2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IoTSMS 2019)

Blockchain Enabled Rooms Implementation For Internet of Things

Joice Joseph, Dr Keivan Navaie
Department of Science and Technology
Lancaster University
Lancaster, United Kingdom
Email: {K.Navaie, Josephj1}@lancaster.ac.uk

***Abstract*—The number of IoT devices is growing at an exponential rate. It is expected that by 2020, there will be approximately 30 billion internet-connected devices and 500 billion by 2030. Not only, does it increase security concerns but will give rise to interoperability issues. In this paper, the recently introduced Ethereum network will be utilised with respect to Internet of Things to create an infrastructure compatible with IoT devices. With an addition of an automated, immutable smart contract that will aid the interoperability of various devices through the heterogenous-friendly Ethereum network. The true potential of Ethereum, when combined with IoT, will be explored and demonstrated. Demonstrated through an implementation in which power will be provided to a room embedded with various IoT devices upon payment (in the form of rent) into a landlord's smart contract [1].**

Keywords—Blockchain, Smart Contracts, Public Network, Landlord, Ethereum, Proof of Work, Private Network, IoT

I. Introduction

With the rising concerns for security, interoperability, and privacy, there was a dire need for an innovative new technology with the capacity to revolutionise industries and streamline users lives all while providing them with the anonymity they deserve. The concept of blockchain was introduced dating far back as 1976, in which a block chaining technique involving ciphers was proposed- known as cipher block chaining (CBC). In CBC, a small group of bits is converted to hash code, then stored as a 'block' before being assigned a unique key. Other concepts were also proposed, one presented by Wei Dai suggested rewarding nodes for solving computational puzzles. This ultimately formed the framework around the Proof of Work algorithms that exist today [2] [3].

A. Concept

The concept described throughout this paper forms the foundations of a real-life application. Though the model designed is simplistic, the concept it embodies has the potential to form the basis for a practical smart home in the future.

The combination of IoT and blockchain technology as used within this use case can streamline rent payments, control of home appliances and create a private network within each home comprising of various (IoT) appliances, whereby the network admin is the landlord. A smart contract can then be deployed onto this network that monitors and controls the appliances, electrical supply, and water supply subjective to the landlord's conditions. Smart contracts can be programmed to automatically execute routine payments for resources used i.e electricity, water, etc.

B. Blockchain

When stripped down to its core, blockchain consists of a chain of digital signatures. Together, it forms a distributed, immutable ledger with a unique hash assigned to each signature that expands across a vast network of computers. During transaction processing, the hash of – format SHA-256 – the former transaction and its associated public key is automatically signed and timestamped. This information is saved as a block on the end of the chain [4]. Refer to figure 1.

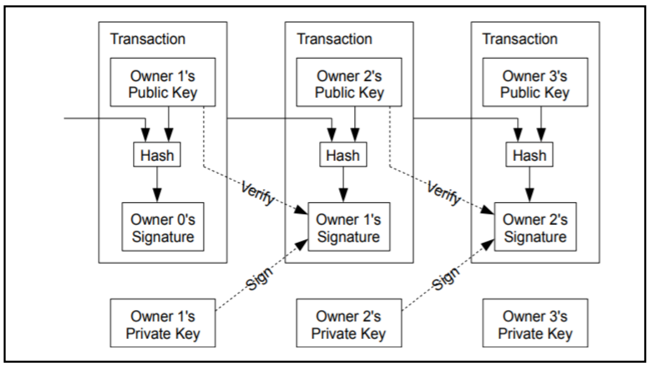


Figure 1 *Hash of previous block points to hash of most recent blocks and is used to sign the most recent transactions (https://bitcoin.org/bitcoin.pdf)*

Authentication of transactions is accomplished through a consensus algorithm named Proof of Work. Here, specialised nodes named miners solve complex computational puzzles using a hash function to complete transactions prior to placing onto the blockchain. This consensus algorithm is currently in use by Ethereum.

In 2014, Ethereum was launched. Providing an open platform on which decentralised applications, known as 'DApps', can be deployed with ease. It utilises the blockchain technology to its full potential. With the addition of smart contracts and Ethereum VM, the capacity of Ethereum had immensely increased. Of the many application of Ethereum, IoT seemed the most promising. The Ethereum enabled smart contract, can automate interactions between IoT devices within a network. For instance, a supply chain can track the condition and journey of their goods using an Ethereum application that communicates with sensors embedded within the packaging.

With the exponentially increasing number of devices and shift towards automating services, the movement to an IoT world has already begun. It is accelerated by the global investments into this field – estimated to exceed $1 trillion by 2020 – whereby the internet no longer exists behind a monitor, but as a separate tangible object serving a very specific purpose and exists as part of a system of systems [4] [5].

C. Aims and Objectives

1. *Conduct research into Ethereum's implementation, tools available for development and capabilities for innovation.*
2. *Understand how Ethereum was derived from blockchain 1.0 and existing deficiencies within Ethereum.*
3. *Understand the functionalities of Raspberry Pi and its suitability with Ethereum.*
4. *Create a functional room model implemented using Raspberry Pi modules, whereby 'tenants' pay into a smart contract using an interface and electricity is provided to the smart room – signalled through activation of IoT devices within the room.*

II. Background

The drastic increase in IoT devices brings forth the rise in security concerns, verifiability, and interoperability problems. Thus, the need for an infrastructure with the capacity to nurture this rise in IoT devices is therefore essential, and blockchain satisfies this condition.

Blockchain introduces a new means for firms to automate procedures between stakeholders such as businesses ordering shipment from a supplier. These are monitored using IoT devices without needing a complicated infrastructure with individuals such as landlords to automate procedures and eliminate the middlemen. Similarly, the data security provided by blockchain will help to cultivate the relationship between organisations and individuals which ultimately increases the efficiency of processes further [6]. IoT devices are inherently insecure, low-cost, often developed with one sole purpose and possess weak computational power with little security safeguard. This makes IoT devices vulnerable to attackers. It is even possible to attack IoT devices through side-channel attacks, in which electromagnetic emanations can be exploited to extract the sensitive data i.e private keys [7] [8].

Due to the distributed nature of blockchain, the network is prone to 51% attacks. As a result, private networks with fewer blocks and small chain are more susceptible to attacks. Similarly, this smart room implementation makes use of a private Ethereum network where nodes are added to the network individually after approval. Due to transaction validation procedures and scalability issues, Ethereum is only able to process approximately 15 transactions per second and thus, will cause queues when many transactions need to be processed. In comparison to Visa, that can process 24,000 transactions/second. However, this issue is avoided in the smart room implementation with the private blockchain network. As there are only a small number of approved nodes, comprising of a thermostat, speakers, and an LED light in each of the 2 rooms. Thus, a transaction speed of 15/seconds offered by Ethereum is adequate for this experiment [10].

A. Related Research Projects

Previous research includes a smart home implemented using blockchain technology, however, the primary focus was on security and privacy issues. The smart home was divided into 3 tiers: cloud storage, overlay, and smart home. Each smart home was provided with an internet device monitoring all messages and transactions passed to/from the home. To support this, a blockchain framework was implemented. Yet to make the framework more flexible to suit the IoT ecosystem, Proof of Work system was removed. Proof of Work is crucial to reaching legitimate, error-free consensus with regards to transactions. Therefore, removing or replacing it with another system will likely create vulnerabilities [11]. In comparison to this Ethereum room model which makes use of "dummy" miners to run a proof of work system, for verifying all transactions before being recorded onto to the private distributed ledger.

IoT devices often have small storage space because their main function is simply to transfer data seamlessly between other devices, which makes blockchain technology difficult to implement. As such, research has been done into creating a blockchain based on 'Hypergraphs', so that the issue of memory and security can be averted (in a smart home setting at least), in which the network itself has been divided into multiple parts (multiple 'Hyperedges') using the proposed theory [12].

B. Ethereum Mining Fundamentals

Each block on the Ethereum blockchain contains 3 vital components: most recent state of the chain, block number, and difficulty value. Fundamentally, the difficulty value refers to the complexity of the computational puzzle miners are required to solve during block creation. Meaning the higher the difficulty, the longer the time taken to create a block and thus more secure transactions. During the initialisation of the private Ethereum network, difficulty value for the genesis block will be intentionally set to a low value of "0x20000" to guarantee that transactions can be completed more promptly [13] [14].

Although high transaction security has been sacrificed, transactions made on the network are still secure due to the authorisation needed to join the network initially. Control over nodes joining the network will fall on the landlord's node i.e the admin. The steps to validating a block during block creation are as follows:

1. *Parent block is proven to exist and is verified.*
2. *Timestamp of current block > timestamp of previous block and < 15 minutes into the future.*
3. *Block number, difficulty, transaction on root, gas limit and uncle root are validated.*
4. *Validate PoW on current block.*
5. *Error is returned if gas limit has been reached.*
6. *Miners are rewarded with Ether after block has been added to the chain.*
7. *If root of the previous block is the same as state root; the block is valid. Otherwise invalid.*

C. Account Creation and Functionality

On Ethereum there are 2 forms accounts: user accounts, each with a unique private key and contract accounts that are managed by code automatically. Both types of accounts have been used within this project, with 7 external accounts each on a separate device. However, 6 of the 7 external accounts are owned by the tenants through which they transfer payments into the contract account, thereby activating the programmed tenancy contract controlling the contract account. Tenants are required to sign off the transaction with their private keys to authenticate the payment.

However, contract accounts are subject to a cost when used, called ‘gas’, as it utilises the computational resources provided by the network. In short, the higher the gas value, the higher the priority given by miners for they receive larger rewards.

Mist wallet was chosen as it offers a highly accessible web interface providing the functionality to create external and contract accounts. Mist can connect to the Ethereum blockchain, enabled through the web3 library embedded in Mist which interacts with nodes by connecting to an HTTP connection. In contrast to creating accounts through the terminal, which many non-tech savvy users will find challenging. The abstraction provided by Mist will increase the efficiency of users.

D. Role of Smart Contracts within Ethereum Network

In this use case, the smart contract outlines the rules and conditions for rent payment between a tenant and landlord, which has been deployed onto the private blockchain accessible by tenants. They will pay into the smart contracts through their Ethereum accounts. Afterward, the contracts will forward the payments directly into the landlords account if pre-defined conditions have been met. Hence, streamlining the payment process without the need for intermediaries e.g. agencies, banks.

Smart contract will need to be validated by miners prior to deployment delaying the release of the contract. An implication that will be dealt with during the creation of the tenancy smart contract. Thus, the higher the gas value set, the quicker the contract will be deployed onto the blockchain. Ethereum together with smart contracts is vastly more capable than many other platforms, e.g. Bitcoin and Ripple because of the turing-completeness, alterable state transitions, blockchain-awareness, and value-awareness [13].

Solidity has a similar syntax to JavaScript; therefore, programmers will be able to become proficient in solidity with ease. During the creation of the tenancy smart contract, the Remix IDE will be used to develop the code. The tenancy contract can be compiled into bytecode then executed on the public, Testnet, or private Ethereum network. Using the contract address and JSON code, others i.e tenants are then able to execute payments into the same tenancy contract from other IoT nodes [15].

E. Consensus Algorithms

Ethereum makes use of hashing procedure, in which the difficulty value, is dynamic and constantly increasing the limit to govern the rate at which Ethereum blocks are mined. If there is a larger number of miners in the network mining a block, the difficulty value is decreased, thus lowering the chances of a miner discovering a valid block hash and vice versa [16].

Consensus algorithms are not limited to Proof of Work: Proof of Stake, Delegated Proof of Stake, Proof of Authority, Proof of Weight and Byzantine Fault Tolerance. Ethereum is moving from PoW to the more efficient and economical Proof of Stake. PoW was found to be very computationally intensive (pollution) and provided a slow throughput.

Proof of Stake will increase the cost incurred during attacks and is less computationally intensive. With Proof of Stake, blocks are generated through miners as they wager on which block hashes are approved. But in the future, Ethereum will not require miners, as the transactions will be validated and protected from interference by the token owners. However, this transition can create an opportunity for unforeseen, critical issues to rise within the Ethereum system architecture [9] [17].

However, the Smart room model makes use of a permissioned Ethereum blockchain and so the PoW process will be made redundant considering the private blockchain can only be accessed by the approved nodes. Consequently, the reliability of the Raspberry Pi-IoT nodes in each of the 2 rooms will dictate the security of the private blockchain [18].

F. Main Network, Testnets and Private Network

There are 3 main types of networks provided by Ethereum: main network, test networks, and private networks. Transactions that occur on the main network are validated by authentic miners. Secondly, there exists the test network providing a secure network ran by test ether which is paid out when deploying beta applications or smart contracts on the network to test functionality.

On the testnet, it is possible for anyone to interact with a smart contract or distributed application, which may add unnecessary complications for the landlord when creating contracts that cater to specific individuals e.g. a tenancy contract. With private networks, only the approved nodes will be permitted to pass through the built-in access control layer, this is the only gateway into the private network dissimilar to main networks or testnets with no access control procedures.

During the development of the tenancy contract, 2 clear options were presented, to deploy the contract onto the Testnet (e.g. Rinkeby) or to create a private net. Although the Testnet supplied higher security, via the PoW algorithm. The reliability of the rent payments made on the private network are directly influenced by approved IoT nodes participating in the network who validate transactions but as there are only 3 users within the network (1 landlord and 2 room tenants), the identity of existing nodes will be known and consequently, nodes are expected to be reliable.

G. Raspberry Pi for IoT Implementation

IoT devices can be embedded into certain objects, forming a “smart object”, e.g. smartwatch or into the environment itself. In the following use case, the approach taken replicates that of a smart environment, in which a room has been integrated with small Raspberry Pi module(s) controlling the electricity input into the room.

Access to information from the physical world is made possible through services offered, as stated in the article by Stephan Haller: “Resources may offer a service interface directly, or services inside the network act as proxies for the actual resources, possibly providing additional levels of abstraction”. If this smart room model was implemented in the real world, it would follow a model like the one displayed in figure 2 [19].

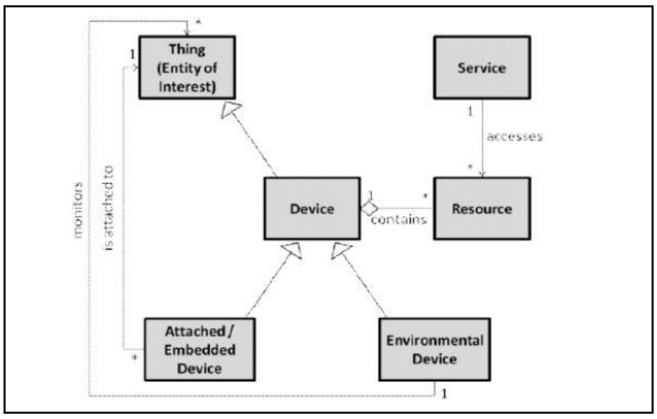


Figure 2 Haller, Stephen "Relationship between things, devices, resources, and services" The Things in the Internet of Things, ResearchGate, January 2010 (https://www.researchgate.net/publication/228488111_The_Things_in_the_Internet_of_Things)

The electricity data collected by the sensor would constitute a "resource" while the IoT device embedded within the room monitoring the electricity usage would be regarded as the 'service' that is being provided. The power supply is controlled by the landlord who will monitor the electricity usage and will have the responsibility of commanding the power supply to provide electricity to the rooms after rent payment.

Raspberry Pi's provides a small and powerful device with an impressive capacity for interoperability which naturally makes it perfect for use with Ethereum (a platform promoting interoperability). In addition, Raspberry Pi fits effortlessly in an IoT ecosystem, with its built-in WiFi connectivity, USB ports, USB interface supporting various devices and simple configuration. As a result, the task of configuring the Raspberry Pi device and connecting to peripherals was simple. Raspberry Pi also supported Geth software transforming the device into an Ethereum node.

III. Design

A. *Private Network Constraints*

By utilising a private blockchain network, the Raspberry Pi devices exchange data effectively. As there are only a small number of the IoT nodes in this use case, a private blockchain will aid the landlord's desire to monitor each tenant that interact with the smart contract more accurately, in contrast to the public or test network with 10 million+ nodes. One constraint regarding private networks remains, in which scalability will become a rising concern, as each it will be difficult to authorise every node as the network scales up.

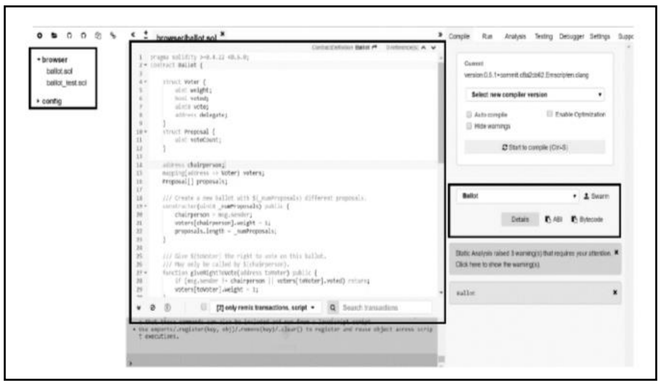
B. *Choice of Tools*

Raspberry Pi provides the option of 2 main operating systems with which the device can be set up: Noobs, Raspbian. Noobs offer a simplistic OS designed for beginners, with a selection menu much like Raspbian, without the need for network access, distinct imaging software's and demands less memory on the SD card [20].

Raspbian OS contains a desktop image and despite consuming more than 4GB in memory. Consequently, it is possible for the Raspberry Pi device to interact with the smart contract deployed on the private network and support the development of python script controlling the connected breadboard [21].

Secondly, Ethereum provided the option to utilise their main network, test networks or initialise a private net. Utilising the main network or test network would cause additional complications i.e unidentified nodes on the blockchain interacting with the smart contract.

Smart contracts can be developed through Remix IDE in this scenario. Remix provided a user-friendly web page with an integrated compiler containing a record of all smart contract creations.



```
pragma solidity >=0.4.22 <0.6.0;
contract Ballot {

    struct Voter {
        uint weight;
        bool voted;
        uint8 vote;
        address delegate;
    }
    struct Proposal {
        uint voteCount;
    }

    address chairperson;
    mapping(address => Voter) voters;
    Proposal[] proposals;

    /// Create a new ballot with $(_numProposals) different proposals.
    constructor(uint8 _numProposals) public {
        chairperson = msg.sender;
        voters[chairperson].weight = 1;
        proposals.length = _numProposals;
    }

    /// Give $(toVoter) the right to vote on this ballot.
    /// May only be called by $(chairperson).
    function giveRightToVote(address toVoter) public {
        if (msg.sender != chairperson || voters[toVoter].voted) return;
        voters[toVoter].weight = 1;
```

Figure 3 Online Remix IDE interface displaying an example smart contract and selection of options to aid the development of smart contract

Upon clicking on the wallet icon, users will be taken to a page providing them with the option to add an existing account or create a new account. The balance of the selected account is displayed on the right-hand side of the page.

Clicking the contract icon will display the page providing users with the option to deploy a new contract to the connected private network. Additionally, the user will be able to monitor previously deployed contracts.

Remix interface displays all existing smart contracts under development, saved on the IDE automatically. To the center of the page, smart contracts can be created and edited.

Errors will be highlighted after compilation. Remix auto generates the Application Binary Interface (ABI) code – which will be essential for the IoT nodes to access the contract on the network. Bytecode is a set of instructions created from the source code such that Ethereum VM can comprehend the code. Contract bytecode will be necessary for deploying a contract. Remix provides an additional array of tools for the development, debugging and testing purposes.

C. *Ethereum- IoT Ecosystem Architecture*

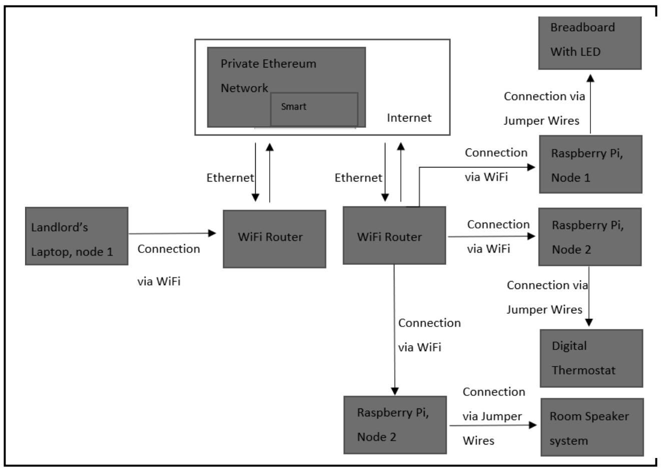


Figure 4 High-level architecture of the blockchain-IoT ecosystem

The above diagram describes the scenario whereby the landlord node sets up a private Ethereum network, hosted over the internet through the WiFi connection to the router. This private Ethereum network hosts an environment where IoT nodes make transactions to the unique smart contract deployed by the landlord. Although the IoT nodes are in a different location, they can gain access to the private Ethereum network. Initialising a geth node with the private network ID and corresponding port number will gain the IoT nodes access to the private network.

Following this, IoT nodes will be able to interact with the contract deployed on the private blockchain. The contract ABI code can be used to search for the contract, parallel with the contract address. After which tenants can send transactions to the contract over the private network. If the conditions have been met, the contract will communicate with the python script running on the IoT nodes – controlling the breadboard LED, room speakers and thermostat to activate. Connections between the breadboard and IoT device is via male to female jumper wires.

Since each of the IoT nodes will be running Geth and the python scripts, the smart contract will be accessible from each IoT node. Therefore, it will be possible to execute payments into the contract from any of the devices e.g. if desired the tenant(s) will be able to make the payments through the thermostat Raspberry pi-IoT. After the payment has been processed and authorised, the landlord can then proceed to provide power to the room(s).

D. *Smart Contract Requirements*

Although the purpose of the smart contract is simplistic in nature, additional functionality has been added to the contract to provide supplementary information to the landlord. The contract will be activating the LED lights for approximately 30 seconds upon correct payment into the contract. The expected payment value of 100 Ether (e.g. £100/week) will be programmed into the contract, such that the incoming payments will have to be compared against the programmed expected payment values, forming a condition that will have to be met for the rent payments to be valid i.e payments exceeding 100 and below 100 will be rejected.

While the condition is checked, the contract will be expected to retain the transferred payments within the contract – transactions will be held in the contract account until further notice. Depending on the outcome of the condition, retained funds will be expected to be either transferred back to the tenant accounts or forwarded to the landlord account (will be programmed into the contract).

E. *Expected Flow of Processes*

1) *Initialise private net through geth.*
2) *Create tenancy contract through Remix.*
3) *Create account through Mist/terminal and deploy.*
4) *Create tenant accounts.*
5) *Add tenant to the private network.*
6) *Access contract by searching contract address.*
7) *Transfer payment into contract.*
8) *Authorise the activation of IoT devices.*
9) *IoT devices are activated.*

IV. Implementation

A. *Determining a Suitable Ethereum Wallet*

Mist was chosen to be the most suitable wallet application for this implementation. Mist provides a platform with a simple GUI supporting smart contract, deployment, account creation and clear records of all transactions to the deployed smart contract. Providing advanced features such as multi-signature accounts and withdrawal limit options, Mist provides a wide array of options that were not possible through the conventional JavaScript command line interface [22].

B. *Smart Contract Generation*

The tenancy contract created for the private network was developed in Ethereum Remix IDE. Through Remix, smart contracts can be compiled and deployed onto the public network, test network that your node is connected to or an external blockchain network. During the design phase, the requirements for the smart contract was outlined. The requirements were as follows:

1. Incoming payments will have to be compared against the programmed expected payment of 100 Ether.

2. Payments exceeding 100 Ether or below 100 Ether will be rejected.

3. The contract will hold the payments within the contract account until transaction payments have been validated.

4. Rejected payments will be sent back to the tenants' accounts with immediate effect.

5. Validated payments will be forwarded from the contract account to the landlord's account address – programmed into the smart contract.

6. Tenancy contract shall activate the LED lights a fixed amount of time before deactivating.

An external payable function will be initialised within the contract thus enabling the tenancy contract to accept payments from external addresses. Absence of this function would mean transactions made to the contract address would bounce. An implemented contract balance function will allow others to check the balance the contract account (including pending transactions).

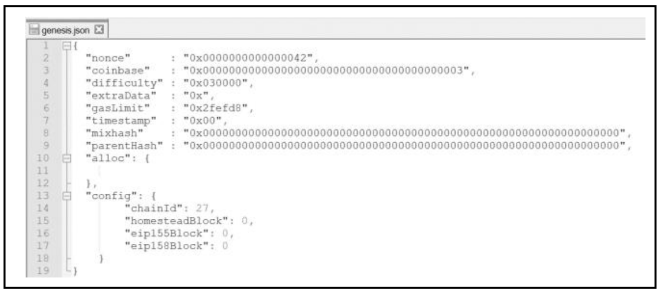
The function to send payments will contain a conditional if-else statement checking whether the transaction made to the contract matches the programmed weekly amount of 100 Ether (dependent on the landlord's terms). If the payment matches pre-programmed amount, payments will be forwarded from the contract address to the landlord's address. Contained within that conditional is a loop that will keep the Breadboard LED, room speakers and thermostat activated for a given amount of time, thus satisfying requirements 1,3, 5 and 6. The else condition will forward the payment back to the sender's address if the payment does not match 100 Ether, thereby meeting requirement 4. Together the if-else conditional satisfies requirement 3.

An additional function 'kill' will be included. If an updated contract is released or the existing tenancy contract is not adequate for one reason or another, the kill function will permanently terminate the contract. After the contract has been developed and tested on Ethereum Remix, the source code can then be copied and pasted into the Mist. The account address and gas limit will be detailed before deploying the contract onto the private net connected to the Mist wallet.

C. *Private Network Initialisation Procedure*

Go Ethereum was essential to the initialisation of an Ethereum network, Geth will transform the PC and IoT devices into an Ethereum node and enabled the nodes to communicate and exchange data within an Ethereum network. As a result, the first steps of the initialisation procedure will be to install Go Ethereum software onto the IoT devices and PC.

The second step of the procedure is to configure a genesis block, achieved by creating a file of type JSON and initialising geth on the terminal specifying the genesis block file. The genesis block will lay the foundations for the private network, outlining the unique parameters of the private network.



```
genesis.json
{
    "nonce"      : "0x0000000000000042",
    "coinbase"   : "0x0000000000000000000000000000000000000003",
    "difficulty" : "0x030000",
    "extraData"  : "0x",
    "gasLimit"   : "0x2fefd8",
    "timestamp"  : "0x00",
    "mixhash"    : "0x0000000000000000000000000000000000000000000000000000000000000000",
    "parentHash" : "0x0000000000000000000000000000000000000000000000000000000000000000",
    "alloc": {

    },
    "config": {
          "chainId": 27,
          "homesteadBlock": 0,
          "eip155Block": 0,
          "eip158Block": 0
      }
}
```

Figure 5 Figure displaying the details of the genesis block and specific parameters set for a private Ethereum network.

After the creation of the genesis block file, the private blockchain will be initialised by running geth and specifying the folder in which genesis file lies, i.e. *>geth –datadir privateBlockchain init genesis.json*. Chain data will be saved into this folder. By doing so, a private Ethereum blockchain can be established from the genesis block. Every block that chains from the genesis block specified will be subject to the parameters set in the genesis block file.

The third step will consist of opening another terminal window and running another instance of geth while setting parameters for the private network: 'port', 'networkid', 'nodiscover', 'datadir' and 'rpcport'. However, certain network ids are reserved for specific networks provided by Ethereum: main Ethereum network – network id 1, Morden public Ethereum test network – network id 2, Ropsten cross-client test network – network id 3, Rinkeby public test network – network id 4.

The IoT nodes will have to be added to the network manually, reducing the possibility of an anonymous node interfering with the actions between the tenant and landlord nodes This is done using the following command: > *geth --port 3000 --networkid 5081 --nodiscover --datadir=./privateBlockchain --rpcport 8757*. A specific data directory is mentioned, into which the blockchain data will be saved. RPC commands permits http-rpc server and specified the port to which the rpc messages will be forwarded to.

D. *Initialising & Adding IoT Nodes to Private Network*

The process of configuring the Raspberry Pi devices will begin with formatting an external SD card of size 4GB on a PC to which the Raspbian OS will be installed from the Raspberry Pi website. Raspberry Pi provided 2 main operating systems for public use: Noobs, and Raspbian. Raspbian was chosen as the most appropriate OS to be installed onto the SD cards. After Raspbian has been downloaded onto the formatted SD cards, they will be inserted into the Raspberry Pi devices.

Once the Raspberry Pi has been booted and internet connection has been established, the Raspbian desktop image will be downloaded together with the installation files and any additional programs that have been specified. Upon successful configuration of the Raspbian desktop, it will then be possible to run the device as an Ethereum node. Therefore, the latest version of geth – arm7 geth 1.8.1.tar package, will be downloaded onto all Raspberry Pi devices through the Raspbian desktop browser. The downloaded geth package will then be untarred using the 'tar -xvf' command and moved into the local /bin folder.

The landlord node is then booted. It will be initialised with the following parameter values explicit to the private network: port number of 3000, network id of 5081, and rpc port number of 8757. From the terminal, the IoT nodes can begin their own geth instance by starting geth and specifying a networkid that matches the geth instance running on the landlord's node. However, each tenant node will be required to specify a unique port and rpc port number. E.g.:

Landlord Node:

> *geth --port 3000 --networkid 5081 --nodiscover --datadir=./privateBlockchain -- rpcport 8757*

IoT Node 1 (Thermostat):

> *geth –port 3012 --networkid 5081 --nodiscover --rpcport 8750 --datadir=./iotBlockchain*

IoT Node 2(Speaker):

> *geth --port 3032 --networkid 5081 --nodiscover --rpcport 8751 --datadir=./iotBlockchain2*

IoT Node 3(LED light):

> *geth --port 3042 --networkid 5081 --nodiscover --rpcport 8752 –datadir=./iotBlockchain3*

By running the following commands on each of the devices appropriately, 3 separate geth instances can be initiated. Each of the nodes can participate in the same network. The first node will be required to copy the encoded URL of its geth instance by using the command 'admin.nodeInfo'. The encoded URL can then be used to establish a connection between the IoT nodes and the landlord node by calling the 'admin.addPeer(*landlord's encode URL*)' command from within the tenant's geth instance. The landlord can check the details of tenants/nodes that are connected to the private network using the command 'admin.peers' [23].

V. Proposed System in Operation

A. *Robustness of Ethereum Network*

The durability provided by Ethereum is reliant on the number of nodes partaking in the network. Due to the distributed nature of Ethereum, there is no single point of failure and additionally, any effort to alter an existing block on the blockchain network would cause all following blocks to become invalid as each block points to the hash of the previous block.

In a real-world application, to alter a block on the blockchain network, miners/participants will need to control more than 50% of the networks mining and computational power – 51% attack. Only then will malicious miners be able to manipulate the transactions and confirmation procedures, e.g. make double spending possible. However, with a globally distributed network such as Ethereum, this is extremely improbable. It is also possible to attempt to manipulate without possessing more than 50% of the mining hash rate, though it is computationally impractical, the cost incurred with mining would far exceed the gains made from manipulating the transaction [24] [25].

The private network used within this use scenario only contains 7 nodes: 1 landlord and 6 of the IoT nodes. As the landlord holds full knowledge of the IoT nodes he is responsible for manually adding the nodes to the network, it is unlikely that the network will be compromised even though the reliability of the network is fully dependent on the nodes participating. One of the negatives and admittedly greatest advantages was complete transparency. Sensitive information should not be accessible by any node, but as a private network is used, any information shared within the network is hidden with minute possibility of leakage and only accessible to connected nodes, adding to its reliability.

B. *Blockchain-IoT Ecosystem Features*

By using Mist as the chosen wallet for transactions, all users of the wallet can benefit from the additional features provided. Aside from the wallet functions provided by Mist, it enables the tenants to create smart contracts and deploy it to another Ethereum network if necessary, by utilising the built-in workspace and configuration screens that have been designed to appeal to non-tech savvy individuals. Thus, it is well suited to the tenants of a housing agency. Tenants are also able to create pools of money, like that of crowdfunding which may have tremendously practical applications in the case of rent payments.

Hypothetically, tenants of a shared household will be able to generate pools of money – such that each tenant will contribute an equal amount of money towards their rent before transferring the money as a lump sum to the landlord/agency at an agreed time. By doing so, the rent payments procedure can be streamlined as the time and effort taken to pursue each individual tenant regarding weekly/monthly payments can be avoided. Another feature provided by Mist and one that is not widely appreciated is its ability to allow users to generate a database of information only alterable by select users but accessible by the public. In the real world, this has the potential to streamline communication between landlord and tenant among many other applications. The landlord can create a database (only alterable by the landlord) presenting important information or updates which can be viewed by tenants.

However, certain features of Mist are of higher importance in this use case, such as the multi-signature accounts which is extremely beneficial if there are 2 or more landlords who are responsible for the same tenants. Multiple landlords can oversee the same account and manage all transactions associated with the account. Tenants can also view all transactions – past and present presented in a simple format through the Mist interface. Withdrawal/payment limits can also be set on accounts, which will aid with regulating rent payments i.e ensuring that rent payments do not exceed the amount set in the smart contract.

Aside from Mist, the contract itself will provide additional functionality for the tenants. The contract will allow the landlord to check the balance of the contract, giving the landlord a clear indication of the number of rent payments that have been made into the contract. Another function provided by the contract is the kill function.

VI. EVALUATION & CONCLUSION

A. *Review of Original Aims*

Aim 1: In pages 1, 2 and 3 exhaustive research and analysis were carried out into blockchain technology, and in doing so, discovered the tools accessible for building smart contracts and constructing a private blockchain network. The deficiencies that existed in blockchain 1.0 that formed the basis for the development of blockchain 2.0 were identified and studied.

Aim 2: Research on Raspberry Pi operating systems were covered in page 4 in which a comprehensive analysis was carried out into the various aspects of Raspberry Pi and the role it will play within this use case. Applications of Raspberry Pi within an IoT ecosystem was also explored and the packages provided by Raspberry that encourage the use of Raspberry Pi with Go Ethereum.

Aim 3: The smart room model was created by hand after the system was created during the system in operation and testing process – described in chapter 5.

Aim 4: As this is a highly specific use case, there are very few research materials concerning this area. Nonetheless, the focus was on security and risk, therefore the results obtained were not relevant to this project as were the objectives. Comparisons were still made with the methodology used in the related research, but there were no practical uses to be gained from the comparison.

B. *Design & Implementation Revisions*

As Mist is a public software, it has been created to serve a common function – support exchange of transactions and smart contract development. If the house owner required additional functionality, e.g. supporting tenant-landlord communication, Mist's interface will be inadequate. Therefore, a small-scale distributed application (Dapp) with an interface and functionality unique to the landlord's specifications/ tenant needs can be built. This application can utilise the private Ethereum network set up by the landlord and will prove to be more effective.

C. *Future Works*

An Ethereum powered door lock to monitor room entry can be implemented, the conventional door lock will be replaced with a smart lock. The lock will contain an Arduino with network access, connected to the private blockchain network. An electronic card reader will be connected to the Arduino board. Therefore, once the tenant scans their card, the card reader will signal the Arduino to open the electronic lock. Connection to the private Ethereum network will be through the ethernet controller. Every entry and exit will be recorded to the private blockchain. Additionally, each card held by the user will have a unique ID that is known to the landlord, hence if a break-in is attempted, it will be recorded on the blockchain containing a smart contract that can be programmed to automatically contact law enforcement if such an event occurs.

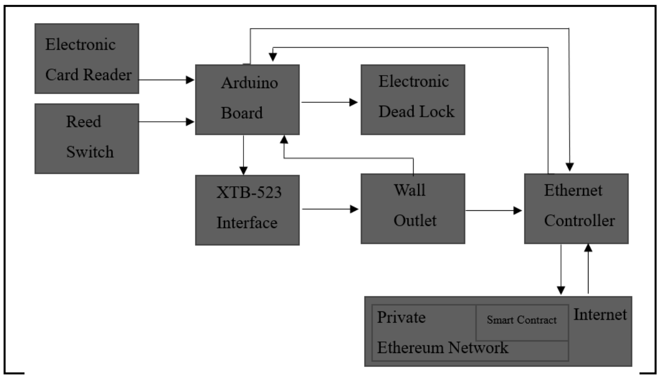


Figure 6 High-level architecture of the smart door lock system

With regards to the future of blockchain and IoT, it is assumed that it will effectively utilise the increasing size of big data. This will streamline the services provided to consumers, as blockchain will act as the platform that enables the interoperability and effective exchange of information between various IoT devices. Yet, there is no tamper-proof glue to bind blockchain technology with many IoT devices currently. The issues that exist within blockchain such as low bandwidth and waste-full consensus algorithms shows that it is not ready for industry adoption. However, as more companies experiment with blockchain and IoT implementations, it will eventually be refined [26].

D. *Conclusion*

The emphasis of this project was on the capabilities of blockchain when combined with IoT. This project presents tangible proof of one of the many applications that a blockchain-IoT system enables. The Ethereum enabled a rent payment network to act as a model of a potential real-life use case with very practical applications as evidenced by this project. As blockchain technology continues to develop, its capabilities and potential applications will grow. The existing weaknesses will eventually diminish and will rival the traditional systems in its efficiency and convenience such that blockchain technology may become the norm. However, there will be risks in moving from the trusted, traditional systems to one that is remarkably different. But there is no innovation without risk.

References

[1] Quora.com. (2018). What are the applications of Ethereum to the Internet of Things? - Quora. [online] Available at: https://www.quora.com/What-are-the-applications-of-Ethereum-to-the-Internet of Things

[2] Smartym - Mobile and Web App Development. (2018). Blockchain practical use cases: Internet of Things, insurance, healthcare, digital rights. [online] Available at: https://smartym.pro/blog/blockchain-practical-use-cases-Internet of Things-insurance-healthcare-digital-rights/

[3] Medium. (2018). Usage of the word "blockchain" – richbodo – Medium [online] Available at: https://medium.com/@richbodo/common-use-of-the-word-blockchain-5b916cecef29

[4] Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. [pdf] Available at: https://bitcoin.org/bitcoin.pdf

[5] Medium. (2019). How Blockchain and Smart Contracts Can Impact Internet of Things – Smartz Platform Blog – Medium. [online] Available at: https://medium.com/smartz-blog/how-blockchain-and-smart-contracts-can-impact-Internet of Things-f9e77ebe02ab

[6] I-scoop.eu. (2019). Blockchain and the Internet of Things: the IoT blockchain picture. [online] Available at: https://www.i-scoop.eu/blockchain-distributed-ledger-technology/blockchain-iot/

[7] Orcutt, M. (2019). How secure is blockchain really?. [online] MIT Technology Review. Available at: https://www.technologyreview.com/s/610836/how-secure-is-blockchain-really/

[8] O'Neill, M. (2016). Insecurity by Design: Today's IoT Device Security Problem. Engineering, 2(1), pp.48-49.

[9] Blockgeeks.com. (2019). [online] Available at: https://blockgeeks.com/guides/proof-of-work-vs-proof-of-stake/

[10] Blocksplain. (2019). Blockchain speeds & the scalability debate | Blocksplain. [online] Available at: https://blocksplain.com/2018/02/28/transaction-speeds/

[11] Dorri, A., Jurdak, R., Gauravaram, P. and Kanhere, S. (2017). Blockchain for IoT Security and Privacy: The Case Study of a Smart Home. [online] ResearchGate. Available at: https://www.researchgate.net/publication/312218574_Blockchain_for_IoT_Security_and_Privacy_The_Case_Study_of_a_Smart_Home

[12] Qu, C., Tao, M. and Yaun, R. (2018). A Hypergraph-Based Blockchain Model and Application in Internet of Things-Enabled Smart Homes. [online] ResearchGate. Available at: https://www.researchgate.net/publication/327217691_A_Hypergraph-Based_Blockchain_Model_and_Application_in_Internet_of_Things-Enabled_Smart_Homes

[13] Buterin, V. (2013). A Next Generation Smart Contract & Decentralized Application Platform. Ethereum White Paper, p.18.

[14] DLTlabs. (2019). How Difficulty Adjustment Algorithm Works in Ethereum - DLTlabs. [online] Available at: https://dltlabs.com/how-difficulty-adjustment-algorithm-works-in-ethereum/

[15] Mathew Kurian, A. (2018). Interacting With Ethereum Smart Contracts Through Web3.js. [online] Medium. Available at: https://medium.com/coinmonks/interacting-with-ethereum-smart-contracts-through-web3-js-e0efad17977

[16] Mycryptopedia. (2019). Ethash Explained - Mycryptopedia. [online] Available at: https://www.mycryptopedia.com/ethash-explained/

[17] Witherspoon, Z. (2019). A Hitchhiker's Guide to Consensus Algorithms – Hacker Noon. [online] Hacker Noon. Available at: https://hackernoon.com/a-hitchhikers-guide-to-consensus-algorithms-d81aae3eb0e3

[18] Thompson, C. (2019). Private Blockchain or Database? – The Blockchain Review – Medium. [online] Medium. Available at: https://medium.com/blockchain-review/private-blockchain-or-database-whats-the-difference-523e7d42edc

[19] Haller, S. (2010). The Things in the Internet of Things. [ebook] ResearchGate, pp.1, 2. Available at: https://www.researchgate.net/publication/228488111_The_Things_in_the_Internet_of_Things

[20] Raspberry Pi. (2019). Introducing the New Out Of Box Software (NOOBS) - Raspberry Pi. [online] Available at: https://www.raspberrypi.org/blog/introducing-noobs/

[21] Raspberry Pi. (2019). Download Raspbian for Raspberry Pi. [online] Available at: https://www.raspberrypi.org/downloads/raspbian/

[22] Hess, T. (2019). What's the difference between Accounts and Wallets in Mist?. [online] Ethereum Stack Exchange. Available at: https://ethereum.stackexchange.com/questions/212/whats-the-difference-between-accounts-and-wallets-in-mist

[23] Ethereum.gitbooks.io. (n.d.). Setting up a cluster | Ethereum Frontier Guide. [online] Available at: https://ethereum.gitbooks.io/frontier-guide/content/cluster.html#

[24] Frankfield, J. (2019). 51% Attack. [online] Investopedia. Available at: https://www.investopedia.com/terms/1/51-attack.asp

[25] Thumar, C. (2018). How to Control Blockchain Durability and Robustness. [online] technology.org. Available at: https://www.technology.org/2018/01/11/how-to-control-blockchain-durability-and-robustness/

[26] Hauser, A. (2018). What is the future of Internet of Things (IOT) - Blockchain? - Dataconomy. [online] Dataconomy. Available at: https://dataconomy.com/2018/10/what-is-the-future-of-internet-of-things-iot-blockchain/