Digital Communications and Networks(DCN)

HOSTED BY

ELSEVIER

Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/dcan

Blockchain-based DDoS attack mitigation protocol for device-to-device interaction in smart home

Bello Musa Yakubu[a], Majid Iqbal Khan[b], Abid Khan[c], Farhana Jabeen[b], Gwanggil Jeon*[d]

[a]*Department of Mathematics and Computer Science, Chulalongkorn University,Bangkok 10330, Thailand*
[b]*Department of Computer Science, COMSATS University Islamabad (CUI),Islamabad 45550, Pakistan*
[c]*College of Science and Engineering, University of Derby, Derby DE22 1GB, United Kingdom*
[d]*Department of Embedded Systems Engineering, Incheon National University, Incheon 22012, Korea*

Abstract

Smart home devices are vulnerable to a variety of attacks. The matter gets more complicated when a number of devices collaborate to launch a colluding attack (e.g. Distributed-Denial-of-Service (DDoS)) in a network (e.g., Smart home). To handle these attacks, most studies have hitherto proposed authentication protocols that cannot necessarily be implemented in devices, especially during Device-to-Device (D2D) interactions. Tapping into the potential of Ethereum blockchain and smart contracts, this work proposes a lightweight authentication mechanism that enables safe D2D interactions in a smart home. The Ethereum blockchain enables the implementation of a decentralized prototype as well as a peer-to-peer distributed ledger system. The work also uses a single server queuing system model and the authentication mechanism to curtail DDoS attacks by controlling the number of service requests in the system. The simulation was conducted twenty times, each with varying number of devices chosen at random (ranging from 1 to 30). Each requester device sends an arbitrary request with a unique resource requirement at a time. This is done to measure the system's consistency across a variety of device capabilities. The experimental results show that the proposed protocol not only prevents colluding attacks, but also outperforms the benchmark protocols in terms of computational cost, message processing, and response times.

KEYWORDS: DDoS attack, Device-to-device, Smart homes, Blockchain, Authentication, Ethereum smart contract

1. Introduction

Devices in a smart home-based networks are vulnerable to a variety of attacks, many of which can be highly malicious [1, 2, 3]. These attacks are divided into two types, "external node-based attack" where an adversary node that is not the legitimate member of the network tries to gain an unauthorized access to the network and "internal node-based attacks" where the adversary node is a registered member of a network, and thus can access network operations and has cryptographic credentials [2, 4, 5]. A more advance form of such attacks are colluding attacks, where multiple nodes collaborate to disrupt a network service or target a particular node in a network [1, 6, 7]. The devices that launch colluding attacks are referred to as colluding nodes. An example of a colluding attack is the DDoS attack [1, 8, 9]. DDoS attacks are more apparent in smart home-based networks [10], where nodes are deemed legitimate members by default [1, 11, 12, 13, 14, 15].

Most of the existing schemes addressing DDoS attacks are based on authentication mechanisms that are effective against a DDoS attack launched by the external adversaries. However, they do not provide adequate protection against the DDoS attack from internal adversaries [2, 3]. Once a node is authenticated and registered as a trustworthy node in a network (becomes an internal node), the authentication mechanism does not check the node activities during D2D interactions. Consequently, multiple internal adversary nodes may collaborate to launch a DDoS attack in a given smart home network [3]. Moreover, most of the schemes rarely consider that smart home devices are used daily [16]. This means that these schemes require too many resources to compute operations carried out by smart home devices [1, 2, 3].

*Gwanggil Jeon (Corresponding author)
Email:gjeon@inu.ac.kr

1.1. Motivation of the study

Many schemes such as [17] and [18] introduced authentication and access control mechanisms to achieve secure and better D2D interactions. In [17], a secure session key and Diffie-Hellman (DH) key exchange mechanisms were used to prevent DDoS attacks originating from external adversaries. Their method enables exchange of a public key between the interacting devices with the help of a trusted service provider. The public key is used to provide mutual authentication between the devices using a short authentication token and a secure session key during each device interaction. Other techniques involve the use of an anonymous client verified session key negotiation framework and a three-factor verification protocol to prevent potential external threats [18]. The technique leverages the Elliptic Curve Diffie-Hellman key exchange (ECDH) scheme that allows public key exchange between the devices in the smart home.

With respect to the use of the above mentioned techniques, the following issues have been identified:

- Both the DH and ECDH techniques do not offer session key protection. Therefore, they are prone to secret key leakage attacks. Adversaries could exploit this security breach to intentionally disrupt the operations and behavior of the affected device. With the possession of the secret key, the adversaries can send a number of malicious requests (packets) to the victim device continuously, namely, initiate a DDoS attack.
- Due to the use of bilinear pairing, the approach described by [18] is computationally expensive. Similarly, the technique presented by [17] may sustain impersonation attacks because the key generation center can track the device identity, and it is also prone to central point of failure.
- Cryptographic and public key exchange processes increase the length of the message by a factor of two. This means the ciphertext is twice as long as the plaintext. The computational complexity of the three basic cryptographic families, namely, of those based on integer factorization, (e.g. Rivest–Shamir–Adleman (RSA)), discrete logarithm, (e.g. DH), and elliptic curve, (e.g. ECDSA) increases steadily as the cubic bit length increases [19]. Therefore, the application of such cryptographic techniques on modern personal computers (PCs) does not cause any harm since their execution time ranges from 10 to a few 100 msec. However, their use in smart home devices may not be feasible.

To tackle the above mentioned limitations, we propose a semi-centralized scheme using blockchain and smart contracts to provide reliable and secure D2D interactions in a smart home network. In addition, our approach leverages a single server queuing system and a D2D authentication mechanism to curtail DDoS attacks by considering finite number of service requests in the network.

1.2. Research contributions

- We introduce DDoS attack mitigation protocol (referred here as D2D protocol) in smart homes by leveraging a single server queuing system and a D2D authentication mechanism.
- We present a security analysis to assess the credibility of the proposed DDoS attack mitigation protocol.
- Through usability assessments on the Ethereum client using testnets and solidity Integrated Development Environment (IDE), we evaluate the efficiency of the proposed D2D protocol on a smart home scenario in terms of computational costs, message processing latency, and message response time.
- To verify the efficacy and security of our protocol, we carry out a vulnerability analysis test on our smart contract. The results of the analysis show that our smart contract is secure and free of all established bugs of smart contract vulnerabilities such as reentrance vulnerability, timestamp dependency, transaction ordering dependency, parity multisig bug, and assertion failure.

2. Background and related work

This section presents a concise summary of the background and context, namely, the issues that the study addressees, and the basic concept of the study. Moreover, it explores the relevance and limits of the most recent studies in the field by examining their discussions.

2.1. Basic concepts

The Ethereum blockchain, which was founded by Vitalik Buterin in 2005, represented a fresh perspective on already established Bitcoin platforms [20]. The Ethereum platform facilitates financial transactions, and can also be used for non-financial purposes, such as Internet-of-a-Thing (IoT) security, smart home security, and privacy safeguards [21]. Ether is the cryptocurrency used in Ethereum. Smart contracts (self-executing programs) are the fundamental building blocks of Ethereum-based applications. Bitcoin does not support smart contracts [22].

These contracts are basic programs recorded on the blockchain that are used to initiate asset exchange, like coins, in response to particular situations. Smart contracts are programs that run on their own when certain circumstances are met. Smart contract execution is proportional to the quantity of gas utilized [23].

Table 1: Summary of some existing approaches

Techniques	Objectives	Achievements	Limitations
One-way hash function [17].	Anonymous user authentication and session key agreement	Secure under BAN-logic and AVISPA.	Timestamp might involve the challenges in clock synchronization.
Diffie-Hellman (DH) key exchange [18].	Mutual authentication and session key agreement	Secure under AVISPA.	Impersonation attacks, central point of failure and secret leakage attacks.
Physically Unclonable Functions [29]	Authentication and privacy preservation	Resist DoS and cloning attacks.	Inferential attack.
One-way hash function. physically unclonable function (PUF) and bitwise exclusive-OR (XOR) operations [31]	Anonymous user authentication.	Resist various attacks under Random Oracle model.	Impersonation attacks.
Diffie-Hellman protocol [32].	Mutual authentication, key exchange, forward secrecy, and privacy protection.	Secure under BAN-logic and AVISPA.	Inferential attack.

Gas is an element of the Ethereum blockchain token that the contract uses to compensate miners for confirming transactions. Gas is required in all Ethereum blockchains to calculate and execute smart contracts and transactions. Enough gas must be provided to miners to guarantee the transactions run smoothly; otherwise, the transactions will be cancelled or fail to process [24].

Nowadays, computers can calculate hundreds of thousands of hashes per second. Relying solely on hashes does not help preventing tampering, since it is easy to tamper with a block and recalculate all the hashes of subsequent blocks to restore the blockchain's validity. To mitigate this, the blocks on a blockchain must reach an agreement with each other before any changes to the given block are made. This is referred to as the consensus algorithm. There are many different consensus algorithms, but in the context of this article, we will focus on the Proof-of-Authority (PoA) algorithm. This consensus method selects a participating node based on its identification, which must be publicly visible to all other network members. The selected one is granted authority (power) to check and approve each transaction added to the block. When a transaction occurs, this validator gets compensated or receives incentives [25].

New advances in blockchain technology provide feasible and effective methods for achieving secure node-to-node connections in a smart home [26]. Blockchain technology builds trust between nodes by ensuring the transparency and immutability of transactions inside the smart environment. Owing to its trustworthiness, security, traceability, and immutability, blockchain technology can be utilized successfully to handle secure device-to-device communications in a smart home [27], [28].

2.2. *Literature review*

Many studies have addressed issues related to users access and authentication during information sharing in networks. Few studies have focused on secure, reliable and effective D2D interactions in smart home networks [26]. However, it is essential to decrease the danger of threats and vulnerabilities such as DDoS attacks in these networks, especially in applications like home security monitoring systems [17, 29].

A typical method for protecting smart home devices from colluding adversaries is to remove these nodes from the network after malfunctioning or detection of malicious activity [30]. Various approaches presented in studies [17, 18, 31, 32] uses cryptographically signed acknowledgment, mutual authentications, and other intermediary nodes for monitoring defective and illegitimate nodes. However, the approaches are particularly vulnerable to impersonation, inferential, and secret key leakage attacks [30]. The transparency, traceability, and accountability features in blockchain technology [33, 34, 35] enable the sharing of requests and services. This sharing eliminates the need of a trusted entity or a shared secret key and helps in detecting defected nodes effectively [36, 37, 38].

Another important issue facing smart home devices is the issue of how to achieve trust [39, 40]. This can be solved by developing a technique that enables participants to trust a shared record of transaction or event, even though they have no knowledge or trust of each other [41]. Approaches such as [18, 32] uses the Diffie-Hellman key exchange mechanism to achieve trust between the participating devices. Both server and requester devices mutually authenticate each other. While approaches like [17, 29, 31] employ one-way functions and physically unclonable functions to achieve trust among the participants. However, both approaches are affected by impersonation, inferential and secret key leakage attacks [3] blockchain technology can largely help in addressing these issues. Blockchain eliminates fault and errors while also helping detecting fraudulent activities associated with shared records within the entire network [33, 42]. Similarly, the technology can also handle many commonly encountered issues (e.g.

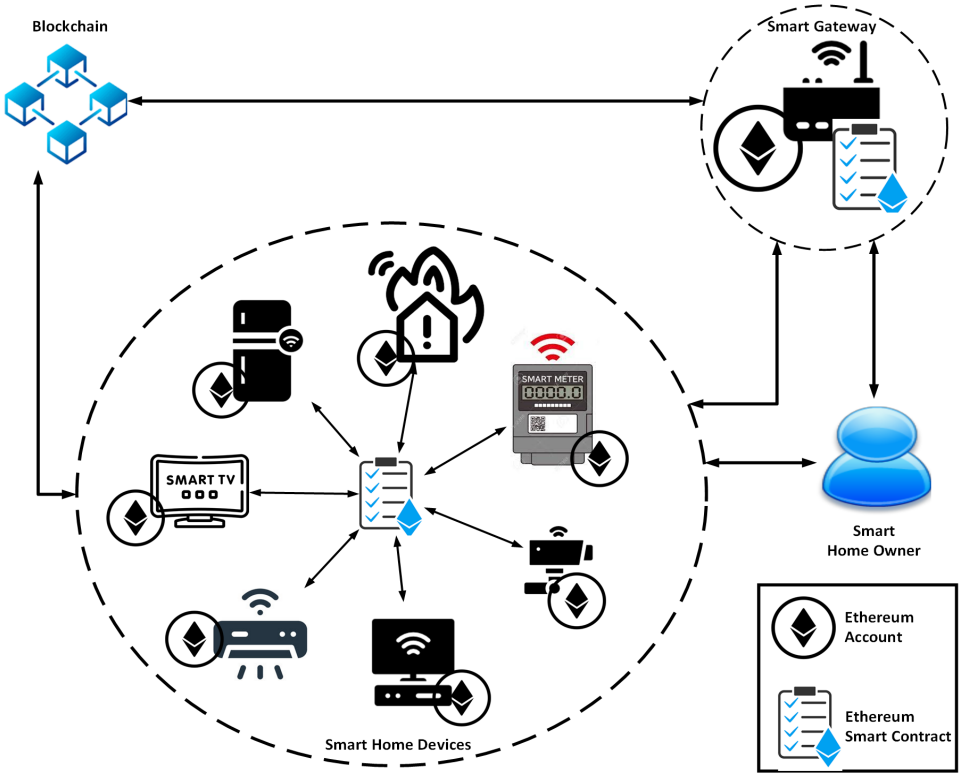

Figure 1: Smart home network model.

DDoS attacks) associated with smart home devices [37, 41, 43].

Unique identity management and secure verification of each participant are other factors that need to be addressed in smart homes [33]. Blockchain technology, solutions can be provided for secured identity management, such as validation of device profiles, user profiles, and other public profile, and digital signatures [33, 44]. Thus, blockchain smart contracts can help automating transaction or communication agreements, enabling them to be automatically executed when certain conditions are met [45, 46]. Table 1 summarizes the most studies that address the issues and solutions proposed in this paper.

3. System model

This section discusses our system models, which comprises a network model and a data model. The section also presents attacker model and the security requirements.

3.1. Network model

The model is based on a semi-centralized private network running on Ethereum blockchain with PoA consensus mechanism as depicted in Figure 1. The Ethereum blockchain allows users to build and operate smart contracts and distribute autonomous applications without third-party censorship [7, 45]. The PoA is semi-centralized consensus algorithm that can run under a distributed blockchain ledger such as Ethereum [7, 26, 48]. From Figure 1, all participating devices use separate Ethereum accounts to participate in the smart home via a smart contract. Each Ethereum account has an associated Ethereum Address (EA) by default as per Ethereum blockchain architecture [20]. Therefore, they all have a direct connection to the blockchain where individual transaction and communications are recorded and easily retrieved. Similarly, we presume that the devices continuously perform real-time services to the inhabitant of the building. From Figure 1, the services include electricity meter reading, security surveillance, entertainment, fire alarm and other smart services. The fundamental roles of the main constituents of the system are described as follows:

- Home Gateway: It is a high speed device with high computing power. All smart home devices are authenticated and mapped to the home gateway. In this work, the gateway is also considered as the validator that runs on PoA.
- Smart contract: This is code that will be executed when a device calls for a function (e.g. request for service). It is used as the key validator if conditions are fulfilled, while also allowing authorized devices to interact with the blockchain network.
- Device(s): They can call for a function (e.g. request for service). They may contain resources that can be requested by other devices during interaction and may communicate with any device

in the network only if they are valid and trusted devices.

3.2. Data model

We assumed that the devices in a network generate two types of data: periodic and sporadic [49]. Each device generates its own periodic data such as reporting temperature variations. Sporadic data is generated when a certain incident occurs, for example, a security breach alert by the smart alarm. The smart home devices can relay the sensed data to the blockchain periodically or during an event. Furthermore, the smart home gateway can aggregate the collected data to externally communicate with the owner of the smart home as depicted in Figure 1.

3.3. Adversary model and assumptions

The goal of the adversaries is to overwhelm and overload any targeted device with a flood of network traffic, creating outages. Given this, the following assumptions are made:

- The colluding nodes (compromised devices) can collectively transmit unlimited amounts of data to any targeted device in the smart home including the home gateway.
- The devices including the gateway are not constrained in terms of space, memory, and processing power. The devices are also protected from spoofing threats.
- Both the home gateway and the blockchain cannot be compromised.

Importantly, this study does not address the issue of what happens if the gateway fails or is not trusted. However, home gateways are susceptible to DDoS attacks just like any other network device, therefore the proposed protocol protects both home devices and the gateway.

3.4. Security requirements

The proposed scheme aims to satisfy the following security requirements in order to defend devices against colluding attacks such as DDoS:

- Validate the legitimacy of the devices before they join the network and the two devices exchange messages.
- Limit the number of concurrent requests arriving at a node to prevent a possible DDoS attack

4. DDoS attack mitigation protocol for D2D interaction (D2D protocol)

This section proposes the DDoS attack mitigation protocol for D2D communication (D2D protocol) between devices in smart home. The model serves as a proactive measure against DDoS attacks. We use it to achieve the following: 1) address attacks in external nodes by presenting a registration and authentication protocol and, 2) address attacks in internal nodes by presenting a protocol to restrict the request arrival rate during D2D interactions in the smart home.

4.1. Registration and authentication protocol

In this section, we discuss device registration in a smart home and device authentication during D2D interactions.

4.1.1. Registration

The process of registration is illustrated in lines 1-3 of Algorithm 1. The smart home owner initially establishes the smart contract and registers all devices credentials, such as the EA of the devices' Ethereum accounts and connects (maps) them to the gateway of the smart home, which also uses a different EA. All registered device credentials are stored in the blockchain ledger within the private network.

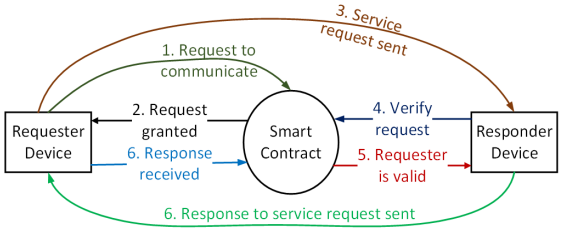

Figure 2: D2D communication and interaction.

4.1.2. D2D authentication

As depicted in Figure 2, a background check requires to be carried out on the requester device to communicate with a valid device (responder) within the network. For example, the requester device must first send a request to communicate using the function *Request2Communicate* in the algorithm, specifying its EA, the corresponding responder device EA, and that of the smart gateway. The smart contract will search for the requester device in the list of the registered devices in the blockchain ledger. If found, then the smart contract will check whether the device is mapped to the home gateway as depicted in line 8 of Algorithm 1. If the mapping is true, then the smart contract deems that the device exist and that it is valid (line 8 to 11, Algorithm 1), otherwise it considers it a false and untrusted device (line 14, Algorithm 1). The smart contract considers that valid existing devices are authenticated (line 16-18, Algorithm 1) and then triggers an acceptance event with an interaction token $Token_{Device}$

Algorithm 1: Registration and authentication of devices in a smart Home

Input: device, homeGateway
Output: Token, Connection

```
1  //Register and map a device in a homeGateway
2  homeGateway ← device_i;
3  Return : Device has been added in
     homeGateway Mapping!
4  // Check if device exists in homeGateway
     mapping
5  Function
     Request2Communicate(New-Request):
6      for DeviceExists = True do
7          // check for a particular device
8          if ((homeGateway ← device_i) = True)
             then
9              // device is mapped to a smart home
10             homeGateway ← device_i;
11             Return : deviceExists = True!
12             break;
13         else
14             Return : deviceExists = False!
15         end
16         if deviceExists = True then
17             deviceAuthentication = True;
18             Return : Device Authenticated!
19         else
20             deviceAuthentication = False;
21             Return : Device not Authenticated!
22         end
23     end
24     while deviceAuthentication = True do
25         //Generate Token
26         Return : Token Created!
27         //Establish Open SSL connections
28         Return : Open SSL Connections
             established!
29     end
```

(line 23-28, Algorithm 1) provided by equation (1). Otherwise, a denial message is issued, and no connection is established (line 20-21, Algorithm 1).

$$Token_{Device} = (UID, Timestamp, \Delta, Device_{responder}, Device_{requester}, SmartGW) \quad (1)$$

The information in the token includes the following: Unique Identification (UID), access time duration (Δ), responder device EA, requester device EA, smart gateway EA (SmartGW) and block timestamps (timestamp). The UID is created by hashing the smart gateway EA, and block timestamps are created using the keccak256 cryptographic hash function.

$$Signed_{Device}(Token_{Device}, message_{packet}) \quad (2)$$

After acquiring the acceptance token from the smart contract, the requester device will include the token in the service request message (signed packet-message) and send it to the responder device in order to request connection permission, as illustrated in Equation (2). The responder device uses the token to verify the requester device validity through the smart contract and then respond to the request accordingly (granting connection permission or otherwise). Once permission is granted, the connection time is defined as the duration of one complete device-to-device interaction (i.e. request and response). A regular, secured SSL link for services, resource, and information exchanges is established between the requesting device and the targeted device.

4.2. *Protocol for addressing randomly continuous malicious requests arrivals*

Consider the case when some authenticated devices collude (not necessarily deliberate), sending continuous random requests (packets) to an arbitrary device (responder device) in the smart home as depicted in Figure 3. The requests are randomly sent to the responder device (like a Poisson distribution), whether deliberately or accidentally. As a result, the responder device resources become overwhelmed and eventually cannot render services. To address this problem, we present a protocol that depends on a single server queuing system (M/M/1) [47], where all packets arrival follow a Poisson distribution, with a mean inter-arrival time given as $\bar{r}$. The responder device service time is assumed to have an exponential distribution, with an average service time given as $\bar{s}$. All arrived requests are served based on a First-In, First-Out basis (FIFO). The order of request arrival and response is the same.

4.2.1. *The protocol*

In this protocol, the process comprises of a finite number of request arrival queue capacity and a threshold proportional to the responder device computing capability. The threshold parameter is determined from the optimal resource committed by of the responder device during the process, this is referred here as optimum resource commitment. Likewise, the service nature is non-preemptive. Hence, once the process has been initiated, requests will be serviced until completion. The service is ultimately conservative; therefore, the server will not be idle if there is at least one request in the queue.

Thus, in a given i-number of requests (i-request),

1. The request arrival time is given as: a_i
2. The delay in the queue is given as: d_i
3. The time that the service process begins is given as: $b_i = a_i + d_i$
4. The service processing time is given as: s_i
5. The wait in the device is given as: $w_i = d_i + s_i$

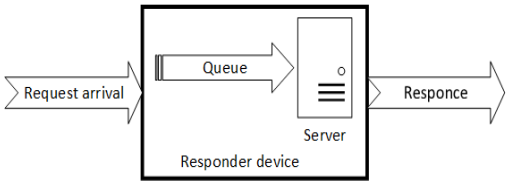

Figure 3: Responder device structure.

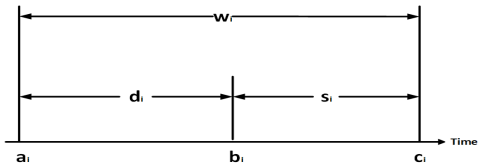

Figure 4: Request processing and modelling.

6. The response time is given as: $c_i = a_i + w_i$

From the request arrival end, the interarrival time between requests $i-1$ and i is given as follows:

$$r_i = a_i - a_{i-1} \tag{3}$$

whereby definition, $a_0 = 0$ Therefore, $a_i = a_{i-1} + r_i$
By induction,

$$a_i = r_1 + r_2 + ... + r_i \tag{4}$$

where $i = 1, 2, 3, ..., n$

The delay time (d_i) is an important factor during request and service sharing and can be computed using the FIFO approach, i.e., the d_i can be determined when a_i occurs relative to c_{i-1}. For this argument, we can consider two cases:

1. If $a_i < c_{i-1}$, request i arrives before request $i-1$ completes.

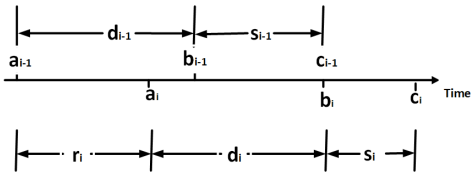

Figure 5: Request i arrives before request $i-1$ completes.

2. If $a_i \geq c_{i-1}$, request i arrives after request $i-1$ completes.

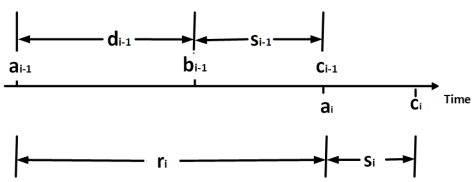

Figure 6: Request i arrives after request $i-1$ completes.

From the above cases, we can deduce that the service processing time s_i depends on n, a_i, and r_i. where n is the number of requests received periodically. Thus, during request and service sharing in the protocol, the request-service average statistics can be arithmetic computed:

1. The average inter-arrival time of requests can be computed as follows:

$$\bar{r} = \frac{1}{n}\sum_{i=1}^{n} r_i = \frac{a_n}{n} \tag{5}$$

and $\frac{1}{\bar{r}}$ is the arrival rate.

2. Average service time (average request processing time) can be computed as follows:

$$\bar{s} = \frac{1}{n}\sum_{i-1}^{n} s_i \tag{6}$$

and $\frac{1}{\bar{s}}$ is the service rate (request processing rate).

To address the continuous request (not necessarily malicious) arrivals during D2D communications and service sharing in a smart home network, the relation $\bar{r} < \bar{s}$ should hold true. The protocol process is explained in Algorithm 2. From line 1 of Algorithm 2, the protocol is designed to ensure that the $\bar{r} < \bar{s}$ is always true by; 1) following a Poisson distribution during request arrivals, 2) serving the request based on the FIFO approach, and 3) using a non-preemptive service process.

If the average resource requirement *rho* available for processing the queue population at a time t equals the responder device's optimum resource commitment (threshold (%)), the system triggers a locking event to prevent buffer overflow. This means that, the resource requirement to process a queue population including the newly arrived request at a given time frame, must be lower than the given threshold (%) of the device optimum resource commitment before the newly arrived request can be processed.

Given a continuous set of requests $i_1, i_2, i_3, ..., i_n$ arriving at a given time t and being processed in an average service time $\bar{s}$, the resource requirement at a given time ρ_t is defined as the average maximum computing power required by a responder device to process requests in a given queue population at a given time t. If n_t is the queue population that do not include the newly arrived request at time t, then the resource requirement of the responder device when the newly arrived request is considered can be computed as follows:

$$\begin{aligned}\rho_t &= \frac{1}{2}.\frac{\partial}{\partial n}\left(n_{t+1}^2 + \frac{\bar{s}(n_{t+1}-n_t)(n_{t+1}+n_t)}{100}\right)\\ &= \frac{1}{2}.\frac{\partial}{\partial n}\left(n_{t+1}^2 + \frac{\bar{s}(n_{t+1}^2-n_t^2)}{100}\right)\\ &= \frac{1}{2}\left(2n_{t+1} + \frac{\bar{s}(2n_{t+1}-2n_t)}{100}\right)\\ &= n_{t+1} + \frac{\bar{s}(n_{t+1}-n_t)}{100}\end{aligned} \tag{7}$$

where n is the queue population at a time t. With this no further request will be accepted until the current requests are served and the ρ_t becomes lower than the processing power ($threshold(\%)$ of the device).

Algorithm 2: Protocol for addressing continuous malicious requests arrivals

Input: $n, \{a_1, a_2, ..., a_i\}, \{s_1, s_2, ..., s_i\}, \rho$
Output: $\{c_1, c_2, ..., c_i\}$

```
1  while ρ_t < threshold(%) and r̄ < s̄ and i ! = 0
   do
2      if (min(a_i, CT, R(i).s_i) = a_i) then
3          // Update Arrival();
4          //next event is a service for arrival a_i
5          i − −;
6      end
7      if min(a_i, CT, R(i).s_i) = CT then
8          //next event represents the end of a
             service
9          // Update finish_service();
10         //next event is a return
11     else
12         Update Retrial();
13     end
14 end
```

From Algorithm 2, CT is the service termination time of a request i. It is also defined as a server clock. When there is no request to serve, $CT = \infty$. The vector; $R(i)$, with $(i = 1, ..., n)$, corresponds to the i^{th} request in the queue and comprises three fields: the number of remaining returns R_i, the next return time V_i and the request service time s_i as given in equation (8). The algorithm works based on the 'next event' rule or 'minimum time' rule [50]. This means that the requests in the queue are sorted and served based on the FIFO approach and the time for the next return as given in equation (9) and elaborated in lines 2-5 and 7-10 of Algorithm 2.

$$R(i) = (R_i, V_i, s_i) \tag{8}$$

Thus,

$$R(i).V_i < R(i+1).V_i, with(i = 1, ..., n-1) \tag{9}$$

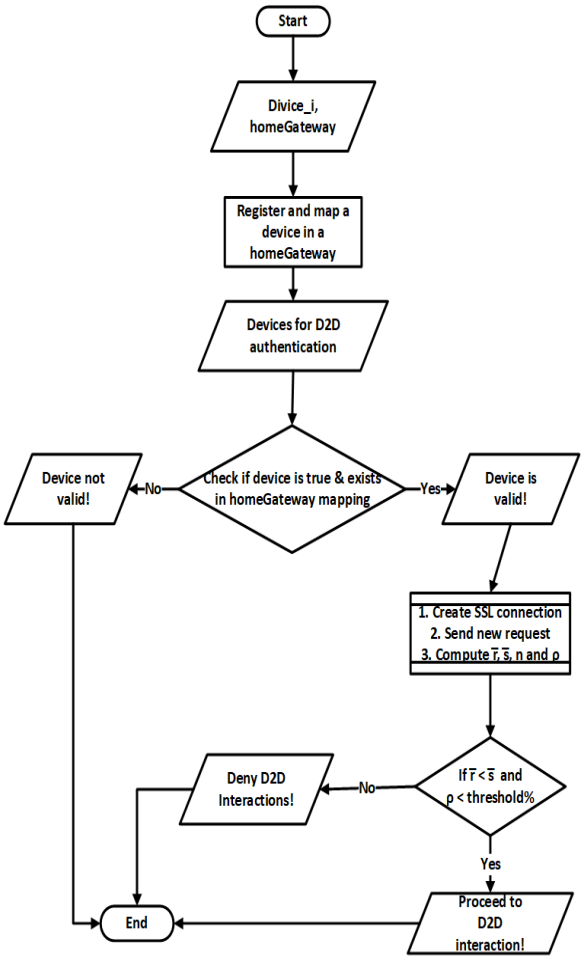

Figure 7: Overall working principle of the proposed model.

In summary, the overall working principle of the proposed model is described in Figure 7. It details the flow of the whole model operation, from registration through DDoS mitigation methods, which, when correctly implemented, enable safe D2D interactions in the system.

5. Security analysis

In this section, we provide a security-based formal analysis for the security requirements highlighted earlier, while also considering the resource-constraint nature of the IoT devices.

5.1. Authentication

The following theorems and their proofs provides the security analysis for authentication requirements of the model.

Theorem 5.1. *When a smart home device interacts or communicates with another device in a smart home, then it has done it with a legitimate device.*

Proof. The framework is protected from replay and Man-In-The-Middle (MITM) attacks, since every

message communication within the framework is digitally signed off in the blockchain network. In addition, the use of UIDs and block timestamps in the verifications prevents MITM and replay threats. Even when the intruder replaces a device EA with another EA and a public key, they cannot sign it accurately. Additionally, the events produced are tamper-resistant and are verified by a smart contract, which protect them against MITM and replay threats. □

Theorem 5.2. *In the proposed protocol, the communication between smart home devices is confidential and the provenance of the data is authentic and secure. Thus, an intruder cannot access the IoT devices nor their data.*

Proof. The Ethereum blockchain-based protocol uses a special 20-byte EA that can be allocated instantaneously to every device in the network with almost no collision. This strong blockchain feature mitigates the need for costly key distribution mechanisms such as Public Key Infrastructure (PKI). Ethereum comes with asymmetric pairs of keys in each EA, it can also be used to create a Secure Socket Layer (SSL) session for communication between all authenticated IoT devices. This technique enables achieving high confidentiality. Using the secure SSL session to encrypt and decrypt messages after successful device authentication, the technique impedes illegal access to the device and its associated data. Similarly, the adversary may attempt to imitate a legitimate device and submit manipulated data to other network devices. This problem can be addressed using Theorem 5.1. □

5.2. Preventing continuous request arrival

The following theorem and its proof provides the security analysis for concurrent requests limitation requirements of the model.

Theorem 5.3. *The colluding adversaries cannot overwhelm a victim's resources by sending number of malicious requests continuously.*

Proof. Given a set of arrival times $a_1, a_2, a_3, ..., a_i$ of a malicious request, each spending at least d_i time in the queue before being processed. The protocol ensures that the request average interarrival time $\bar{r}$ is less than the average service time $\bar{s}$ (average request processing time), i.e. $\bar{r} < \bar{s}$. This is because of all the arrived requests follows a Poisson distribution and are served in a non-preemptive manner, which is also based on a FIFO basis. □

6. Experimental setup, analysis and results

In this section we present the experimental setup and simulation results and compare our technique's efficiency with that of the state-of-the-art techniques presented in [17] (S. Dey) and [18] (S. Banerjee et al). Solutions presented in both of these studies are based on blockchain where interactions are based on transactions. We have implemented these studies on the Ethereum blockchain, where all interactions are conducted as transactions, which can be quantified in terms of gas per unit transaction in Ethereum.

6.1. Simulation setup and performance metrics

The simulations were implemented using Python and Solidity with MetaMask Ethereum wallet and Chrome plugin to allow user-friendly connections with Ethereum networks. Furthermore, a PoA based on Rinkeby testnet was used to alternatively emulate the blockchains ledger and network for testing. The Truffle Framework was used to speed up testing of the smart contract and testnet switches.

In this model, each device must be linked to the smart home network through Ethereum accounts in the real world. We use Ethereum accounts to represent devices on the network and simulate real-world scenarios. To mimic the smart home devices, we established Ethereum accounts each having a predefined EA. Therefore, the devices are connected in the same way as various Ethereum accounts.

Each device in the smart home network is simulated using a separate EA. Thus, we quantify the computational cost that each device may incur by comparing it to the amount of gas used in each transaction, which is feasible using the MetaMask Chrome plugins. The gateway as a network device must also be validated by the owner to ensure its authenticity.

A separate device serving as a responder device was used as a test subject, while 30 other devices were registered to serve as request senders during the simulation. To eliminate bias and obtain the best observation and performance, our model (D2D protocol) and the base line models [17] and [18] were implemented in the same context.

In this model, implementation is quantified in terms of all framework testing, which clearly depends on a sequence of transactions. Notably, the concept of what could happen if a device runs out of fund is beyond the scope of this study; therefore, this can be regarded as one of our limitations. However, we have funded all accounts during system implementation, so we do not anticipate that devices will run out of funds during system operation.

The D2D protocol employs devices that support interrupt endpoints and transfers, which are reliable for transmitting small amount of data [51]. The transfer technique employs polling, which ensures that the responder device reviews data at regular intervals. In this case, the maximum packet size is determined by device speed. The smart gateway is considered a high-speed device with a maximum packet size of 1024 bytes in this model, whereas the other devices are high

and low-speed devices with maximum packet sizes of 64 bytes and 8 bytes, respectively.

The simulation was conducted twenty times each time using a random number of devices ranging from 1 to 30 at a time serving as request sender(s). Each requester device sends an arbitrary request with a unique resource requirement at a time. In addition, each requester device is programmed to transmit a separate packet size of request at a time. This helps better monitoring the point of buffer saturation and better observation of the models' behaviours as the queue population increases. The simulations were evaluated by considering an optimum resource commitment (threshold) of the responder device of 70%, 80%, and 90% . This is done to measure the system's consistency across a variety of device capabilities. In this work, we will concern ourselves with the disparity on the following performance metrics of the D2D protocol and that of [17] and [18]:

- Computational cost,
- Request turnaround time

6.2. *Smart contract vulnerability analysis*

To ensure that our smart contract is protected from bugs and security vulnerabilities, such as reentrancy vulnerabilities, timestamp dependency, transaction ordering dependency and other vulnerabilities [52, 53], we use the Oyente smart contract security analyzer tool [54]. Table 2 shows that all results were "False", indicating that our smart contract is safe and does not have security bugs from all the above-mentioned attacks with Ethereum Virtual Machine (EVM) code coverage of 63.90%.

Table 2: Smart contract vulnerability analysis report

VULNERABILITIES	**RESULTS**
EVM Code Coverage	63.90%
Parity Multisig Bug 2	FALSE
Callstack Depth Attack Vulnerability	FALSE
Transaction-Ordering Dependence (TOD)	FALSE
Timestamp Dependency	FALSE
Re-Entrancy Vulnerability	FALSE
Assertion Failure	FALSE

6.3. *Results and evaluations*

During simulation, the logic of the models was checked and several events were observed and recorded based on the defined performance metrics. From Figure 8, the saturation point was noticed in the D2D protocol after the 16th, 20th and 23rd request in each experiment. Therefore, the remaining subsequent requests from the other 14, 10 and 7 devices were dropped. This is attributed to the lock event that was triggered when the resource requirement to process the queue population, which included the newly arrived requests, was equal or greater than 7o%, 80% and 90% of the responder device capability (optimum resource commitment) respectively. For instance, Figure 8 shows that the average number of packets received by the responder device from each threshold and beyond were dropped. This indicates that the average number of packets received are beyond the maximum processing capacity of the responder device at the given time frame; therefore, they are not processed. Considering the average allowable number of requests at a time in the experiment (i.e. 20 requests), the computational cost and request turnaround time (queue latency, processing delay and response time) of the responder device are observed and recorded.

6.3.1. *Computational cost:*

Taking advantage of the proposed protocol to perform authentication and prevent DDoS attacks will have some cost from both the IoT devices and the network. Hence, in this work, computational costs are derived from the average transaction expenses and the average execution expenses recorded on the smart contract during D2D interactions. The gas consumed when the smart contract is deployed and implemented in the blockchain is defined as a transaction expense. The execution expense refers to the amount of gas used to execute the smart contract.

Initially, we used a different number of IoT devices to study how the proposed protocol processes the recorded expenses on the IoT device and the network as compared to state-of-the-art techniques. The expenses were measured based on the number of requests received and handled by the responder device and the number of devices submitting these requests. The expenses in the systems increased as the number of IoT device request increased in the network. However, the results presented in Table 3 show that the average gas consumption for the D2D protocol (129380.8) is far lower than that presented in other studies [17] and [18], namely 138511.5 and 150800.4, respectively. Thus, as compared to the existing approaches, our proposed solution is more scalable and effective in terms of gas consumption.

6.3.2. *Request turnaround time:*

Considering that all the packets arrive to the responder device in a Poisson distribution order, the processing time has an exponential distribution, and the arrival, processing and service are based on a FIFO, this section presents the experiment results based on the overall turnaround time of all the received requests in the system. The turnaround time is the time required from the moment the request is queued for processing to the time the whole output is returned to the requester. The following observations are made from Figure 9:

Firstly, the request processing time in [17] and [18] approaches is relatively higher than that of our proposed protocol, which is in line with the fact that the

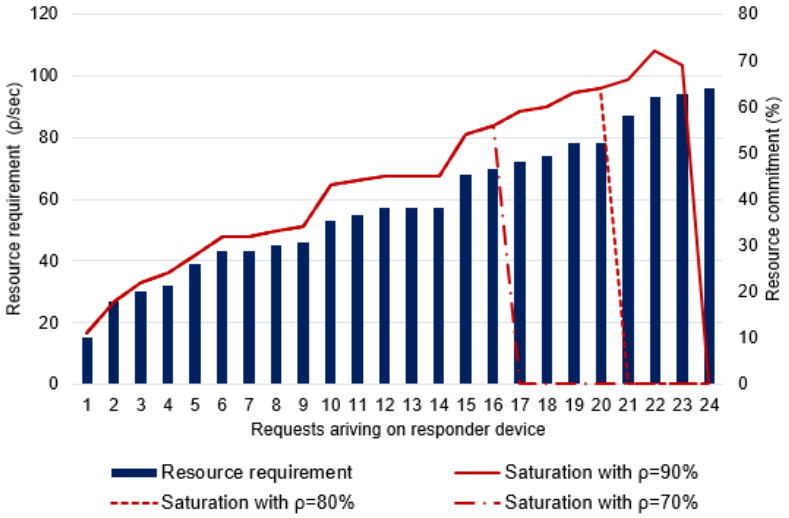

Figure 8: Colluding requests control with 70%, 80% and 90% thresholds

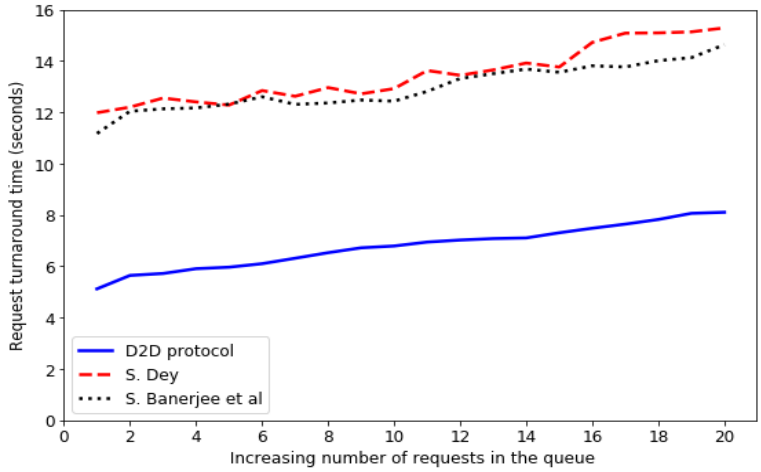

Figure 9: Request turnaround time.

maximum block mining time is less than 6 seconds in our protocol according to Etherscan, the Ethereum blockchain explorer, and testnet explorers. In comparison, it takes roughly up to 12 seconds to validate a message block both in [17] and [18].

Secondly, the [17] and [18] appears to be more unstable in terms of request processing delay per device compared to our protocol. This is because both techniques drive their instability by being reliant on a central server and D2D key agreement time variation. In our proposed protocol, the PoA chains are maintained by trusted parties and keys are broadcasted via the blockchain network. Thirdly, even if the unstable delay issue from [17] and [18] is disregarded, it is evident that the change in connectivity delay in our protocol is negligible and lower than the state-of-the-art models regardless of the number of connected devices. This result indicates that the average time delay for each processing is slightly increased as the total number of device requests increases.

As described in Figure 9, our proposed protocol demonstrates a rapid turnaround time to requests as compared to those presented in [17] and [18]. This is attributed to the low queue waiting time and low processing time of the system and its ability to filter out colluding attack requests. This is also due to the non-preemptive nature of the system and the FIFO order of request processing. The system forces all arriving requests to queue up and stop arriving requests after the threshold is reached so that to avoid buffer overflow attack.

7. Conclusions

This work presents a lightweight authentication mechanism and request queue management to combat DDoS attack in a smart home network. The scheme does not incur high computational costs, which are quantified on the Ethereum blockchain as the amount of gas used in each transaction. This feature (low computational costs) also affects the Central Processing Unit (CPU), memory, and battery consumption of each

Table 3: Computational cost per gas consumption

Requests	D2D protocol	S. Dey	S. Banerjee et al
1	110464	133429	142455
2	112499	120464	134578
3	114534	122499	135433
4	116569	124534	137446
5	118604	126569	139456
6	120639	128604	141043
7	122674	130639	143233
8	124709	132674	145122
9	118429	134709	147433
10	128779	136744	149023
11	130814	138779	151001
12	132849	140814	153480
13	134884	142849	155213
14	136919	144884	157211
15	138954	146919	159322
16	140989	148954	161234
17	143024	150989	163233
18	145059	153024	164984
19	147094	155059	166234
20	149129	157094	168873

D2D protocol Average computational cost:	**129380.8**
S. Dey Average computational cost:	**138511.5**
S. Banerjee et al Average computational cost:	**150800.4**

device, thereby making the system more adaptable. Security analysis results showed that the proposed solution is resilient to DDoS attack. Security vulnerability analysis results of the smart contract showed that our solution is protected from bugs and is not vulnerable to threats. Experimental analysis results of the proposed scheme revealed that the overhead incurred when tackling DDoS attack is significantly less compared to the existing approaches.

References

[1] R. Almadhoun, M. Kadadha, M. Alhemeiri, M. Alshehhi and K. Salah, A User Authentication Scheme of IoT Devices using Blockchain-Enabled Fog Nodes, in: Proceedings of 2018 IEEE/ACS 15th International Conference on Computer Systems and Applications (AICCSA), 2018, pp. 1-8.

[2] U. Javaid, A. K. Siang, M. N. Aman, and B. Sikdar, Mitigating IoT Device based DDoS Attacks using Blockchain, in: Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems - CryBlock'18, 2018, pp. 71–76.

[3] I. Yaqoob, E. Ahmed, M.H.ur Rehman, A.I.A. Ahmed, M.A. Al-garadi, M. Imran, and M. Guizani, The rise of ransomware and emerging security challenges in the Internet of Things. Computer Networks, 129 (2017), pp. 444-458.

[4] J. Vora, A. Nayyar, S. Tanwar, S. Tyagi, N. Kumar, M.S. Obaidat, J.J.P.C. Rodrigues, BHEEM: A blockchain-based framework for securing electronic health records, in: Proceedings of 2018 IEEE Globecom Workshops (GC Wkshps), pp. 1-6.

[5] T. Alladi, V. Chamola, J.J.P.C. Rodrigues, S.A. Kozlov, Blockchain in smart grids: A review on different use cases, Sensors 19(22) (2019), pp. 4862.

[6] M. A. Rahman and M. S. Hossain, M-Therapy: A multisensor framework for in-home therapy management: A social therapy of things perspective, IEEE Internet Things J., 5(4) (2018) pp. 2548–2556.

[7] M. A. Khan and K. Salah, IoT security: Review, blockchain solutions, and open challenges, Futur. Gener. Comput. Syst., 82(2018) pp. 395–411.

[8] S. Garg, K. Kaur, G. Kaddoum, F. Gagnon and J. J. P. C. Rodrigues, An Efficient Blockchain-Based Hierarchical Authentication Mechanism for Energy Trading in V2G Environment, in: Proceedings of 2019 IEEE International Conference on Communications Workshops (ICC Workshops), 2019, pp. 1-6,

[9] N. Garg, M. Wazid, A.K. Das, D.P. Singh, J.J.P.C. Rodrigues, Y. Park, BAKMP-IoMT: Design of blockchain enabled authenticated key management protocol for internet of medical things deployment, IEEE Access, 8 (2020), pp. 95956-95977.

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

[11] M.N. Jamal, S.A. Hassan, D.N.K. Jayakody, J.J.P.C. Rodrigues, Efficient nonorthogonal multiple access: Cooperative use of distributed space-time block coding, IEEE Vehicular Technology Magazine, 13(4)(2018) pp. 70-77.

[12] D. Chattaraj, B. Bera, A.K. Das, J.J.P.C. Rodrigues, Y. Park, Designing Fine-grained Access Control for Software Defined Networks using Private Blockchain, IEEE Internet of Things Journal, 9(2)(2022) pp. 1542-1559.

[13] R. Saha, G. Kumar, T. Devgun, W.J. Buchanan, R. Thomas, M. Alazab, T.H. Kim, J.J.P.C. Rodrigues, A Blockchain Framework in Post-Quantum Decentralization, IEEE Transactions on Services Computing, ISSN: 1939-1374.

[14] A. Lakhan, M.A. Mohammed, S.Kozlov, J.J.P.C. Rodrigues, Mobile-Fog-Cloud Assisted Deep Reinforcement Learning and Blockchain-Enable IoMT System for Healthcare Workflows, Transactions on Emerging Telecommunications Technologies (ETT), Wiley, ISSN: 2161-3915, p.e4363.

[15] S. Wang, S. Sun, X. Wang, Z. Ning, J.J.P.C. Rodrigues, Secure Crowdsensing in 5G Internet of Vehicles: When Deep Reinforcement Learning Meets Blockchain, IEEE Consumer Electronics Magazine, IEEE, Vol. 10, N. 5, 10(5) (2021), pp. 72-81.

[16] S. Huh, S. Cho and S. Kim, Managing IoT devices using blockchain platform, in: Proceedings of 2017 19th International Conference on Advanced Communication Technology (ICACT), 2017, pp. 464-467.

[17] S. Dey, Session-Key Establishment and Authentication in a Smart Home Network using Public Key Cryptography, IEEE Sensors Lett., 3(4) (2019), pp.1-4.

[18] S. Banerjee et al., A Provably-Secure and Lightweight Anonymous User Authenticated Session Key Exchange Scheme for Internet of Things Deployment, IEEE Internet Things J., 6(5) (2019), pp.8739-8752.

[19] C. Paar and J. Pelzl, Understanding cryptography a textbook for students and practitioners. Springer Science & Business Media, 2013.

[20] V. Buterin, Ethereum White Paper, http://buyxpr.com/build/pdfs/EthereumWhitePaper.pdf. (accessed 2020-07-07).

[21] Investforesight.com, Benefits of Ethereum blockchain, Invest Foresight-Online business magazine, https://investforesight.com/benefits-of-ethereum-blockchain/#: :text=Ethereum allows its users to,tools to build decentralized applications. (accessed 2019-03-02).

[22] N. Atzei, M. Bartoletti, and T. Cimoli, A survey of attacks on Ethereum smart contracts (SoK), in: Proceedings of 2017 International conference on principles of security and trust, Springer, 2017, pp. 164-186.

[23] Y. Zhang, S. Kasahara, Y. Shen, X. Jiang, and J. Wan, Smart contract-based access control for the internet of things, IEEE Internet Things J., 6(2) (2019) pp. 1594–1605.

[24] Y.C. Hu, T.-T. Lee, D. Chatzopoulos, and P. Hui, Hierarchical interactions between Ethereum smart contracts across Testnets, in:Proceedings of CRYBLOCK '18: 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems, 2018, pp. 7–12.

[25] Y. Xu, G. Wang, J. Yang, J. Ren, Y. Zhang, and C. Zhang, Towards Secure Network Computing Services for Lightweight Clients Using Blockchain, Wirel. Commun. Mob. Comput., 2018 (2018), pp. 1–12.

[26] T. M. Fernandez-Caramas and P. Fraga-Lamas, A Review on the Use of Blockchain for the Internet of Things, IEEE Access, 6 (2018) pp. 32979–33001.

[27] K. Leng, Y. Bi, L. Jing, H. C. Fu, and I. Van Nieuwenhuyse, Research on agricultural supply chain system with double chain architecture based on blockchain technology, Futur. Gener. Comput. Syst., 86 (2018), pp.641-649.

[28] W. Lin et al., Blockchain Technology in Current Agricultural Systems: From Techniques to Applications, IEEE Access, 8 (2020), pp.143920-143937.

[29] M. N. Aman, M. H. Basheer, and S. Member, Data Provenance for IoT with Light Weight Authentication and Privacy Preservation, IEEE Internet Things J., 6(6) (2019), pp.10441-10457.

[30] J. Eriksson, M. Faloutsos, and S. V. Krishnamurthy, Routing amid colluding attackers, in: Proceedings of 2007 IEEE International Conference on Network Protocols pp. 184-193 IEEE.

[31] Y. Park, Physically Secure Lightweight Anonymous User Authentication Protocol for Internet of Things Using Physically Unclonable Functions, 7(2019) pp. 85627–85644.

[32] D. Shin et al., A Security Protocol for Route Optimization in DMM-based Smart Home IoT Networks, IEEE Access, 7 (2019), pp.142531-142550.

[33] M. S. Hossain, G. Muhammad, W. Abdul, B. Song, and B. B. Gupta, Cloud-assisted secure video transmission and sharing framework for smart cities, Futur. Gener. Comput. Syst., 83 (2018), pp. 596–606.

[34] C. Lin, D. He, N. Kumar, X. Huang, P. Vijayakumar and K. R. Choo, HomeChain: A Blockchain-Based Secure Mutual Authentication System for Smart Homes, IEEE Internet of Things Journal, (2)(2020) pp. 818-829.

[35] Lee, Y., Rathore, S., Park, J.H. et al. A blockchain-based smart home gateway architecture for preventing data forgery. Hum. Cent. Comput. Inf. Sci. 10(9) (2020) pp.1-14.

[36] M. A. Rahman, M. M. Rashid, M. Shamim Hossain, E. Hassanain, M. F. Alhamid, and M. Guizani, Blockchain and IoT-Based Cognitive Edge Framework for Sharing Economy Services in a Smart City, IEEE Access, 7(2019), pp. 18611–18621.

[37] R. Guo, H. Shi, Q. Zhao, and D. Zheng, Secure Attribute-Based Signature Scheme with Multiple Authorities for Blockchain in Electronic Health Records Systems, IEEE Access, 6(2018), pp. 11676–11686.

[38] A. Yassine, S. Singh, M. S. Hossain, and G. Muhammad, IoT big data analytics for smart homes with fog and cloud computing, Futur. Gener. Comput. Syst., 91(2019), pp. 563–573.

[39] H. Yin, D. Guo, K. Wang, Z. Jiang, Y. Lyu, and J. Xing, Hyperconnected Network: A Decentralized Trusted Computing and Networking Paradigm, IEEE Netw., 32(1) (2018), pp. 112–117.

[40] A. Qashlan, P. Nanda and X. He, Security and Privacy Implementation in Smart Home: Attributes Based Access Control and Smart Contracts, in: Proceedings of 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), 2020, pp. 951-958.

[41] R. Zhang, F. R. Yu, J. Liu, T. Huang and Y. Liu, Deep Reinforcement Learning (DRL)-Based Device-to-Device (D2D) Caching With Blockchain and Mobile Edge Computing, IEEE Transactions on Wireless Communications, 19(10) (2020), pp. 6469-6485.

[42] H. -N. Dai, Z. Zheng and Y. Zhang, Blockchain for Internet of Things: A Survey, IEEE Internet of Things Journal, 6(5) (2019), pp. 8076-8094.

[43] M. Singh and S. Kim, Branch based blockchain technology in intelligent vehicle, Comput. Networks, 145(2018), pp. 219–231.

[44] M. A. Rahman, E. Hassanain, M. M. Rashid, S. J. Barnes, and M. Shamim Hossain, Spatial Blockchain-Based Secure Mass Screening Framework for Children with DysleRZhang, IEEE Access, 6(2018), pp. 61876–61885.

[45] Y. Zhang, S. Kasahara, Y. Shen, X. Jiang, and J. Wan, Smart contract-based access control for the internet of things, IEEE Internet Things J., 6(2) (2019), pp. 1594–1605.

[46] I. Makhdoom, I. Zhou, M. Abolhasan, J. Lipman, and W. Ni, PrivySharing: A blockchain-based framework for privacy-preserving and secure data sharing in smart cities, Comput. Secur., 88(2020), p. 101653.

[47] CS. Departement, A Single-Server Queue Section 1.2: A Single-Server Queue arrivals queue service node, Willliam Mary Univ. CS Study Guid., 2006.

[48] O. Novo, Blockchain Meets IoT: An Architecture for Scalable Access Management in IoT, IEEE Internet of Things Journal, 5(2) (2018), pp. 1184-1195.

[49] X. Li, M. Wang, and S. Member, Toward Secure and Efficient Communication for the Internet of Things, IEEE/ACM Transactions on Networking, 27(2) (2019), pp.621-634.

[50] I. Florea and C.-Ş. Nănău, A simulation algorithm for a single server retrial queuing system with batch arrivals, Analele Univ. "Ovidius" Constanta - Ser. Mat., 2017.

[51] S. Mishra, N. K. Singh, and V. Rousseau, System on Chip Interfaces for Low Power Design. Morgan Kaufmann, 2015.

[52] P. Praitheeshan, L. Pan, J. Yu, J. Liu, and R. Doss, Security Analysis Methods on Ethereum Smart Contract Vulnerabilities: A Survey, arXiv preprint arXiv:1908.08605, pp. 1–21, 2019.

[53] N. Nizamuddin, H. Hasan, K. Salah, and R. Iqbal, Blockchain-Based Framework for Protecting Author Royalty of Digital Assets, Arab. J. Sci. Eng., 44(4) (2019), pp. 3849–3866.

[54] L. Luu, D. H. Chu, H. Olickel, P. Saxena, and A. Hobor, Making smart contracts smarter, in: Proceedings of the ACM Conference on Computer and Communications Security, 2016, pp. 254–269.