Blockchain for Smart Home Data Privacy: Ethereum and Hyperledger Fabric

MSc Research Project
MSc in Cyber Security

Udith Ragav Saravana Kumar
Student ID: x23140348

School of Computing
National College of Ireland

Supervisor: Kamil Mahajan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Udith Ragav Saravana Kumar		
Student ID:	x23140348		
Programme:	MSc in Cyber Security	**Year:**	2024
Module:	MSc Research Project		
Supervisor:	Kamil Mahajan		
Submission Due Date:	29-01-2025		
Project Title:	Blockchain for Smart Home Data Privacy: Ethereum and Hyperledger Fabric.		
Word Count:	7318	**Page Count**	24

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Udith Ragav Saravana Kumar
Date:	29-01-2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Blockchain for Smart Home Data Privacy: Ethereum and Hyperledger Fabric

Udith Ragav Saravana Kumar
x23140348

Abstract

Smart homes, being one of the commonly adopted technologies in today's world, generate a large amount of sensitive data that requires proper data privacy, security and management. The traditional systems which are currently used for smart home data storage often suffer from various vulnerabilities such as unauthorised access and lack of transparency, which makes blockchain a good alternative for data privacy. This paper evaluates the ability of Ethereum and Hyperledger Fabric for managing smart home data, focusing on privacy, performance and cost efficiency. Using a simulation environment for creating a smart home dataset, which is encrypted and uses IPFS on Ethereum Sepolia test network with Alchemy and Hyperledger Fabric in a two-organization setup within a Windows Subsystem for Linux (WSL) environment. The findings provide critical insights into the trade-offs between performance, privacy, and cost in blockchain-based IoT systems, advancing the development of secure and scalable smart home solutions.

1. Introduction:

Majority of the people in today's world are adopting smart home technology. It is critical to understand if blockchain can potentially enhance the security of the data produced by the devices in a smart home environment. In this paper, we primarily aim to compare and evaluate two blockchain platforms namely Ethereum and Hyperledger Fabric. Despite the fact that people know Ethereum for being a cryptocurrency, its potential is far more since it can provide a solid network for building powerful Decentralized Applications (DApps). Similarly, Hyperledger Fabric is a modular and permissioned blockchain infrastructure that can help with enterprise level applications.

1.1 Blockchain Technology:

Although being a decade old, blockchain technology can address various existing problems associated with the traditional storage systems. It can prevent various illegal activities and can also achieve comprehensive privacy by introducing a decentralized and immutable ledger system that records all the transactions that take place in the distributed network. The process of decentralization eliminates the need for a central authority which directly reduces single points of failure and enhances security and privacy(Nakamoto Satoshi, 2008).

Many developers are finding new ways to implement multiple types of blockchains and networks together. Internet with intelligent applications and interconnected data in a decentralized manner is not so far away. This goal can be achieved using blockchain, which allows secure peer to peer transactions without any central authority or intermediary.

The most common type of blockchain are public blockchains, which is open to anyone to participate and view the transactions taking place in the network like Ethereum. Next are the private blockchains, which are only available to specified participants and has restricted networks and often used within organizations. At last, the permissioned blockchain which combines both the abilities of private and public blockchain, creating a controlled participant access while maintain some degree of transparency. The use case and required balance between privacy and transparency are the key factors which help the developers to choose a specific type of blockchain. The key feature of blockchain which is similar across all types is the distributed ledgers, which make sure that all the nodes in the blockchain network have an identical copy of the ledger, which enhances data integrity and security.

1.2 Ethereum:

Finance is one of the most prominent sectors which use Ethereum. This sector uses this to build decentralized finance (DeFi) applications since it supports smart contracts. Smart contracts are self-executing contracts with terms that are directly integrated into the code. These contracts can execute complex transactions and enforce rules without the need of an intermediary and enables the developers to create better decentralized applications(Buterin, 2013).
Ethereum used to operate on a Proof of Work (PoW) consensus algorithm, but from 2022, it started adopting the Proof of Stake (PoS) model which is because PoW is resource intensive and may not be suitable for IoT applications due to energy consumption issues as well. PoS on the other hand has improved scalability, efficiency and security.

1.3 Hyperledger Fabric:

Being a permissioned blockchain, Hyperledger Fabric is designed for enterprises that require enhanced privacy and performance. The modular architecture allows the organization with full customizations and allow them to plug in their preferred components such as consensus mechanisms. Fabric supports private channels, enabling confidential transactions between specific network participants(*A Blockchain Platform for the Enterprise — Hyperledger Fabric Docs Main Documentation*, n.d.).
Furthermore, Hyperledger Fabric does not rely upon any cryptocurrency, making it much more suitable for applications where tokens are unnecessary like the smart home implementation. Also, the design of this blockchain aligns well with the use cases where access control and data privacy are crucial, making it much more suitable for smart home scenarios.

1.4 Challenges in Blockchain and IoT integration:

Even though blockchain technology has many advantages over the traditional storage methods, it also has certain weaknesses that must be noted. The common attacks on blockchain include attacks like the 51% attack, where a single entity will control the majority of the networks power which will allow them to potentially alter the ledger. When smart contracts are poorly coded, they might act as an attack surface. The eclipse attack in which an isolated node will be able to control all the network traffic, can effectively disrupt the network's consensus. Additionally, when integrating blockchain with IoT devices, majority of the problems arise due to resource constraints. They lack the computational power and often cannot fully participate in the blockchain network. Scalability issues also arise in these environments due to the volume of data produced by these environments. But successful integration of blockchain and IoT devices, specifically smart home devices, will be able to provide various advantages like improved data integrity, decentralized control and powerful security policies.

2. Research Question:

Can Ethereum and Hyperledger Fabric blockchains provide viable solutions for securing data generated by smart homes while ensuring privacy? By implementing the application which can handle smart home data on both of the blockchain platforms, it was clear that public and permissioned blockchains have huge differences and has their own pros and cons when it comes to handling smart home data. Handling various challenges across the implementation, beginning with handling the gas costs in an effective way using testnets in a public blockchain to the time-consuming process of setting up the permissioned blockchain, a deep understanding of the platforms was developed.

2.1 Contribution of the research:

This research has directly contributed to many increasing smart homes around the world and the data produced by them. Even though there are many existing research about smart homes and blockchain, none of them specified the advantages of one blockchain platform over another. Also, none of them showed a large-scale deployment of smart home data into blockchain and evaluated them. All of the existing papers helped me map out a custom architecture and help in defining various key performance indicators for the evaluation. This research will help many smart homes adopt blockchain technology and selecting their platform cautiously.

3. Related Work

3.1 Why Blockchain?

The paper about hybrid blockchain for IoT networks by (Alkhateeb et al., 2022) discusses that hybrid blockchain platforms are emerging as promising solutions for IoT, enhancing security, transparency, and data integrity across applications such as energy and healthcare. By combining public and private chains, they balance privacy with data accessibility, though challenges in scalability and interoperability remain. Future work could focus on energy-efficient protocols and AI-driven optimizations to address these issues. Overall, hybrid blockchains offer a viable path toward secure and scalable IoT systems.

To further solidify the fact that blockchain can be a good option to enhance data privacy in smart homes, (Yang & Wang, 2021) introduces a blockchain-based system for transactive energy management in IoT-enabled smart homes, focusing on secure and privacy-preserving energy transactions. Key advantages include enhanced privacy for users and efficient peer-to-peer and grid-based energy trading, leveraging smart contracts for decentralized operation. Regardless of the advantages, the limitations include potential computational challenges on IoT devices and scalability concerns as the system expands.

A comprehensive survey of blockchain applications across various IoT domains such as smart cities, smart homes and energy systems, emphasize the role of blockchain in enhancing IoT security(Panarello et al., 2018). This paper categorizes many blockchain based approaches into device manipulation and data management, which highlights Ethereum and Hyperledger's role in enabling smart contracts. Secure communication, decentralized access controls and integration of peer-to-peer energy trading platforms are the key contributions made in this paper. Resilience to various cyber-attacks, improved data privacy are the advantages of blockchain mentioned in this paper. The paper also identifies various drawbacks like computational overhead , energy consumption and scalability issues in blockchain based IoT systems.

3.2 Existing blockchain solutions:

(Saraf & Sabadra, 2018) gives a detailed overview of the blockchain platforms that are available in the market. This paper was crucial in helping to understand how blockchain technology transformed the security of data using transparency in various sectors including finance, IoT, healthcare and much more using various platforms such as the Ethereum, Hyperledger and Corda. Each of these platforms have different functionality which serves different purposes. For instance, Hyperledger supports permissioned environments, Ethereum can help with building decentralized applications and Corda specializes in financial agreements. However, challenges remain in scalability, interoperability, and energy consumption. (Alkhateeb et al., 2022) showed how blockchain is being integrated into various sectors and addressed various research questions including 'What blockchain platforms are used in the IoT and Blockchain integration'. This research shows that Ethereum is one of the prominent blockchain platforms that was being used for IoT integration because of the smart contracts. Moreover, this paper also outlines the challenges that were reported in the integration of blockchain and IoT.

On the other hand, the paper by (Uddin et al., 2021) conducts a survey which gives a comprehensive review of blockchain's integration into various IoT applications, highlighting blockchain's potential for securing decentralized IoT ecosystems and addressing data privacy concerns. Key advantages include resilience against single points of failure and enhanced trust without third-party reliance, particularly beneficial in applications like healthcare and smart cities. However, challenges such as high computational costs and limited scalability pose significant constraints, especially for resource-constrained IoT devices.

3.3 Smart home and Blockchain:

To begin with the paper published by (Moniruzzaman et al., 2020) states the basic requirements for a smart home to be able to adopt blockchain technology. They also explore a few of the case studies involving smart homes and blockchain technology. Particularly the 'Smart Home Data Sharing' case study highlights the use of a private Hyperledger Fabric blockchain platform to collect and store smart home data. Finally, this paper also discusses many challenges and open research ideas which includes ideas like 'Smart Home Security and Privacy Leaks' which highlights the importance of addressing challenges with respect to blockchain features and attacks. (Lee et al., 2020) proposed a blockchain based smart home gateway which could potentially prevent data forgery. In this research they have successfully implemented the proposed smart home network on Ethereum platform and evaluated it in terms of security and accuracy. However, the architecture which was proposed was limited to additional computational complexity by blockchain operation.

3.4 Blockchain security for smart home data:

(Dorri et al., 2017) discusses about the evaluation of blockchain based smart homes with respect to security, privacy and performance. They argue that the overheads produced by blockchain are worth the weight given the security and privacy they provide. Given that this was one of the first works that aimed to use blockchain in smart homes, it was limited to only the basic Bitcoin blockchain platform. (Guo & Yu, 2022) analyses the security provided by blockchain technology by surveying consensus algorithms, smart contracts and cryptography. Ultimately, this provides an overview onto Zero-Knowledge proofs and hash functions used in blockchain.

To show how blockchain can help with improving smart home environments, (Arif et al., 2020) explores a consortium blockchain model to secure smart home environments, allowing IoT devices to participate directly in blockchain processes, thus decentralizing data storage and verification. Key advantages include enhanced security, privacy, and resistance to single-point failures, making it more resilient for smart home applications. However, limitations such as processing delays and computational constraints in IoT devices are noted.

Next, learning about a privacy-preserving mechanism for smart home using blockchain and edge computing helped this research in many ways. (Qashlan et al., 2021) shows the integration of Ethereum smart contracts, attribute-based access control, and differential privacy which could enhance security and privacy. The architecture includes IoT devices, edge servers and cloud servers which enables the decentralized management of smart home ecosystems. This paper addresses many challenges like unauthorized access, DoS attacks and data modification. The performance evaluation from this paper highlights its efficiency in real-time smart home scenarios. The challenges from this paper include computational overhead and scalability limitations for large-scale IoT deployments.

3.5 Ethereum and smart home:

To show how Ethereum could improve a smart home, (Xu et al., 2018) presents a decentralized smart home system using an Ethereum-based private blockchain to securely store and manage IoT sensor data, specifically temperature and humidity levels. This paper shows how data privacy can me improved in a smart home with the help of the blockchain immutability and the elimination of centralized data vulnerabilities, making it suitable for secure, automated home environments. However, limitations include the high cost and impracticality of the 15-second block time for real-time applications.

To build upon the previous paper, in 2021, (Murphy, 2021) investigates the use of an Ethereum-based private blockchain to secure IoT devices, specifically focusing on integrating IoT devices like Raspberry Pi with Ethereum for enhanced security. By implementing smart contracts in a Proof of Authority (PoA) setup, the study demonstrates improved data security and device control without relying on centralized authorities, which is advantageous in preventing common cyber threats. However, the challenges include interoperability issues and the high computational demand for small-scale devices.

To have a clear understanding of the privacy provided by Ethereum in IoT environments, (Patruni & Saraswathi, 2022) suggests a methodology which integrates IoT devices with blockchain through Ethereum based systems like Ganache, Truffle and MetaMask. The key features of this paper are public key cryptography, which helped attain confidentiality and automated smart contract verification which ensures availability. Moreover, the methodology was able to mitigate DoS attacks, which pose significant threat to IoT systems. The downside of this paper lies in addressing the resource constraints of IoT devices and maintaining scalability at the same time in a distributed environment.

3.6 Hyperledger and smart home:

The paper by (Liao, 2022) proposes a secure smart home system using a combination of blockchain (Hyperledger Fabric) and cloud services to provide decentralized and fine-grained access control. The advantages from this paper include enhanced privacy, user autonomy in access control, and prevention of unauthorized access, addressing key security concerns in smart home data management. But the challenges remain in optimizing system latency and handling computational overhead due to blockchain operations.

(Ren et al., 2022) also introduces a Hyperledger Fabric based security control system which is designed specifically for smart home environments. This implementation is highly focused on performance optimization to meet the high demands from the IoT systems. This paper integrates Hyperledger fabric's permissioned blockchain to provide an access control and security controls for smart home devices. The key components of this paper include certificate-based device authentication and integrity verification during device operations, which ensures tamper proof management of device configurations. The optimization made to the blockchain enhances efficiency in transaction processing, with results showing better performance with device registrations, which was up to 6000 transactions under specified hardware conditions. The challenges from this paper highlight the scalability limitations and resource demands of blockchain operations in IoT setups.

Furthermore, (Ammi et al., 2021) proposes a custom blockchain based architecture which uses the integration of Hyperledger Fabric and Hyperledger Composer to address various security challenges like confidentiality, integrity, availability, authorization and mainly privacy in a smart home which is using lightweight IoT systems. The architecture of this implementation includes four main layers which include a cloud storage, Hyperledger Fabric (blockchain), the smart home application and the Hyperledger Composer which handles the orchestration. The paper emphasizes the benefits of permissioned blockchain. It clearly shows how transparency can improve trustworthiness in a IoT environment. The main challenges highlighted in this paper include resource constraints in IoT environments and managing the computational overhead.

3.7 Comparative study of blockchain platforms for IoT privacy:

This part of the literature review is important since it shed light on how to compare different blockchain platforms. Firstly (El-Sayed & Ibrahim, 2023) shows a comparative study of different blockchain based methods for securing IoT based devices. This analyses public, private and hybrid blockchain architectures. This paper evaluates key performance and security metrics, which include latency, throughput and energy efficiency. It also highlights the trade-offs that are involved in the approaches, mainly the scalability challenges of public blockchain versus the better performance of private blockchain while having reduced decentralization

abilities. This research focuses on the fact that the IoT ecosystems need a tailored blockchain which can balance security, scalability and energy efficiency.
Traditional centralized IoT systems involved many challenges like vulnerability to attacks and high setup costs, which proves why blockchain's decentralized architecture might prove good for IoT systems (Urmila et al., 2019). Various use cases like implementation of smart contracts for medical IoT ownership and data exchange is also examined to illustrate practical applications. Moreover, this paper also provides a comparative analysis of blockchain solutions like Ethereum and Hyperledger for managing IoT device ownership and inter-device communication. It highlights blockchain's advantages such as decentralization, immutability and resistance to single-point failures. However, the limitations in this paper include constraints in performance analysis and the need for tailored implementation techniques for different IoT use cases.
On the other hand, (Pancari et al., 2023) has provided a systematic comparison of Ethereum and Hyperledger Fabric blockchain platforms for attribute-based access control (ABAC) in smart home environments. This paper outlines the development of an ABAC smart contract for Ethereum and modifies a Hyperledger contract for comparison. The findings from the research highlight how Ethereum's public blockchain is able to provide flexibility while Hyperledger Fabric is all about modular and ideal for organizational settings. The scalability issue in Ethereum is one of the main challenges underlined in this paper.

4. Research Methodology:

There are many examples that show successful integrations of smart home devices and blockchain platforms (Liao, 2022; Murphy, 2021; Xu et al., 2018). These papers show how both Hyperledger Fabric and Ethereum can secure smart home data effectively. Given that Ethereum is a public blockchain and Hyperledger Fabric is a private blockchain, it comes down to which platform would be more efficient in terms of privacy, performance and cost with respect to smart home data. The primary goal of this research is to simulate a smart home environment and use the data from it to evaluate both the specified blockchain platform. The final step is evaluating both the blockchain platforms with respect to the privacy, performance and cost. For privacy, I have outlined three criteria such as data encryption, access controls and data exposure. For performance I have measured the time taken to perform a certain transaction and the cost is the total fees for the entire transaction. To avoid any expensive or resource constraint setbacks, I have divided the dataset into batches and evaluated the blockchain platforms with two different batches. A diagram representing the research methodology for this research is specified in the appendix.

5. Design specification and Implementation:

5.1 Dataset simulation and preprocessing:

Dataset generation using a smart home simulation was one of the integral parts of this research, since it enables us to test the blockchain platforms more effectively for their privacy and other metrics. For this part of the research, I used OpenSHS, which is an open-source smart home simulation tool(Alshammari et al., 2017). The simulation was also carried out in a virtual environment since the application is more compatible with a Linux machine rather than a Windows machine.

5.1.1 Simulation Environment:

The virtual machine was setup in VirtualBox for the dataset simulation. Apart from being able to provide compatibility for the OpenSHS application, the virtual machine setup also ensured that it provided a controlled environment for testing and generation of the dataset. The specification of the virtual machine is as follows:

- Operating system: Ubuntu 20.04 LTS (64-bit)
- Base memory: 4GB
- Processors: 2 virtual processors
- Disk space: 35 GB (virtual hard disk)
- Graphics controller: VBoxSVGA (128 GB video memory)

It was also important to install Blender in this virtual machine since it is one of the dependencies for the OpenSHS application(Alshammari et al., 2017). After ensuring all the important dependencies are installed, the application was run in order to generate the dataset. Aggregating all the contexts (weekday morning, weekday evenings, weekend morning, weekend evenings), I got a CSV file which had 30 days' worth of data from the smart home. This CSV file was then imported back to the Windows machine for further preprocessing.

5.1.2 Preprocessing:

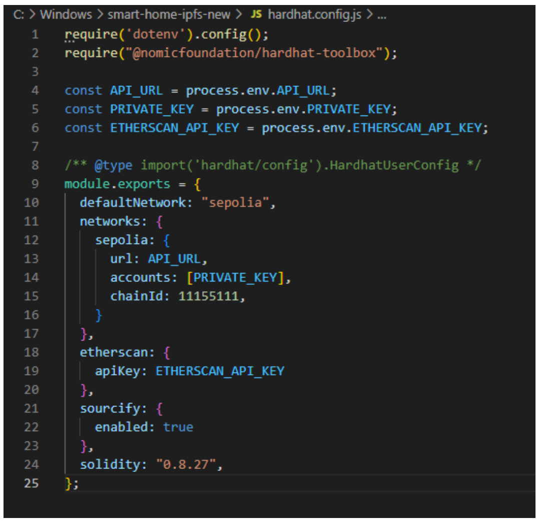
To make the dataset compatible with Ethereum, it must be processed into a structured JSON file. Besides that, there are a few steps that was followed in order to make sure that the dataset was fully ready for the blockchain implementation. To begin with, the timestamp from the CSV file was converted into UNIX timestamp which verifies all the data point on the timestamp column is uniform and verifies usability in blockchain transactions. Also, the names of the devices were simplified to reduce cost when being processed by the blockchain platforms. On the other hand, the activity column, which had information about the activity about what the person was doing in the smart home at the specified time, was hashed using SHA-256, ensuring privacy. It was also made certain that the content was in 32-byte format which is essential for Ethereum environment. Next, this processed dataset was converted into a JSON file with the timestamp, activityHash and deviceStates, which was a dictionary containing the states of the devices. Finally, the dataset was divided into batches which contained 500 records each and was saved separately, ready for the blockchain integration.

5.2 Ethereum:

To implement the dataset in Ethereum, I utilized Alchemy, a web3 development platform, which is used for deployment and interaction with the Ethereum blockchain. I decided to use Sepolia, which is the most cost effective and reliable testing environment. This was necessary because implementing this setup on the actual Ethereum blockchain can be really expensive, given the rise in the gas prices. On the other hand, Alchemy provides a reliable infrastructure with secure RPC endpoints and various tools to monitor performance and debugging solutions. To further optimize cost, an off-chain data storing solution (IPFS) was used and the blockchain integration ensures storage and retrieval of the IPFS CIDs, which are the data references.

5.2.1 Setting up the Sepolia network:

To be able to interact with the Ethereum Sepolia test network from the local machine, hardhat which is a Ethereum development environment was used. Using hardhat, I created a configuration file which includes the to the Sepolia network with the Alchemy RPC endpoint. In addition to this, the Alchemy's dashboard also gives insight into various metrics such as the success rate, total number of requests, median response and much more.



C: > Windows > smart-home-ipfs-new > JS hardhat.config.js > ...

```
require('dotenv').config();
require("@nomicfoundation/hardhat-toolbox");

const API_URL = process.env.API_URL;
const PRIVATE_KEY = process.env.PRIVATE_KEY;
const ETHERSCAN_API_KEY = process.env.ETHERSCAN_API_KEY;

/** @type import('hardhat/config').HardhatUserConfig */
module.exports = {
  defaultNetwork: "sepolia",
  networks: {
    sepolia: {
      url: API_URL,
      accounts: [PRIVATE_KEY],
      chainId: 11155111,
    }
  },
  etherscan: {
    apiKey: ETHERSCAN_API_KEY
  },
  sourcify: {
    enabled: true
  },
  solidity: "0.8.27",
};
```

Figure 1 : Hardhat Configuration file

Besides this, a MetaMask wallet was also created and connected to the Sepolia network. The wallet's public and private keys are necessary for signing transactions and also interacting with the blockchain. In order to get the Sepolia ETH from the Alchemy faucet, the wallet must have 0.001 ETH (Appendix 1.1). After this, we can request 0.1 Sepolia ETH every 72 hours. This Sepolia ETH currency can then be used to make transactions in the Sepolia network.

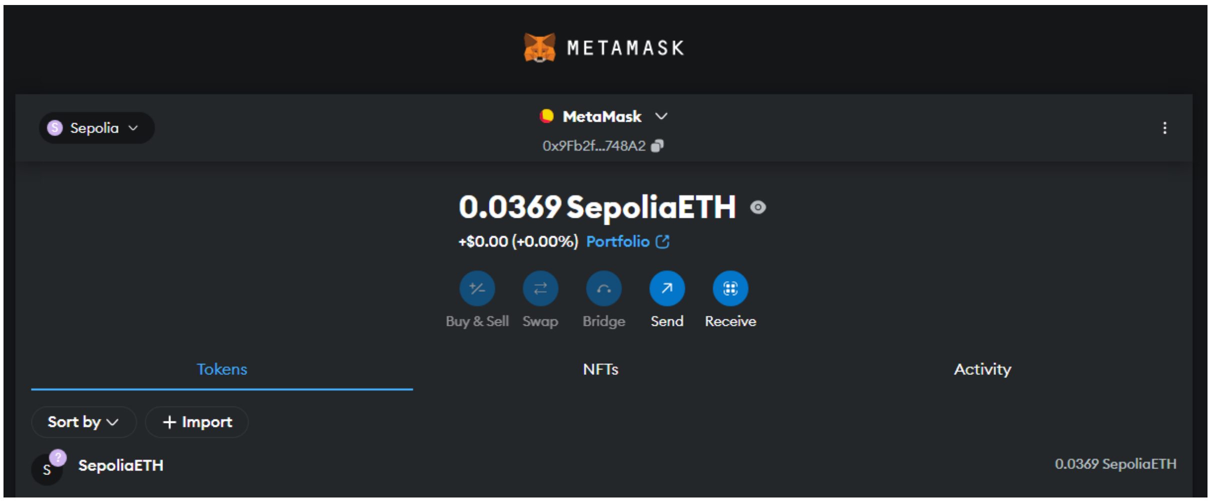
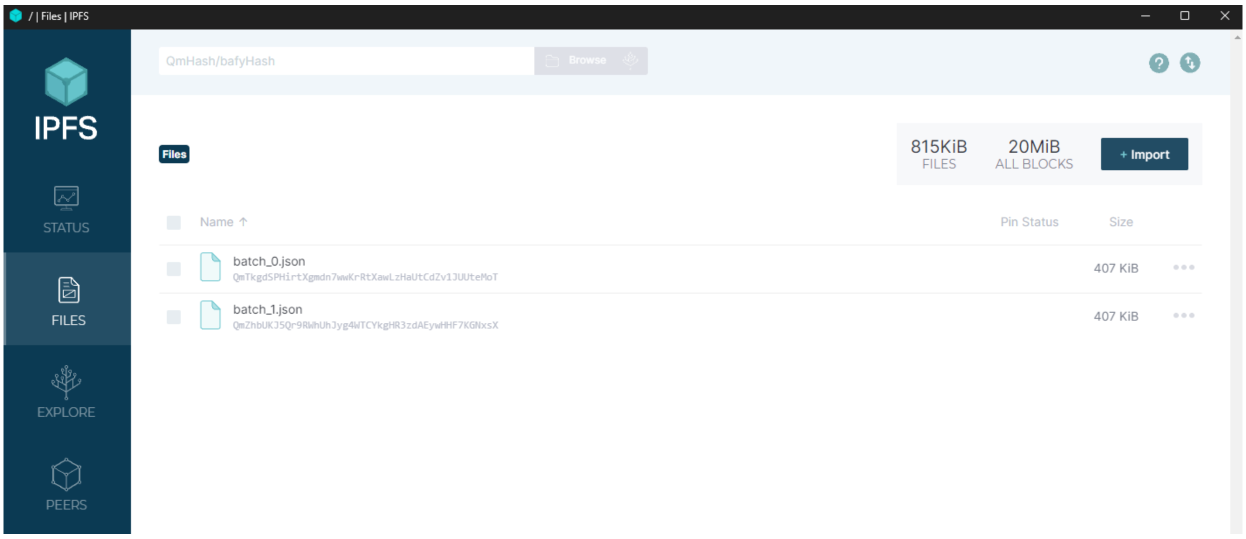


Figure 2 : Wallet with Sepolia ETH

5.2.2 IPFS Integeration:

As said before, the InterPlanetary File System was used as the off-chain storage solution since it provides an efficient and secure data storage solution for storing the encrypted datasets that was generated by the smart home simulation. The content identifier (CID) from the IPFS was then stored on the Ethereum blockchain. As mentioned before, the dataset was also encrypted using the AES-256-CBC algorithm, which ensures confidentiality of the data before it gets stored in the IPFS. This encrypted dataset was then stored in the IPFS system using the IPFS desktop application. This process would generate a unique CID for every dataset that is being uploaded. The data retrieval process ensures decentralized access since the data stored in the IPFS can be retrieved using the CID, which in turn ensures data security.



Name ↑	Pin Status	Size
batch_0.json QmTkgdSPHirtXgmdn7wwKrRtXawLzHaUtCdZv1JUUteMoT		407 KiB
batch_1.json QmZhbUKJ5Qr9RWhUhJyg4WTCYkgHR3zdAEywHHF7KGNxsX		407 KiB

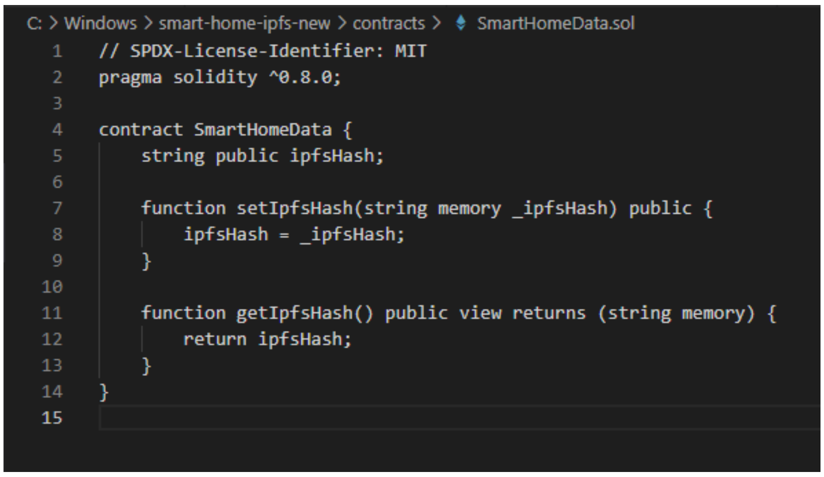
Figure 3 : Dataset uploaded to IPFS Desktop.

5.2.3 Smart Contract:

The smart contract implementation for the Ethereum platform is straightforward. It is written in Solidity, which is a programming language primarily used for developing smart contracts. This smart contract is deployed to the Sepolia test network with the help of Hardhat. Between these frameworks, the MetaMask wallet serves as the secure wallet which is used to manage the transactions with the help of public and private keys. The smart contract is minimalistic, and its core function is to securely link the references to the data stored in the off-chain storage to the blockchain. This optimizes cost and storage.

The two main functions on the smart contract include the setIpfsHash(), which allows the authorised user to store the IPFS CID on the Sepolia testnet. It updates the state variable ipfsHash with the CID given by the user, which ensures secure referencing to off chain encrypted data. Furthermore, this function can be easily modified in order to add more access control to the blockchain application, showcasing the flexibility of Ethereum platform.

Secondly, the getIpfshash() function allows the user to retrieve the IPFS hash stored on the blockchain. This function will be able to return the current state of the ipfsHash variable, which then allows the user to fetch the CID and access the data corresponding to it on the IPFS storage. The advantage of this being a view function is that it allows to user to retrieve the data using the function without any gas costs.



C: > Windows > smart-home-ipfs-new > contracts > SmartHomeData.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract SmartHomeData {
    string public ipfsHash;

    function setIpfsHash(string memory _ipfsHash) public {
        ipfsHash = _ipfsHash;
    }

    function getIpfsHash() public view returns (string memory) {
        return ipfsHash;
    }
}

```

Figure 4 : Smart contract written in Solidity.

5.2.4 The Ethereum application:

The main application, which is written in JavaScript, has various important implementation steps. Firstly, the encryption step which uses the AES-256-CBC algorithm for the encryption. It generates a unique 32-byte key for every dataset that is passed to the application. The application then uses this key to encrypt the dataset. The result from this function is the combination of the initialization vector and the cipher text. The encrypted dataset is then stored in the IPFS using storeDataOnIpfs() function, which will return a CID. To verify if the dataset is successfully stored in IPFS, the dataset can be retrieved using the CID and decrypted using the same key.

Furthermore, the application also has the functions from the smart contract such as the setIpfshash() and getIpfsHash() which stores and retrieves the hash from the blockchain. The application also helps in deploying the contract using the Hardhat framework. Hardhat will help in automating compilation and deployment of the contract. This script also contains functions that evaluates the blockchain and will be further explained in the evaluation section.

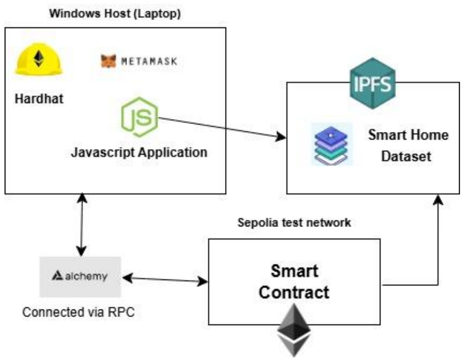


Figure 5: Network diagram of the Ethereum implementation.

5.3 Hyperledger:

The environment to do the Hyperledger implementation was configured using Windows Subsystem for Linux (WSL) on the host Windows machine. This provided a lightweight Linux system that was used to run the Fabric network, Docker containers and all the other tools. This setup with the WSL was useful in providing a seamless development and deployment environment without requiring a dedicated Linux machine. This method was chosen over the virtual machine since the virtual machine seemed to work a bit slow with all the requirements for the Fabric network.

5.3.1 WSL Configuration:

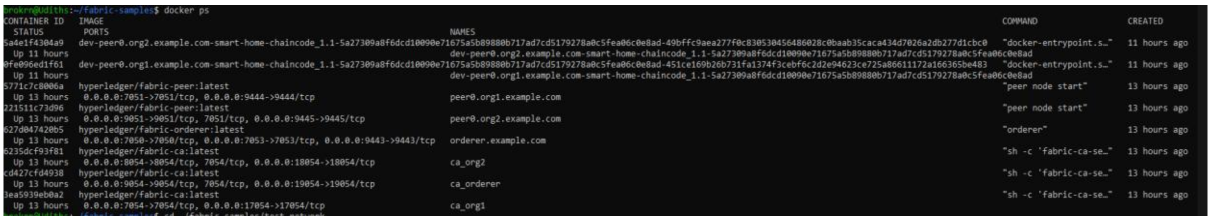
The WSL setup in the Windows machine bridges the gap between the Windows and Linux development ecosystems. For the fabric implementation, the WSL2 was used. Ubuntu v24.04.1 LTS was installed and used as the WSL operating system. Various other necessary tools that were required for the implementation was also installed and set up on the environment. The tools that were set up include:

- Docker, which is used to put the Hyperledger Fabric components into a container, like the peers, orderers and the Certificate Authorities(CA).
- Node.js, which is the primary coding language used for developing the chaincode and the application scripts.
- Fabric CLI, which is a command line tool that can be used to manage the Hyperledger Fabric network .
- And the Fabric Network SDK, which is a node-based SDK that helped in integrating the application with the blockchain.

Using WSL simplified the workflow of the Fabric implementation, easing the process of data sharing and it also helped the resource constraints, making the progress of the implementation fast and efficient without the need for optimizing the virtual machine.

5.3.2 Fabric Samples:

Docker is one of the important dependencies for the Hyperledger fabric implementation since it helps in arranging the components of the blockchain framework. The official fabric Samples repository from GitHub provided me with various preconfigured scripts which included the test-network, which was used as the primary foundation for deploying the custom chaincode that I wrote. The network.sh script inside the test-network directory helped me initialize a channel with two organizations and a single orderer node, which was responsible for ordering the transactions. The two organizations namely, Org1, which was the primary organization, and it was responsible for managing the smart home data, while Org2 was a responsible for validating the same. Both of these organizations were put in a shared channel (mychannel) which was used for the communication between the two organizations. Furthermore, only one user identity was created (user3), which was used for all transactions, and it was created for Org1 and stored in wallet in the smart-home-application directory.



```
brokrn@Udiths:~/fabric-samples$ docker ps
CONTAINER ID   IMAGE                                                                                                                                                                                COMMAND                  CREATED
  STATUS        PORTS                                                                    NAMES
5a4e1f4304a9   dev-peer0.org2.example.com-smart-home-chaincode_1.1-5a27309a8f6dcd10090e71675a5b89880b717ad7cd5179278a0c5fea06c0e8ad-49bffc9aea277f0c830530456486028c0baab35caca434d7026a2db277d1cbc0   "docker-entrypoint.s…"   11 hours ago
  Up 11 hours                                                                            dev-peer0.org2.example.com-smart-home-chaincode_1.1-5a27309a8f6dcd10090e71675a5b89880b717ad7cd5179278a0c5fea06c0e8ad
0fe096ed1f61   dev-peer0.org1.example.com-smart-home-chaincode_1.1-5a27309a8f6dcd10090e71675a5b89880b717ad7cd5179278a0c5fea06c0e8ad-451ce169b26b731fa1374f3cebf6c2d2e94623ce725a86611172a166365be483   "docker-entrypoint.s…"   11 hours ago
  Up 11 hours                                                                            dev-peer0.org1.example.com-smart-home-chaincode_1.1-5a27309a8f6dcd10090e71675a5b89880b717ad7cd5179278a0c5fea06c0e8ad
5771c7c8006a   hyperledger/fabric-peer:latest                                                                                                                                                       "peer node start"        13 hours ago
  Up 13 hours   0.0.0.0:7051->7051/tcp, 0.0.0.0:9444->9444/tcp                           peer0.org1.example.com
221511c73d96   hyperledger/fabric-peer:latest                                                                                                                                                       "peer node start"        13 hours ago
  Up 13 hours   0.0.0.0:9051->9051/tcp, 7051/tcp, 0.0.0.0:9445->9445/tcp                 peer0.org2.example.com
627d047420b5   hyperledger/fabric-orderer:latest                                                                                                                                                    "orderer"                13 hours ago
  Up 13 hours   0.0.0.0:7050->7050/tcp, 0.0.0.0:7053->7053/tcp, 0.0.0.0:9443->9443/tcp   orderer.example.com
6235dcf93f81   hyperledger/fabric-ca:latest                                                                                                                                                         "sh -c 'fabric-ca-se…"   13 hours ago
  Up 13 hours   0.0.0.0:8054->8054/tcp, 7054/tcp, 0.0.0.0:18054->18054/tcp               ca_org2
cd427cfd4938   hyperledger/fabric-ca:latest                                                                                                                                                         "sh -c 'fabric-ca-se…"   13 hours ago
  Up 13 hours   0.0.0.0:9054->9054/tcp, 7054/tcp, 0.0.0.0:19054->19054/tcp               ca_orderer
3ea5939eb0a2   hyperledger/fabric-ca:latest                                                                                                                                                         "sh -c 'fabric-ca-se…"   13 hours ago
  Up 13 hours   0.0.0.0:7054->7054/tcp, 0.0.0.0:17054->17054/tcp                         ca_org1
```

Figure 6: The containers running in the Docker.

5.3.3 Chaincode development:

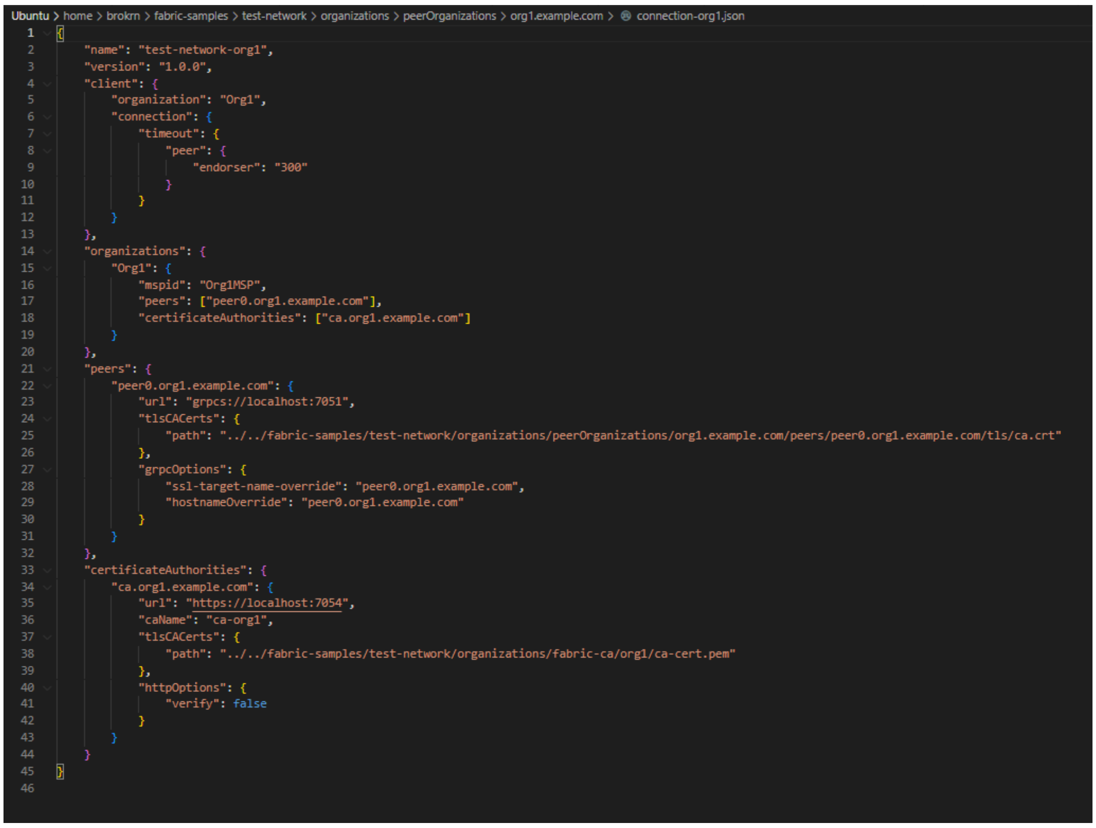
The custom chaincode was written in Node.js and was installed and approved on peers from both Org1 and Org2, which made sure that both the organizations on the channel would be validating the transactions. The chaincode will be able to manage storage, retrieval of the encrypted smart home dataset and evaluation of the blockchain. The key features of the chaincode include the data encryption, which encrypts the data from the dataset using AES-256-CBC algorithm before it is stored on the ledger. It enforces privacy by using functions like checkEncryption() and checkDataExposure(), which is also used to evaluate the Hyperledger Fabric blockchain. Finally, even though the implementation uses only user3, the chaincode is designed to support multiple users. Furthermore, the various other chaincode functions are listed below:

- As the name suggests, storeEntry() function is used to store a data entry on the ledger with a unique ID.
- The checkEncryption() function verifies if the data associated with the unique ID is properly encrypted or not.
- evaluatePerformance() function allows us to measure the time taken to retrieve a data entry form the ledger.
- Finally, the evaluateCost() function helps us to estimate the storage cost for a data entry based on the size of the entry.

5.3.4 Hyperledger Fabric application:

The application for Hyperledger Fabric was developed with the help of Node.js using the Fabric Network SDK and was successfully executed with the help of the user3 identity that was verified by the Fabric CA. This code was used to perform various data operations that were specified on the chaincode. The important features of the application script are as follows:

- The application was connected to the Hyperledger Fabric network via the connection-org1.json configuration file. Also, the user3 identity that exists in the wallet will authenticate all the interactions the application is doing with the network.


Ubuntu > home > brokrn > fabric-samples > test-network > organizations > peerOrganizations > org1.example.com > connection-org1.json

```json
{
    "name": "test-network-org1",
    "version": "1.0.0",
    "client": {
        "organization": "Org1",
        "connection": {
            "timeout": {
                "peer": {
                    "endorser": "300"
                }
            }
        }
    },
    "organizations": {
        "Org1": {
            "mspid": "Org1MSP",
            "peers": ["peer0.org1.example.com"],
            "certificateAuthorities": ["ca.org1.example.com"]
        }
    },
    "peers": {
        "peer0.org1.example.com": {
            "url": "grpcs://localhost:7051",
            "tlsCACerts": {
                "path": "../../fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt"
            },
            "grpcOptions": {
                "ssl-target-name-override": "peer0.org1.example.com",
                "hostnameOverride": "peer0.org1.example.com"
            }
        }
    },
    "certificateAuthorities": {
        "ca.org1.example.com": {
            "url": "https://localhost:7054",
            "caName": "ca-org1",
            "tlsCACerts": {
                "path": "../../fabric-samples/test-network/organizations/fabric-ca/org1/ca-cert.pem"
            },
            "httpOptions": {
                "verify": false
            }
        }
    }
}
```

Figure 7: connection-org1.json file which shows how the application is connected to the application.

- Another main feature of the application is that it will process the dataset (in JSON) and will assign a unique ID for each entry before it submits the transaction to the chaincode. This will make sure that each entry that is being sent to the blockchain is easily identifiable.
- Finally, the application also has many functions that will help in evaluating the blockchain. These functions will be explained in detail in the evaluation part of this report.

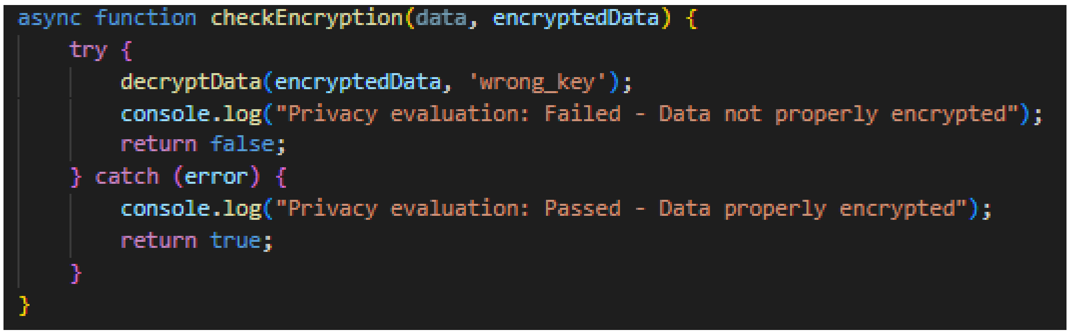
6. Evaluation:

6.1 Ethereum Evalution:

The Ethereum platform's performance is evaluated upon three key metrics as said before, namely, transaction latency, gas usage and privacy assurance. Sepolia testnet and Etherscan serve as the primary tools for this evaluation. Etherscan is used for monitoring transaction details and primarily calculate gas costs. The different functions used for conducting evaluation is explained in detail below:

6.1.1 Privacy Evaluation:

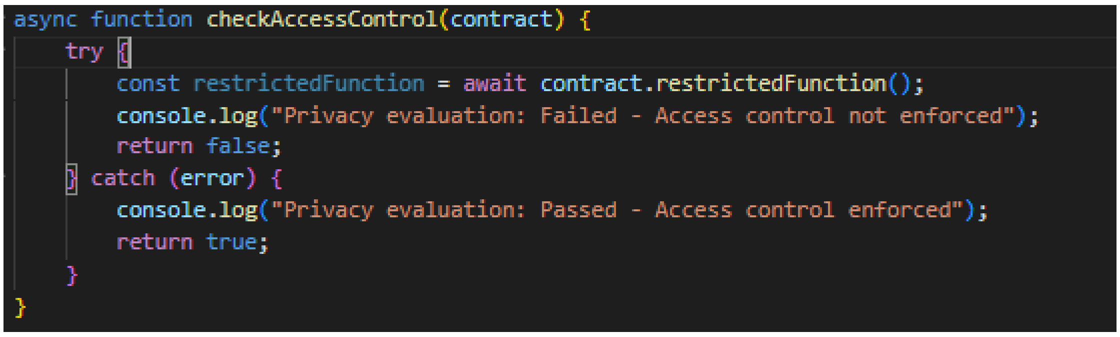
This is the key evaluation criteria of this research. There are three main privacy evaluations that take place for the Ethereum blockchain. This evaluation ensures that sensitive data from the smart home is securely stored, accessed and managed without unauthorised breaches or exposure. Firstly, to perform encryption validation, the dataset is encrypted using AES-256-CBC algorithm and then the **checkEncryption()** function attempts to decrypt the encrypted dataset using a wrong key and if the decryption fails, it is evident that the encryption process is robust.



```
async function checkEncryption(data, encryptedData) {
    try {
        decryptData(encryptedData, 'wrong_key');
        console.log("Privacy evaluation: Failed - Data not properly encrypted");
        return false;
    } catch (error) {
        console.log("Privacy evaluation: Passed - Data properly encrypted");
        return true;
    }
}
```

Figure 8: checkEncryption() function.

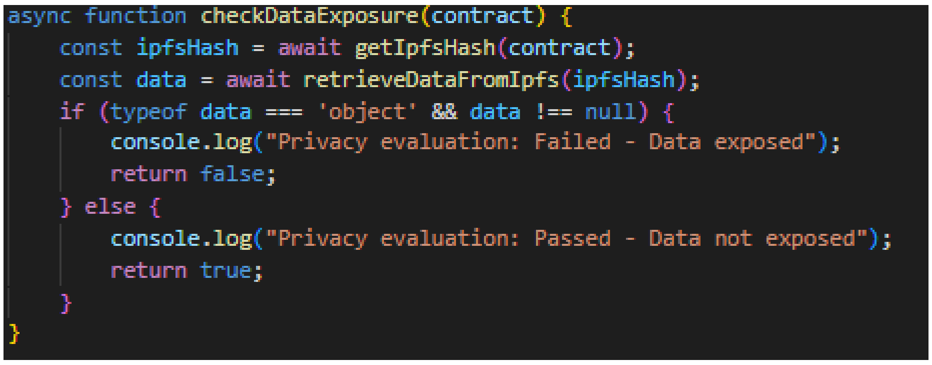
Next, to verify that no unauthorized user has access to restricted functions or power to modify data on the blockchain, the **checkAccessControl()** function attempts to use a restricted function. If the attempt is successfully blocked, it confirms that access control measures are effectively placed in the blockchain.



```
async function checkAccessControl(contract) {
    try {
        const restrictedFunction = await contract.restrictedFunction();
        console.log("Privacy evaluation: Failed - Access control not enforced");
        return false;
    } catch (error) {
        console.log("Privacy evaluation: Passed - Access control enforced");
        return true;
    }
}
```

Figure 9: checkAccessControl() function.

Finally, to confirm that no sensitive data is stored in plaintext on the blockchain or exposed via the IPFS, the **checkDataExposure()** retrieves the IPFS CID from the blockchain and attempts to access the data. If the data retrieved by this function is encrypeted and cannot understood without decryption, this function will be success. Even if the CID can be accessed without authorization, the encryption will secure the data.

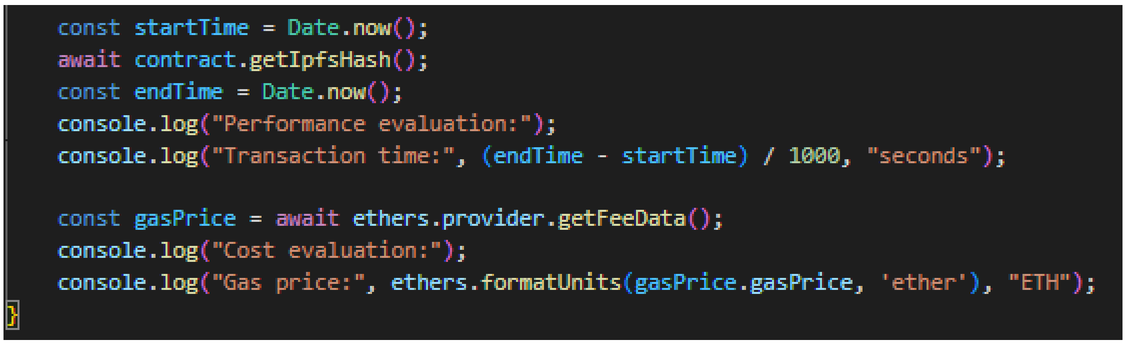


```
async function checkDataExposure(contract) {
    const ipfsHash = await getIpfsHash(contract);
    const data = await retrieveDataFromIpfs(ipfsHash);
    if (typeof data === 'object' && data !== null) {
        console.log("Privacy evaluation: Failed - Data exposed");
        return false;
    } else {
        console.log("Privacy evaluation: Passed - Data not exposed");
        return true;
    }
}
```

Figure 10: checkDataExposure() function.

6.1.2 Transaction Latency:

To measure the transaction speed of the Ethereum blockchain, I have measured the time taken to execute the important smart contract function such as the setIpfsHash() and getIpfsHash(). The objective of this metric is to provide information about the responsiveness of the blockchain under the defined network condition for the smart home dataset. I measured this metric by recording the timestamp before and after calling the getIpfshash(). The difference between these timestamps can give the execution time of the blockchain to process the implementation.



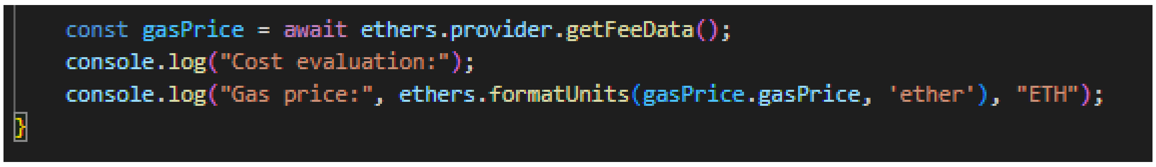
```
    const startTime = Date.now();
    await contract.getIpfsHash();
    const endTime = Date.now();
    console.log("Performance evaluation:");
    console.log("Transaction time:", (endTime - startTime) / 1000, "seconds");

    const gasPrice = await ethers.provider.getFeeData();
    console.log("Cost evaluation:");
    console.log("Gas price:", ethers.formatUnits(gasPrice.gasPrice, 'ether'), "ETH");
}
```

Figure 11: Code for evaluating the performance of Ethereum.

6.1.3 Gas price calculation:

To evaluate the cost of implementing the Ethereum blockchain, the application uses the ether.js library which is able to retrieve and display the gas price form the Sepolia test network. This part of the evaluation is crucial for this research since it helps to understand the transaction cost involved in interacting with the blockchain. The evaluation is performed using **ethers.provider.getFeeData()** to fetch the current gas price from the network. This method will help in retrieving real-time gas costs for transactions on the test network. After retrieving the price, which is typically measured in wei (smallest unit of ETH), it is converted to ETH with the help of ethers.formatUnits().



```
    const gasPrice = await ethers.provider.getFeeData();
    console.log("Cost evaluation:");
    console.log("Gas price:", ethers.formatUnits(gasPrice.gasPrice, 'ether'), "ETH");
}
```

Figure 12: Gas price evaluation for Ethereum.

4.2 Hyperledger Fabric Evaluation:

Similar to Ethereum, Hyperledger Fabric was also evaluated on the same key metrics. As discussed before, these evaluations were conducted using a chaincode that was deployed on a Fabric network, which was interacting with the dataset using a Node.js application. But unlike Ethereum, the evaluation script is written inside the chaincode and called in the application script. The key evaluation functions are explained below.

4.2.1 Privacy Evaluation:

The Fabric evaluation is very similar to that of the Ethereum. The encryption validation is done by calling the **checkEncryption()** function which retrieves the data from the blockchain and checks if it is encrypted using the AES-256-CBC algorithm. If the data is properly encrypted, the test passes. It shows that the data remains encrypted while in the blockchain and cannot be accessed without decryption. Secondly, to verify access control and ensure that unauthorised users cannot access restricted functions, the **checkAccessControl()** function compares the identity of the invoking user with a list of the authorised user (which for now is only user3). Finally, to validate data exposure, the **checkDataExposure()** function retrieves data associated with a ledger entry and ensures that no plain data is exposed.

4.2.2 Performance Evaluation:

For the performance evaluation, I used the **evaluatePerformance()** function which will record the execution time taken for storing or retrieving an entry from the ledger. The timestamp before and after the transaction is done is recorded to calculate the total latency. The result is displayed after all the transactions are completed in the dataset and the output is shown in the console.

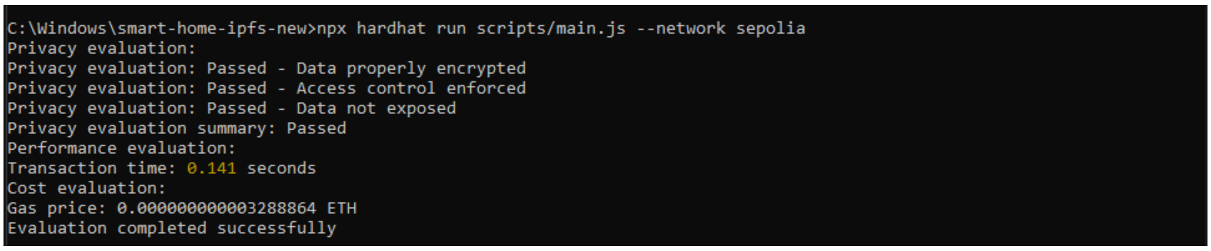
4.2.3 Cost Evaluation:

Finally, for the cost evaluation of the Hyperledger fabric, the **evaluateCost()** function calculates the cost of storing a smart home data entry based on the size of the entry in bytes. An important thing note is that Hyperledger Fabric is a permissioned blockchain and it does not incur any cost to make the transactions. This function is merely an assumption or a place holder. The cost of each entry is then summed up to complete the total cost of storage for the dataset. All these evaluations helped me reach the key requirements to complete the comparison between the blockchains.

4.3 Summary of evaluation:

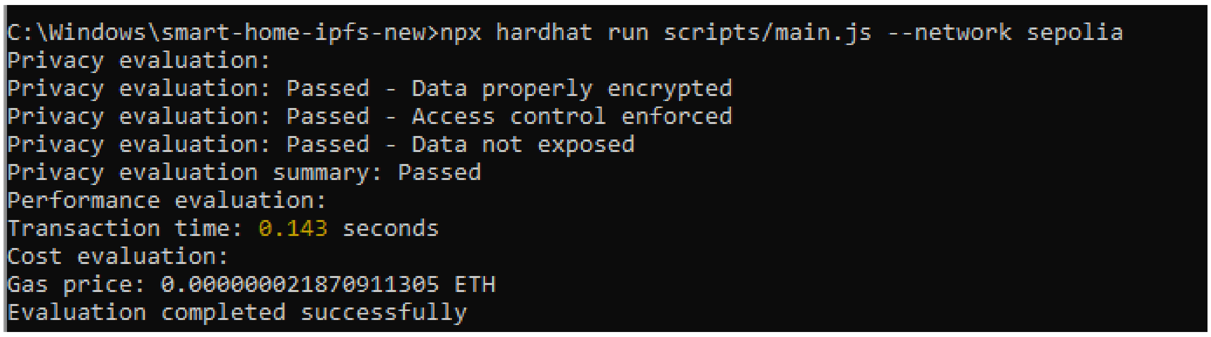
The evaluation of both the Ethereum and Hyperledger Fabric platforms for managing the smart home data was conducted with two different batches of the dataset, batch0 and batch1. The result from this evaluation is summarised below and then the comparative analysis will be done in the next section. To begin with Ethereum's privacy assurance passed all evaluations for both the batches including the encryption validation, access control and data exposure. In case of Hyperledger Fabric, encryption validation passed for all entries in both the batches. Data exposure also passed since no plaintext data was exposed on the ledger. But access control failed for all entries, and this indicates that the access control mechanisms were not enforced as intended. In terms of performance, the latency that was recorded in Ethereum is approximately 0.141 seconds for batch0 and 0.143 seconds for batch1. In Hyperledger Fabric, batch0 had 583 milliseconds and batch1 had 675 milliseconds. The reason for Ethereum having high performance maybe because, in Ethereum, I only stored the IPFS CID on the chain. Finally, the cost evaluation for Ethereum batch0 came around 0.0000000003288864 ETH and batch1 to 0.00000002187911305 ETH. In case of Hyperledger Fabric, the total cost for both the batches came to 4165.00 units since both the batches had same amount of entries.

In addition to the evaluation summary, I will also attach the results of the application down below,



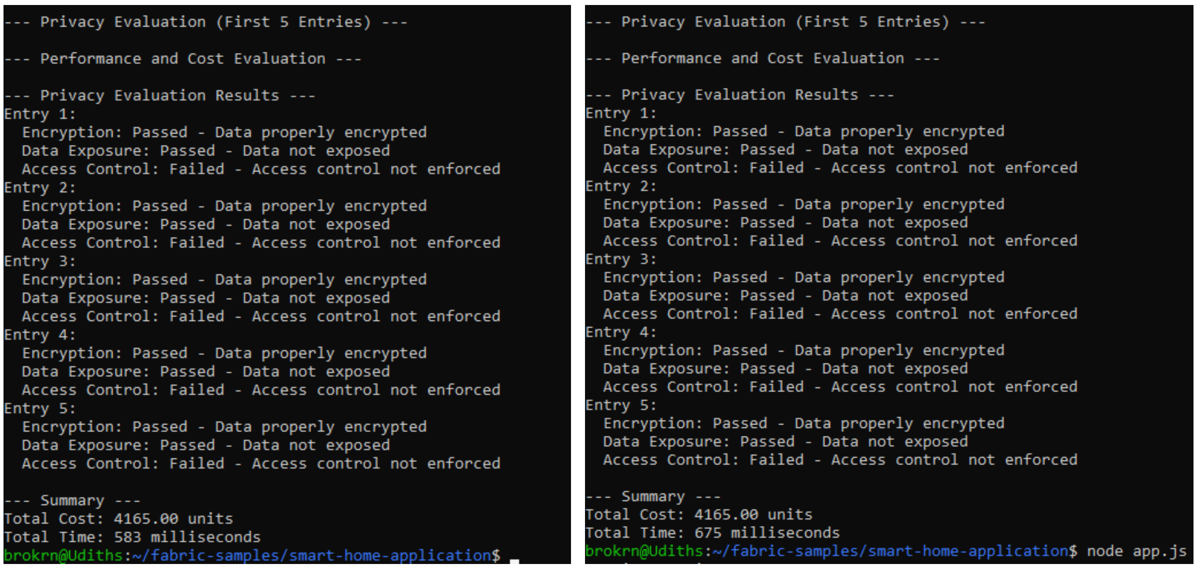
```
C:\Windows\smart-home-ipfs-new>npx hardhat run scripts/main.js --network sepolia
Privacy evaluation:
Privacy evaluation: Passed - Data properly encrypted
Privacy evaluation: Passed - Access control enforced
Privacy evaluation: Passed - Data not exposed
Privacy evaluation summary: Passed
Performance evaluation:
Transaction time: 0.141 seconds
Cost evaluation:
Gas price: 0.000000000003288864 ETH
Evaluation completed successfully
```

Figure 13: Batch0 implementation in Ethereum.



```
C:\Windows\smart-home-ipfs-new>npx hardhat run scripts/main.js --network sepolia
Privacy evaluation:
Privacy evaluation: Passed - Data properly encrypted
Privacy evaluation: Passed - Access control enforced
Privacy evaluation: Passed - Data not exposed
Privacy evaluation summary: Passed
Performance evaluation:
Transaction time: 0.143 seconds
Cost evaluation:
Gas price: 0.000000021870911305 ETH
Evaluation completed successfully
```

Figure 24: Batch1 implementation in Ethereum.



```
--- Privacy Evaluation (First 5 Entries) ---

--- Performance and Cost Evaluation ---

--- Privacy Evaluation Results ---
Entry 1:
  Encryption: Passed - Data properly encrypted
  Data Exposure: Passed - Data not exposed
  Access Control: Failed - Access control not enforced
Entry 2:
  Encryption: Passed - Data properly encrypted
  Data Exposure: Passed - Data not exposed
  Access Control: Failed - Access control not enforced
Entry 3:
  Encryption: Passed - Data properly encrypted
  Data Exposure: Passed - Data not exposed
  Access Control: Failed - Access control not enforced
Entry 4:
  Encryption: Passed - Data properly encrypted
  Data Exposure: Passed - Data not exposed
  Access Control: Failed - Access control not enforced
Entry 5:
  Encryption: Passed - Data properly encrypted
  Data Exposure: Passed - Data not exposed
  Access Control: Failed - Access control not enforced

--- Summary ---
Total Cost: 4165.00 units
Total Time: 583 milliseconds
brokrn@Udiths:~/fabric-samples/smart-home-application$
```

```
--- Privacy Evaluation (First 5 Entries) ---

--- Performance and Cost Evaluation ---

--- Privacy Evaluation Results ---
Entry 1:
  Encryption: Passed - Data properly encrypted
  Data Exposure: Passed - Data not exposed
  Access Control: Failed - Access control not enforced
Entry 2:
  Encryption: Passed - Data properly encrypted
  Data Exposure: Passed - Data not exposed
  Access Control: Failed - Access control not enforced
Entry 3:
  Encryption: Passed - Data properly encrypted
  Data Exposure: Passed - Data not exposed
  Access Control: Failed - Access control not enforced
Entry 4:
  Encryption: Passed - Data properly encrypted
  Data Exposure: Passed - Data not exposed
  Access Control: Failed - Access control not enforced
Entry 5:
  Encryption: Passed - Data properly encrypted
  Data Exposure: Passed - Data not exposed
  Access Control: Failed - Access control not enforced

--- Summary ---
Total Cost: 4165.00 units
Total Time: 675 milliseconds
brokrn@Udiths:~/fabric-samples/smart-home-application$ node app.js
```

Figure 35: Batch0 and Batch1 implementation in Hyperledger Fabric.

7. Comparative Analysis:

This part of the report explores the performance of the blockchain platforms and conducts a comparative analysis. As mentioned in the previous sections of the report, the evaluation is based upon privacy controls, cost and transaction efficiency. I have also plotted various graphs which will make the comparison more elaborate.

7.1 Privacy evaluation comparison:

When it comes to encryption, Ethereum provides basic encryption but to enhance it, I have used IPFS which is an off-chain solution to ensure data privacy in smart homes. The IPFS was originally deployed to reduce gas costs but later also helped in data privacy. On the other hand, Hyperledger Fabric supports robust built-in encryption techniques that are further tailored by the user in order to enhance security and secure sensitive data produced by smart homes.

Being a public blockchain, Ethereum will always expose all transaction data unless additional privacy layers are implemented. Hence, it might require extra setup out of the box for smart home applications. But the data inside the blockchain is completely secure and does not show any sign of exposure. On the contrary, Hyperledger Fabric's permissioned model ensures that the data in it is only accessible to users that are authorised to view it. This minimizes data exposure.

Finally, access controls mechanisms from the evaluation showed that Ethereum passed but it lacks granular access control mechanisms which could potentially expose the smart home data to unauthorized access. But in the case of Hyperledger, it provides strong access control which ensures secure interactions in the network within roles that are predefined.

7.2 Gas Price:

For Ethereum, the gas price involved in processing both the batches (batch0 and batch1) approximately come around 0.0000000025 ETH. This is because Ethereum platform incurs gas fees as the transaction cost because it is a public blockchain with decentralization. This can be seen as a significant overhead in handling frequent data transactions generated by smart homes.

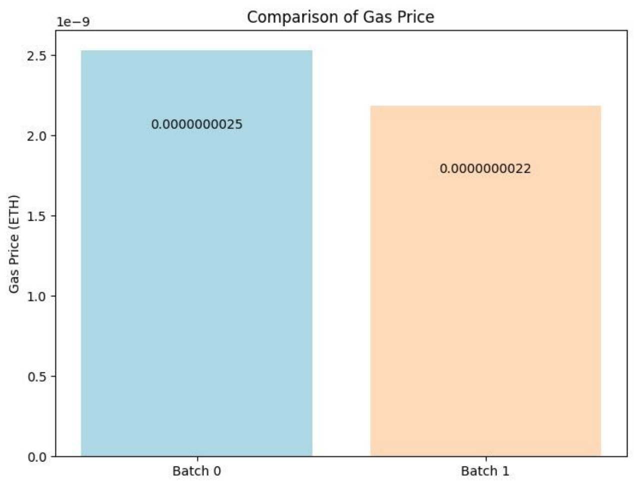


Figure 16: Ethereum gas fees comparison.

On the other hand, Hyperledger Fabric does not involve any gas fees since it is a permissioned blockchain model in which operations are free. Ethereum's gas fees make it a less viable option because of its gas fees when it comes to smart home applications because they produce a huge amount of data. Hyperledger Fabric's cost-free architecture might serve as a better option when it comes to smart home environments.

7.3 Transaction Time:

The transaction time of the two batches in Ethereum implementation came to 0.162 seconds for batch0 and 0.143 for batch1. Even though both the batches had equal number of entries, the transaction time in Ethereum was variable and was majorly influenced by network congestion. This could affect the real time data needs of smart homes. Due to its private and controlled network, Hyperledger was able to be consistently faster across both the datasets. Hyperledger's predictable and faster transaction time makes it more suitable for real-time smart home data processing compared to Ethereum.

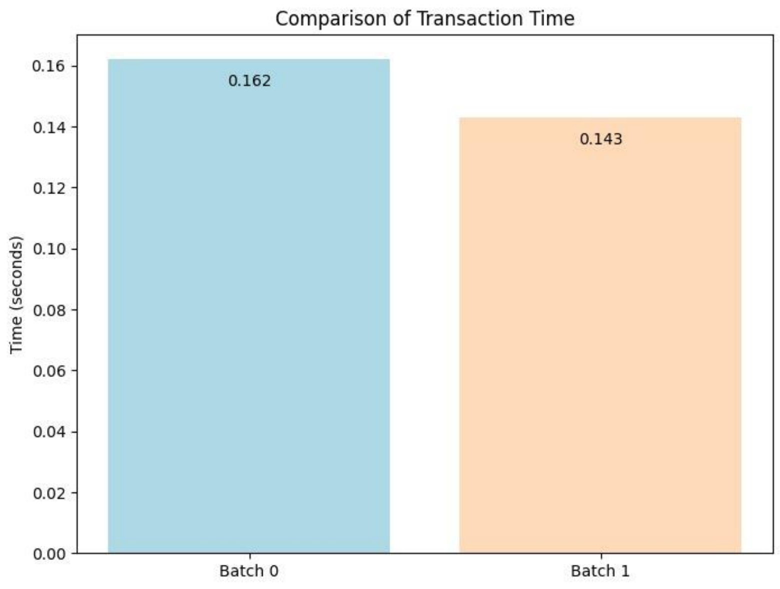


Figure 17: Ethereum transaction time.

8. Future Work:

This research has provided a proper comparative analysis of two prominent blockchain solutions available in todays market, Ethereum and Hyperledger Fabric. While this study shows that blockchain can effectively enhance data privacy in smart home scenarios, there were several challenges which was worked around instead of being given a permanent solution. These challenges can server as the future work from this research. To begin with, Ethereum public architecture and the gas costs proved to be the greatest obstacle in the implementation. The dataset had to be split into batches and further had to be deployed into an off-chain storage to reduce the overhead. Future work can address this issue and help in deploying the large-scale dataset into the public infrastructure and evaluate it in depth. Furthermore, various techniques could be implemented in Ethereum like Zero-Knowledge Proofs (ZKPs) to enhance the privacy protocols. Other than that, ethical considerations were one of the key factors for choosing a simulated dataset. Testing the blockchain solutions in real-world smart home environments, rather than simulated datasets, can provide more accurate insights into their performance and usability. Real-world deployments will also help identify unforeseen challenges and refine the implementation.

9. Conclusion:

This research provided me with a great learning curve. Apart from cybersecurity, various other domains such as computer architecture, application development and blockchain technology was also involved in completing this research successfully. In the beginning, the aim was to evaluate three blockchain platforms, but realistically it was impossible given the learning curve and the time in hand. Considering the other setbacks like budget and finding the right tool for smart home simulation, the end result is satisfying. Other than this, the findings of this study underline the critical trade-offs between the decentralization and openness of public blockchains like Ethereum and the controlled, performance-oriented nature of private blockchains like Hyperledger Fabric. If I could go back and start over, the first thing that I will do is find a better way to simulate the smart home setup and get better dataset. Moreover, I would also find a way to test the Ethereum public network with lesser gas fees by finding a cheaper and more optimal test network.

10. References:

A Blockchain Platform for the Enterprise — Hyperledger Fabric Docs main documentation. (n.d.). Retrieved 12 December 2024, from https://hyperledger-fabric.readthedocs.io/en/release-2.5/

Alkhateeb, A., Catal, C., Kar, G., & Mishra, A. (2022). Hybrid Blockchain Platforms for the Internet of Things (IoT): A Systematic Literature Review. *Sensors 2022, Vol. 22, Page 1304*, *22*(4), 1304. https://doi.org/10.3390/S22041304

Alshammari, N., Alshammari, T., Sedky, M., Champion, J., & Bauer, C. (2017). OpenSHS: Open Smart Home Simulator. *Sensors 2017, Vol. 17, Page 1003*, *17*(5), 1003. https://doi.org/10.3390/S17051003

Ammi, M., Alarabi, S., & Benkhelifa, E. (2021). Customized blockchain-based architecture for secure smart home for lightweight IoT. *Information Processing & Management*, *58*(3), 102482. https://doi.org/10.1016/J.IPM.2020.102482

Arif, S., Arif Khan, M., Rehman, U. R., Kabir, M. A., & Imran, M. (2020). Investigating Smart Home Security: Is Blockchain the Answer?; Investigating Smart Home Security: Is Blockchain the Answer? *IEEE Access*, *8*. https://doi.org/10.1109/ACCESS.2020.3004662

Buterin, V. (2013). *A NEXT GENERATION SMART CONTRACT & DECENTRALIZED APPLICATION PLATFORM [White Paper]*.

Dorri, A., Kanhere, S. S., Jurdak, R., & Gauravaram, P. (2017). Blockchain for IoT security and privacy: The case study of a smart home. *2017 IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops 2017*, 618–623. https://doi.org/10.1109/PERCOMW.2017.7917634

El-Sayed, A., & Ibrahim, F. (2023). Comparative Study of Blockchain-Based Approaches for Securing Internet of Things (IoT) Devices. *Journal of Intelligent Connectivity and Emerging Technologies*, *8*(3), 72–93. https://questsquare.org/index.php/JOUNALICET/article/view/20

Guo, H., & Yu, X. (2022). A survey on blockchain technology and its security. *Blockchain: Research and Applications*, *3*(2), 100067. https://doi.org/10.1016/J.BCRA.2022.100067

Lee, Y., Rathore, S., Park, J. H., & Park, J. H. (2020). A blockchain-based smart home gateway architecture for preventing data forgery. *Human-Centric Computing and Information Sciences*, *10*(1), 1–14. https://doi.org/10.1186/S13673-020-0214-5/TABLES/1

Liao, K. (2022). Design of the Secure Smart Home System Based on the Blockchain and Cloud Service. *Wireless Communications and Mobile Computing*, *2022*(1), 4393314. https://doi.org/10.1155/2022/4393314

Moniruzzaman, M., Khezr, S., Yassine, A., & Benlamri, R. (2020). Blockchain for smart homes: Review of current trends and research challenges. *Computers & Electrical Engineering*, *83*, 106585. https://doi.org/10.1016/J.COMPELECENG.2020.106585

Murphy, J. (2021). *Blockchain and IOT Security – Can an Ethereum Blockchain be the solution for securing IOT devices.*

Nakamoto Satoshi. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*.

Panarello, A., Tapas, N., Merlino, G., Longo, F., & Puliafito, A. (2018). Blockchain and IoT Integration: A Systematic Survey. *Sensors 2018, Vol. 18, Page 2575*, *18*(8), 2575. https://doi.org/10.3390/S18082575

Pancari, S., Rashid, A., Zheng, J., Patel, S., Wang, Y., & Fu, J. (2023). A Systematic Comparison between the Ethereum and Hyperledger Fabric Blockchain Platforms for Attribute-Based Access Control in Smart Home IoT Environments. *Sensors 2023, Vol. 23, Page 7046*, *23*(16), 7046. https://doi.org/10.3390/S23167046

Patruni, M. R., & Saraswathi, P. (2022). Securing Internet of Things devices by enabling Ethereum blockchain using smart contracts. *Building Services Engineering Research and Technology*, *43*(4), 473–484. https://doi.org/10.1177/01436244221078933

Qashlan, A., Nanda, P., He, X., & Mohanty, M. (2021). Privacy-Preserving Mechanism in Smart Home Using Blockchain. *IEEE Access*, *9*, 103651–103669. https://doi.org/10.1109/ACCESS.2021.3098795

Ren, L., Zhou, H., Hang, X., Yang, B., & Su, L. (2022). Research on Performance Optimization and Application in Smart Home for Hyperledger Fabric. *Sensors 2022, Vol. 22, Page 3222*, *22*(9), 3222. https://doi.org/10.3390/S22093222

Saraf, C., & Sabadra, S. (2018). Blockchain platforms: A compendium. *2018 IEEE International Conference on Innovative Research and Development, ICIRD 2018*, 1–6. https://doi.org/10.1109/ICIRD.2018.8376323

Uddin, M. A., Stranieri, A., Gondal, I., & Balasubramanian, V. (2021). A survey on the adoption of blockchain in IoT: challenges and solutions. *Blockchain: Research and Applications*, *2*(2), 100006. https://doi.org/10.1016/J.BCRA.2021.100006

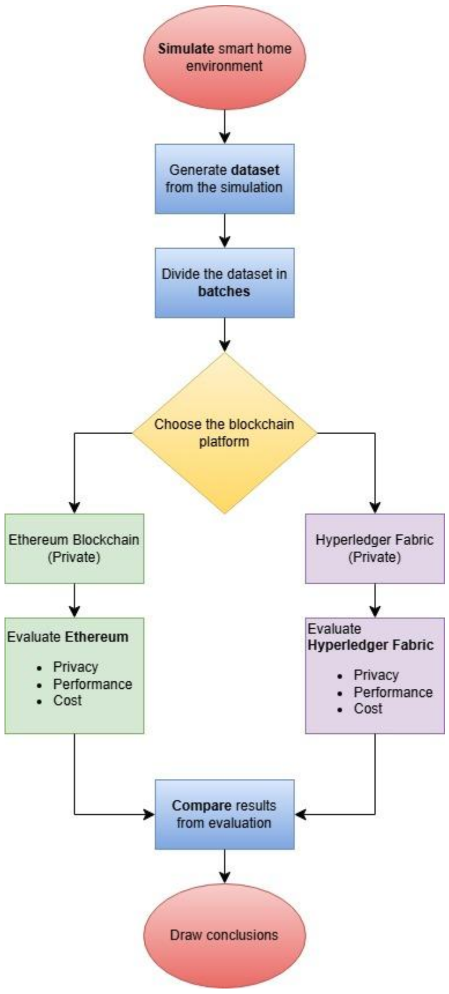
Urmila, M. S., Hariharan, B., & Prabha, R. (2019). A Comparitive Study of Blockchain Applications for Enhancing Internet of Things Security. *2019 10th International*

Conference on Computing, Communication and Networking Technologies, ICCCNT 2019. https://doi.org/10.1109/ICCCNT45670.2019.8944446

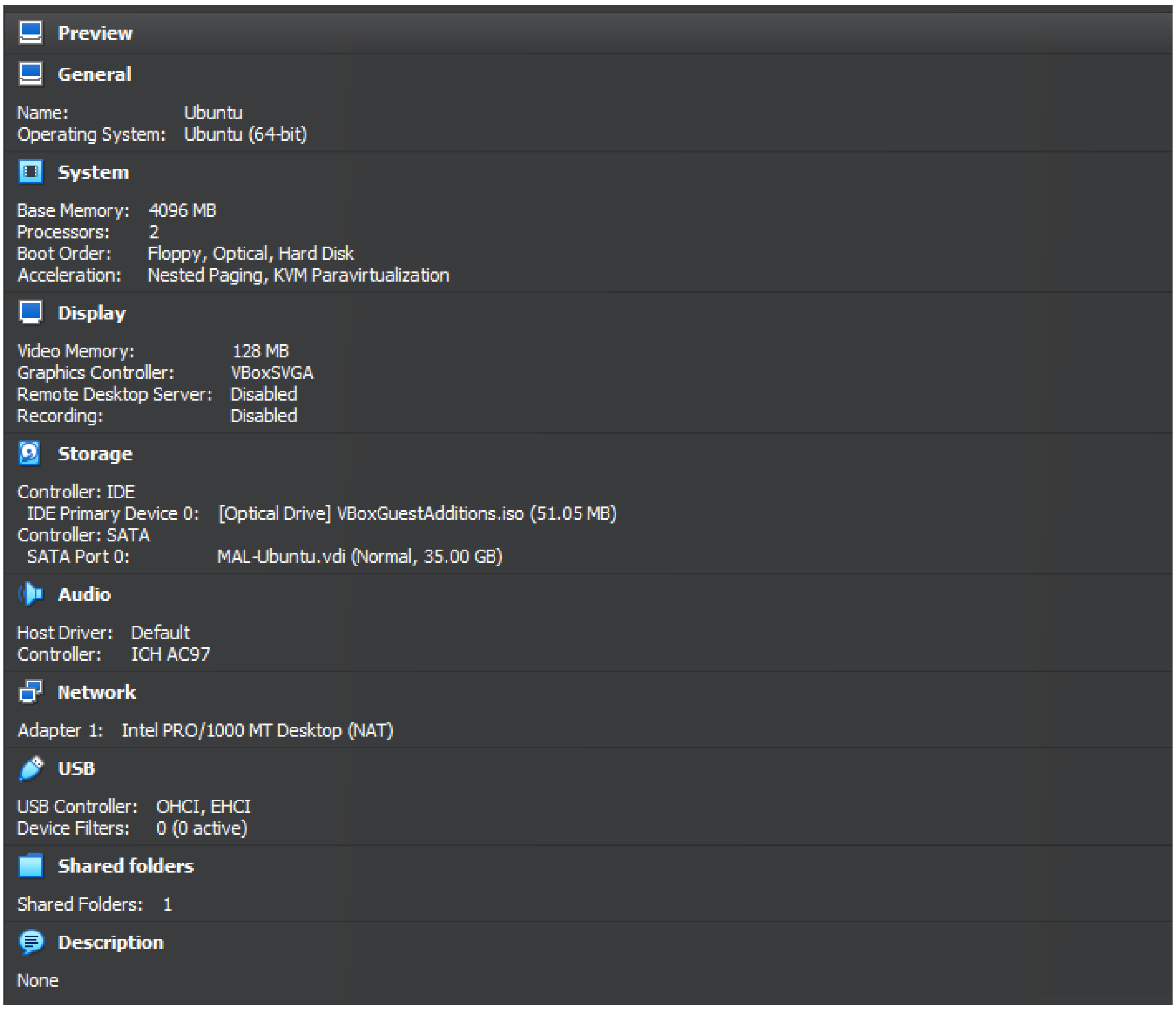
Xu, Q., He, Z., Li, Z., & Xiao, M. (2018). Building an Ethereum-Based Decentralized Smart Home System. *Proceedings of the International Conference on Parallel and Distributed Systems - ICPADS*, *2018-December*, 1004–1009. https://doi.org/10.1109/PADSW.2018.8644880

Yang, Q., & Wang, H. (2021). Privacy-Preserving Transactive Energy Management for IoT-Aided Smart Homes via Blockchain. *IEEE Internet of Things Journal*, *8*(14), 11463–11475. https://doi.org/10.1109/JIOT.2021.3051323

11. Appendix:

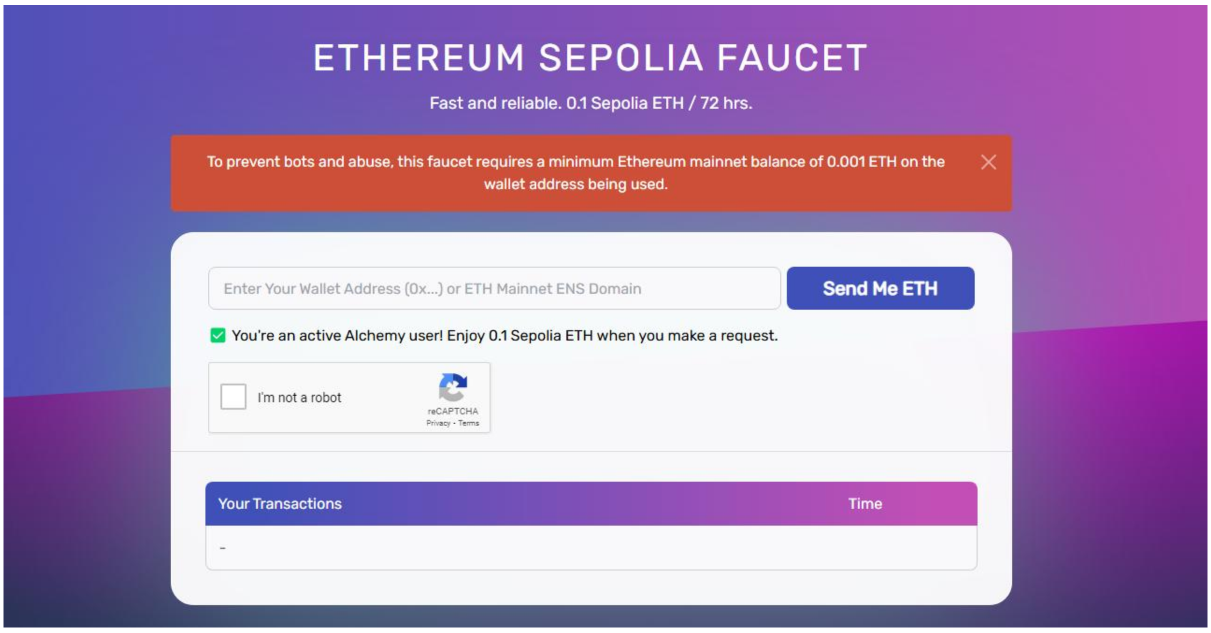

Simulate smart home environment
Generate dataset from the simulation
Divide the dataset in batches
Choose the blockchain platform
Ethereum Blockchain (Private)
Hyperledger Fabric (Private)
Evaluate Ethereum
Privacy
Performance
Cost
Evaluate Hyperledger Fabric
Privacy
Performance
Cost
Compare results from evaluation
Draw conclusions

Appendix 1: Research methodology flowchart



```
Preview
General
Name:              Ubuntu
Operating System:  Ubuntu (64-bit)
System
Base Memory:   4096 MB
Processors:    2
Boot Order:    Floppy, Optical, Hard Disk
Acceleration:  Nested Paging, KVM Paravirtualization
Display
Video Memory:            128 MB
Graphics Controller:     VBoxSVGA
Remote Desktop Server:   Disabled
Recording:               Disabled
Storage
Controller: IDE
 IDE Primary Device 0:   [Optical Drive] VBoxGuestAdditions.iso (51.05 MB)
Controller: SATA
 SATA Port 0:            MAL-Ubuntu.vdi (Normal, 35.00 GB)
Audio
Host Driver:   Default
Controller:    ICH AC97
Network
Adapter 1:   Intel PRO/1000 MT Desktop (NAT)
USB
USB Controller:   OHCI, EHCI
Device Filters:   0 (0 active)
Shared folders
Shared Folders:   1
Description
None
```

Appendix 2: Virtual Machine Configuration.



ETHEREUM SEPOLIA FAUCET

Fast and reliable. 0.1 Sepolia ETH / 72 hrs.

To prevent bots and abuse, this faucet requires a minimum Ethereum mainnet balance of 0.001 ETH on the wallet address being used.

You're an active Alchemy user! Enjoy 0.1 Sepolia ETH when you make a request.

Your Transactions	Time
-	

Appendix 3: Sepolia faucet to buy test coins.