Performance and security evaluation of blockchain-enabled IoT networks

Ayush Sharma

School of Computing Science

Sir Alwyn Williams Building

University of Glasgow

G12 8RZ

A dissertation presented in partial fulfilment of the requirements of the Degree of Master of Science at the University of Glasgow

26 August 2022

Abstract

Internet of Technology (IoT) has become an integral part of society, and it comes with its challenges. Privacy and security issues arise with the large amount of data created and shared by users over the internet. The emergence of the blockchain and its integration with IoT devices is a hot topic in the industry. In this research, we look at the simulation of the blockchain-based IoT device to evaluate the security and performance of the devices after blockchain integration. We use ganache to create a local Ethereum blockchain and integrate it with the Raspberry Pi simulator to create a simulation of the blockchain-based IoT devices. The IoT device is managed using a smart contract deployed in the local blockchain, and a web application is used to interact with it. The simulations were tested against some security attacks, and the results indicate a robust and secure IoT device network with a minimal impact on the device's performance. The performance is evaluated against the transaction execution time and resource usage, which are essential for further research and optimization in integrating both technologies.

Education Use Consent

I hereby give my permission for this project to be shown to other University of Glasgow students and to be distributed in an electronic form.

Name: Ayush Sharma Signature: [illegible]

Acknowledgements

I want to take this opportunity to show my heartfelt gratitude and thanks to my supervisor Dr Dongzhu Liu for the constant guidance and help throughout the project. Also, a big thank you to my family and friends for their support.

Contents

Chapter 1 Introduction

Data is a powerful tool in the modern world. With the wide availability of devices connected to the internet, users share a large amount of data every second. Internet of Things (IoT) devices are a network of embedded small-scale devices with minimal computational powers and hardware constraints connected over the network, which collect, process, and share data over the web. According to studies, the number of IoT devices is expected to reach approximately 19.1 billion in 2025. By 2030, it will observe explosive growth and get 29.4 billion [18], but due to so many constraints, IoT devices fail to implement strict security policies [4]. The IoT devices provide a single point of failure with a centralised architecture.

IoT devices have been employed in many industries and have vastly taken a critical hold in emerging technologies. With such a high spread of devices, the implementation of IoT devices has found its way into our daily lives. IoT devices can control air conditioning at home, turn off tube lights, bulbs or fans, closing the main door, windows, or garage door through a mobile app from the office or any other remote location [10]. An IoT device can have components such as sensors that collect raw data, a gateway that is used as a medium to transfer data, such as the internet, a storage space, usually cloud storage, and a user interface such as a mobile application to view the analysed data in the form of graphs or reports. It is time we think towards a more distributed architecture to increase security and avoid that single point of failure in the system.

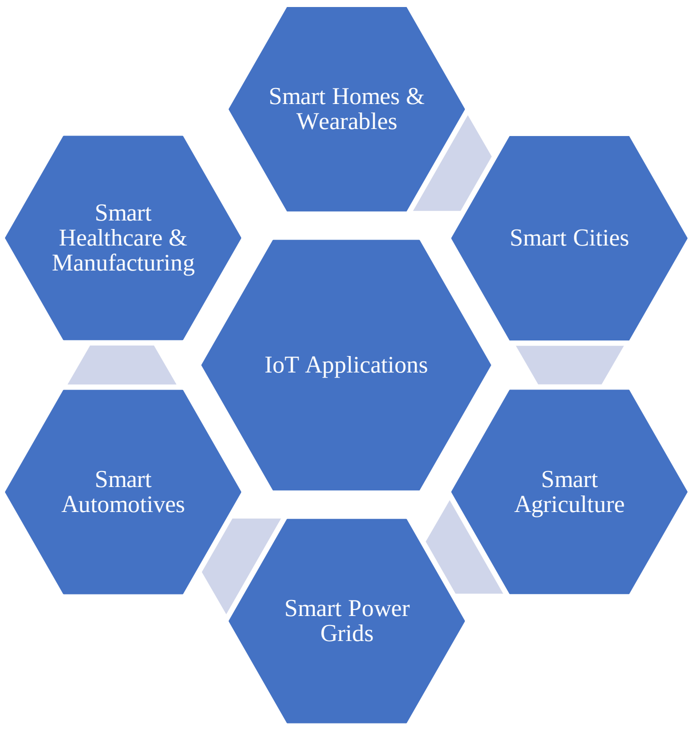


Figure 1: Applications of IoT

Blockchain Technology is a Distributed Ledger Technology (DLT) that secures, verifies, and records all peer-to-peer transactions quickly, safely, and transparently [5]. Blockchain uses a distributed network system to record data, and each list of data is called a block. These blocks store a cryptographic hash of the previous block, a timestamp at which the block was generated, and transaction data, represented in the form of a Merkle root tree [2]. The following blocks reinforce every block, containing the previous block's hash value. Furthermore, each block is sent over the network to each endpoint, creating multiple copies for verification.

The distributed nature of blockchain architecture is the future for the IoT industry, which faces serious security risks. However, blockchain has numerous applications and parameters that affect the system's security, performance, and usability [8]. Although. Blockchain has already found its way into multiple IoT devices such as smart homes, smart cities, smart agriculture, smart power grids, smart transportation and automotive, smart healthcare, and smart manufacturing [4]. Even after such benefits of blockchain in IoT, we have only recently seen much progress in the IoT devices to implement the distributed architecture.

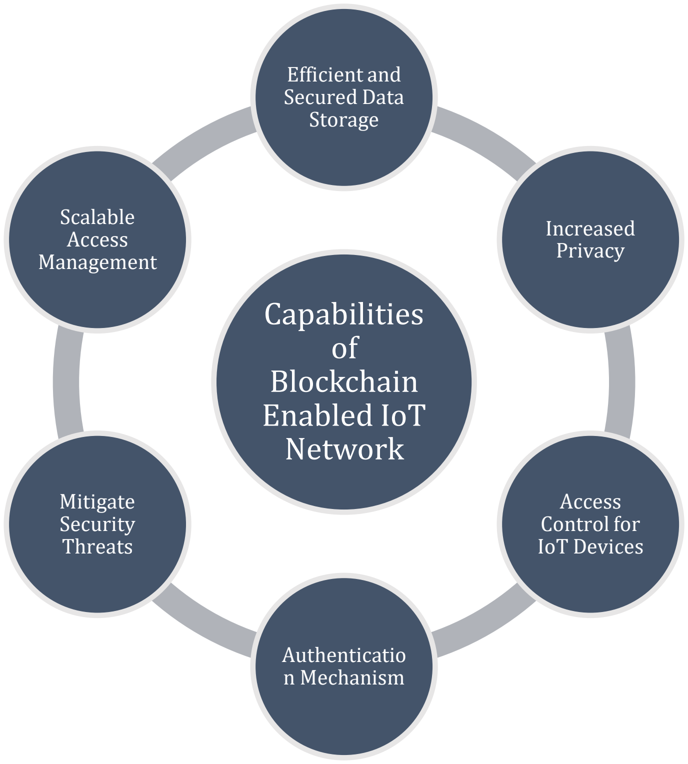

Efficient and Secured Data Storage
Increased Privacy
Access Control for IoT Devices
Authenticatio n Mechanism
Mitigate Security Threats
Scalable Access Management
Capabilities of Blockchain Enabled IoT Network

Figure 2: Capabilities of Blockchain-Enabled IoT Network

1.1 Motivation and Contribution

Generally, IoT devices are plagued by privacy and security problems due to their limited computational powers and storage capacities. Blockchain-enabled IoT devices can overcome these challenges but, in return, generate a few of their challenges, such as high energy consumption and high computational resource requirement. Since blockchain was designed for large systems with increased computational power, this takes a toll on the system's performance as this takes longer to complete a transaction.

In this thesis, we will focus on an explanatory simulation of a blockchain-based IoT application — a secure transaction between two peers and will review the impact on security and privacy. For tractability, we will apply Blockchain's Smart Contracts to manage, control, and secure IoT device. A smart contract is a simple program, a collection of functions and data stored at a specific address on the blockchain. Moreover, we will also evaluate the performance toll of blockchain architecture in IoT applications.

1.2 Security Risk in IoT

This section will discuss a few security issues in the IoT ecosystem. With the sharp increase in IoT devices, most applications have become IoT-based applications embedded in different IoT devices such as smart TV, smart watches, smart home, etc. As discussed earlier, the device limitations of the IoT devices leave a big hole in the security aspect of the IoT devices connected over the web. Mohanta et al. [1] categorise IoT applications with "three-layer architecture, which are the physical, network and application layers." The security attacks should be addressed at the respective layers of the attack instead of changing the compromised IoT device.

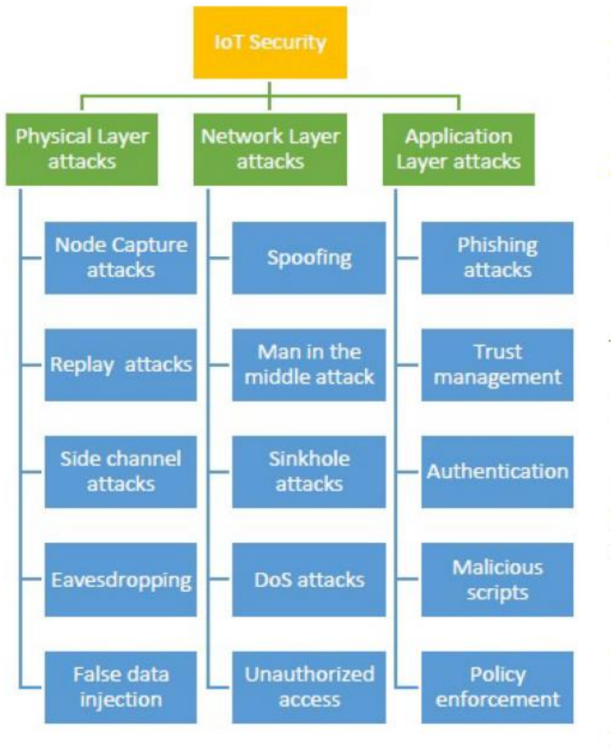


Figure 3: Some IoT Security Issues divided into Three-Layer Architecture [1]

Node Capture Attacks: This is a physical attack where the attacker can change the genuine node with a fake node to access the network and use this access to receive critical data. Early detection tools and techniques are needed to mitigate this physical attack.

Replay Attacks: The attacker captures a message or communication from a node and later sends the same message in the network to misguide or make the system believe the data is coming from an authorised node.

Side-Channel Attacks: These attacks exploit the system information leaked over the network, like timing, power to break the cryptography, and access to keys for encryption and decryption.

Eavesdropping: These attacks happen over an insecure network where the attacker can read messages between the IoT nodes or devices in an IoT ecosystem. This is a passive attack, where the user's privacy is compromised.

False Data Injection: Sensors deployed over different locations can capture the sensor data and share it over the network; due to resource constraints, the attacker can gain access to the communication medium and pass on false information.

Spoofing: In the network layer, the attacker can gain access to a node and act as a legitimate node in the network, passing falsified data.

MITM Attack: Man-in-the-Middle attack is both an active and passive attack. In these attacks, the attacker intercepts the data packets in the communication channel to gain insights into the data that can be critical to the user.

Sinkhole Attack: These attacks use compromised nodes to attract network traffic and compromise the communication medium by sending fake routing routes. These collapse the network and can also be used to send acknowledgement in spoofing attacks.

DoS Attacks: These are the most common types of attacks happening in the IoT ecosystem; in this, the attacker tries to compromise the system's expected functionalities.

Unauthorised Access: As the name suggests, the attacker tries to gain access to the IoT node, and once the attacker is in the network, they can exploit the information in the system.

Phishing Attacks: Social engineering attacks such as Phishing attacks in an extensive user-based IoT network are highly effective. The attacker tries to retrieve important information such as User-Id or passwords to gain access to the IoT network by sending an email or message.

Trust Management: At the application layer in an IoT application, a user shares confidential private information to manage and monitor the application. The information is broadcasted in the network during the analysis of the data, a decentralised environment. Nodes must be monitored for malicious activity to implement proper trust management.

Authentication: Smart devices and sensors in an IoT ecosystem collect and process data all the time. An attacker who captures sensitive data can cause serious harm to the network or services of the ecosystem. Therefore, authenticating and registering all the IoT nodes and devices is critical.

Malicious Scripts: IoT devices depend on the outside world communication mediums to share data. An attacker can access such communication channels or applications and put a malicious script to compromise the device in the channel or the application.

Policy Enforcement: In IoT applications, implementing a security policy to safeguard user privacy is a challenge with the limitations of the resources.

1.3 Blockchain: A Background and Working Principles

Blockchain allows peer-to-peer transfer of digital assets without any intermediaries [19]. Blockchain was a technology initially created to support the famous cryptocurrency Bitcoin. Bitcoin was first proposed in 2008 and implemented in 2009 by Nakamoto [20]. Although Bitcoin is the most prominent blockchain application, it has found its way into various industrial applications apart from cryptocurrencies.

Blockchain can be used mainly in permissionless and permissioned [22]. The permission-less design is commonly used in an open environment like Bitcoin, or Ethereum, which allows anyone to join the system and modify the shared blocks. The permissionless designs also give equal privilege to all the nodes in case of the consensus process. The Permissioned Blockchain design is used by a known set of entities and is established in a closed environment such as Hyperledger fabric. Although all the entities are allowed to perform transactions, only a fixed set of predetermined nodes can participate in the consensus process in a Permissioned Blockchain [22]. Consensus algorithms are fundamental for an efficient and secure Blockchain system. Some popular consensus algorithms are Proof of Work (PoW), Proof of Burn (PoB), Proof of Stake (PoS), Raft, Practical Byzantine Fault Tolerant (PBFT), Paxos, etc. [23]

A. Features of Blockchains

Decentralisation: In a centralised architecture, transactions are validated and authorised by a central third party. On the other hand, in the blockchain architecture, two nodes are independent of a central trust entity and can engage in a transaction. As a result, blockchain avoids any bottleneck.

Immutability: All the transactions in a blockchain are agreed upon by peers using decentralised consensus. To compromise the data, an attacker will have to compromise a significant number of nodes which is easy to detect, making the blockchain data immutable.

Auditability: All the peers hold a copy of the blockchain and, as a result, have all the timestamped transactions; this allows the peers to audit and verify any transaction in the past.

Fault Tolerance: The copies of blockchain with each node due to decentralised consensus help mitigate the risk of data leak or faulty copies by tallying the blockchain copies stored with the peers.

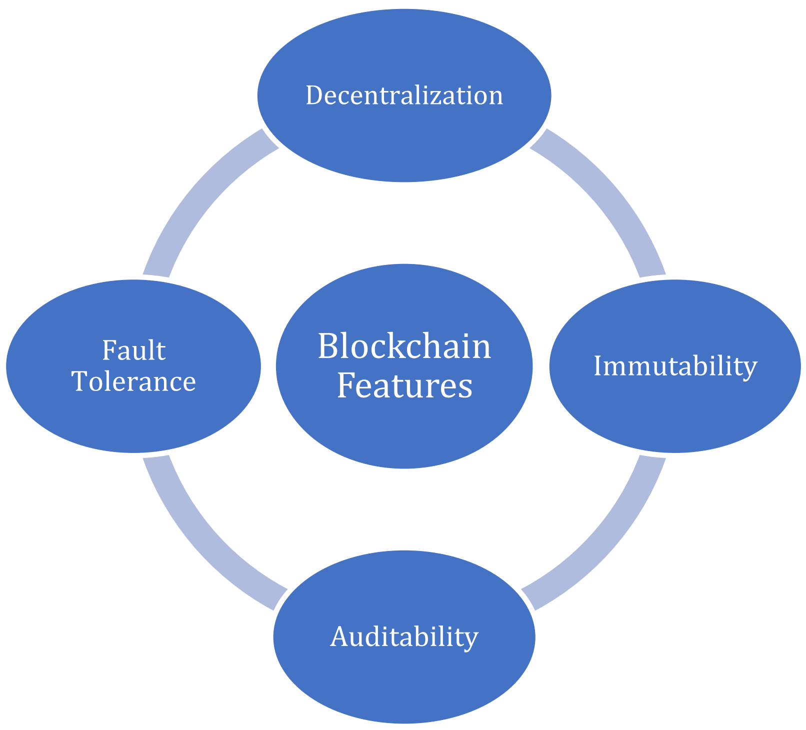


Figure 4: Blockchain Features

B. Blockchain Structure

A blockchain comprises blocks in a network containing details of transactions. Ali et al. [23] describe the blockchain as follows, "Each block is logically divided into two parts, namely, the header and the body. Transactions are stored within the body of the block, while the header of each block contains, among other fields, the identifier of the previous block." The blocks are connected like a chain, as shown in Figure 4; hence the name Block-Chain. The first block in the blockchain is called the "genesis block," which is always hardcoded into the application software using the blockchain [7].

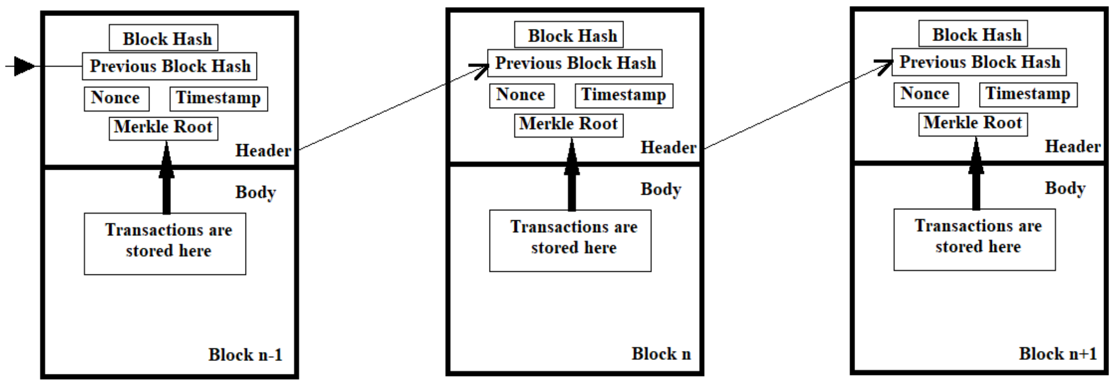


Figure 5: Blockchain Structure

A block is a data structure that typically contains the number, version, hash of the previous block, Merkle root, timestamp, nonce, transaction count, and signed transactions followed by a long list of transactions [7]. The identifier for each block can be obtained by referencing the block hash or height. The block header contains metadata that references the previous block's hash, timestamp, nonce, and Merkle tree root, summarising the block's transactions.

C. Types of Blockchain

Public Blockchain: These are truly decentralised, which means all the entities can participate in creating a new block, and access is allowed to all the entities. Public blockchains are often termed "permissionless" as they allow everyone to keep a blockchain copy and take part in verifying the same. These are designed to hold many entities or nodes, and publishing each node requires a cost. The cost can be solving a puzzle that is computationally expensive or leveraging cryptocurrency. Since each transaction has a processing fee, which is an incentive for the peers helps in keeping the blockchain secure as the cost of tampering with the chain will be massive.

Private Blockchain: These are "permissioned," and each new node coming to the blockchain is a member of a specified organisation. These are suitable for a single organisation that needs to keep track of all transactions between different departments or personnel. These blockchain does not contain any transaction fee and, as a result, are not very secure against tampering. The organisation maintains that the blockchain can roll back its chain to any point in the past without incurring any cost.

Consortium Blockchain (Federated Blockchain): Similar to private blockchains, consortium blockchains are permissioned blockchains spread through multiple organisations, helping maintain the transparency of the parties involved. These also do not include any transaction fee and are used to increase the auditability of the organisation's transactions.

D. Consensus Algorithms

In the blockchain, when a new block is added to the chain, it needs to be verified by all the nodes in the network. Murthy et al. [6] define a consensus algorithm as "a kind of protocol that maintains the nodes involved in the blockchain network to reach a transaction order decision and filter invalid transactions." Alrubei et al. [12] state that the "most popular" consensus algorithm is Proof-of-Work (PoW), where nodes mine a solution for the nonce in the hope of gaining incentive using sufficient computing power. Consensus algorithms are designed with the objective of safety, liveness, and fault tolerance. Table 1 compares permissionless and permissioned blockchain systems and the consensus algorithms used.

In a public blockchain or permissionless blockchain, anyone is allowed to participate, and reaching a consensus using votes can compromise the integrity of the blockchain as anyone can create multiple accounts to get the minimum required votes in their favour. Therefore, in a public blockchain, consensus algorithms are based on the lottery-based selection of a node that publishes the new block in the blockchain [23]. In a private or consortium blockchain where permission is required to join, achieving consensus is low cost and straightforward as it avoids the risk of getting multiple votes from the same miner. Hence, a voting mechanism is employed to reach a consensus.

Table 1: Comparison of Permissionless and Permissioned Blockchain [22]

	Permissionless Blockchain	**Permissioned Blockchain**
Environment Type	Open	Closed
Participation in Consensus	All nodes	Selected nodes
Consensus Type	Lottery based	Voting based
Transaction Processing Speed	Slow	Fast
Consensus Algorithms	Proof of Work, Proof of Stake, Proof of Burn, Proof of Delegated Stake, etc.	Proof of Work, Proof of Stake, Proof of Burn, Proof of Delegated Stake, etc.

Chapter 2 Literature Review

In the last few years, although there has been an increase in the research on the application of blockchain in the resource constraint IoT devices, only a few focus on meeting the IoT security requirements while keeping the performance in consideration. This section will discuss the previous works focusing on applying the blockchain with IoT devices to achieve a secure, good performance-based system.

Satoshi Nakamoto [20] introduced bitcoin in a white paper to solve the trust issue in a decentralised monetary system. Bitcoin works on a decentralised blockchain that has applications other than just cryptocurrency. Ali et al. [23] extensively discussed the blockchains and the characteristics of blockchain in an IoT environment, emphasising the security benefits of integrating the blockchain in an IoT environment. Christidis et al. [30] described how blockchain could be combined with IoT and provided a list of the advantages and limitations of the distributed architecture. They concluded that using the blockchain and smart contracts allows the sharing of IoT devices and resources cryptographically. The concept of security enhancement needed further discussion focusing on the benefits of distributed architecture in IoT devices.

Smart contracts are an integral part of any blockchain and are autonomous programs that can be used to perform actions and transactions in a blockchain when certain conditions are met, Hammi et al. [29] exploited the concept of smart contracts in the blockchain to build a new authentication scheme for the IoT environment called Bubbles-of-Trust. They proposed to create secure virtual zones where the IoT devices can communicate securely, relying on the implementation of the public blockchain to verify the devices in the blockchain and, in turn, using all the security benefits of the public blockchain. They extensively discussed the security benefits of smart contracts in authenticating IoT devices and creating a more secure IoT environment. However, implementing the authentication system in the live application and the cost of authenticating devices of multiple users in the public blockchain is an overhead that needs to be considered.

By suggesting decentralised web authentication in blockchain, Mohanta et al. [28] proposed AuthKey, a 160-bit hash key capable of defending against cyberattacks in the IoT environment. They used the Ethereum platform and smart contracts, where web authentication happens through decentralised Ethereum authentication, DecAuth. DecAuth overcomes the issue of live monitoring of the application, which is fundamental to IoT devices and provides a method to authenticate IoT devices in a decentralised way. Alrubei et al. [12] implemented the Proof of Authority Ethereum blockchain to do a quick study of the performance within the IoT and understand the real-world use case of implementing blockchain in the IoT. Similarly, Huh et al. [14] proposed using Ethereum and smart contracts to manage IoT devices using Proof of Concept (PoC) over public key infrastructure to make it more secure.

Various blockchain simulators in the market can be used to simulate blockchain in the IoT environment, Zheng et al. [4] reviewed 18 blockchain simulators and discussed the advantages and challenges of the specific simulators, and suggested

Ethereum and Ganache as the popular choice and similar findings were also addressed by Alkhateeb et al. [5], they also suggest Ethereum is considered as a mature blockchain platform for smart contracts development. They discussed the applications of the blockchain with data warehouses in multiple domains, pointing out that security, data integrity, and efficiency are the top motivations behind integrating the blockchain and other technologies, such as data warehouses.

Similarly, Okegbile et al. [9] combined cloud-edge computing-based technologies and blockchain to avoid high transaction outage rates when offloading to cloud servers. They proposed a blockchain-enabled health data-sharing model (BeHDS). Modified PBFT protocol to validate the transactions, which captured the data-sharing system's block generation and blockchain-building stages. They emphasised the privacy of the shared data in the data-sharing system and evaluated the system's performance with live edge nodes. However, with the implementation of the modified PBFT protocol, it was complicated to calculate the average validation time for each step in the protocol and estimate the time of the transactions. Ochôa et al. [27] proposed a similar blockchain storage architecture focusing on the privacy of license plate recognition using a private blockchain and smart contracts. The proposed system was secure against multiple attacks and identified the need for using a side chain to make the system scalable.

With the computation power of cloud servers increasing, Rehman et al. [2] proposed a secure cloud-based service, where the cloud nodes are introduced to maintain the validity state of edge-based nodes, and the reputation of the nodes is stored in the cloud nodes, and updated over time to ensure secure service for IoT devices. A proof of Authority (PoA) consensus algorithm is used to enhance the system's performance in the Ethereum blockchain instead of the Proof of Work (PoW) algorithm.

Ahmed et al. [7] proposed a secured framework for IoT using the Ethereum blockchain and its smart contracts to enforce a set of rules established by the system administrator. They discussed the mathematical analysis of the system's security on a hardware-based setup for three common types of cyber-attacks. They backed the framework's robustness and security with numerical results. Kreku et al. [8] evaluated the efficiency of blockchain implementation in the IoT with simulations on embedded devices using the existing simulation tool - ABSOLUT. They suggested that the simulation tool can be used to find the limits and possibilities of blockchain implementation in the IoT.

Mohanta et al. [1] addressed the security and privacy issues of the resource constraint IoT devices using the blockchain technique. Application of smart contracts using the solidity platform over the Ethereum blockchain, they suggested that only authentication and authorisation should take place in the blockchain network, that is, the devices such as smart-watch, smart-TV, etc., and the users are to be registered and authenticated in the blockchain network initially, and all the intermediate devices are authenticated using the decentralised Decauth technique [28].

Chapter 3 Design and Implementation

This chapter will discuss the implementation of the principles of blockchain in the IoT environment. It illustrates the execution of smart contracts in the Ethereum blockchain to manage IoT devices. The project implements Ethereum blockchain simulation in Python, HTML, CSS, JavaScript, and Solidity for smart contracts using Ganache and Truffle suite. A smart contract in Ethereum is used to manage the GPIO console pin (Raspberry Pi) configuration to simulate IoT device endpoints.

The data for the time and gas expenditure on every transaction for changing the status of the GPIO pin is published over the console. This will help address the transaction performance with various devices on the local blockchain network. Implementing the consensus algorithm and gas expenditure features of the blockchain through smart contracts will also address the security issues faced by the resource constraint IoT devices.

3.1 Design

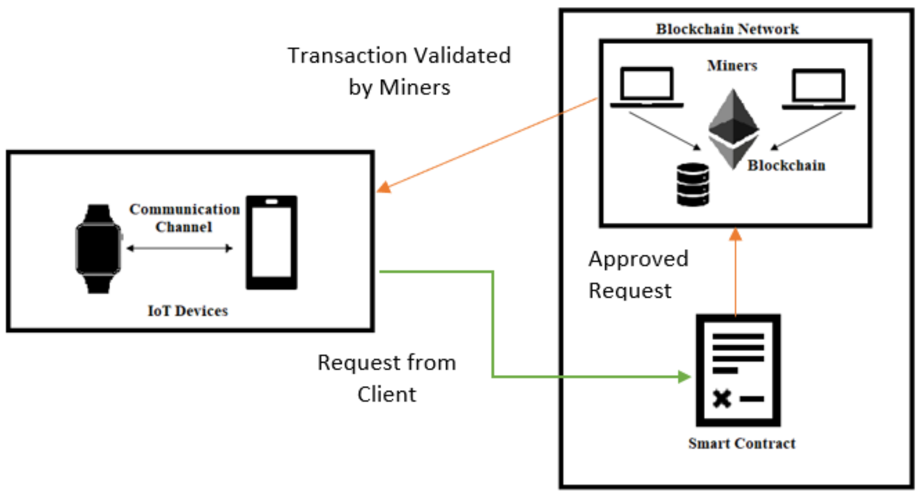


Figure 6: Smart contract implementation in IoT Network

3.1.1 Ethereum Blockchain

Ethereum is the most used mature blockchain [5] platform for developing smart contracts. "In the Ethereum universe, there is a single, canonical computer called Ethereum Virtual Machine or EVM, whose state everyone on the Ethereum network agrees on [31]." Every node keeps a copy of the EVM's state, and any network participant can request a computation from the EVM. Whenever such a request is made, other participants verify, validate and execute the computation, which causes a state change in the EVM, which is broadcasted throughout the network [34]. These transactions are confirmed valid, added to the blockchain through a cryptographic mechanism, and made tamper-proof. The exact mechanism also ensures that all the transactions are signed and executed with appropriate permissions.

3.1.2 Ether

Ether or ETH is the native cryptocurrency of the Ethereum network, which provides an economic incentive for participants to verify and execute transactions and provide computational resources to the Ethereum network. When a transaction occurs, the participant offers ether to the network as a bounty to whoever verifies and executes the transaction, commits it to the blockchain, and broadcasts it to the Ethereum network.

3.1.3 Gas

Gas is the amount of computational work required by the Ethereum network to perform a specific operation or transaction. Every Ethereum transaction requires resources and the resources require a transaction fee. This fee is the gas needed in Ethereum to execute a transaction successfully. The costs are paid in Ether or ETH and denoted in gwei, where each gwei equals 10^{-9} ETH.

In Ethereum, if a new block is to be added, there is a base fee calculated on demand for block space in the network. Users are also expected to provide a tip for the miners to execute the transaction, and this fee is usually automatically set by most of the wallets. The gas fee helps keep the Ethereum network safe, as, for every computation, we require a payment in the form of gas, preventing malicious users from spamming the Ethereum network.

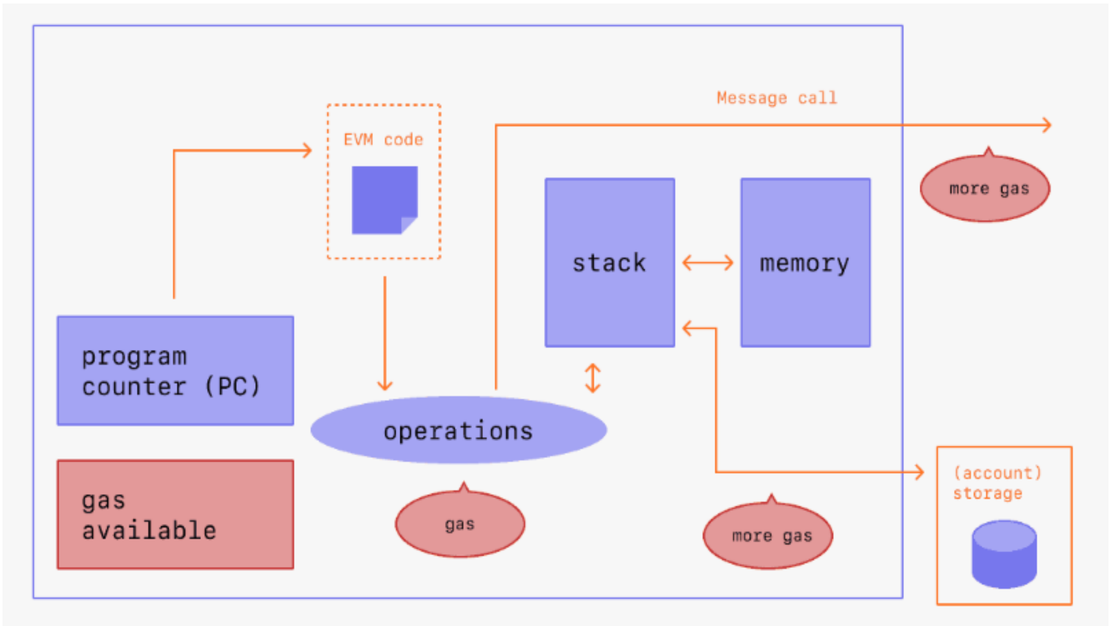


Figure 7: Ethereum EVM and Gas Fee [31]

3.1.4 Proof-of-Work (PoW) Consensus Algorithm

Proof-of-work is a consensus algorithm that lets the Ethereum network come to a consensus and is the underlying algorithm that sets the difficulties and rules for miners' work. Mining is the process of creating and adding valid blocks of transactions to the Ethereum blockchain and is the "work" itself [32]. PoW prevents users from duplicating transactions and safeguards the Ethereum chain by making it tremendously challenging to attack or manipulate. Mining is important because the more "work" done, the longer the chain will be, and the higher the block number, the more confident the Ethereum network can be of its current state.

Ethereum transactions occur inside the blocks, and each block has a block difficulty, mixHash, nonce ("number used once"). Ethash is the proof-of-work protocol used by the miners to carry out the trial-and-error race to find the nonce of the block. While trying to create a block and racing against other miners, a miner repeatedly put a dataset, only available through downloading and running the existing complete chain, through a complex mathematical function. By trial and error, the dataset generates a mixHash below a nonce as suggested by the block difficulty. The difficulty dictates the target hash, which is verified by other miners, any minor change in the transaction can lead to a completely different hash value, signalling fraud.

Blockchain relies on a single state of truth; that is, the work done on the main chain is incentivised more than the sub-chain of the miners. Users always choose the main or heavy, or longest chain. The objective of the proof-of-work algorithm is to extend the chain, as the longest chain is the one with the most computational work and is the most believably valid. For a malicious miner, it becomes impossible to tamper with the transactions in the block as every time, the attacker will have to solve the nonce faster than everyone else. To consistently create valid and malicious blocks, the attacker will need 51% of the network mining power to beat everyone else [32]. To achieve such an amount of work, the energy and resources spent will outweigh the benefit of the attack.

3.1.5 Smart Contracts

The term "smart contract" was coined by N. Szabo with the objective of "securing relationships on public networks" [24]. Smart contracts are programmable applications stored in the blockchain that manage transactions under specific terms and conditions [22]. In a blockchain network, smart contracts are independent of a centralised third party to ensure the requirements are met. Smart Contracts are often termed "autonomous agents" because they have their accounts on the blockchain and their blockchain address [25].

Generally, smart contracts are written in a non-standard or domain-specific language such as Solidity to reach a consensus among all of the peers [13]. Smart contracts have a predictable behaviour to drive any logic expressed as a function, introducing the concept of decentralised autonomous organisations [12]. With proper secure coding guidelines followed, a smart contract can be used to provide multiple functionalities and utilities in the blockchain network.

3.2 Implementation

Smart contracts in Ethereum are written using Solidity, and majorly VS Code is used as the code interpreter. The simulation of a local blockchain is carried out using ganache in the truffle suite, a development and testing framework for blockchains using the Ethereum Virtual Machine.

3.2.1 Ganache

Ganache is a personal blockchain for developers to test and develop a local blockchain for a distributed application. Ganache can be used throughout the development cycle allowing the developer to create, deploy, and test distributed applications in a safe and deterministic environment. Ganache has two versions, UI and CLI, ganache UI is a desktop application and allows development in Ethereum and Corda, and the ganache command line (also called ganache-cli) enables the development in Ethereum. Ganache CLI is a part of the Truffle suite of Ethereum development tools and uses ethereumjs to simulate complete client behaviour.

Ganache CLI is started in a terminal with the command "ganache-cli," and this will create ten virtual accounts and their private keys, each starting with pre-funded ether. The keys are used to sign the transactions. On executing any transaction or smart contract, the ganache-cli produces a transaction receipt used to collect the performance data in the local blockchain. Ganache is a versatile and helpful virtual server tool that enables the developer to test smart contracts and helps understand the process of writing a transaction on the blockchain.

3.2.2 Web3.js

Transactions must be validated in the chain to communicate with the blockchain components. For an input node to communicate with the IoT device endpoint, which is an offline framework, the transaction has to be relayed to the peers in the network. Web3 is a collection of libraries that enables the communication between the Ethereum nodes using HTTP, IPC, or WebSocket. Web3.js is a library that contains specific modules for specific functionalities of the Ethereum ecosystem. Most web3.js objects support returning promises and a callback as the final parameter for function chains; web3.eth is a module designed to develop smart contracts and the Ethereum blockchain.

Web3 in the project uses the HTTP provider to connect with the smart contract and Ethereum nodes, which can be the Ethereum wallet or the GPIO pin endpoint simulating the IoT device endpoint for executing a transaction. Web3 can also connect with web applications using Metamask or TrustWallet (Ethereum-based browser extensions), which allows operating the Ethereum accounts. With a combination of Ethereum-based browser extensions such as Metamask, web3.js, and Ethereum, along with a web interface, an application can communicate easily with the back-end and the front-end.

3.2.3 Transactions in Blockchain Simulation

Transactions are cryptographically signed instructions from accounts [33]. An Ethereum transaction is an action that updates the state of the Ethereum network or EVM initiated by an external account, owned explicitly by a user and not by a contract. Any change in the state of EVM needs to be broadcast throughout the network. For example, a simple transaction will transfer ETH from one account to another; any node can request a transaction to be executed, which requires a transaction fee and is needed to be mined to be valid. Once a request is made, a miner will execute the transaction and broadcast the result to the whole network.

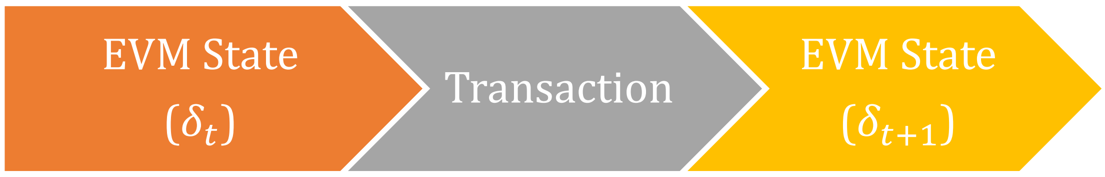


Figure 8: EVM as a state-based machine

Most transactions access a contract from an external account and interpret their data field according to the application binary interface (ABI) [33]. The first four hashed bytes specify the function call and the rest of the call data is the arguments encoded as per ABI specifications. Ethereum has a few different kinds of transactions, a regular transaction, a transaction from one account to another, a contract deployment transaction, where the data field is used for the contract code, and the execution of a contract, a transaction that deals with a deployed contract, and the "to" address is the smart contract's address. Ganache allows mining the transaction blocks immediately by default but can also provide options such as to mine the block after a specified time or at a fixed interval.

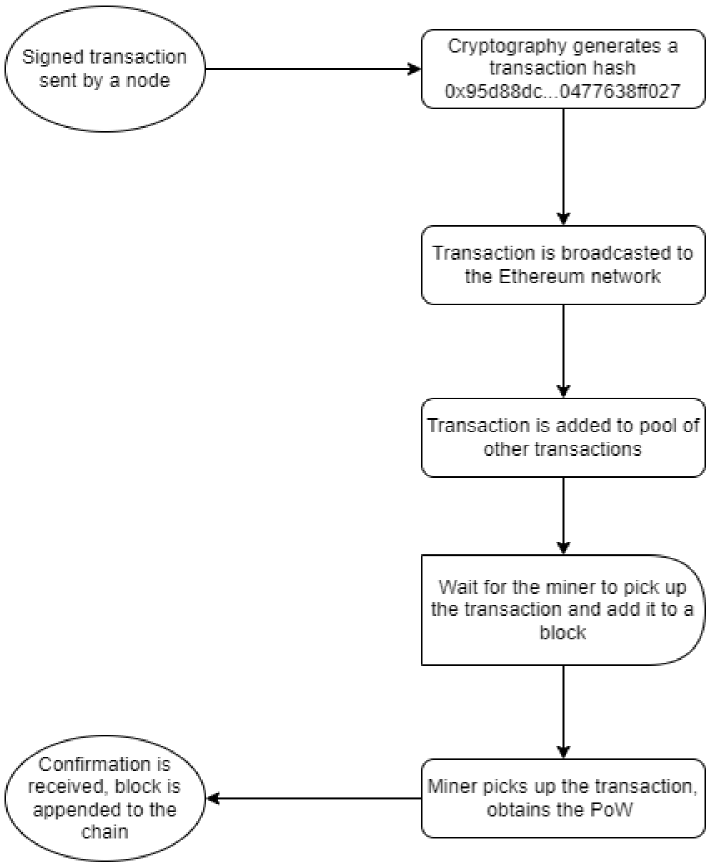

Signed transaction sent by a node
Cryptography generates a transaction hash 0x95d88dc...0477638ff027
Transaction is broadcasted to the Ethereum network
Transaction is added to pool of other transactions
Wait for the miner to pick up the transaction and add it to a block
Miner picks up the transaction, obtains the PoW
Confirmation is received, block is appended to the chain

Figure 9: Lifecycle of a transaction

3.2.4 Ethereum Blockchain in IoT Simulation

Raspberry Pi emulator (GPIO) is used in the program to simulate the endpoints of an IoT device. A smart contract written in Solidity is interpreted by the program to interact with the pin configuration of the GPIO module to simulate blockchain implementation in the IoT environment. Any modification done in the status of the pin (ON/OFF) is characterised as a transaction in the Ethereum network and requires a fee in the form of "gas," which is collected from the dummy account created by the ganache-cli instance.

Ganache-cli instance creates ten accounts with their private keys and compiles the smart contract written in Solidity. The program uses the web3.js HTTP provider to access the GPIO module, interact with the smart contract, and sign the transactions using the private keys of the accounts. The web3 object instantiates and deploys the smart contract and then stores the transaction hash from the deployed contract to receive the transaction's receipt to calculate the transaction's gas usage and time consumption. The status of the pin can be changed using the website written using Flask, a python framework on the default port 8545. The transaction details are printed on the ganache-cli terminal.

Chapter 4 Evaluation

The chapter is divided into two parts, discussing the approach for the two research questions of performance and security evaluation of a blockchain-based IoT network. The evaluation was carried out quantitatively and covered the performance and security of the blockchain in an IoT network.

4.1 Experimental Simulation Setup

We deploy the instance of the ganache-cli, which deploys the local Ethereum blockchain on the network and run the python program, which deploys the smart contract written in Solidity to the network. Multiple instances of the program are initiated to increase the load on the network by deploying multiple smart contracts on the network.

The number of smart contract instances can vary from 1 to 20, and each deployment consumes gas to be deployed on the blockchain system. No network latency is introduced in the system design. Ideally, it takes 14 seconds on the Ethereum blockchain [29] for a block to be deployed to the end nodes once consensus is reached.

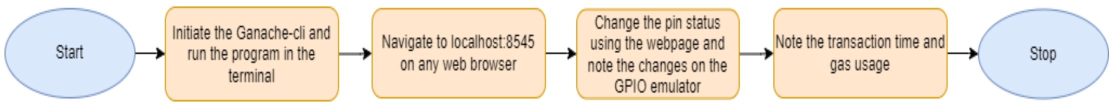
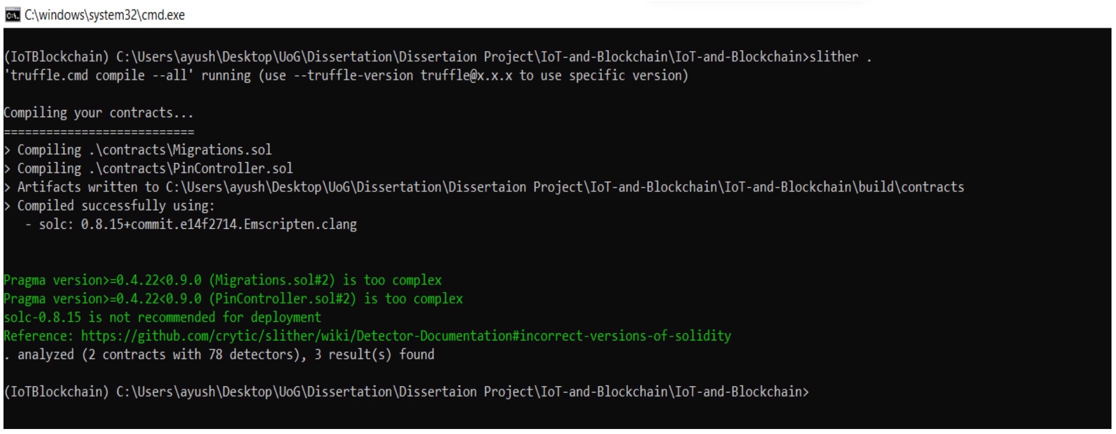

Start
Initiate the Ganache-cli and run the program in the terminal
Navigate to localhost:8545 on any web browser
Change the pin status using the webpage and note the changes on the GPIO emulator
Note the transaction time and gas usage
Stop

Figure 10: Program Flow

4.2 Security Evaluation

We used a security audit tool called Slither to assess the smart contract developed to control the gpio pin status. Slither is a security tool that identifies the vulnerabilities in smart contracts and uses security analysis techniques like taint analysis and symbolic execution to detect various vulnerabilities. This tool can detect security susceptibilities in smart contracts for Ethereum and various other blockchains and provides an extensive list of errors in the documentation. Figure 11 shows the screenshot of the report with only three informational vulnerabilities in the smart contract. The incorrect solidity version error is because the latest stable version of Solidity for deployment to the production server is still 0.8.7, and we have used the newest version for development.

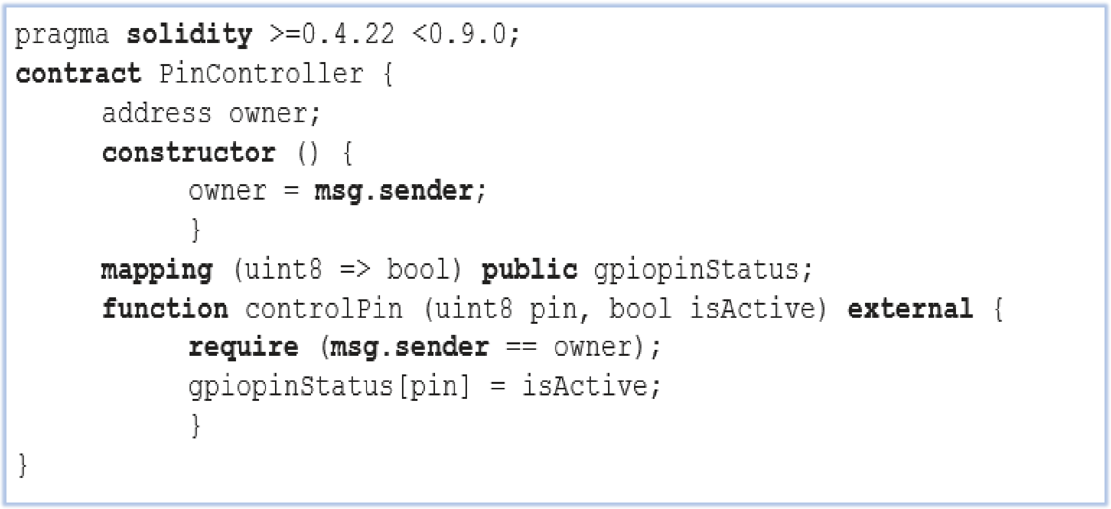


```
(IoTBlockchain) C:\Users\ayush\Desktop\UoG\Dissertation\Dissertaion Project\IoT-and-Blockchain\IoT-and-Blockchain>slither .
'truffle.cmd compile --all' running (use --truffle-version truffle@x.x.x to use specific version)

Compiling your contracts...
===========================
> Compiling .\contracts\Migrations.sol
> Compiling .\contracts\PinController.sol
> Artifacts written to C:\Users\ayush\Desktop\UoG\Dissertation\Dissertaion Project\IoT-and-Blockchain\IoT-and-Blockchain\build\contracts
> Compiled successfully using:
   - solc: 0.8.15+commit.e14f2714.Emscripten.clang


Pragma version>=0.4.22<0.9.0 (Migrations.sol#2) is too complex
Pragma version>=0.4.22<0.9.0 (PinController.sol#2) is too complex
solc-0.8.15 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
. analyzed (2 contracts with 78 detectors), 3 result(s) found

(IoTBlockchain) C:\Users\ayush\Desktop\UoG\Dissertation\Dissertaion Project\IoT-and-Blockchain\IoT-and-Blockchain>
```

Figure 11: Slither Security Analysis Report

We also assessed the security of the Ethereum blockchain-enabled IoT network on the possible attacks on the smart contracts and network as discussed in section 1.1.1:

DoS Attack – A transaction fails to submit if the user invokes multiple transactions to exceed the gas limit of a block, and as a result, the transaction gets rejected. This can be achieved by exploiting the smart contract code and invoking the *controlPin* function available in the contract code. Due to the simplicity of the smart contract, multiple function calls in a short period are processed instantly and avoid the DoS attack. However, Ochôa et al. [27] stated that "the complexity of the algorithm directly influences this security issue." The complexity of the smart contract function is classified as O(1), which makes the attack impossible here. Table 2 gives the probability of the DoS attack with the complexity of the algorithm/code.



```
pragma solidity >=0.4.22 <0.9.0;
contract PinController {
      address owner;
      constructor () {
            owner = msg.sender;
            }
      mapping (uint8 => bool) public gpiopinStatus;
      function controlPin (uint8 pin, bool isActive) external {
            require (msg.sender == owner);
            gpiopinStatus[pin] = isActive;
            }
}
```

Figure 12: Smart Contract Function of O(1)

Table 2: DoS probability in relation to contract complexity [27].

Complexity	Probability of DoS
O(1)	Impossible
O(n)	High
O(n^2)	Extremely high
O(2^n)	Extremely high
O(n!)	Extremely high

We performed a DOS attack on the blockchain-based IoT application simulation and found that with the increase in the attack rate, the transaction loss percentage increased rapidly after 200 transactions. The initial loss rate remained at 0% for 20 transactions and rose to 8% for 50 transactions which later decreased to 4% at 100 transactions. The results for the transaction loss due to a DOS attack are in figure 13.

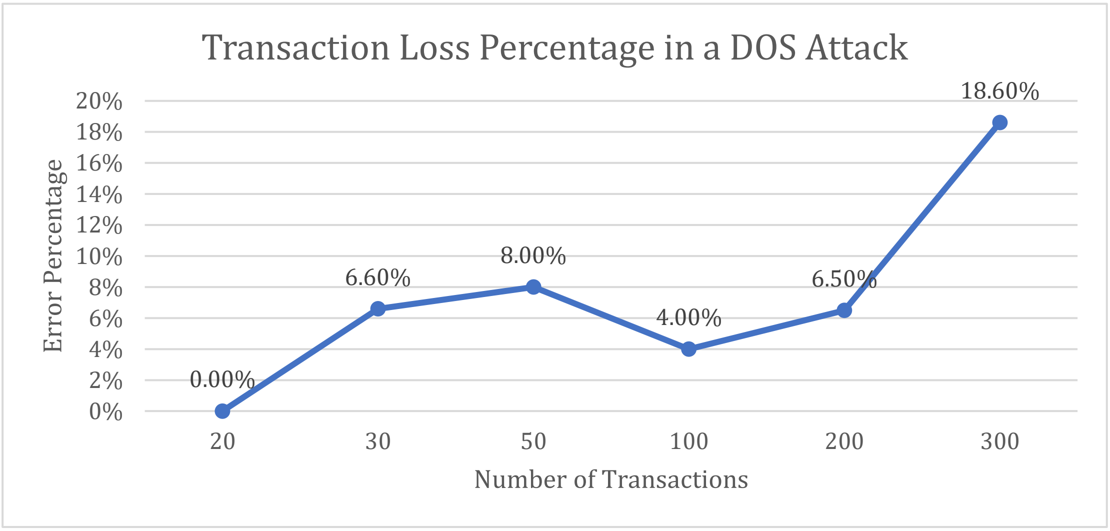


Figure 13: Transaction Loss Percentage in a DOS attack

The loss percentage increase indicates that with the rise in the complexity of the smart contract function, the attack's success rate will increase rapidly. However, sending malicious transactions to disrupt the network is very costly as every transaction requires a base gas price and bounty or tip for miners to mine the transaction. This extra cost for mining each transaction reduces the chances of a DOS attack at such a large scale to cause any disruption with a smaller number of transactions.

Re-entrancy Attack – Repeated calls to the smart contract function from multiple users can be fatal for the smart contract. The attacker can exploit the state where the smart contract fails to update the state of the chain before executing the transaction completion notification and hence, can have an inconsistency.

In this test scenario, we are testing the flow of multiple function calls to a smart contract function through n number of users can generate inconsistency in the output. As can be seen in Table 3, it was found that there was no inconsistency in the execution of the *controlPin* function (figure 12), and Figure 14 depicts the scenario of the testing of the re-entrancy attack. The attack was not successful due to the simplicity of the function of the smart contract.

Table 3: Re-entrancy attack result with varying nodes

Number of Nodes	Expected Result	Final Result	Inaccuracy
1	OFF	OFF	0%
3	ON	ON	0%
5	OFF	OFF	0%
10	ON	ON	0%
20	OFF	OFF	0%
30	ON	ON	0%
50	OFF	OFF	0%

The results from table 3 indicate that no inaccuracy was observed even with a higher attack rate. The program kept performing as it should, and the smart contract developed was not affected by the re-entrancy attack and produced 0% inaccuracy at 50 nodes or users.

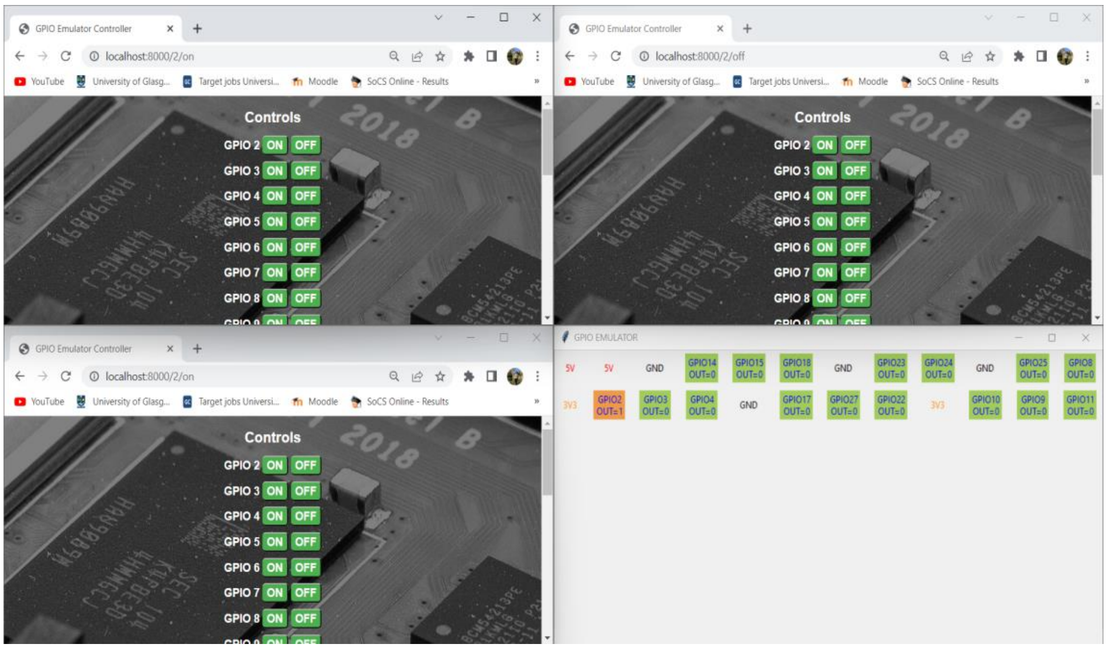

GPIO Emulator Controller
localhost:8000/2/on
localhost:8000/2/off
Controls
GPIO 2 ON OFF
GPIO 3 ON OFF
GPIO 4 ON OFF
GPIO 5 ON OFF
GPIO 6 ON OFF
GPIO 7 ON OFF
GPIO 8 ON OFF
GPIO EMULATOR
5V 5V GND GPIO14 OUT=0 GPIO15 OUT=0 GPIO18 OUT=0 GND GPIO23 OUT=0 GPIO24 OUT=0 GND GPIO25 OUT=0 GPIO8 OUT=0
3V3 GPIO2 OUT=1 GPIO3 OUT=0 GPIO4 OUT=0 GND GPIO17 OUT=0 GPIO27 OUT=0 GPIO22 OUT=0 3V3 GPIO10 OUT=0 GPIO9 OUT=0 GPIO11 OUT=0

Figure 14: Re-entrancy attack testing

False Data Injection and Malicious Script Attack – Transactions in a blockchain-based IoT network are cryptographically hashed and timestamped, making them immutable. With the smart contract application, a self-executing program deployed securely in the blockchain, the attacker cannot make any changes in the data or the code of the smart contract sent over the network and avoids any chances of compromising the integrity of the transaction.

Blockchain uses 160 bits hash value to store the address, which is generated using the public key and needs a private key for decryption. The total address space, therefore, equals 2^{160}, which is nearly $1.47*10^{48}$. So, in IoT applications, each device could be assigned a unique address accessible globally with an address collision probability of 1 in 10^{48}. This makes the system secure to communicate and use secure unique identification to identify the nodes for data exchange. Table 3 shows a few risks that can be resolved by blockchain implementation.

Table 4: Decentralised IoT network potential solution of some security risk

Attack	Potential Solution
Man-in-the-Middle	The data packets or transactions in a blockchain are hashed using pair of public and private keys, which makes it very hard for the attacker to gain any insight.
Unauthorised Access	Implementing simple, smart contracts to execute pre-defined tasks in managing IoT devices makes it impossible for the attacker to exploit the network to gain unauthorised access to the system.
Spoofing	The transactions should be mined to become a legitimate node in the IoT network. They must reach the 51% consent rate, which makes it very expensive for an attacker and outweighs the benefits of the attack as the attacker will have to overtake and change the transaction values to act like a legitimate node.

4.3 Performance Evaluation

The performance evaluation is based on metrics of gas usage per transaction, the time taken to complete the transactions and GPU usage per transaction. The results from the gas and GPU usage are in Table 4. The simulation assumes that the network latency is minimal (<1ms), and transaction time depends on the block size and is carried out instantaneously.

The results show that while the gas usage remains the same for the transactions where the pin status changes to "ON" irrespective of the number of transactions but the gas usage average value increases with the number of transactions overall with the pin status change to "OFF" starts consuming more gas.

Table 5: Performance metrics of blockchain-based IoT environment

Number of Transactions	Gas Usage per Transaction (average)		GPU Usage (per cent)
	ON	OFF	
1	43454	14421	4
5	43454	14421	4
10	43454	14421	4
20	43454	16123	5
50	43454	16332	6
100	43454	16335	8

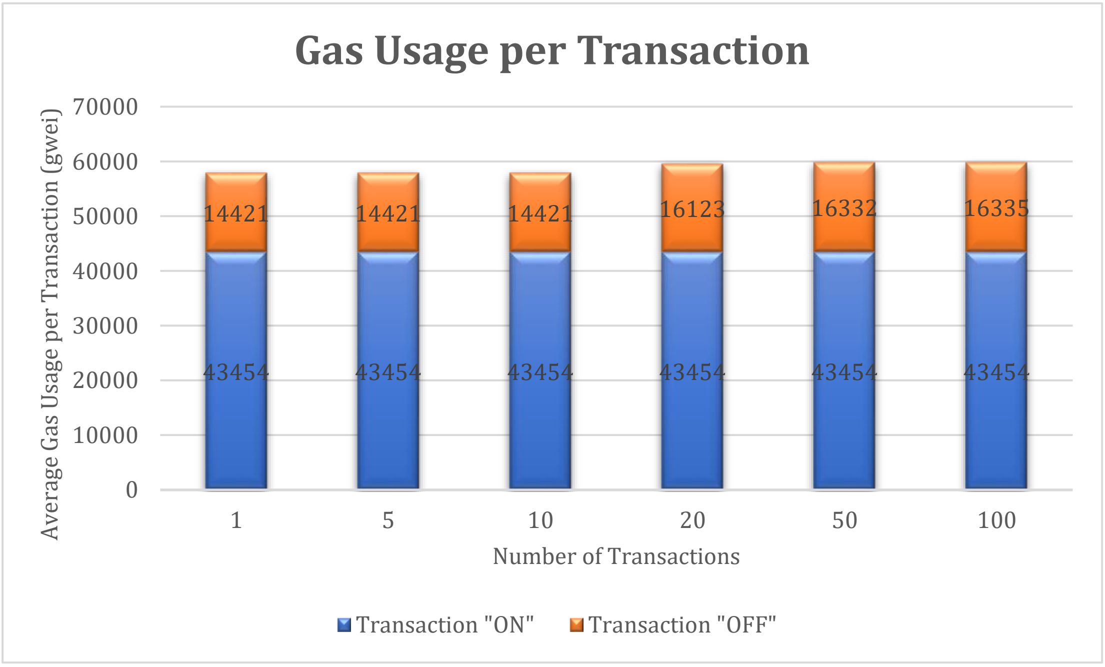
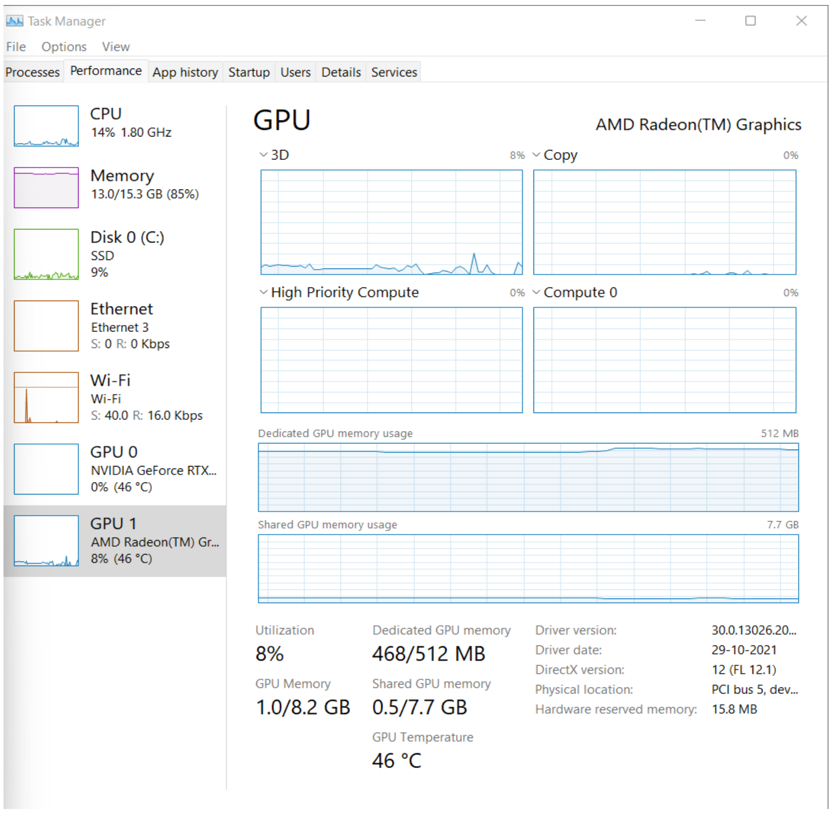


Figure 15: Gas consumed (gwei) per transaction



Task Manager
File Options View
Processes Performance App history Startup Users Details Services

CPU
14% 1.80 GHz

Memory
13.0/15.3 GB (85%)

Disk 0 (C:)
SSD
9%

Ethernet
Ethernet 3
S: 0 R: 0 Kbps

Wi-Fi
Wi-Fi
S: 40.0 R: 16.0 Kbps

GPU 0
NVIDIA GeForce RTX...
0% (46 °C)

GPU 1
AMD Radeon(TM) Gr...
8% (46 °C)

GPU AMD Radeon(TM) Graphics

3D 8% Copy 0%

High Priority Compute 0% Compute 0 0%

Dedicated GPU memory usage 512 MB

Shared GPU memory usage 7.7 GB

Utilization
8%

Dedicated GPU memory
468/512 MB

GPU Memory
1.0/8.2 GB

Shared GPU memory
0.5/7.7 GB

GPU Temperature
46 °C

Driver version: 30.0.13026.20...
Driver date: 29-10-2021
DirectX version: 12 (FL 12.1)
Physical location: PCI bus 5, dev...
Hardware reserved memory: 15.8 MB

Figure 16: GPU specifications

The block size is kept the same in the simulations, and we can observe a slight increase in gas consumption with the rise in the number of blocks in the blockchain. Figure 16 depicts the specifications of the GPU used to host the simulations. The GPU usage follows a similar trend and increases with the number of transactions sent over the network, assuming there is no latency in the network. The increase in block size will mean an increase in the network latency [12] of the system, and as a result, gas consumption and GPU usage will also increase.

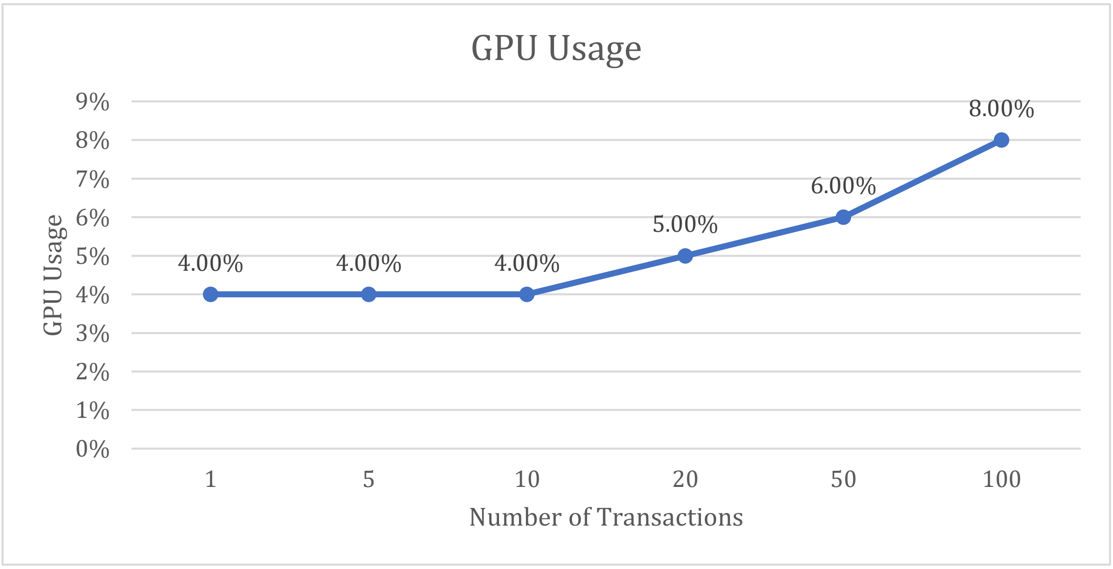


Figure 17: GPU usage per transaction

Smart contracts deployed over the blockchain also consumed gas. They were observed to consume the same amount of gas irrespective of the number of smart contracts deployed on the blockchain, as observed in Table 5, and the increase is almost linear, as seen in figure 18.

Table 6: Smart Contracts and gas consumption

Number of Smart Contracts	Gas Consumption per Contract (gwei)
1	153,130
5	765,650
10	1,531,300
20	3,062,600

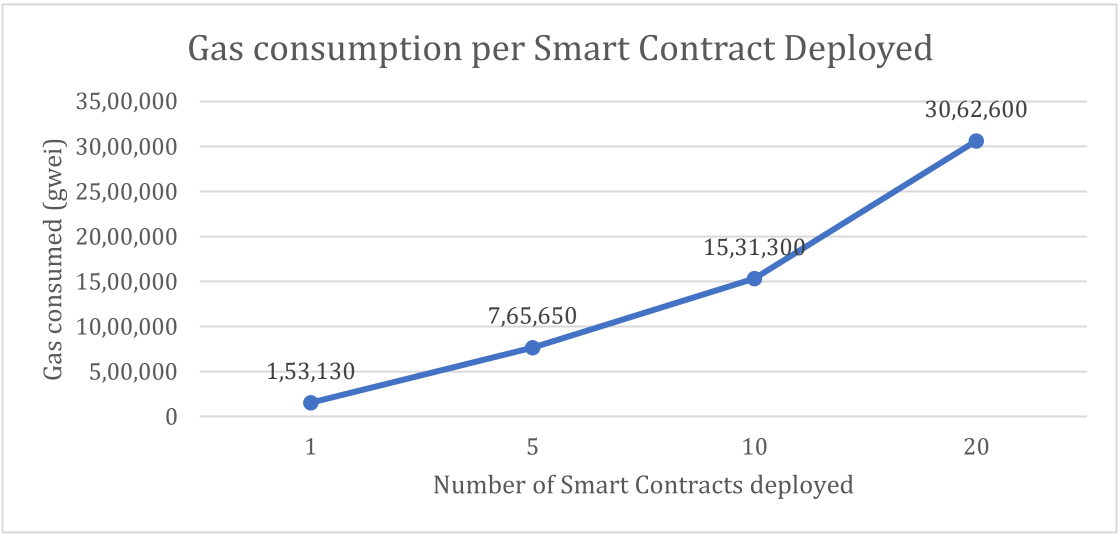


Figure 18: Gas consumption per Smart Contract deployed

During the execution of the transaction, the time consumed is proportional to the block size and latency in the network. Transactions in the simulation are executed instantaneously and face minimal latency, so we tested the time consumed to complete a transaction with one smart contract and multiple smart contracts in the blockchain to test the performance by introducing load in the system. The results in figure 19 indicate that the number of smart contracts deployed in the blockchain has minimal impact on the transaction execution time.

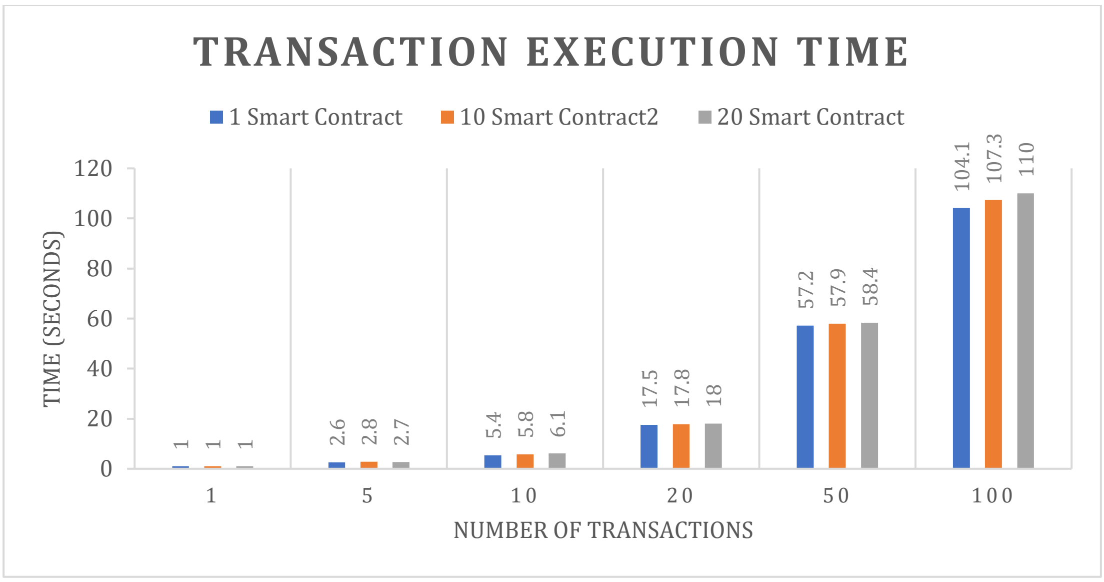


Figure 19: Transaction execution time with different numbers of smart contracts deployed

The smart contract deployment in the IoT environment has minimal impact on the execution of the transactions. Hence, we can use multiple simple smart contracts to carry out complex IoT functionalities.

Chapter 5 Discussion

This section will discuss the results obtained for the security and performance evaluation of the blockchain-based IoT device simulation.

With the implementation of the distributed architecture of the Ethereum blockchain and the smart contract application to manage the IoT device simulation, device security has increased rapidly. Tables 2, 3, and figure 13 show the attack rate and the impact of the attacks on the simulation. Significant security risks are neutralised or managed because of the distributed architecture of the blockchain in the simulation. Similarly, the simple smart contract to manage the IoT device simulation also helps increase security measures and works against attacks like the malicious script and re-entrancy attacks. Blockchain uses the pair of a public and private key to hash the 160-bit address to store; this makes it immune to brute force attacks and makes it hard for the attackers to cause a collision as it allows each device a unique address.

The simulation works well against DOS attacks, we performed DOS attacks with no transaction loss for 20 transactions, and the project kept working well till the attack rate reached 300 transactions, where the loss rate reached 18.6% (Table 7). The application will perform significantly well in a public blockchain where the miner nodes are greater in number providing higher computational powers. As a result, the transaction loss rate will dramatically decrease. Each transaction in the blockchain also consumes ETH, and this also deters attackers from causing attacks like denial of service or re-routing attacks where the attacker needs to create many transactions.

Table 7: Transaction Loss in a DOS attack

Number of Transactions	Transactions Lost (Percentage)
20	0%
30	6.6%
50	8%
100	4%
200	6.5%
300	18.6%

With the implementation of the security measures in the resource constraint, IoT device is not a simple feat and impacts the devices' performance. The network latency is a function of the block size, and the block size will increase with more complex functionalities. Hence, the network latency will also increase, taking more time to complete a transaction which is not ideal for an IoT device. Ideally, it takes

14 seconds in the Ethereum blockchain [29] to deploy a block in the blockchain, which means the transaction completion time will increase, impacting the system's overall performance. The transaction delay is not ideal and not economical for any IoT ecosystem that performs many actions every second.

The deployment of several smart contracts in the blockchain had minimal effect on the application's performance, indicating that we can create many smart contracts in the ecosystem to perform separate tasks for the IoT devices. The transaction execution time exceeded the 100-second mark for 100 transactions without introducing any delay in the simulation. This value will be much more significant in a real-life scenario as one block needs 14 seconds to be mined in the blockchain. Hence, deploying a private blockchain will be more suitable for the practical application of the blockchain in the IoT ecosystem. The GPU usage remains constant and only increases slightly with the number of transactions. However, Ethereum's mining algorithm is more suitable for dedicated hardware platforms and is not optimal for resource-constrained IoT devices. Hence, blockchains must operate the mining operations elsewhere, and only light transaction verification functionalities should be deployed on resource-constrained platforms.

Chapter 6 Conclusion

IoT devices and networks are resource-constrained and lack high-performing security protocols but cover billions of devices over the web, introducing scalability issues. The project discusses the development and evaluation of a solution based on blockchain to tackle the security issues in an IoT network while keeping the IoT devices' performance in mind. The application of blockchain in the IoT device ecosystem is not new. Still, more discussion is required to understand such security measures' impact on performance and security efficiency, which were the research questions discussed in the project.

Implementing the distributed architecture of blockchain and the application of smart contracts in controlling IoT devices has significantly improved security measures. However, it has an impact on the performance of IoT devices. Blockchain-based IoT devices' performance is subject to many parameters and requirements. Still, the results indicate that implementing a distributed architecture in a resource-constrained IoT ecosystem increases the security of the ecosystem manifolds. Evaluating performance suggests a stable performance and effective transaction execution but requires further work to implement this solution in a real-life setting. The performance can be improved by applying a private blockchain with different consensus algorithms, which are more economical and suitable for quick transaction mining as future work.

6.1 Future Work

As part of future work, we plan to implement real-life network loads and compare various consensus algorithms with each other. This research direction will help generate a comparative analysis toward a more feasible blockchain solution in IoT devices, tackling the issue of interoperability and scalability. Integrating decentralised authentication systems such as Decauth [28] or Bubbles of Trust [29] is also a step towards a higher level of security. Also, more resource-constrained and hardware IoT devices such as raspberry pi or ESP32 microcontrollers can be implemented to investigate the problems that may cause issues in the integration of blockchain and IoT.

Chapter 7 References

[1] Mohanta BK, Jena D, Ramasubbareddy S, Daneshmand M, Gandomi AH. Addressing security and privacy issues of IoT using blockchain technology. IEEE Internet of Things Journal. 2020 Jul 13;8(2):881-8.

[2] Rehman M, Javaid N, Awais M, Imran M, Naseer N. Cloud based secure service providing for IoTs using blockchain. In2019 IEEE Global Communications Conference (GLOBECOM) 2019 Dec 9 (pp. 1-7). IEEE.

[3] Han H, Fei S, Yan Z, Zhou X. A survey on blockchain-based integrity auditing for cloud data. Digital Communications and Networks. 2022 May 5.

[4] Zheng J, Dike C, Pancari S, Wang Y, Giakos GC, Elmannai W, Wei B. An In-Depth Review on Blockchain Simulators for IoT Environments. Future Internet. 2022 Jun 10;14(6):182.

[5] Alkhateeb A, Catal C, Kar G, Mishra A. Hybrid blockchain platforms for the internet of things (IoT): A systematic literature review. Sensors. 2022 Feb 9;22(4):1304.

[6] Murthy CV, Shri ML, Kadry S, Lim S. Blockchain based cloud computing: Architecture and research challenges. IEEE Access. 2020 Nov 9;8:205190-205.

[7] Ahmed AH, Omar NM, Ibrahim HM. Performance Evaluation of a Secured Framework for IoT Based on BlockChain. J. Commun.. 2022 Jan;17(1):1-0.

[8] Kreku J, Vallivaara VA, Halunen K, Suomalainen J, Ramachandran M, Muñoz V, Kantere V, Wills G, Walters R. Evaluating the Efficiency of Blockchains in IoT with Simulations. IoTBDS. 2017 Apr 24;820:216-23.

[9] Okegbile SD, Cai J, Alfa AS. Performance analysis of blockchain-enabled data sharing scheme in cloud-edge computing-based IoT networks. IEEE Internet of Things Journal. 2022 Jun 9.

[10] Desai S. Measuring Performance of Blockchain Enabled IoT over Edge Computing System. International Journal for Research in Applied Science and Engineering Technology. 2020;8(7):175-182.

[11] Wang Z, Dong X, Li Y, Fang L, Chen P. Iot security model and performance evaluation: A blockchain approach. In2018 international conference on network infrastructure and digital content (ic-nidc) 2018 Aug 22 (pp. 260-264). IEEE.

[12] Alrubei S, Rigelsford J, Willis C, Ball E. Ethereum Blockchain for Securing the Internet of Things: Practical Implementation and Performance Evaluation. In2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security) 2019 Jun 3 (pp. 1-5). IEEE.

[13] Hang L, Kim DH. Design and implementation of an integrated iot blockchain platform for sensing data integrity. Sensors. 2019 May 14;19(10):2228.

[14] Huh S, Cho S, Kim S. Managing IoT devices using blockchain platform. In2017 19th international conference on advanced communication technology (ICACT) 2017 Feb 19 (pp. 464-467). IEEE.

[15] Koohang A, Sargent CS, Nord JH, Paliszkiewicz J. Internet of Things (IoT): From awareness to continued use. International Journal of Information Management. 2022 Feb 1;62:102442.

[16] Avci İ, Dakhil YH. Survey About Green IoT: Applications, Technologies, Challenges, and Future Directions. In2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA) 2022 Jun 9 (pp. 1-7). IEEE.

[17] Vangala A, Sutrala AK, Das AK, Jo M. Smart contract-based blockchain-envisioned authentication scheme for smart farming. IEEE Internet of Things Journal. 2021 Jan 11;8(13):10792-806.

[18] IoT connected devices worldwide 2019-2030 | Statista [Internet]. Statista. 2022 [cited 24 August 2022]. Available from: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/

[19] Aste T, Tasca P, Di Matteo T. Blockchain technologies: The foreseeable impact on society and industry.

[20] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system. Decentralized Business Review. 2008 Oct 31:21260.

[21] Monrat AA, Schelén O, Andersson K. A survey of blockchain from the perspectives of applications, challenges, and opportunities. IEEE Access. 2019 Aug 19;7:117134-51.

[22] Panda SS, Mohanta BK, Satapathy U, Jena D, Gountia D, Patra TK. Study of blockchain based decentralized consensus algorithms. InTENCON 2019-2019 IEEE Region 10 Conference (TENCON) 2019 Oct 17 (pp. 908-913). IEEE.

[23] Ali MS, Vecchio M, Pincheira M, Dolui K, Antonelli F, Rehmani MH. Applications of blockchains in the Internet of Things: A comprehensive survey. IEEE Communications Surveys & Tutorials. 2018 Dec 18;21(2):1676-717.

[24] Szabo N. Formalizing and securing relationships on public networks. First monday. 1997 Sep 1.

[25] Fairfield JA. Smart contracts, Bitcoin bots, and consumer protection. Wash. & Lee L. Rev. Online. 2014;71:35.

[26] Baliga A. Understanding blockchain consensus models. Persistent. 2017 Apr 4;4(1):14.

[27] Sestrem Ochôa I, Reis Quietinho Leithardt V, Calbusch L, De Paz Santana JF, Delcio Parreira W, Oriel Seman L, Albenes Zeferino C. Performance and Security Evaluation on a Blockchain Architecture for License Plate Recognition Systems. Applied Sciences. 2021 Jan 29;11(3):1255.

[28] Mohanta BK, Sahoo A, Patel S, Panda SS, Jena D, Gountia D. Decauth: Decentralized authentication scheme for iot device using ethereum blockchain. InTENCON 2019-2019 IEEE Region 10 Conference (TENCON) 2019 Oct 17 (pp. 558-563). IEEE.

[29] Hammi MT, Hammi B, Bellot P, Serhrouchni A. Bubbles of Trust: A decentralized blockchain-based authentication system for IoT. Computers & Security. 2018 Sep 1;78:126-42.

[30] Christidis K, Devetsikiotis M. Blockchains and smart contracts for the internet of things. Ieee Access. 2016 May 10;4:2292-303.

[31] Intro to Ethereum | ethereum.org [Internet]. ethereum.org. 2022. Available from: https://ethereum.org/en/developers/docs/intro-to-ethereum/

[32] Proof-of-work (PoW) | ethereum.org [Internet]. ethereum.org. 2022. Available from: https://ethereum.org/en/developers/docs/consensus-mechanisms/pow/

[33] Ethereum development documentation | ethereum.org [Internet]. ethereum.org. 2022. Available from: https://ethereum.org/en/developers/docs/

[34] Yi, X., Yang, X., Kelarev, A., Lam, K.Y., Tari, Z. (2022). Bitcoin, Ethereum, Smart Contracts and Blockchain Types. In: Blockchain Foundations and Applications. SpringerBriefs in Applied Sciences and Technology. Springer, Cham. https://doi.org/10.1007/978-3-031-09670-9_2