Modeling of Trusted Public Emergency Services for Smart Cities Using Blockchain and IoT-based Cognitive Networks

Bhawana Bhawana
Jawaharlal Nehru University
Sushil Kumar (skdohare@yahoo.com)
Jawaharlal Nehru University https://orcid.org/0000-0001-9113-2890

Research Article

Keywords: Internet of Things, Blockchain, Queuing theory, Public emergency service

Posted Date: August 17th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-742472/v1

Modeling of Trusted Public Emergency Services for Smart Cities Using Blockchain and IoT-based Cognitive Networks

Bhawana [a], Sushil Kumar [b]

[a] School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, India. (e-mail:bhawan_scs@jnu.ac.in).
[b] School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, India. (e-mail:skdohare@mail.jnu.ac.in).
Corresponding Author: Sushil Kumar, skdohare@mail.jnu.ac.in

Abstract

The Internet of Things (IoT) recently gained attention from the last few years due to various smart city applications deployment. The existing literature discusses different public emergency service (PES) aspects from smart-healthcare to smart-home automation. However, less work explores for the smart-fire-brigade system. The PESs require high computation, timely service fulfillment, service transparency, and trust, which are difficult to achieve through a centralized system. In recent years, blockchain technology has gained enormous popularity for immutable data management that ensures transparency, reliability, and data integrity using distributed storage. This paper presents a blockchain based model for secure and trusted public emergency service in IoT-enabled smart cities (BMSTP) to handle the PES requests in real-time fairly. An edge compute server (ECS) is introduced to enhance data processing speed and local data storage. Simultaneously, a queuing theory model is used to process PES requests quickly. The ECS manages an access control list (ACL) for smart-home IoT devices to protect against the illegal placement of any new IoT devices near smart-home to misguiding public emergency service departments (PESDs). Further, a reputation model is designed for PESDs to scale their service quality. We explored the BMSTP for smart-homes placed under different sub-areas of a smart-city. The experiment results show the proposed system model is efficient in scheduling the smart-home PES requests to an appropriate PESD and minimizing the delay to reaching the smart-home location.

Keywords

Internet of Things; Blockchain; Queuing theory; Public emergency service,

1. Introduction

The smart city covers urban areas equipped with Internet of Things (IoT) devices. These IoT devices provide convenience to individuals through different smart applications. The IoT devices also help in data collection for various smart-city applications [1], including smart-home automation, smart-healthcare, smart-transportation, robotics, smart-agriculture, and many more, as shown in Fig. 1. The demand for these IoT devices increases day-by-day and reaches 50 billion worldwide by 2030 [2]. With the increasing number of such IoT devices, the data generated from them grows exponentially, and handling this massive data becomes more challenging soon. The IoT devices generally have low computation power, limited storage, restricted network capabilities, and vulnerable to attack. Hence, fewer options are currently available to systematically process and manage these enormous connected IoT devices and their massive data. Therefore, to provide fast access and excess storage for these IoT devices, we require to exploit the edge compute server (ECS) functionality. The ECS removes this hurdle by residing near IoT devices and protect against performing malicious activity by supporting the access control list (ACL) based on access control policies [3]. Hence it ensures better performance, fast computation, device authenticity, and scalability.
Existing public emergency services (PESs) depend on a centralized system that involves social

agencies and government regulation to break data sharing barriers [4]. Due to the centralized system, PESs is suffering from high congestion and a single point of failure. Whereas citizens face misusing of their personal information without any prior knowledge. In this situation, a distributed and reliable system is essential while accessing PES resources. Hence, the importance of blockchain technology is highlighted in this paper for the smart-city implementation to overcome these contradictions.

Blockchain technology supports various features such as immutability, trust, transparency, and synchronization, using multiple distributed ledger maintained by blockchain nodes. Therefore, it eliminates a single point of failure. The blockchain smart contract adds a trust layer between citizens and PES providers and vice-versa. The public blockchain takes a long time to maintain consistency, so it is not worth adding every single piece of information. Hence, an ECS [5] acts as an intermediary between IoT devices and the blockchain. The ECSs are deploying near IoT devices, which require fast connection and low latency for PESs in the smart-city. The ECS calls blockchain application programming interface (API) to transfer PES requests gathered from IoT devices with high traffic load in a desirable time. Allocating PES provider in real-time with minimum waiting is a significant challenge. Executing these operations with a conventional approach adds additional delay. Introducing the queuing model is suitable for handling several PES requests generated from citizens and providing on-time emergency services with low latency to protect against catastrophe.

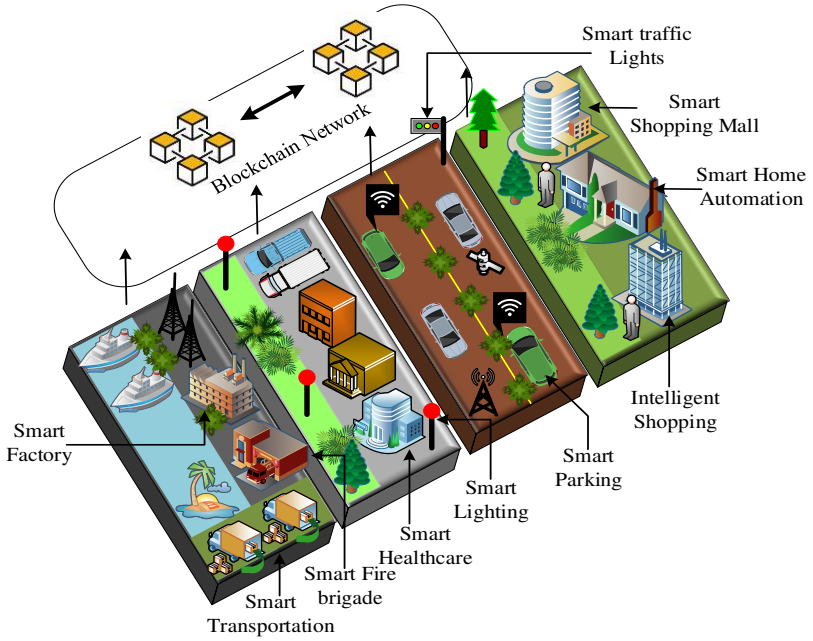

Fig 1: Smart-city applications

1.1. Our contributions

To overcome the challenges mentioned above, we propose a blockchain based model for secure and trusted public emergency service in IoT-enabled smart city (BMSTP). The proposed model detect fire in a smart-home and provide PES by suggesting the nearest fire brigade department to protect smart-homes from catastrophe. The functionality of the proposed system architecture is enhanced by implementing the smart contract logic. In brief, the main contributions of this paper are as follows.

1.) A three-layered architecture is proposed to define the working of each entity involved in the BMSTP. With this, a blockchain network is designed using a private blockchain platform.

2.) This proposed model has two types of entities at the second layer, i.e., IoT controller and service controller. These two behaves as an ECS. The IoT controller manages smart-homes data and forwards PES requests on behalf of smart-homes. Whereas the service controller keeps PESDs data and performs different tasks such as distance evaluation between smart-homes and PESDs and selecting a minimum request queue PESD. To fairly balance the PES

requests in real-time, a queuing theory is utilized at PESDs and an ACL at IoT controller to maintain smart-home IoT devices information.

3.) A smart contract is used to implement various functions. These functions include registration of IoT controllers, service controllers, smart-homes and PESDs, confirmation of PESD service provider, and reputation generation and updation of PESDs.
4.) Implement a reputation model for PESDs to bring transparency and trust in the proposed system architecture while serving PES requests.

1.2. Organization

The rest of the paper is organized as follows. Section 2 discusses the background knowledge of blockchain technology. In section 3, previous work on blockchain and IoT based smart-city applications and uses-cases are presented. A detailed view and implementation of the proposed BMSTP is described in section 4. Section 5 presents the simulation and results, and finally, section 6 is followed by the conclusion and future scope to extend the current work.

2. Background

This section provides blockchain technology fundamentals and private blockchain components to understand the flow of PES requests and transaction commitment, as describes below.

2.1. Blockchain

The term blockchain came up with bitcoin in 2008 [6] by unknown people called satoshi nakamoto. After that, a satoshi is considered the smallest denomination of the bitcoin blockchain. The first transaction on the bitcoin blockchain was recorded in May 2010. The success of the bitcoin blockchain is hashing algorithm, consensus mechanism, mining technique, and longest chain rule that remove the double-spending problem. The bitcoin blockchain aims to overcome the financial crisis issues to transfer digital assets among individuals at different locations. The bitcoin blockchain individuals are connected through a peer-to-peer network to publish financial transactions based on encryption using a public-private key.

In later years the term blockchain is widely used to indicate different public and private blockchain platforms such as Ethereum, Rootstock, R3 Corda, Quorum, and Hyperledger for developing real-world applications. The blockchain is an append-only data structure to maintain immutable transactions in distributed ledgers between unknown and untrusted individuals. This leads to the removal of a centralized system that is using to hold useful information. The blockchain block is an essential component that contains a block header and a block body, as represents in Fig. 2. These two have several pieces of information such as header number, nonce, current-hash, previous-hash, signed transactions data, etc. [7]. The first block on the blockchain is called as genesis block, which holds blockchain information. The upcoming blocks build on top of the genesis block, connected through a hash function to form a chain of blocks.

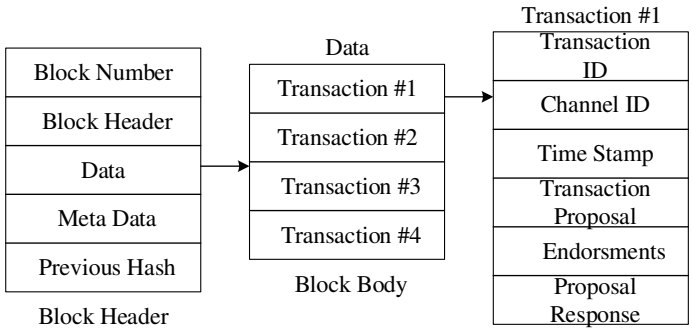

Fig 2: Block structure of private blockchain

2.2 Smart Contract

The smart contract was considered an impractical concept in the early years due to less advancement in information and communication technologies. Later, nick szabo, in 1994 [8], provide a conceptual view of the smart contract. In short, a smart contract is using to write the business rules or business agreements between two or more transacting parties in the form of a self-executable code. The smart contract definition changed after the invention of blockchain technology and became more famous. The smart contract [9] is a turing complete logic to write application codes which store at permanent blockchain address to roll out third party involvement. These smart contracts cannot modify at later stages, and they execute automatically in a distributed environment when certain conditions meet. Some examples of turing complete language support by different blockchain platforms are solidity powered by Ethereum blockchain. Hyperledger blockchain encourages multiple languages such as Python, Rust, JavaScript, Java, and Go as per developer comfort to write business logic. Similarly, some other blockchain platforms support modeling languages that are platform dependent.

2.3 Private Blockchain

The blockchain platform subdivides into two categories such as public blockchain and private blockchain. The public blockchain sometimes calls permissionless blockchain, whereas private blockchain is known as permissioned blockchain. The Hyperledger blockchain platform comes under the private blockchain. The Linux Foundation [10] hosts Hyperledger blockchain platform as a cross-industry collaboration project. The blockchain platform was designed to enable the customized networking rules to operate different consensus protocols and process thousands of requests per second with low latency. Compared to the public blockchain, the private blockchain allows only known entities to participate in blockchain relating operations. The consensus protocol [11] in the private blockchain requires mutual agreement among involved entities to commit transactions on the blockchain. According to application requirements, the private blockchain supports various consensus protocols such as Byzantine Fault Tolerance, Kafka, RAFT, Practical Byzantine Fault Tolerance, plenum, etc.

2.3.1. Private Blockchain Components

The private blockchain [12] is a collection of fabric organizations that contains multiple entities to build a blockchain network. These entities are membership service provider (MSP), fabric certificate authority (fabric-CA), orderer, peer, client, and channel. The functionality of each entity is describing below.

1.) *MSP:* In a private blockchain, the MSP is responsible for authorizing organization's fabric-CA. The MSP generates the digital certificates for each fabric-CA and maintains their information in the certification list.

2.) *Fabric-CA*: The fabric-CA resides under an organization and uses to generate digital certificates for peers.

3.) *Peers:* The peers are sub-divided into two types such as endorsing peer and committing peer. The endorsing peer performs various functionality, including consensus achievement and endorsement of transactions. Whereas the committing peer checks transaction validity and manages transaction block in their ledger.

4.) *Client:* The client is an entity that interacts with the blockchain network to execute smart contract functions. The client creates a transaction (i.e., read-write set of information) and sends it to endorsing peer. The endorsing peer validates the transaction by verifying the

client's public key and a time-stamp. The endorsing peer signs the received transaction and sends it to the client, and maintains a local copy in its ledger. The client waits for an ample number of endorsements and sends the endorsed transaction to the orderer. The orderer generates a valid block of transactions and sends it to committing peers to update their leger with the new block. The committing peer informs the client of the successful transaction. In last, the endorsing peer discards the local copy available in its ledger.

5.) *Orderer:* The orderer bundles the time-stamp transactions in a specific order to create a valid block. The transaction includes a header and a payload. The header comprises transaction and channel identification. Whereas payload contains time-stamp transactions and endorsing peer's signature. The block creates according to the maximum number of transactions limit and time out period.

6.) *Channel:* The channel is a medium to connect multiple organizations to receive the same information on the blockchain network. The channel also provides data privacy and confidentiality among organizations.

3. Related work

3.1. Blockchain Smart City

In [13] author proposed an IoT based smart manufacturing system for quality assurance application. The blockchain is utilized for building a trust relationship and for improving security concerns for manufacturing life-cycle processes. Various trust factors are mentioned with lesser security without practical guidelines. In [14] author proposed a local lightweight expandable blockchain model for a smart factory. Two defensive techniques are presented, such as whitelist and dynamic authentication to provide security and privacy. Also, an ACL is designed using Bell-La-Padula (BLP) and Biba models to prevent malicious activities. The use of bitcoin based local blockchain development limits the transactions per second and wastage resources. In [15] author proposed a useful resource utilization model for IoT devices in smart-city. The edge and miner nodes are placed together in a single blockchain network for the proper functioning of IoT devices which are connected with an edge network. The miner nodes are responsible for performing high computational tasks. The Proof-of-Work consensus is used to bring transparency and security. The utilization of consensus wastage more energy and requires heavy computational resources. In [16] author proposed the smart-city and blockchain notion by linking it with sharing economy services. The distributed technology brings trust, transparency, and privacy in the service relationship and eliminates intermediaries. Hence, it results in low operational costs and increases efficiency of sharing services. A theoretical viewpoint is provided with no practical implementation. In [17] author proposed a three-tier architecture for supporting scalable sharing economy services in the mega smart-city. The blockchain nodes perform data synchronization with the backend cloud-tier. The architecture is extended by adopting artificial intelligence models to capture the information and feed it into an artificial intelligence engine to identify the pattern through deep and convolutional neural networks. These patterns are used to share various economy services, depending on the need. In [18] author mentioned decentralized authentication and trust management for sensor-based IoT networks and designed a human like knowledge based trust model. This model determines the reputation of nodes and used pretty good privacy (OpenPGP) model for the authentication process.

3.2. Blockchain Emergency Services

In [19] author designed a decentralized authentication system using identity based signature scheme with multiple authorities (MA-IBM) approach and proposed a blockchain-based electronic health record (EHR) system. To identify MA-IBM security, a random oracle model is designed using deffie-helman assumption. The model may cause excessive communication overheads due to multiple

authority signatures. In [20] author proposed a patient centric access control for securing protected health information (PHI) using blockchain. For medical healthcare record security, a lightweight double encryption algorithm (i.e., ARX ciphers) is used and deffie-helman key exchange utilizes to transfer public keys. To bring anonymity and authenticity, a lightweight privacy preserving ring signature approach is proposed. In [21] author proposed a blockchain based secure and privacy preserving EHR sharing protocol. The EHR data sharing and privacy preservation is achieved through keyword search encryption and proxy re-encryption technique while sharing EHR information between different medical institutions data requestors. The proof-of-authentication is proposed as a consensus mechanism to build consortium blockchain regulation for efficient operation. The keyword search in the system may bring to endless search.

In [22] author proposed a lightweight access control system for IoT network based on blockchain. The management hub node holds the access control policies to permit the access for registered IoT devices. For security analysis STRIDE (i.e., spoofing, tempering, repudiation, information disclosure, denial of service (DoS) attack, and elevation of privileges) model is used to check against the presence of threats. The proposed architecture may suffer from overhead due to waiting of blockchain to permit access control information. Also, the presence of malicious management hub node could temper, repudiate and disclose information of IoT devices. In [23] author proposed a blockchain based emergency service architecture for smart home. To ensure security and privacy different authentication mechanisms such asymmetric key, digital signatures with interplanetary file system, and QR code through one-time password is used. In the proposed system, the public blockchain limits the smart contract security, and a secure file system is required. In [24] author mentioned a private blockchain-based access control (PBAC) model for smart home to protect against illegal access from service providers. The administrator use two-way secure authentication and token based access control policies to grant access of smart devices in a smart home to service providers. A certain time-interval is created for service provider during session creation to get home access may cause incompletion of service. In [25] author introduced a blockchain based remote user authentication system for smart home. For authentication between user and smart home gateway a group of signature and message authentication code techniques is proposed. An elliptic curve integrated encryption (ECIES) scheme is used for data transmission. The author suggests improving the access control policy by using ABAC in the future. In [26] author proposed an intelligent agriculture system based on blockchain. The bilinear pairing and dark web technology is used to create an agriculture network and private blockchain, respectively. The identity authentication mechanism is added to verify the legitimacy of any identity and hash-based message authentication code to determine message authenticity.

In [27] author proposed a blockchain-based secure firmware management framework for heterogeneous device management. The unidirectional, bidirectional, firmware update, and update propagation protocol are proposed for secure device management. The private blockchain is used to record the firmware transmission and update history. In [28] author designed a microgrids architecture for smart energy grid (SEG) using blockchain in smart-city. The blockchain_SEG application is developed based on the proposed model for information exchange and to buy or sell energy between energy distributors. The blockchain record the quantity of energy stored, selection of available energy supplier, and visualization of final sales.

3.3. Queuing Theory Model

In [29], the author proposed a M/M/n/L queuing theory model for the mining process simulation in the blockchain. In [23], the author presented a closed loop control system to evaluate the optimal number of blocks present in the queue of miner networks for IoT system using $\mathrm{M/M/1}$ queue theory model. In [31] author proposed a $M/M/c$ queuing theory model, which is utilized by hospital managers to guide nursing staff decisions. This model identifies the nurse-to-patient ratio needed to achieve patient services. The limitation is the assumption of a homogeneous workforce and not mentioned the use of distributed technology to bring more reliability into the proposed system model design. In [32] author mentioned an emergency service system to provide medical service at different geographical locations.

A hypercube queuing model with a multi-server is implemented for server-to-consumer services.
So far, a lot of work has been discussed for smart-city implementation, which includes many applications, i.e., smart-healthcare, smart-homes automation, firmware management, authentication system, etc., as shown in Table 1. After observing all these approaches and solutions mention by many authors, we propose a new model for calling PES in unusual environmental conditions. A private blockchain and reputation management are utilized to bring transparency and trust between user and PES provider. To handle the massive data generated through IoT devices, the ECS is used to store and process this data.

Table1: Comparison of different properties between the proposed system model and others

Paper	Blockchain	Smart Contract	Reputation Management	Authentication (access control)	Queuing Model
BMSTP	✓	✓	✓	✓	✓
[20]	✓	✓	×	✓	×
[25]	✓	✓	×	✓	×
[30]	✓	✓	×	×	✓
[33]	✓	✓	✓	×	✓

*Note that notation ✓ represent involved feature, × not involved feature

4. Blockchain Based Model For Secure and Trusted Public Emergency Service

In this section, we present the design of BMSTP. First, a three-layered architecture is presented, and then the components used in each layer with their roles are specified. Next, the working of BMSTP w.r.t. private blockchain and calling of smart contract functions are explained. In this paper, we focus on fire detection in a smart-home and distributing the PES request at minimum request queue length PESD to reach the smart-home location in minimum time.

4.1. System Architecture

The BMSTP architecture comprises three layers: infrastructure layer, edge layer, and blockchain layer, as shown in Fig. 3. The infrastructure layer is the collection of smart-homes and PESDs. Whereas the edge layer is comprising IoT controller and service controller. The role of blockchain layer is to receive and transfer PES requests and store their information.

1.) *Infrastructure Layer:* The infrastructure layer consist smart-homes and PESDs. The smart-home holds smart IoT devices includes a fire detector, a smoke detector, a fire alarm, and an IoT gateway. On the other side, the PESD manages multiple PESD service providers (i.e., fire brigades). Each PESD maintain their request queue to provide instant service to smart-homes in the smart-city.

2.) *Edge Layer:* The edge layer keeps IoT controller and service controller. The IoT controller performs multiple operations. These operations are data gathering and management of IoT devices and IoT gateway, continuously checking for IoT device's threshold values, and maintaining ACL for IoT devices and IoT gateway. The benefit of ACL is to detect the placement of new IoT devices near any smart-home by an adversary to misguide IoT controller. The service controller stores the information of multiple PESDs with their request queue. The service controller performs some local computation while sending the PES request to a particular PESD. This local calculation is useful to identify a suitable PESD while forwarding PES requests, which minimizes waiting time for smart-homes.

3.) *Blockchain Layer:* The blockchain layer is a combination of fabric organization. These fabric organizations are associated with either an IoT controller or a service controller connected through a common channel. The fabric organization store the smart contract information. This

smart contract triggers itself once the condition meets and generates a transaction on the blockchain.

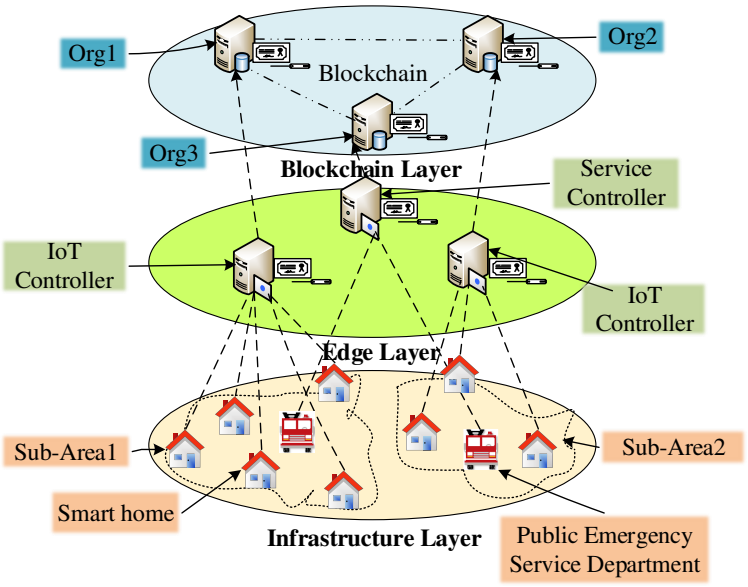

Fig 3: System Architecture of BMSTP

4.2. Working Architecture

This section describes, the functionality of each entity involved in the BMSTP from data generation to data processing. The following section provides an overview of network initialization, queuing model implementation, and reputation management for PESDs.

4.2.1. Network Initialization

The blockchain network initialization and smart contract installation are the basic operations for proper functioning of BMSTP as shown in Fig. 4. It is assumed that a smart-city is sub-divided into multiple sub-areas, and in each sub-area there is a PESD. According to the total number of sub-areas, an IoT controller is created to handle their allocated sub-area's PES requests. A single service controller is generated to manage all PESDs and their information. The fabric organization is represented as Org^F. Whereas the IoT controller and service controller is indicated as IoT^C and S^C.

Step1: A blockchain consists of N^{Org^F} fabric organization, where ($N^{Org^F} \in Org_1^F, Org_2^F, \dots, Org_n^F$) and each $IoT^C, S^C \in N^{Org^F}$. It is assumed that no two IoT controller or service controller belongs to same fabric organization. The IoT controller and service controller connects with the client in fabric organization to send and receive blockchain related information. The MSP creates certificates for each fabric-CA indicated as $Cert\_Org_{CA}^F$ to make fabric organization valid on the blockchain is given by Eq. (1).

$$MSP \xrightarrow{generate} \{Cert\_Org_{CA_1}^F, Cert\_Org_{CA_2}^F, \dots, Cert\_Org_{CA_n}^F\} \quad (1)$$

After receiving certificate from MSP, the fabric-CA generate certificates for their fabric organization peers is given by Eq. (2).

$$Org_{CA}^F \xrightarrow{generate} \{Cert\_Org_{peer_1}^F, Cert\_Org_{peer_2}^F, \dots, Cert\_Org_{peer_n}^F\} \quad (2)$$

Where, $Cert\_Org^{F}_{peer_n}$ is the certificate of n^{th} peer in the fabric organization, and it is assumed as per requirement a fabric organization may contain multiple peers. Once all fabric organization entities receive their certificates, they connect with a common cannel as given by Eq. (3).

$$\{Org^{F}_{1}, Org^{F}_{2}, \dots, Org^{F}_{n}\} \xrightarrow{connect} Channel \tag{3}$$

So far, the basic blockchain network is established. Now, install a smart contract on all fabric organization peers and channel through the software development kit (SDK) is given by Eq. (4).

$$Install\ Smart\ Contract \xrightarrow{SDK} \{Org^{F}_{peer_1}, Org^{F}_{peer_2}, \dots, Org^{F}_{peer_n}, channel\} \tag{4}$$

Step2: The IoT controller and service controller invoke register IoT controller ($register\_IC$) and register service controller ($register\_SC$) smart contract function, respectively describe in section 4.3. They both provide registration information to their respective fabric organization and in return receive a pair of public-private keys, which uniquely identify them on the blockchain.
Step3: After successful registration, the IoT controller and service controller starts registering smart-homes and PESDs, respectively. The smart-home calls register smart home ($register\_SH$) function, and PESD invokes register public emergency service department ($register\_PESD$) function.

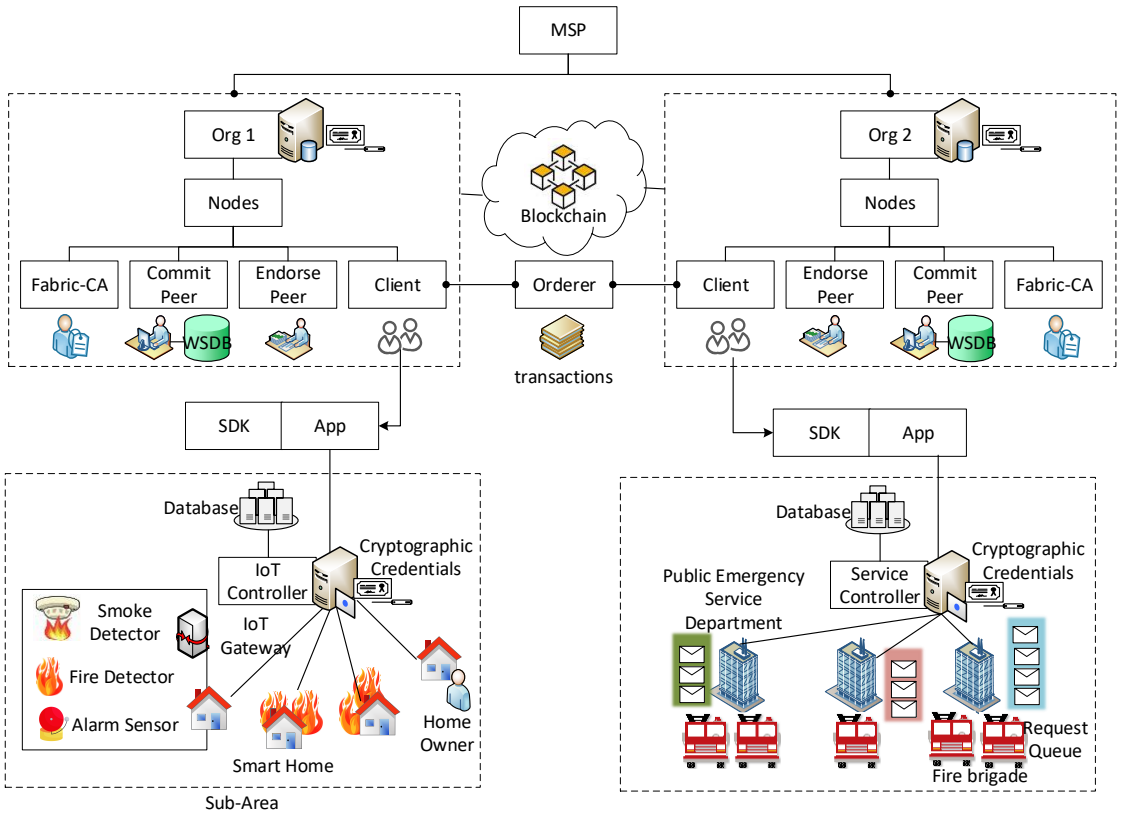

Fig 4: Working Architecture of BMSTP

4.2.2. Selection of Public Emergency Service Department using Queuing Model

In this section, to select the PESD, a queuing theory model for igniting smart-home is defined as shown in Fig. 5. The queuing model helps the service controller to identify an appropriate PESD with a minimum request queue [33]. The selected PESD receive a PES request for an igniting smart-home and reach at the smart-home location to protect against catastrophe. Let the classical $M/M/c$ queuing

theory model is used to represent the above scenario, where smart-home PES request follow the first-come-first-serve (FCFS) queuing discipline. Recall from Kendall's notation, the first and second M represents the inter-arrival time and service time, respectively. The inter-arrival time follows the Poisson distribution, whereas service time is expressed using Markovian Exponential distribution. In the proposed queuing theory scheme, c denotes the number of PESDs according to sub-areas in a smart-city. It is assumed that each PESD contain their own request queue to handle PES requests and this information is centrally maintained at service controller. The waiting time for igniting smart-home to receive PES request confirmation depends on two parameters. These are the local computation of service controller to identify a PESD with minimum request queue and service rate of PESD. The PESD request queue length is represented as $\mathbb{L}^{queue}$, and waiting time of smart-home is indicated as ω^{time}. Evaluating request queue and waiting time some mathematical notations [34, 35] is derived for the BMSTP model. The utilization of i^{th} PESD is represented using $\rho(i)$ to assist igniting smart-homes PES requests is given by Eq. (5).

$$\rho(i) = \frac{\lambda(i)}{\mu(i)} \tag{5}$$

Where $\lambda(i)$ and $\mu(i)$ is the inter-arrival rate and service rate of i^{th} PESD, respectively. The queue length of i^{th} PESD is given by Eq. (6) and (6a).

$$\mathbb{L}_i^{queue} = \frac{\mathcal{P}_i \times (\rho(i))^{N^{PESD}} \times \rho(i)}{N^{PESD}! \times (1-\rho)^2} \tag{6}$$

$$\mathcal{P}_i = 1 \Big/ \left[\sum_{k=0}^{N^{PESD}-1} \frac{(N^{PESD} \times \rho)^k}{k!} + \frac{(N^{PESD} \times \rho)^{N^{PESD}}}{N^{PESD}! \times (1-\rho)} \right] \tag{6a}$$

Where, $\mathbb{L}_i^{queue}$ and $\mathcal{P}_i$ are request queue length and probability of idleness of i^{th} PESD where N^{PESD} is the total number of PESD. The waiting time of igniting smart-home PES requests before getting any confirmation of PESD is given by Eq. (7).

$$\omega_{j,i}^{time} = \frac{\mathbb{L}_i^{queue}}{\lambda(i)} \tag{7}$$

Where, $\omega_{j,i}^{time}$ is the waiting time of j^{th} igniting smart-home PES request residing in i^{th} PESD request queue and waiting for confirmation of preceding PES requests to get its chance. To specify the functionality of queuing theory model in more detail, two cases are considered as discussed below.

Step1: The IoT controller continuously fetches their sub-area smart-homes IoT devices (i.e., fire detector and smoke detector) data. These IoT devices are connected with IoT gateway, which sends this data to IoT controller. Let Th^{α} and Th^{β} represents fire detector and smoke detector threshold value, respectively. If fire detector and smoke detector values reach to Th^{α} and Th^{β} a PES request for igniting smart-home is forwarded from IoT gateway to IoT controller to the blockchain.
Step2: The service controller receives the PES request via a blockchain. It first checks the smart-home sub-area represented as SH^{SA} with all PESD sub-area indicated as $PESD^{SA}$. After comparison, two cases are considered as follows.

1.) *Case1:* If matched sub-area PESD request queue is shorter than all others, the service controller select that PESD and sends the blockchain request to it. The PESD receive the blockchain request and send its service provider to igniting smart-home location
2.) *Case2:* If the matched sub-area PESD request queue is longer than others, the service controller compares the request queue of all PESD and select the one with the minimum request queue. The selected PESD receives a blockchain request from service controller, and PESD sends its service provider to igniting smart-home location.

Step3: A transaction omits from service controller to confirm the arrival of selected PESD service provider at igniting smart-home location.
Step4: After completing the PES request by PESD service provider, the IoT controller sends a transaction on behalf of igniting SH on the blockchain. This transaction contains a rating for PESD service provider to indicate the satisfactory level of igniting smart-home, which is later used to generate reputation value for PESD.

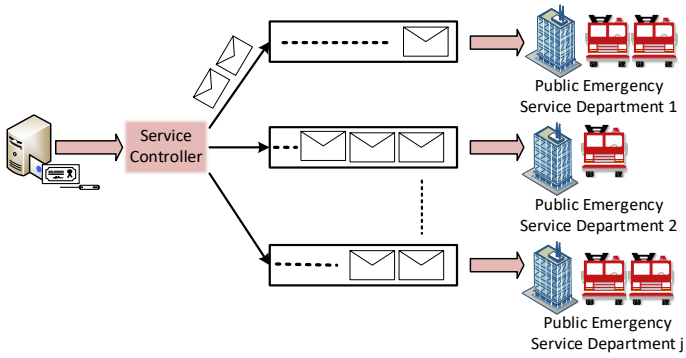

Fig 5: Queuing Model for BMSTP

Algorithm 1: Selection of public emergency service department using queuing model

Input: Threshold value: Th^{α}, Th^{β}, PESD request queue: $\mathbb{L}^{queue}$, smart-home sub-area: SH^{SA}, PESD sub-area: $PESD^{SA}$
Output: allocation of i^{th} PESD to j^{th} ignited smart-home
begin
$Fire_{value}^{detector} = rand1; = Smoke_{value}^{detector} = rand2;$ /* consider random value*/
1. ***For*** $j = 1$ *to* N^{SH} /*N^{SH} indicate the total number of smart-homes*/
2. ***For*** $i = 1$ *to* N^{PESD}
3. ***if*** $\left(Fire_{value_j}^{detector} \geq Th^{\alpha} \;\&\&\; Smoke_{value_j}^{detector} \geq Th^{\beta}\right)$
4. ***if*** $\left(SH_j^{SA} == PESD_i^{SA} \;\&\&\; \mathbb{L}_i^{queue} \leq \mathbb{L}^{queue}(N^{PESD} - i)\right)$
5. ***Print*** *Same Sub − Area* i^{th} *PESD is selected*
6. ***Call Algorithm 2*** *to generate rating for selected PESD*
7. ***else***
8. ***min***$(\mathbb{L}^{queue}(N^{PESD} - i))$ /*identify PESD with minimum queue length*/
9. ***Print*** *Other Sub − area* i^{th} *PESD is selected*
10. ***Call Algorithm 2*** *to generate rating for PESD*
11. ***else***
12. ***Print*** *Everything Under Control*
13. ***End if***
14. ***End if***
15. ***End for***
16. ***End for***
End

4.2.3. Reputation Management for Public Emergency Service Department

To evaluate the reputation value for a PESD, a simple reputation management model is used [36]. The reputation management value is highly dependent on smart-homes. After getting service from a selected PESD service provider, a smart-home generate a rating for it. These ratings are used to evaluate the final reputation value for PESD. This reputation management model helps the smart-homes to see the reputation value of different PESD. It is also beneficial for PESD to see their

performance and take necessary action to improve their PES in the future if required. To provide a reputation value for PESD through an igniting smart-home, we assumed two scenarios. These are in-time-PES and delayed-PES as described below.

Step1: The distance between selected i^{th} PESD and j^{th} smart-home is represented as $D_{i,j}$, is given by Eq. (8).

$$D_{i,j} = \sqrt{\left(X_i^{PESD^{LOC}} - X_j^{SH^{LOC}}\right)^2 + \left(Y_i^{PESD^{LOC}} - Y_j^{SH^{LOC}}\right)^2} \tag{8}$$

Where, the location coordinate of i^{th} PESD and j^{th} igniting smart-home are denoted as $\left(X_i^{PESD^{LOC}}, Y_i^{PESD^{LOC}}\right)$ and $\left(X_j^{SH^{LOC}}, Y_i^{SH^{LOC}}\right)$, respectively. To evaluate the reaching time indicated as $RT_{i,j}$ of selected i^{th} PESD service provider to j^{th} igniting smart-home is calculated using Eq. (8) is given by Eq. (9).

$$RT_{i,j} = \frac{D_{i,j}}{Avg_{PESD_i}^{speed}} \tag{9}$$

Where, $Avg_{PESD_i}^{speed}$ is assumed as an average speed of i^{th} PESD.

Step2: The rating for i^{th} PESD generated by j^{th} igniting smart-home is represented as $R_{i,j}$ is given by Eq. (10).

$$R_{i,j} = b + e^{-\gamma.D_{i,j}} \tag{10}$$

Where, b and γ are two parameters that control lower bound, and change in rate for rating value, respectively. To evaluate the expected reaching time of i^{th} PESD at j^{th} igniting smart-home is represented as $ERT_{i,j}$ is given by Eq. (11).

$$ERT_{i,j} = \omega_{j,i}^{time} + RT_{i,j} + \mathbb{T}_{i,j} \tag{11}$$

Where, $\omega_{j,i}^{time}$ is the waiting time of j^{th} igniting smart-home PES request in i^{th} PESD before receiving any confirmation of PESD service provider and $\mathbb{T}_{i,j}$ is time duration taken by i^{th} PESD during traveling at j^{th} igniting smart-home location due to high traffic. The ERT is calculated and attached by service controller while sending a confirmation of PESD service provider through the blockchain. Once the i^{th} PESD service provider reached at j^{th} igniting smart-home location, an actual reaching time is recorded represented as $ART_{i,j}$, which is similarly calculated using Eq. (11) by varying $\mathbb{T}_{i,j}$ value.

1.) *Case1: In-time-PES:* In this procedure, $ART_{i,j}$ is compared with $ERT_{i,j}$, if $ART_{i,j}$ is shorter then $ERT_{i,j}$ a positive rating generate for i^{th} PESD by j^{th} igniting smart-home is represented as $PR_{i,j}$ is given by Eq. (12).

$$PR_{i,j} = PR_{i,j} + \left(R_{i,j} \times (+1)\right) \tag{12}$$

Where, $R_{i,j}$ is rating generated by i^{th} igniting smart-home for j^{th} PESD is multiplied by $+1$ to compute a positive rating.

2.) *Case2: Delayed-PES:* In delayed-PES, if $ART_{i,j}$ value is greater than $ERT_{i,j}$ a negative rating propose for i^{th} PESD indicated as $NR_{i,j}$ is given by Eq. (13).

$$NR_{i,j} = NR_{i,j} + \left(R_{i,j} \times (-1)\right) \tag{13}$$

Here, $R_{i,j}$ is multiplied by -1 to generate a negative rating.

Step3: Using Eq. (12) and (13), the service controller obtains a final reputation value represented as FR_i for i^{th} PESD is given by Eq. (14).

$$FR_i = \sum_{t=0}^{\chi}\left(\sum_{j=1}^{N^{SH}} PR_{i,j}\right) + \sum_{t=0}^{\chi}\left(\sum_{j=1}^{N^{SH}} NR_{i,j}\right) \quad (14)$$

Where, χ and N^{SH} represents time-interval of twenty-four hours and the total number of smart-homes, respectively.

Step4: The service controller upload this FR_i on the blockchain for further use.

Algorithm 2: Reputation Management for Public Emergency Service Department

Input: Smart-home coordinate: $X_j^{SH^{LOC}}, Y_i^{SH^{LOC}}$, PESD coordinate: $X_i^{PESD^{LOC}}, Y_i^{PESD^{LOC}}$, Expected reaching time of i^{th} PESD at j^{th} smart-home: $ERT_{i,j}$, Average speed of i^{th} PESD.

Output: Positive rating for i^{th} PESD: $PR_{i,j}$, Negative rating for i^{th} PESD: $NR_{i,j}$, Final reputation for i^{th} PESD: $D_{i,j}$

Begin

1. $Calculate\ \ distance = D_{i,j} = \sqrt{\left(X_i^{PESD^{LOC}} - X_j^{SH^{LOC}}\right)^2 + \left(Y_i^{PESD^{LOC}} - Y_j^{SH^{LOC}}\right)^2}$
2. $Calculate\ expected\ reaching\ time = \ ERT_{i,j} = \omega_{j,i} + \frac{D_{i,j}}{Avg_{PESD_i}^{speed}} + \mathbb{T}_{i,j}$
3. $Calculate\ rating = R_{i,j} = b + e^{-\gamma . D_{i,j}}$
4. $\boldsymbol{if}\left(ART_{i,j} \le ERT_{i,j}\right)$
5. $\quad PR_{i,j} = PR_{i,j} + \left(R_{i,j} \times (+1)\right)$
6. $\quad \boldsymbol{Print}\ j^{th}\ PESD\ get\ the\ positive\ rating$
7. ***else***
8. $\quad NR_{i,j} = NR_{i,j} + \left(R_{i,j} \times (-1)\right)$
9. $\quad \boldsymbol{Print}\ j^{th}\ PESD\ get\ the\ negative\ rating$
10. ***End if***
11. $FR_i = \sum_{t=0}^{\chi}\left(\sum_{j=1}^{N^{SH}} PR_{i,j}\right) + \sum_{t=0}^{\chi}\left(\sum_{j=1}^{N^{SH}} NR_{i,j}\right)$
12. $\boldsymbol{Print}\ final\ rating\ for\ j^{th}\ PESD\ is\ FR_i$

End

4.3. Smart Contract

In this section, the functionality of different smart contract functions is defined. These functions are $register\_IC, register\_SC, register\_SH, register\_PESD, call\_PESD\_serviceProvider$, $ratingGeneration\_PESD$ and $finalReputaionUpdation\_PESD$.

4.3.1. Registration of IoT Controller (register_IC)

Step1: The IoT controller call register IoT controller ($register\_IC$) smart contract function through client to become the legitimate entity. For completing the registration process, it passes required information includes IoT controller valid identity (IoT^{C_ID}), and sub-area (IoT^{C_SA}) is given by Eq. (15).

$$register\_IC = < IoT^{C\_ID}, IoT^{C\_SA} > \quad (15)$$

Step2: The endorsing peer receives registration request and process. The endorsing peer check the provided information and use its digital certificate to sign the registration request and send back to client using blockchain transaction (Tx) is given by Eq. (16).

$$Tx = Cert\_Org_{peer}^{F} < register\_IC > \quad (16)$$

The client collects the signed transaction and forwards it to the orderer. The orderer verifies collected information and broadcasts the new block of valid transactions to committing peers of every fabric organization is given by Eq. (17).

$$Block = Orderer < Tx^{ID} || register\_IC || Cert\_Org_{peer}^{F} >, \text{ where, } Tx^{ID} \text{ is transaction identity} \quad (17)$$

Step3: The committing peer informs the client of successful registration and generates a pair of public-private key for IoT controller $\left(PK^{IoT^C}, SK^{IoT^C}\right)$. The public key is used to uniquely identity the IoT controller on the blockchain.

4.3.2. Registration of Service Controller (register_SC)

Step1: The service controller invokes register service controller ($register\_SC$) smart contract function via client. The service controller provides necessary information for completing registration. This includes service controller valid identity (S^{C_ID}), category (i.e., fire brigade as PES), and predetermined threshold values (Th^{α}, Th^{β}) is given in Eq. (18).

$$register\_SC = < S^{C\_ID}, \text{category}, Th^{\alpha}, Th^{\beta} > \quad (18)$$

Step2: The endorsing peer collects the registration request and sign the registration request using its digital signature. This signed registration request is returned to client through Tx is given by Eq. (19).

$$Tx = Cert\_Org_{peer}^{F} < register\_SC > \quad (19)$$

The client receives the signed transaction and address to the orderer. The orderer check received information and broadcast the new block to committing peer to update their ledger with updated information is given by Eq. (20).

$$Block = Orderer < Tx^{ID} || register\_SC || Cert\_Org_{peer}^{F} > \quad (20)$$

Step3: The commit peer update the client and obtain a pair of public-private key for the service controller $\left(PK^{S^C}, SK^{S^C}\right)$.

4.3.3. Registration of Smart Home (register_SH)

Step1: The registration of smart-home perform indirectly through IoT controller. The smart-home call API of register smart home ($register\_SH$) smart contract function via IoT controller. The smart-home provide necessary information includes IoT controller public key (PK^{IoT^C}), smart-home location ($X^{SH^{LOC}}, Y^{SH^{LOC}}$), smart-home sub-area ($SH^{SA}$), category, smart-home owner phone number (SH^{PH}), fire detector identity ($Fire^{detector^{ID}}$), smoke detector identity ($Smoke^{detector^{ID}}$), fire alarm identity ($Fire^{alarm^{ID}}$), and IoT gateway identity ($IoT^{gateway^{ID}}$) is given by Eq. (21).

$$register\_SH =< PK^{IoT^C}, X^{SH^{LOC}}, Y^{SH^{LOC}}, SH^{SA}, category, SH^{PH}, Fire^{detector^{ID}}, Smoke^{detector^{ID}}, Fire^{alarm^{ID}}, IoT^{gateway^{ID}} > \quad (21)$$

Step2: The IoT controller receives this information and sign registration request using SK^{IoT^C}. This signed information is forward to endorsing peer is given by Eq. (22).

$$Tx = SK^{IoT^C} < register\_SH > \quad (22)$$

The endorsing peer verifies the IoT controller PK^{IoT^C} and sign registration request using digital signature. This signed transaction is sent back to client through Tx is given by Eq. (23).

$$Tx = Cert\_Org_{peer}^{F} < register\_SH || PK^{IoT^C} > \quad (23)$$

The client forwards this signed transaction to the orderer. The orderer validates information and generates a new block. This block is broadcast to the committing peer is given by Eq. (24).

$$Block = Orderer < Tx^{ID} || register\_SH || Cert\_Org_{peer}^{F} > \quad (24)$$

Step3: The committing peer inform the client and return a pair of public-private key for smart-home IoT gateway ($PK^{SH^{IoT\_G}}, SK^{SH^{IoT\_G}}$). The IoT controller informs the smart-home for successful registration and provides the same key pair. The IoT controller store $PK^{SH^{IoT\_G}}$ of smart-home IoT gateway and identities of multiple IoT devices in its ACL.

4.3.4. Registration of Public Emergency Service Department (register_PESD)

*Step1: T*he PESD invokes register public emergency service department ($register\_PESD$) smart contract function API via service controller. The PESD send the desired information includes service controller public key (PK^{S^C}), PESD location ($X^{PESD^{LOC}}, Y^{PESD^{LOC}}$), PESD sub-area ($PESD^{SA}$), PESD valid identity ($PESD^{ID}$), and PESD request queue ($\mathbb{L}^{queue}$) is given by Eq. (25).

$$register\_PESD =< PK^{S^C}, X^{PESD^{LOC}}, Y^{PESD^{LOC}}, PESD^{SA}, PESD^{ID}, \mathbb{L}^{queue} > \quad (25)$$

Step2: The service controller receive PESD registration request information and sign it using SK^{S^C}. The service controller transfer this signed registration request to endorsing peer is given by Eq. (26).

$$Tx = SK^{S^C} < register\_PESD > \quad (26)$$

The endorsing peer checks the received information to sign the transaction using its digital signature and return it to client using Tx is given by Eq. (27).

$$Tx = Cert\_Org_{peer}^{F} < register\_PESD || PK^{S^C} > \quad (27)$$

The client forwarded this signed request transaction to the orderer. The orderer generates a new block and pass it to committing peer to update their ledger information is given by Eq. (28).

$$Block = Orderer < Tx^{ID} || register\_PESD || Cert\_Org_{peer}^{F} > \quad (28)$$

Step3: The committing peer notifies the client about successful registration of PESD and returns a pair of public-private key (PK^{PESD}, SK^{PESD}). The service controller informs the PESD and forwards the same key pair to PESD and store PK^{PESD} and $\mathbb{L}^{queue}$ in its local database.

4.3.5. Call Public Emergency Service Department Service Provider (call_PESD_serviceProvider)

Step1: The IoT gateway use $SK^{SH^{IoT\_G}}$ and send the IoT device data to the IoT controller. The IoT controller continuously monitors these smart IoT device data. When threshold reaches the IoT controller invokes call public emergency service department service provider ($call\_PES\_servicProvider$) smart contract function on behalf of smart-home. The function contains necessary information includes $PK^{SH^{IoT\_G}}, X^{SH^{LOC}}, Y^{SH^{LOC}}, SH^{SA}, Th^{\alpha}, Th^{\beta}$. This information is encrypted using IoT controller SK^{IoT^C} is given by Eq. (29).

$$call\_PESD\_serviceProvider = SK^{IoT^C} < PK^{SH^{IoT\_G}}, X^{SH^{LOC}}, Y^{SH^{LOC}}, SH^{SA}, Th^{\alpha}, Th^{\beta} > \quad (29)$$

The PES request of smart-home is broadcast to the blockchain through fabric organization is given by Eq. (30).

$$Tx = Cert\_Org^{F}_{peer} < call\_PESD\_ServiceProvider > \quad (30)$$

Step2: The service controller retrieve required information from Tx to avail PESD service provider with minimum waiting, described above in section 4.2.2
Step3: After selecting PESD, the service controller proposes a transaction that includes ERT and PK^{PESD} is given by Eq. (31).

$$Tx = Cert\_Org^{F}_{peer} = < ERT, PK^{PESD} > \quad (31)$$

The orderer receives transaction information and generates a new block and broadcast. The other fabric organization receives this information and uses it later for rating generation is given by Eq. (32).

$$Block = Orderer < Tx^{ID} || call\_PESD\_serviceProvider || Cert\_Org^{F}_{peer} > \quad (32)$$

4.3.6. Reputation Generation for Public Emergency Service Department (reputationGeneration_PESD)

Step1: The IoT controller use SK^{IoT^C} to generate either a positive or negative rating for PESD by calling call reputation generation for public emergency service department ($reputationGeneration\_PESD$) smart contract function on behalf of igniting smart-home. This function contains necessary information such as PK^{PESD}, $PK^{SH^{IoT\_G}}$, and either a PR or a NR, described in section 4.2.3 is given by Eq. (33)

$$reputationGeneration\_PESD = SK^{IoT^C} < ART, ERT, PK^{PESD}, PK^{SH^{IoT\_G}}, PR/NR > \quad (33)$$

Step2: This information is forward on the blockchain through IoT controller's fabric organization to take further action is given by Eq. (34).

$$Tx = Cert\_Org^{F}_{peer} < reputationGeneration\_PESD > \quad (34)$$

Step3: The orderer receives reputation updation information and generates a new block to broadcast th information to other fabric organizations is given by Eq. (35).

$$Block = Orderer < Tx^{ID} || reputationGeneration\_PESD || Cert\_Org^{F}_{peer} > \quad (35)$$

4.3.7. Final Reputation Updation for Public Emergency Service Department (finalReputationUpdatation_PESD)

Step1: At the end of the day, the service controller evaluates the final reputation using ratings generated by multiple igniting smart-homes after fulfilling PES requests by calling the final reputation updation

for public emergency service department ($finalReputationUpdation\_PESD$) smart contract function. The function parameters include PK^{PESD} and FR, which is encrypted using SK^{S^C} is given by Eq. (36).

$$finalReputationUpdataion\_PESD = SK^{S^C} < PK^{PESD}, FR > \quad (36)$$

Step2: The service controller's fabric organization receives this information and forward the signed transaction to the blockchain is given by Eq. (37).

$$Tx = Cert\_Org_{peer}^{F} < finalRputationUpdataion\_PESD > \quad (37)$$

Step3: The orderer process this information to create a new block and broadcast it to other fabric organizations is given by Eq. (38).

$$Block = Orderer < Tx^{ID} || finalReputationUpdataion\_PESD || Cert\_Org_{peer}^{F} > \quad (38)$$

Algorithm 3: Smart Contract Functions

Begin

1. ***register_IC*** *function*
2. ***For*** $\mathcal{M} = 1$ *to* m
3. ***generate*** $< IoT_m^{C\_ID}, IoT_m^{C\_SA} >$
4. ***Registration Sucessful*** & ***Return pair of public − private key***
5. ***End For***
6. ***register_SC*** *function*
7. ***For*** $\mathcal{N} = 1$ *to* n
8. ***generate*** $< S_n^{C\_ID}, category, Th_n^{\alpha}, Th_n^{\beta} >$
9. ***Registration Sucessful*** & ***Return pair of public − private key***
10. ***End For***
11. ***register_SH*** *function*
12. ***For*** $\mathcal{P} = 1$ *to* p
13. ***generate*** $< PK^{IoT^C}, X_p^{SH^{LOC}}, Y_p^{SH^{LOC}}, SH_p^{SA}, category, SH_p^{PH}, Fire_p^{detector^{ID}}, Smoke_p^{detector^{ID}}, Fire_p^{alarm^{ID}}, IoT_p^{gateway^{ID}} >$
14. ***Registration Sucessful*** & ***Return pair of public − private key***
15. ***End For***
16. ***register_PESD*** *function*
17. ***For*** $\mathcal{Q} = 1$ *to* q
18. ***generate*** $< PK_q^{S^C}, PX_q^{PESD^{LOC}}, PY_q^{PESD^{LOC}}, PESD_q^{SA}, PESD_q^{ID}, \mathbb{L}_q^{queue} >$
19. ***Registration Sucessful*** & ***Return pair of public − private key***
20. ***End For***
21. ***call_PESD_serviceProvider*** *function*
22. ***For*** $\mathcal{R} = 1$ *to* r
23. ***generate*** $< PK_r^{SH^{IoT\_G}}, X_r^{SH^{LOC}}, Y_r^{SH^{LOC}}, SH_r^{SA}, Th_r^{\alpha}, Th_r^{\beta} >$
24. ***Call Algorithm 1***
25. ***End For***
26. ***reputationGeneration_PESD*** *function*
27. ***For*** $\mathcal{S} = 1$ *to* s
28. ***generate*** $< ART, ERT, PK_s^{PESD}, PK^{SH^{IoT\_G}}, PR/NR >$
29. ***Call Algorithm 2***
30. ***End For***
31. ***finalReputationUpdation_PESD*** *function*
32. ***For*** $\mathcal{U} = 1$ *to* u

33. $\boldsymbol{generate} < PK_u^{PESD}, FR_u >$
34. $\boldsymbol{Call\ Algorithm\ 2}$
35. ***End For***
End

5. Simulation Results and Discussion

In this section, various simulations are performed to evaluate the performance of the proposed BMSTP to demonstrate PESD functionality. Hyperledger fabric 1.2v is used to implement smart contract function and python is utilized to call blockchain API. We assumed an eight-digit identification code using a random number generator for IoT devices and IoT gateway connected with smart-home. In this way, the IoT controller identifies the IoT device and IoT gateway connected with the smart-home and stores information in ACL.
The proposed system model consists of eight fabric organization docker nodes. Seven fabric organization is associated with IoT controllers while the eighth is connected with service controller. In Hyperledger fabric, the cryptogenic tool is useful for generating the certificates and a pairs of public-private key for multiple entities, including endorsing peer, committing peer, orderer, client, etc. Whereas the configtxgen tool is used to generate the genesis block, which contains a blockchain configuration with a channel. After the successful setup of Hyperledger fabric, install the smart contract to various functions.

Table 2. Parameter Settings

Parameters	Value
Sub-area	7
Smart-homes in each sub-area	47
IoT Controller	7
Service Controller	1
Total PESD and request queue	7
b	0.5
γ	0.014
Th^{α} and Th^{β}	7.25 and 8.0
Distance between smart home and public emergency service department (D)	The uniform distribution between 15 to 500 meters
$\mathbb{T}$	The uniform distribution between 15 to 30 minutes
χ	Time interval of 24 hours
Avg_{PESD}^{speed}	40-60 Km/hr.
Data Size	Up to 1000 bytes

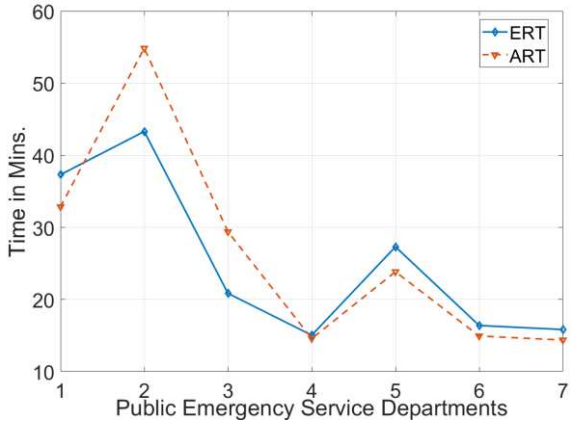

Fig. 5. Comparison of ERT and ART for different PESDs

A comparison between the expected reaching time and the actual reaching time of PESDs is represented using blue and red line in given Fig. 5. The value of these two parameters is evaluated by using Eq. (11). The information is generated by igniting smart-home located in different sub-areas for seven PESDs. As indicated in graph, the second and third PESD is unable to reach before expected reaching time. Hence, they receive a negative rating from allocated igniting smart-homes. Whereas, the other PESDs get a positive rating from respective smart-home after fulfillment of PES request.

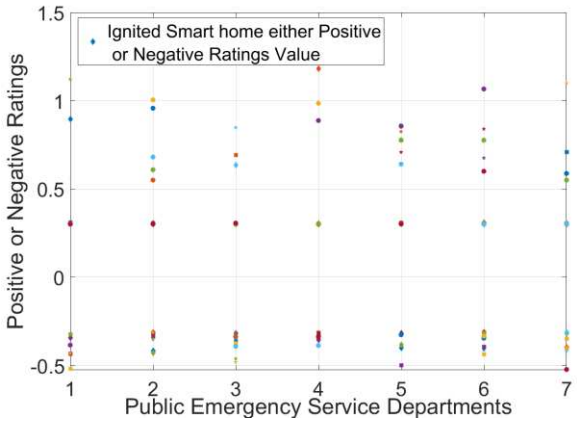

Fig 6: Positive or negative rating by multiple smart-homes to PESDs

In Fig. 6, a one-to-one relationship of the rating between igniting smart-home and PESD is represented. The seven PESD receive either a positive rating or a negative rating according to their service fulfillment time. For the given graph, the data is generated using Eq. (12) and (13) by considering parameter values of Table II. The rating for PESD lies between positive and negative range. The positive rating indicates the PESD fulfills the PES request of igniting smart-home in-time and vice-versa. After successful completion of PES request of igniting smart-home an IoT controller generates a rating for allocated PESD on behalf of smart-home on the blockchain. Later, this range of ratings is useful to calculate the final reputation for PESDs.

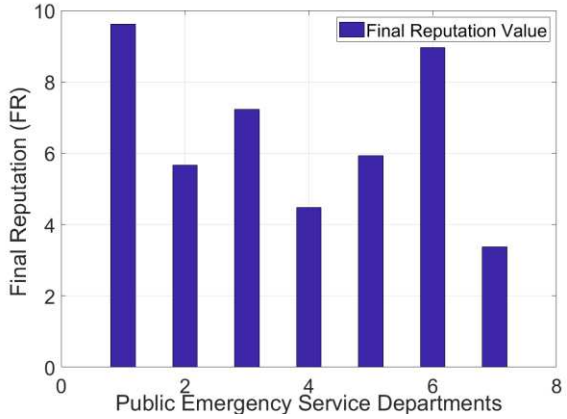

Fig 7: Final reputation for PESDs

A comparison of final reputation between different PESDs can observe through Fig. 7. To evaluate the final rating for each PESD, we used the data of Fig. 6 and inserted it in Eq. (14). According to the given Fig., the first PESD has the highest reputation among all PESDs due to more number of in-time-PES fulfillment. Whereas the seventh PESD has the lowest rank compared to other PESDs because it serves more number of delayed-PES to igniting smart-homes. Due to delayed-PES, the igniting smart-homes generate a negative rating for PESD. This reputation management is useful to analyze the reason for the low rank of PESD and can take necessary action to improve the performance in real-time.

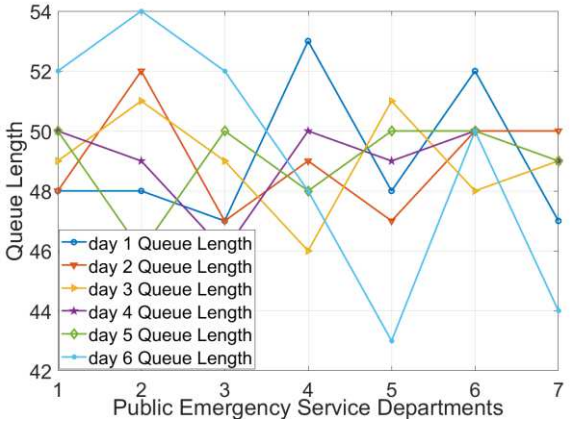

Fig 8: Request queue length of PESDs at different days

Fig. 8 shows the relation between request queue length and PESDs. The data for request queue length at PESDs location is generated for six days indicated through multiple color lines. The request queue length is obtained by inserting inter-arrival rate and service rate in Eq. (6) and (6a). For the upcoming PES requests of multiple igniting smart-homes, this request queue is used to decide which PESD accepts the next PES request. For example, the request queue length for day five is $\{50, 46, 50, 48, 50, 50, 49\}$. When a new PES request reaches at IoT controller, the IoT controller first match the sub-area. If the matched PESD has a minimum request queue length, then the corresponding PESD is selected otherwise, the request forward to the PESD with the shortest request queue among all (i.e., second PESD).

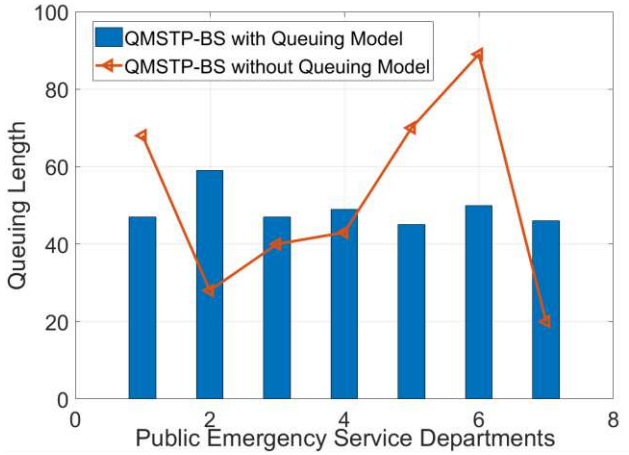

Fig 9: Comparison between BMSTP with and without queuing model

The comparison between BMSTP architecture with and without queuing model is presented in Fig. 9. It is evident from the results that PES request's load is adequately distributed between available PESDs according to their request queue length. Hence, it helps in minimizing waiting time of igniting smart-home with incrementing number of PES requests. Whereas due to lack of queuing model, every PESD only handles their own sub-area PES requests. Therefore, it could lead to the high waiting time for igniting smart-home and congest a PESD under high PES requests load. As shown in the given Fig., there is a drastic change in the request queue of each PESD using the without queuing model. For example, the second and seventh PESD in without queuing model has less load than others, which can be utilized in a better way to handle massive PES requests to protect against disaster in real-time.

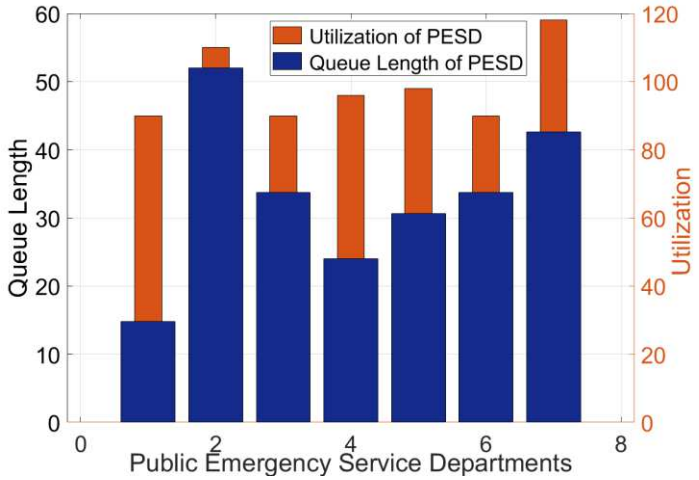

Fig 10: PESD processing time and queue length.

Fig. 10 represents a relation between queue length and processing time or utilization for PESDs. We assumed a service rate, which is randomly distributed between $\{5-7\}/hr$ for PESDs. To evaluate the processing time and request queue length Eq. (5) and (6) are utilized. As shown in the given Fig., the blue bar indicates the request queue whereas the orange bar indicates each PESD. Due to more number of service providers (i.e., fire brigade) the first PESD request queue length is shorter and can maximize their resource utilization for future PES requests. Whereas, the second PESD has longer request queue because of fewer service providers or the worst utilization of its available resources. The processing speed for each PESD varies according to the number of PESD service provider's availability. The rest of PESDs show a balance between request queue and processing speed to fulfill PES requests.

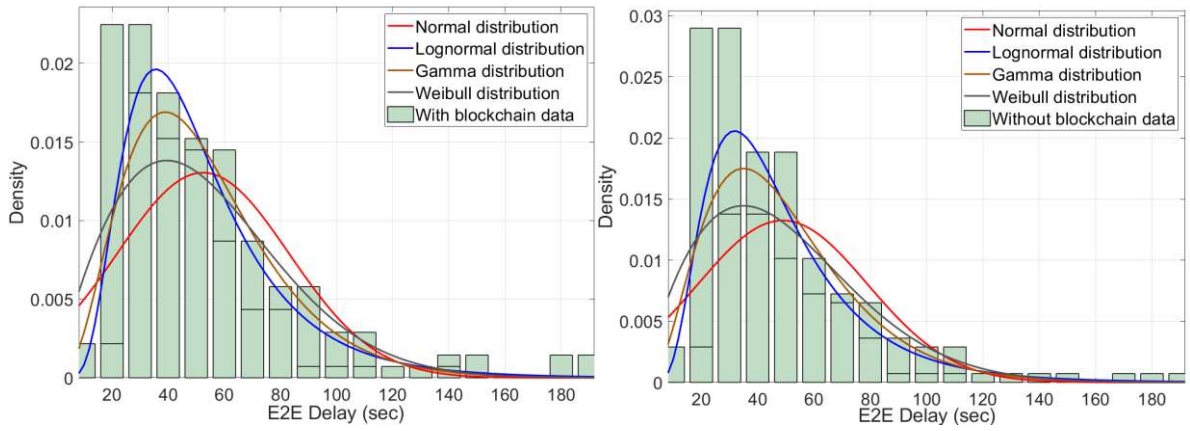

Fig 11 (a): E2E delay with blockchain (b) E2E delay without blockchain

To analyze the behavior of the proposed system model a comparison of End-to-End (E2E) delay with and without blockchain is indicated in Fig. 11. We consider a sum of request and response time, which is known as E2E delay. A request time is calculated to generate with blockchain graph for an igniting smart-home PES request, which sends PES request from IoT gateway to IoT controller to the blockchain. Similarly, for response time estimation, a confirmation from the blockchain to IoT server for PESD service provider's arrival is observed. In comparison, without blockchain, the blockchain time is eliminated for request and response time. Direct communication takes place between IoT controller and service controller for confirmation and arrival of PESD service provider. For the given graph, different distribution functions are considered to identify the proposed system.

6. Conclusion and Future Work

The blockchain holds the promises for transparency, trust, and privacy for IoT-based smart-city. Therefore, applying blockchain directly to IoT networks is not a good option because of numerous challenges, including resource consumption, processing time, storage, and scalability. In this paper, we proposed a three-layered architecture of BMSTP that help in providing reliable PES. In the proposed system model, the ECS and queuing theory model is used to gain fast access to PES resources. The benefit of ECS is off-chain storage, proper management of IoT devices through ACL, and scalability. Whereas, queuing model helps in selecting appropriate PESD. The overall system model is designed using private blockchain, which maintains record of IoT controllers, service controller, smart-homes and PESDs in distributed ledger. The transfer of PES requests and arrival of PESD service provider is ensure using smart contract function implementation. We extended this work by maintaining reputation management for PESDs. The smart-home rate PESD according to their service fulfillment and generate either a positive or negative rating accordingly. The results indicate that our system model is sufficient to handle PES requests in real-time and ensure minimum waiting for igniting smart-homes. In the future, instead of single PES, multiple PESs may add together on a single blockchain platform. So that the users can take advantage of PES with better convenience.

Declarations

1. Data sharing not applicable to this article as no datasets were generated or analysed during the current study.
2. The authors declare that there is no conflict of interests regarding the publication of this paper.

ACKNOWLEDGMENT

This work is supported by the SC&SS, Jawaharlal Nehru University, New Delhi.

References

1. Yin, C., Xiong, Z., Chen, H., Wang, J., Cooper, D., & David, B. (2015). A literature survey on smart cities. *Science China Information Sciences*, *58*(10), 1-18.
2. Online Available: https://www.statista.com/statistics/802690/worldwide-connected-devices-by-access-technology
3. Cruz-Piris, L., Rivera, D., Marsa-Maestre, I., De La Hoz, E., & Velasco, J. R. (2018). Access control mechanism for IoT environments based on modelling communication procedures as resources. *Sensors*, *18*(3), 917.
4. Yu, Y., Li, Y., Tian, J., & Liu, J. (2018). Blockchain-based solutions to security and privacy issues in the internet of things. *IEEE Wireless Communications*, *25*(6), 12-18.
5. Fernández-Caramés, T. M., & Fraga-Lamas, P. (2018). A Review on the Use of Blockchain for the Internet of Things. *IEEE Access*, *6*, 32979-33001.
6. Wright, C. S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. Available at SSRN 3440802.
7. Zhou, E., Sun, H., Pi, B., Sun, J., Yamashita, K., & Nomura, Y. (2019, October). Ledgerdata Refiner: A Powerful Ledger Data Query Platform for Hyperledger Fabric. In *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)* (pp. 433-440). IEEE
8. Szabo, N. (1996). Smart contracts: building blocks for digital markets. *EXTROPY: The Journal of Transhumanist Thought,(16)*, *18*, 2.
9. Gilcrest, J., & Carvalho, A. (2018, December). Smart Contracts: Legal Considerations. In *2018 IEEE International Conference on Big Data (Big Data)* (pp. 3277-3281). IEEE.
10. Online Available: Hyperledger – Open Source Blockchain Technologies. https://www.hyperledger.org/
11. Mingxiao, D., Xiaofeng, M., Zhe, Z., Xiangwei, W., & Qijun, C. (2017, October). A review on consensus algorithm of blockchain. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (pp. 2567-2572). IEEE.
12. Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., ... & Yellick, J. (2018, April). Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the thirteenth EuroSys conference* (pp. 1-15).
13. Zhang, Y., Xu, X., Liu, A., Lu, Q., Xu, L., & Tao, F. (2019). Blockchain-based trust mechanism for iot-based smart manufacturing system. *IEEE Transactions on Computational Social Systems*, *6*(6), 1386-1394
14. Wan, J., Li, J., Imran, M., & Li, D. (2019). A blockchain-based solution for enhancing security and privacy in smart factory. *IEEE Transactions on Industrial Informatics*.
15. Iftikhar, M. Z., Iftikhar, M. S., Jawad, M., Chand, A., Khan, Z., Khan, A. B. M., & Javaid, N. (2019, November). Efficient Resource Utilization Using Blockchain Network for IoT Devices in Smart City. In International Conference on Broadband and Wireless Computing, Communication and Applications (pp. 521-534). Springer, Cham.
16. Sun, J., Yan, J., & Zhang, K. Z. (2016). Blockchain-based sharing services: What blockchain technology can contribute to smart cities. *Financial Innovation*, *2*(1), 26.
17. Rahman, M. A., Rashid, M. M., Hossain, M. S., Hassanain, E., Alhamid, M. F., & Guizani, M. (2019). Blockchain and IoT-based cognitive edge framework for sharing economy services in a smart city. IEEE Access, 7, 18611-18621.
18. Moinet, A., Darties, B., & Baril, J. L. (2017). Blockchain based trust & authentication for decentralized sensor networks. *arXiv preprint arXiv:1706.01730*.
19. Tang, Fei, et al. "An efficient authentication scheme for blockchain-based electronic health records." *IEEE access* 7 (2019): 41678-41689.

20. Dwivedi, A. D., Srivastava, G., Dhar, S., & Singh, R. (2019). A decentralized privacy-preserving healthcare blockchain for IoT. *Sensors*, *19*(2), 326.
21. Wang, Y., Zhang, A., Zhang, P., & Wang, H. (2019). Cloud-Assisted EHR Sharing With Security and Privacy Preservation via Consortium Blockchain. *IEEE Access*, *7*, 136704-136719.
22. Novo, O. (2018). Blockchain meets IoT: An architecture for scalable access management in IoT. *IEEE Internet of Things Journal*, *5*(2), 1184-1195.
23. Aung, Y. N., & Tantidham, T. (2019, February). Ethereum-based Emergency Service for Smart Home System: Smart Contract Implementation. In *2019 21st International Conference on Advanced Communication Technology (ICACT)* (pp. 147-152). IEEE.
24. Xue, J., Xu, C., & Zhang, Y. (2018). Private Blockchain-Based Secure Access Control for Smart Home Systems. *KSII Transactions on Internet & Information Systems*, *12*(12).
25. Lin, C., He, D., Kumar, N., Huang, X., Vijaykumar, P., & Choo, K. K. R. (2019). HomeChain: A Blockchain-Based Secure Mutual Authentication System for Smart Homes. *IEEE Internet of Things Journal*
26. Wu, H. T., & Tsai, C. W. (2019). An intelligent agriculture network security system based on private blockchains. *Journal of Communications and Networks*, *21*(5), 503-508.
27. Gong, S., Tcydenova, E., Jo, J., Lee, Y., & Park, J. H. (2019). Blockchain-based secure device management framework for an internet of things network in a smart city. *Sustainability*, *11*(14), 3889.
28. Pieroni, A., Scarpato, N., Di Nunzio, L., Fallucchi, F., & Raso, M. (2018). Smarter city: smart energy grid based on blockchain technology. *Int. J. Adv. Sci. Eng. Inf. Technol*, *8*(1), 298-306.
29. Memon, R. A., Li, J., Ahmed, J., Khan, A., Nazir, M. I., & Mangrio, M. I. (2018, December). Modeling of blockchain based systems using queuing theory simulation. In *2018 15th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)* (pp. 107-111). IEEE.
30. Casado-Vara, R., Chamoso, P., De la Prieta, F., Prieto, J., & Corchado, J. M. (2019). Non-linear adaptive closed-loop control system for improved efficiency in IoT-blockchain management. *Information Fusion*, *49*, 227-239.
31. Yankovic, N., & Green, L. V. (2011). Identifying good nursing levels: A queuing approach. *Operations research*, *59*(4), 942-955.
32. Galvao, R. D., & Morabito, R. (2008). Emergency service systems: The use of the hypercube queueing model in the solution of probabilistic location problems. *International Transactions in Operational Research*, *15*(5), 525-549.
33. Online Available, Javaria Tahir, Smart Ambulance for Emergency Patient: https://www.researchgate.net/profile/Nadeem_Javaid/publication/334696607_Smart_Ambulance_Blockchain_technology_in_health-care_for_emergency_patients/links/5d3aa87a299bf1995b4b0ed4/Smart-Ambulance-Blockchain-technology-in-health-care-for-emergency-patients.
34. Gosavi, A. (2020). Tutorial for Use of Basic Queueing Formulas. *Department of Engineering Management and Systems Engineering, Missouri University of Science and Technology*.
35. Online Available, Jayeshkumar J. Patel (2012), Queuing model simulation for Big Bazar: http://psrcentre.org/images/extraimages/26%20812007.pdf
36. Yang, Z., Yang, K., Lei, L., Zheng, K., & Leung, V. C. (2018). Blockchain-based decentralized trust management in vehicular networks. *IEEE Internet of Things Journal*, *6*(2), 1495-1505.