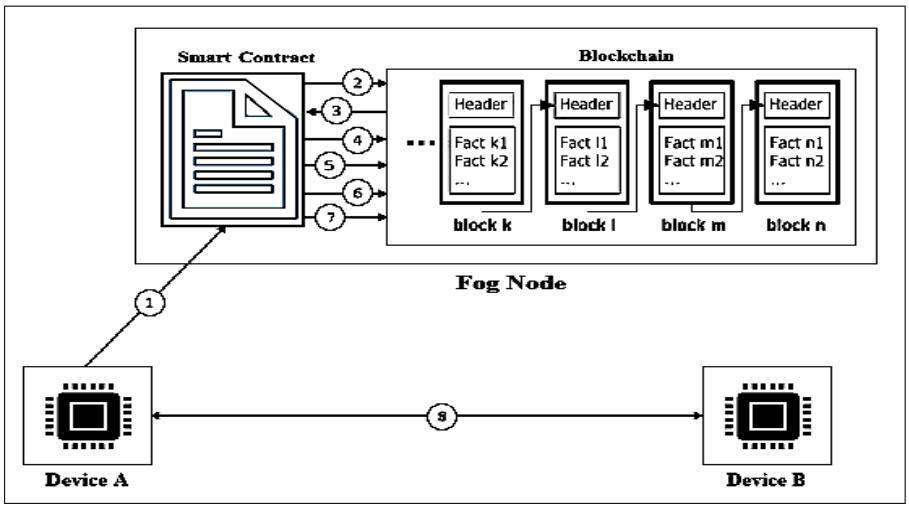


Fig. 7: Communication between IoT Devices

$$T_6 = EID_{AIK}(F_{AIK}(SID_A, EID_A, EIP_A), EID_B)$$

Here EID_{AIK} is the private key of the device A to encrypt the message. The blockchain-enabled fog node extracts the *auth_pass* and device ID EID_B by applying the public key of device A (i.e., EID_{APK}).

$$EID_{APK}(EID_{AIK}(F_{AIK}(SID_A, EID_A, EIP_A), EID_B)) = F_{AIK}(SID_A, EID_A, EIP_A), EID_B$$

Algorithm 4: Device-to-Device Communication Rules for Smart Contract

Parameters:
block_chain: Blockchain
sys: Object
div: Object

```
1 begin
      // Checking if the provided system ID for the 1st device
       exists in the blockchain or not
2     if (SID_exists(sysA.id,block_chain) = true) then
          // Checking if the provided device ID for the 1st device
           exists in the blockchain or not
3         if (EID_exists(divA.id,block_chain) = true) then
              // Checking if the provided public address of 1st
               device exists in the blockchain or not
4             if (divA.pk = EIP_A) then
                  // Checking if the provided authentication details
                   for 1st device exists as mapping in the
                   blockchain
5                 if (vali-
                   date_mapping(sysA.id,divA.id,divA.pk,block_chain)
                   = true) then
                      // Checking if the provided device ID for the
                       2nd device exists in the blockchain or not
6                     if EID_exists(divB.id,block_chain) = true)
                       then
7                         //Secure Communication Established
                           among Devices
8     else
9         return error()    // If any of the above conditions is not
           met than system will return error
```

The blockchain-enabled fog node will verify the *auth_pass* by applying its public key F_{APK}.

$$F_{APK}(F_{AIK}(SID_A, EID_A, EIP_A)) = SID_A, EID_A, EIP_A$$

2. Here, smart contract verifies that the SID_A exists in the blockchain. Algorithm 4 describes this rule for the device-to-device communication phase.
3. The verification is received from the blockchain. If provided SID_A does not exist in the blockchain, the communication cannot be initialized. Otherwise the process moves to next step.
4. The smart contract verifies that the provided EIDs (EID_A and EID_B) exists in the blockchain or not.

Table 2: Specification of Nodes used in the Experimentation

Type of Node	Node Role	Processor Type	Max CPU Speed MHz	Operating System	Quantity	Make	Model
Laptop	Fog Node	64-bit	2800	Ubuntu 18 LTS	3	Dell	Vostro 3750
Raspberry Pi	IoT Node	32-bit	700	Raspbian 5.6.12	3	Raspberry Pi	3B
Laptop	IoT Node	64-bit	2321	Ubuntu 18 LTS	3	HP	ProBook 450

5. If any of the *(EID)* is missing in the blockchain, the communication cannot be initialized, otherwise the process moves to the next step.
6. The smart contract validates that the given mapping (SID_A, EID_A, EIP_A).
7. If the mapping appeared is invalid or not defined in the blockchain, the communication cannot be allowed.
8. Finally, smart contract allows the transaction and a mapping between the device IDs (EID_A, EID_B).

Figure 8 shows the sequence diagram of the device-to-device communication in the proposed mechanism.

4 Experiments and Evaluation

In this section, we first validate the proposed method against the security requirements and attacks it can tackle, similar to the work presented in [11]. We then evaluate the proposed method in terms of execution time and power consumption, and compare our proposed method with the state-of-art mechanism given in [11].

4.1 Experimental setup

In order to provide the validation of the proposed mechanism, evaluation based on 100 experiments is presented in this paper. The experimental setup contained three laptops acting as fog nodes, three laptops and three raspberry pi systems acting as IoT nodes. Each fog node was connected to two raspberry pi systems. Table 2 shows the testing environment for the proposed mechanism. These end nodes were developed using C++ language in order to allow interaction among them. The selection of experimental setup was based on the previous related work [11]. All the communication among the nodes was done using the JsonRPC library.

QTframework [37] used for the implementation of nodes to allow the execution of the application on various platforms. Thus, making the application platform-independent. Ganache-cli was used in the validation process for an emulated environment for Ethereum, which was very close to real Ethereum deployment. Another main reason for using Ganache was to provide

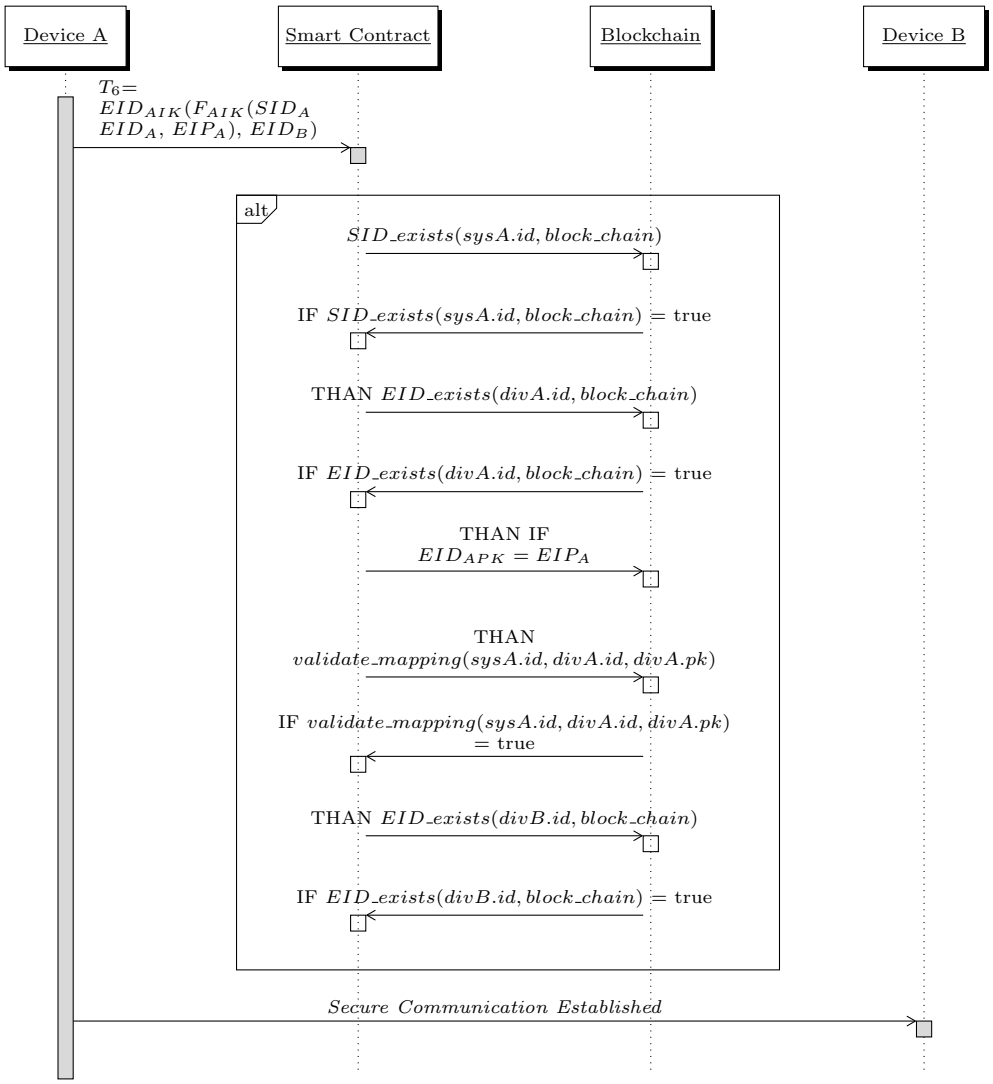


Fig. 8: Sequence Diagram of Device-to-Device Communication

accounts with dummy ether for testing smart contract. For the deployment of the smart contract, Truffle was used to provide the compilation and deployment of a smart contract. Truffle used the latest solidity compiler for the compilation of smart contract. In order to allow procedure calls among the nodes and the Ethereum blockchain, the QJSON-RPC library used, which is a lightweight and efficient library for a resource-constrained environment and supports JSON-RPC 2.0 [43]. Table 3 shows the list of tools used in the validation of the proposed mechanism.

Table 3: Description of the Tools used in the Experimentation

Ganache-cli	Ethereum emulator.
QTframework	IDE for the development of the node interface in C++.
Truffle	For the compilation and deployment of smart contract.
JsonRPC	For the realization of communication between node and blockchain.
Remix	IDE for the development of smart contract.

4.2 Evaluation Against the Security Requirements and Attacks

This section discusses the proposed mechanism compliance with the security requirements. These requirements include confidentiality [29], integrity [38], identification [24], non-repudiation [39], authentication [11] and mutual authentication [28].

4.2.1 Integrity

To achieve data integrity in the system, the data is first signed using the private key of the sender before sending data to the receiver. This is done to create a data packet using the ECDSA algorithm that is supported by Ethereum. The final data packet contains the data, the hash of that data and the signature generated by the sender [29]. The receiver verifies it by using the address of the signer and the hash attached to the received data.

4.2.2 Identification

To achieve this security requirement, device identification (ID) is compulsory for any device that wants to enter into the system. Each device that is registered with the system has a unique ID, and if any device wants to communicate with other devices in the system, it has to provide the identity of the listening device. In addition to the registration of a device, the system is also registered with a unique ID as well. Therefore, the device can extract the information about the ID of the sender device and can also know the system to which the device belongs [29].

4.2.3 Non-Repudiation

All the transactions in the system are signed by their respective private keys. Therefore, a sender cannot deny (repudiate) having performed a transaction [39].

4.2.4 Mutual Authentication

As discussed in section 3, the proposed mechanism contains the incorporation of an auth pass. Each device that is a part of the system has a valid *auth_pass*. This increases trust among other nodes to communicate with each other. The *auth_pass* is generated using the private key of the fog node. Consequently, if

any entity wants to generate fake *auth_passes*, it will require a private key of the fog node that is often secured and hard to retrieve [40].

4.2.5 Authentication

For the authentication of a device, the device must first register with the system. If the device is already registered, the system will haves its associated public address and valid mappings (discussed in section 3). Once the smart contract checks the existence and legitimacy of the details provided by the device, it will allow the device to make transactions in the system.

4.2.6 Spoof Attack

To successfully launch a spoof identity attack, the attacker needs the device ID, the system ID and the private key of that device. If the attacker somehow gets the device and system ID, the attacker will still need the private key to spoof the device identity [39].

4.2.7 Sybil Attack

Sybil attack requires the attacker to create fake identities in order to inject false information in the system. In the proposed mechanism, the devices are not allowed to get multiple identities thus, each device will have only one ID that is already registered on the system. The message generated by a device is first signed with the corresponding private key [42]. Thus, the chances of generating fake identities or injecting false messages into the system have been reduced to zero.

4.2.8 Message Replay Attack

In the proposed mechanism, all transactions (messages) generated in the system are associated with a unique transaction ID and time-stamp. Thus, a replay message with an already accepted transaction ID will be rejected by the system. Hence, providing protection against the message replay attack [18].

4.2.9 Substitution Attack

In the proposed mechanism, all transactions are first signed with the sender's private key and then are allowed to share with other nodes. Thus, if an attacker tries to modify the message, the signature will falsify this and will be caught during signature recovery process [41].

Table 4: Time Comparison of Different Operations

Approach	Laptop				Raspberry Pi			
	Time needed to generate registration request (ms)		Time needed to generate data message (ms)		Time needed to generate registration request (ms)		Time needed to generate data message (ms)	
	Average	SD	Average	SD	Average	SD	Average	SD
Proposed Approach	1.069	0.10	0.03	0.0005	24.77	0.013	0.69	0.0042
Bubble of Trust	1.56	0.13	0.04	0.001	28.03	0.045	0.82	0.029

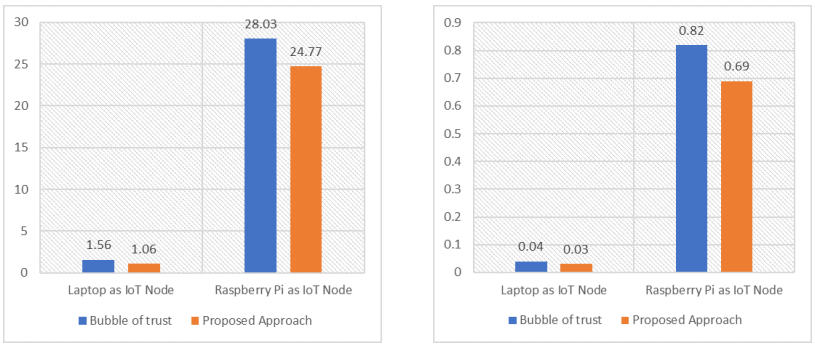
4.3 Evaluation in Terms of Execution Time and Power Consumption

The main concern of this research work is to provide a security mechanism that is readily hosted by resource-constrained nodes in the network. Thus, the time needed for a message to fully propagate in the blockchain network and the network latency issue are out of the scope of this study. The experimentation is performed in order to calculate:

- The time needed by the node for making the registration request.
- The time needed by the node for sending a data message.
- The CPU power consumed by the node for making the request for registration.
- The CPU power utilized by the node for the sending a data message.

4.3.1 Time Consumption

Two important factors for IoT devices are considered during the experimentation: time consumption and power consumption. The first factor is the minimum amount of time required by a device in order to generate the association request for a system. After registration, the device will be given an authentication pass that will only be used when a device wants to re-authenticate with the same or another system. Table 4 shows the average and standard deviation of the time required to generate a registration request. It shows the time comparison of the proposed approach with the bubble of trust [11]. The average time required by a laptop (currently acting as a IoT node) is *1.06 ms* with *0.10 ms* standard deviation while the maximum time is *1.25 ms*. The time required by the IoT node to generate a data message is *0.03 ms* with the standard deviation of *0.0005 ms*. The maximum time required by a laptop acting as a IoT node to send data message is *0.08 ms*. Thus, the standard deviation, in this case is quite low. In [11], the authors also used the same metric for the evaluation of their proposed approach. However, the association time needed by the laptop acting as the IoT node is *1.56 ms* and the time needed to send a data message is *0.04 ms*. In addition to this, the average time required by the raspberry pi node acting as the IoT node for generating association request is *24.77 ms* with the standard deviation of *0.013 ms* in



(a) A (b) B

Fig. 9: A: Time comparison for generating registration request, B: Time comparison for sending data message

Table 5: Power comparison of different operations

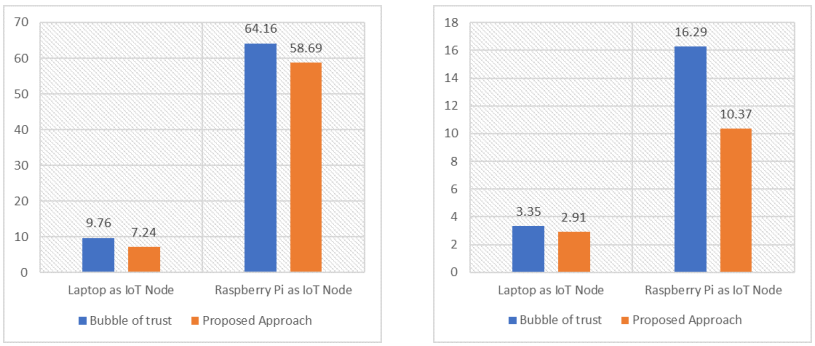
Approach	Laptop				Raspberry Pi			
	Power needed to generate registration request (mWatt)		Power needed to generate data message (mWatt)		Power needed to generate registration request (mWatt)		Power needed to generate data message (mWatt)	
	Average	SD	Average	SD	Average	SD	Average	SD
Proposed Approach	7.24	2.01	2.91	0.82	58.69	7.03	10.37	1.085
Bubble of Trust	9.76	2.04	3.35	0.87	64.16	8.19	16.29	1.10

our mechanism with the maximum number of *26.61 ms*. The time required to generate and to send a data message is *0.09 ms* and the standard deviation is *0.0042 ms*, whereas the maximum time required to perform that operation is *0.13 ms*.Moreover, the association time required in [11] is *28.03 ms*, and the time needed for sending a data message is *0.82 ms* for the raspberry pi. The larger value of time for raspberry is due to the complexity of operations needed to be performed. Moreover, the raspberry pi is less resourceful as compared to a laptop. This comparison can also be visualized in figures 9.

In figure 9 (A), a comparison of time required to generate association request is shown for both the laptop and raspberry pi acting as IoT nodes. Figure 12 (B) shows the comparison of the time needed to send data message for both type of device.

4.3.2 Power Consumption

The second important factor is the amount of power that a resource-constrained device consumes when associated to a system. Table 5 lists the average and standard deviation of the amount of power required by an IoT node to complete



(a) A (b) B

Fig. 10: A: CPU power comparison for a laptop, B: CPU power comparison for a Raspberry Pi

the registration process. It shows the comparative analysis of our mechanism with the mechanism proposed in [11]. The registration process includes the power required by the IoT node to make an association with the system, and, then, to send the data request to the blockchain. The main objective of associating power with the registration phase is due to the fact that IoT devices are equipped with limited power supply. Existing approaches spent more power by investing too many messages during the registration phase. Whereas, in the proposed approach, the average amount of power required by an IoT device is *7.24 mWatt* with a standard deviation of *2.01 mWatt* in order to complete the registration process with the maximum number of *10.32 mWatt*. The average amount of power required to send a message is *2.91 mWatt* with the standard deviation of *0.82 mWatt*. The maximum amount of power consumed to send the data message is *4.12 mWatt*. While in [11], the average power of CPU required by a laptop (IoT node) for the association task is *9.76 mWatt* with standard deviation of *2.04 mWatt*. Likewise, the average power required by the raspberry pi acting as the IoT node for completing the association process is *58.69 mWatt* with standard deviation of *7.03 mWatt*, whereas the maximum value for generating association request is *80.58 mWatt*. The average power required by the raspberry pi node for sending the data message is *10.37 mWatt* with a standard deviation of *1.085 mWatt* and the maximum value is *13.17 mWatt* in our technique. While, the average CPU power required by the raspberry pi (IoT node) for the association is *64.16 mWatt* with the standard deviation of *8.19 mWatt* for [11]. Similarly, the average CPU power required by the laptop (IoT node) to send a data message is *3.35 mWatt* with standard deviation *0.87 mWatt*. The raspberry pi consumed *16.29 mWatt* when sending a data message with standard deviation of *1.10 mWatt*. This comparison is further visualized in figure 10.

4.4 Comparison with the State-of-the-Art

In this section we compare the proposed mechanism with other authentication techniques that are proposed for IoT. Figure 11 illustrates the comparison in terms of the number of messages required for the authentication of IoT nodes (node association) with a system.

As discussed earlier, operations like node association and message exchange involve CPU power consumption. Therefore, as the number of messages between IoT nodes and the systems increases, the amount of power consumption also increases. Moreover, greater the number of messages also negatively affect the latency factor. In [13], the number of messages required to complete the authentication process is 5 seconds. Similarly, the majority of centralized systems require more than two messages for the completion of the authentication process [11]. For example, the minimum amount of time required to complete the authentication process in [49] is 2.137 seconds. But, this time readily increases as the number of connected devices increases in the system. The mechanism proposed in this research work requires only 69 milliseconds to generate and send the authentication request. One limitation of the proposed approach is the use of Ethereum blockchain for the evaluation of the proposed approach. The Ethereum blockchain takes a minimum of 14 seconds to complete a transaction, which may add an additional delay to our proposed authentication mechanism. This time-delay is due to the PoW consensus mechanism used in the Ethereum blockchain. However, using other alternative consensus algorithms (PoS, Ripple, etc.) that are not computationally intensive can improve the time-consuming factor.

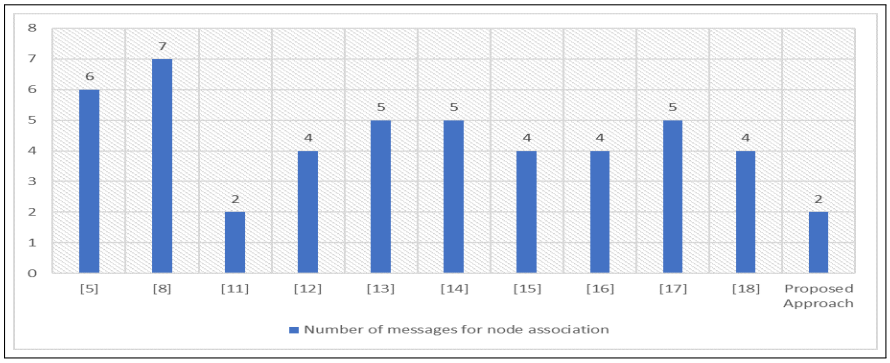


Fig. 11: Comparison based on number of messages required for node association

The proposed model can scale well in terms of the number of devices. As the network expands, the number of connections, device registration requests and number of blocks generated and to be stored increases. The model is not affected by the increase in the number of connections and registration

requests as the registration process is handled by fog nodes and the computation complexity is independent of the number of devices. Ethereum storage requirement is very modest, however, if needed the devices can act as thin clients and will store only the header of the blocks while the actual blocks can reside on resourceful fog devices. High transaction throughput is well handled by Ethereum as it has an upper bound on the transaction time which is achieved by varying the complexity level of consensus mechanism.

5 Conclusion and Future Work

IoT systems are expected to operate in a secure and distributed environment with a stringent minimum delay requirement so that IoT devices can interact with each other in a secure manner to communicate and exchange time-sensitive data. In this research, a novel authentication and access control mechanism is proposed for IoT that permits secure communication between devices from the same IoT system as well as communication between devices from different IoT systems. The proposed mechanism is based on the blockchain technology to benefit from its cryptographic properties and distributed nature, and relies on fog computing to address the latency issues. The proposed mechanism can be applied to a number of IoT scenarios. Moreover, security requirements and an attack model are also defined to evaluate our approach and test its ability to meet these requirements. In order to avoid the huge amount of energy consummation require by PoW to verify each block, the future work would focus on the development of a lightweight consensus protocol where miners will be nominated based on their trust value.

References

1. Baker, Thar and Asim, Muhammad and Tawfik, Hissam and Aldawsari, Bandar and Buyya, Rajkumar, An energy-aware service composition algorithm for multiple cloud-based IoT applications, Journal of Network and Computer Applications, 89, 96–108 (2017)
2. By, Gartner Says, 2020, More Than Half of Major New Business Processes and Systems Will Incorporate Some Element of the Internet of Things, Publicado em Janeiro (2016)
3. Hammi, Badis and Khatoun, Rida and Zeadally, Sherali and Fayad, Achraf and Khoukhi, Lyes, IoT technologies for smart cities, IET Networks, 7, 1–13 (2017)
4. Muhammad, Khan and Hamza, Rafik and Ahmad, Jamil and Lloret, Jaime and Wang, Haoxiang and Baik, Sung Wook, Secure surveillance framework for IoT systems using probabilistic image encryption, IEEE Transactions on Industrial Informatics, 14, 3679–3689 (2018)
5. Feng, Wei and Qin, Yu and Zhao, Shijun and Feng, Dengguo, AAoT: Lightweight attestation and authentication of low-resource things in IoT and CPS, Computer Networks, 134, 167–182 (2018)
6. Abbas, Nadeem and Asim, Muhammad and Tariq, Noshina and Baker, Thar and Abbas, Sohail, A Mechanism for Securing IoT-enabled Applications at the Fog Layer, Journal of Sensor and Actuator Networks, 8, 16 (2019)
7. Roman, Rodrigo and Zhou, Jianying and Lopez, Javier, On the features and challenges of security and privacy in distributed internet of things, Computer Networks, 57, 2266–2279 (2013)

8. Aman, Muhammad Naveed and Chua, Kee Chaing and Sikdar, Biplab, Mutual authentication in IoT systems using physical unclonable functions, IEEE Internet of Things Journal, 4, 1327–1340 (2017)
9. Kothmayr, Thomas and Schmitt, Corinna and Hu, Wen and Brunig, Michael and Carle, Georg, A DTLS based end-to-end security architecture for the Internet of Things with two-way authentication, 37th Annual IEEE Conference on Local Computer Networks-Workshops, 956–963 (2012)
10. Jan, Mian Ahmad and Nanda, Priyadarsi and He, Xiangjian and Tan, Zhiyuan and Liu, Ren Ping, A robust authentication scheme for observing resources in the internet of things environment, 2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications, 205–211 (2014)
11. Hammi, Mohamed Tahar and Hammi, Badis and Bellot, Patrick and Serhrouchni, Ahmed, Bubbles of Trust: A decentralized blockchain-based authentication system for IoT, Computers & Security, 78, 126–142 (2018)
12. Vijayakumar, Pandi and Chang, Victor and Deborah, L Jegatha and Balusamy, Balamurugan and Shynu, PG, Computationally efficient privacy preserving anonymous mutual and batch authentication schemes for vehicular ad hoc networks, Future generation computer systems, 78, 943–955 (2018)
13. Wu, Fan and Li, Xiong and Xu, Lili and Kumari, Saru and Karuppiah, Marimuthu and Shen, Jian, A lightweight and privacy-preserving mutual authentication scheme for wearable devices assisted by cloud server, Computers & Electrical Engineering, 63, 168–181 (2017)
14. Esfahani, Alireza and Mantas, Georgios and Matischek, Rainer and Saghezchi, Firooz B and Rodriguez, Jonathan and Bicaku, Ani and Maksuti, Silia and Tauber, Markus and Schmittner, Christoph and Bastos, Joaquim, A lightweight authentication mechanism for m2m communications in industrial iot environment, IEEE Internet of Things Journal, (2017)
15. Gong, Bei and Zhang, Yu and Wang, Yubo, A remote attestation mechanism for the sensing layer nodes of the Internet of Things, Future Generation Computer Systems, 78, 867–886 (2018)
16. Gope, Prosanta and Sikdar, Biplab, Lightweight and privacy-preserving two-factor authentication scheme for IoT devices, IEEE Internet of Things Journal, 6, 580–589 (2019)
17. Roychoudhury, Probidita and Roychoudhury, Basav and Saikia, Dilip Kumar, Provably secure group authentication and key agreement for machine type communication using Chebyshev's polynomial, Computer Communications, 127, 146–157 (2018)
18. Zhang, Jiansong and Wang, Zeyu and Yang, Zhice and Zhang, Qian, Proximity based IoT device authentication, IEEE INFOCOM 2017-IEEE Conference on Computer Communications, 1–9 (2017)
19. Kshetri, Nir, Blockchain's roles in strengthening cybersecurity and protecting privacy, Telecommunications policy, 41, 1027–1038 (2017)
20. Schwartz, David and Youngs, Noah and Britto, Arthur and others, The ripple protocol consensus algorithm, Ripple Labs Inc White Paper, 5, 8 (2014)
21. Sousa, Joao and Bessani, Alysson and Vukolic, Marko, A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform, 2018 48th annual IEEE/IFIP international conference on dependable systems and networks (DSN), 51–58 (2018)
22. Kang, Jiawen and Xiong, Zehui and Niyato, Dusit and Wang, Ping and Ye, Dongdong and Kim, Dong In, Incentivizing consensus propagation in proof-of-stake based consortium blockchain networks, IEEE Wireless Communications Letters, 8, 157–160 (2018)
23. Jayasinghe, Upul and Lee, Gyu Myoung and MacDermott, Aine and Rhee, Woo Seop, TrustChain: A Privacy Preserving Blockchain with Edge Computing, Wireless Communications and Mobile Computing, 2019 (2019)
24. Reddy, Alavalapati Goutham and Suresh, Devanapalli and Phaneendra, Kolloju and Shin, Ji Sun and Odelu, Vanga, Provably secure pseudo-identity based device authentication for smart cities environment, Sustainable cities and society, 41, 878–885 (2018)
25. Goyal, Tarun Kumar and Sahula, Vineet, Lightweight security algorithm for low power IoT devices, 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 1725–1729 (2016)

26. Lau, Chi Ho and Alan, Kai-Hau Yeung and Yan, Fan, Blockchain-Based Authentication in IoT Networks, 2018 IEEE Conference on Dependable and Secure Computing (DSC), 1–8 (2018)
27. Li, Dongxing and Peng, Wei and Deng, Wenping and Gai, Fangyu, A Blockchain-Based Authentication and Security Mechanism for IoT, 2018 27th International Conference on Computer Communication and Networks (ICCCN), 1–6 (2018)
28. Lee, Kyung Chang and Lee, Hong-Hee, Network-based fire-detection system via controller area network for smart home automation, IEEE Transactions on Consumer Electronics, 50, 1093–1100 (2004)
29. Fotiou, Nikos and Polyzos, George C, Decentralized name-based security for content distribution using blockchains, 2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 415–420 (2016)
30. Mahmoud, Rwan and Yousuf, Tasneem and Aloul, Fadi and Zualkernan, Imran, Internet of things (IoT) security: Current status, challenges and prospective measures, 2015 10th International Conference for Internet Technology and Secured Transactions (ICITST), 336–341 (2015)
31. Alkheir, Ala Abu and Aloqaily, Moayad and Mouftah, Hussein T, Connected and autonomous electric vehicles (caevs), IT Professional, 20, 54–61 (2018)
32. Lee, In and Lee, Kyoochun, The Internet of Things (IoT): Applications, investments, and challenges for enterprises, Business Horizons, 58, 431–440 (2015)
33. Won, Jongho and Seo, Seung-Hyun and Bertino, Elisa, A secure communication protocol for drones and smart objects, Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security, 249–260 (2015)
34. Aloqaily, Moayad and Al Ridhawi, Ismaeel and Salameh, Haythem Bany and Jararweh, Yaser, Data and service management in densely crowded environments: Challenges, opportunities, and recent developments, IEEE Communications Magazine, 57, 81–87 (2019)
35. Baker, Thar and Asim, Muhammad and MacDermott, Aine and Iqbal, Farkhund and Kamoun, Faouzi and Shah, Babar and Alfandi, Omar and Hammoudeh, Mohammad, A secure fog-based platform for SCADA-based IoT critical infrastructure, Software: Practice and Experience (2019)
36. Tariq, Noshina and Asim, Muhammad and Al-Obeidat, Feras and Zubair Farooqi, Muhammad and Baker, Thar and Hammoudeh, Mohammad and Ghafir, Ibrahim, The security of big data in fog-enabled IoT applications including blockchain: A survey, Sensors, 19, 1788 (2019)
37. XU, Dexin and TAN, Zhenfan and GAO, Yanbin, Developing application and realizing multiplatform based on Qt framework [J], Journal of Northeast Agricultural University, 3, 018 (2006)
38. Barua, Mrinmoy and Liang, Xiaohui and Lu, Rongxing and Shen, Xuemin, ESPAC: Enabling Security and Patient-centric Access Control for eHealth in cloud computing, International Journal of Security and Networks, 6, 67–76 (2011)
39. Sultan, Abid and Mushtaq, Muhammad Azhar and Abubakar, Muhammad, IOT Security Issues Via Blockchain: A Review Paper, Proceedings of the 2019 International Conference on Blockchain Technology, 60–65 (2019)
40. Al-Turjman, Fadi and Altrjman, Chadi, IoT Smart Homes and Security Issues: An Overview, Security in IoT-Enabled Spaces, 111–137 (2019)
41. Wu, Mingli and Wang, Kun and Cai, Xiaoqin and Guo, Song and Guo, Minyi and Rong, Chunming, A Comprehensive Survey of Blockchain: from Theory to IoT Applications and Beyond, IEEE Internet of Things Journal (2019)
42. Hassija, Vikas and Chamola, Vinay and Saxena, Vikas and Jain, Divyansh and Goyal, Pranav and Sikdar, Biplab, A Survey on IoT Security: Application Areas, Security Threats, and Solution Architectures, IEEE Access, 7, 82721–82743 (2019)
43. Matt Broadstone, QJSON RPC, Bitbucket, url=https://bitbucket.org/devonit/qjsonrpc/src/master/, (Last Accessed: 08/09/2019)
44. Tonyali, Samet and Akkaya, Kemal and Saputro, Nico and Uluagac, A Selcuk and Nojoumian, Mehrdad, Privacy-preserving protocols for secure and reliable data aggregation in IoT-enabled smart metering systems, Future Generation Computer Systems, 78, 547–557 (2018)

45. Nawir, Mukrimah and Amir, Amiza and Yaakob, Naimah and Lynn, Ong Bi, Internet of Things (IoT): Taxonomy of security attacks, 2016 3rd International Conference on Electronic Design (ICED), 321–326 (2016)
46. Liu, Bin and Yu, Xiao Liang and Chen, Shiping and Xu, Xiwei and Zhu, Blockchain based data integrity service framework for IoT data, Liming, 2017 IEEE International Conference on Web Services (ICWS), 468–475 (2017)
47. Anirudh, M and Thileeban, S Arul and Nallathambi, Daniel Jeswin, Use of honeypots for mitigating DoS attacks targeted on IoT networks, 2017 International Conference on Computer, Communication and Signal Processing (ICCCSP), 1–4 (2017)
48. Li, Fagen and Xiong, Pan, Practical secure communication for integrating wireless sensor networks into the internet of things, IEEE Sensors Journal, 13, 3677–3684 (2013)
49. Lohachab, Ankur and others, ECC based inter-device authentication and authorization scheme using MQTT for IoT networks, Journal of Information Security and Applications, 46, 1–12 (2019)
50. Kumar, Gulshan and Saha, Rahul and Rai, Mritunjay Kumar and Thomas, Reji and Kim, Tai-Hoon, Proof-of-Work consensus approach in Blockchain Technology for Cloud and Fog Computing using Maximization-Factorization Statistics, IEEE Internet of Things Journal (2019)
51. Baliga, Arati, Understanding blockchain consensus models, Persistent, 4, 1–14 (2017)
52. Zhang, Xuyun and Yang, Laurence T and Liu, Chang and Chen, Jinjun, A scalable two-phase top-down specialization approach for data anonymization using mapreduce on cloud, IEEE Transactions on Parallel and Distributed Systems, 25, 363–373 (2013)
53. Raptis, Theofanis and Passarella, Andrea and Conti, Marco, Performance analysis of latency-aware data management in industrial IoT networks, Sensors, 18, 2611 (2018)
54. Rathee, Geetanjali and Sharma, Ashutosh and Iqbal, Razi and Aloqaily, Moayad and Jaglan, Naveen and Kumar, Rajiv, A blockchain framework for securing connected and autonomous vehicles, Sensors, 14, 3165, (2019)
55. Al Ridhawi, Ismaeel and Aloqaily, Moayad and Kotb, Yehia and Jararweh, Yaser and Baker, Thar, A profitable and energy-efficient cooperative fog solution for IoT services, IEEE Transactions on Industrial Informatics, (2019)
56. Ahmad, Awais and Din, Sadia and Paul, Anand and Jeon, Gwanggil and Aloqaily, Moayad and Ahmad, Mudassar, Real-Time Route Planning and Data Dissemination for Urban Scenarios Using the Internet of Things, IEEE Wireless Communications, 26, 50–55 (2019)