

AN ETHEREUM-BASED DEGREE CREDENTIAL STORE

A DISSERTATION SUBMITTED TO THE UNIVERSITY OF MANCHESTER FOR
THE DEGREE OF MASTER OF SCIENCE
IN THE FACULTY OF COMPUTER SCIENCE

2021

10655496

Huanjie Guo

School of Computer Science

CONTENT

LIST OF TABLES	4
LIST OF FIGURES	5
LIST OF ABBREVIATIONS	6
ABSTRACT	7
DECLARATION	8
COPYRIGHT	9
ACKNOWLEDGEMENTS	10
1 INTRODUCTION	11
1.1 MOTIVATION	11
1.2 AIM AND OBJECTIVES	12
1.3 OVERVIEW OF DISSERTATION	13
2 LITERATURE REVIEW	14
2.1 BLOCKCHAIN	14
2.1.1 <i>What is blockchain</i>	14
2.1.2 <i>Classification of blockchain</i>	23
2.1.3 <i>Blockchain Features</i>	25
2.2 ETHEREUM	26
2.2.1 <i>What is Ethereum</i>	26
2.2.2 <i>Ethereum hierarchy</i>	29
2.2.3 <i>Smart Contract</i>	33
2.3 BLOCKCHAIN-BASED STORE	34
3 METHODOLOGY	37
3.1 CONSIDERATIONS	37
3.2 DATA DESIGN	41
3.2.1 <i>Data in the database</i>	41
3.2.2 <i>Data on Ethereum</i>	41
3.3 SYSTEM DESIGN	44
3.4 SMART CONTRACT DESIGN	45
3.4.1 <i>Voting</i>	45
3.4.2 <i>Add token</i>	46
3.4.3 <i>Issue Certificate</i>	47

3.4.4 <i>Revoke Certificate</i>	48
3.4.5 <i>Request Hash</i>	48
4 DEPLOYMENT	49
4.1 MYSQL	49
4.1.1 <i>Introduction</i>	49
4.1.2 <i>setting</i>	49
4.2 GETH	50
4.2.1 <i>Introduction</i>	50
4.2.2 <i>genesis.json</i>	50
4.2.3 <i>Initialisation</i>	51
4.3 METAMASK	52
5 TEST & EVALUATION	54
5.1 TEST	54
5.1.1 <i>Contract Deployment</i>	55
5.1.2 <i>Voting</i>	56
5.1.3 <i>Add University</i>	59
5.1.4 <i>Degree Storage</i>	60
5.2 EVALUATION	62
6 CONCLUSION & FUTURE WORK	65
6.1 CONCLUSION	65
6.2 FUTURE WORK	65
REFERENCES	67

Word Count: 14203

LIST OF TABLES

Table 1 Three types of blockchain	24
Table 2 Some factors need to consider	38
Table 3 The table of degree certificate	41
Table 4 The structure of a certificate on the Ethereum	43
Table 6 The role of five addresses (idx.1)	54
Table 7 The role of five addresses (idx.2)	56
Table 8 The role of five addresses (idx.3)	57
Table 9 The role of five addresses (idx.4)	58
Table 10 The role of five addresses (idx.5)	60
Table 11 Performance of three plans	64

LIST OF FIGURES

Figure 1 The structure of block	14
Figure 2 How to generate an address	15
Figure 3 The structure of Merkel tree	16
Figure 4 The Peer to Peer network	18
Figure 5 Five layers model of Ethereum	29
Figure 6 The process of issuing a degree certificate	40
Figure 7 How the data come	43
Figure 8 An overview of this system	44
Figure 9 The table of degree certificates in MySQL	50
Figure 10 The console of Geth client	52
Figure 11 The Metamask wallet	53
Figure 12 Deploy the smart contract	55
Figure 13 Sending transactions to call the smart contract	57
Figure 14 The result of searchIfAuthority function	58
Figure 15 The result of getUniversity function	59
Figure 16 Degree certificate in MySQL	60
Figure 17 The hash of the degree certificate	61
Figure 18 The result of viewDegreeHash function	61
Figure 19 The result of revokeDegree function	62

LIST OF ABBREVIATIONS

PoW	Proof of Work
PoS	Proof of Stake
DPoS	Delegated Proof of Stake
ETH	Ether coin
BTC	Bitcoin
P2P	Peer to Peer

ABSTRACT

The real-time performance of data verification has long been a concern in the realm of credential storage. Nowadays, conventional centralized verification relies too much on the central node. However, a single server may fail, resulting in data loss and service interruption.

This study presents a degree credential storage mechanism based on Ethereum that makes use of blockchain technology to enable distributed credential storage and verification. In this distributed network, Ethereum wallet addresses will be classified as authority addresses, university addresses, and normal addresses. The private keys of each address are owned by different organizations for a variety of procedures such as university validation, degree issuing, degree revocation, and degree verification. Firstly, the authority address is in charge of authorizing university addresses and cancelling university addresses. This network's authority addresses use a voting mechanism to elect and deprecate other authority addresses. Secondly, the university address is in charge of the issuing and revocation of graduate degrees. Finally, the normal address can be used to query the blockchain and validate the certificate's authenticity.

The system was put through its paces and evaluated. It employs blockchain technology and additional servers to safeguard users' privacy, achieve real-time authentication, and lessen the storage burden on Ethereum nodes.

DECLARATION

None of the work in this thesis has been submitted for the degree application or qualification of this university or other institutions.

COPYRIGHT

i. The author of this dissertation (including any appendices and/or schedules to this dissertation) owns certain copyright or related rights in it (the “Copyright”), and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii. Copies of this dissertation, either in full or in extracts and whether in hard or electronic copy, may be made only in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has entered into. This page must form part of any such copies made.

iii. The ownership of certain Copyright, patents, designs, trademarks and other intellectual property (the “Intellectual Property”) and any reproductions of copyright works in the dissertation, for example, graphs and tables (“Reproductions”), which may be described in this dissertation, may not be owned by the author and may be owned by third parties. Such intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv. Further information on the conditions under which disclosure, publication and commercialization of this dissertation, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see <http://documents.manchester.ac.uk/display.aspx?DocID=24420>), in any relevant Dissertation restriction declarations deposited in the University Library, and The University Library’s regulations (see http://www.library.manchester.ac.uk/about/regulations/_files/Library-regulations.pdf).

ACKNOWLEDGEMENTS

I would like to express my gratitude to my parents and friends for their financial and spiritual support in studying at the University of Manchester. During this time, their encouragement enabled me to overcome the challenges I faced.

Also, I want to thank my tutor, Dr Richard Banach, from the bottom of my heart. Thank you for your direction and advice, which has improved my learning skill. Your assistance and patience mean a lot to me.

Furthermore, owing to the open-source system's authors, I may learn the most advanced expertise. I appreciate them for sharing their accomplishments.

Finally, I'd like to thank all of my classmates and teachers at the University of Manchester for enriching my Master's experience.

1 INTRODUCTION

1.1 Motivation

As technology has advanced, we have gradually progressed from storing paper copies to using computers to store critical texts. This makes information easier to handle and preserve, and it speeds up data transport. However, putting text data on the internet raises a number of concerns. For example, who is the information's source? Do you have the authorization to keep the data? Is the data that has been saved genuine? Has a third party maliciously altered the stored data? There are a number of questions that must be answered.

There have been numerous examples of degree fabrication in recent years. For example, Mary Jones, the president of MIT, resigned in 2007 after revealing that she had faked degrees. Mary Jones forged three different degrees to earn a teaching certificate at MIT. However, this could have been averted if MIT had been able to verify the legitimacy of these certificates. Furthermore, a BBC investigation in 2014 discovered a website that sold counterfeit University of Kent degrees for as little as £500. This happens all the time, and we need to find a solution that works. In today's employment market, where candidates come from good backgrounds, interviewer selection will be more challenging. Usually, most companies will focus their ultimate decision on the interviewer's qualifications. As a result of the value of education, some interviewers began to fabricate their degree certificates to acquire lucrative job opportunities. The result is that employer hires a dishonest employee for the company. On the other hand, those who obtain positions with bogus education deprive their competitors of their efforts. According to one report, 33% of job searchers in the market lie about their schooling. Seventy-three percent of them go undiagnosed.

According to a university professor, many degrees are falsified in the job market since they are difficult to verify. In the current job market, recruiters must contact the university that provided the degree and wait for confirmation. As a result, university faculty members may take longer to react, sometimes up to a month. Obviously, it is a lengthy procedure for

hiring firms. Is it possible to make it easy for recruiters to verify employees' degree certificates?

With the introduction of Bitcoin in 2008, the blockchain, a decentralized and trusted transaction system based on electronic virtual currency, was born. Bitcoin's prominence in the financial and technical areas has sparked broad interest, and academics have delved deeper into blockchain technology. With the continual invention and growth of blockchain technology in recent years, there has been a surge in the study of blockchain. Also, blockchain-based apps expand from the original electronic currency to smart contracts by writing code and deploying it to the blockchain, allowing the programs to execute on several nodes. The introduction of blockchain technology has shattered old transactional understandings, most notably with smart contracts. These scripting languages can be run on Ethereum virtual machines. According to the idea of smart contracts, the authenticity of degree credentials may be accurately confirmed if they are recorded in a completely decentralized blockchain. As a result, recruiters would no longer need to contact universities, and verification processes could be carried out via Ethereum smart contracts. The author proposed this essay based on this concept.

1.2 Aim and Objectives

The goal of this project is to create a system that can rapidly validate the validity of degree credentials on Ethereum. After the students graduate, we want the institution to keep their graduation certificates in this system. When a third party wants to validate the credentials, they simply need to do so through blockchain. Simultaneously, we want to minimize the quantity of data kept on the blockchain while maintaining the privacy of credential information. Because blockchain is a completely decentralized system, we need a set of procedures to guarantee its correct functioning.

To accomplish these aims, we must perform the following tasks.

- To get an understanding of blockchain and Ethereum since the entire architecture is built on Ethereum.

- Understand the current design solutions of the educational system and the technology they use, as well as their benefits and drawbacks.
- Propose my system based on the previous ones' design concepts. In this system, it is essential to optimize the functionality of each component while also introducing new ones.
- Validate the system and see whether it matches the expectations. Compare to other systems and weigh the benefits and drawbacks of each.

1.3 Overview of Dissertation

The first chapter is titled "Introduction." It is an introduction chapter that will provide an outline of the situation and what has to be done. It also introduces the overall structure of the article.

The second chapter is a review of the literature. This chapter will explain and evaluate the evolution of blockchain, Ethereum, and education systems, as well as comparing their benefits and drawbacks, before introducing the project we will perform in this paper.

System Design is covered in Chapter 3. We will discuss many critical issues that must be addressed while building this system, such as storage pressure, privacy, and so on. We will also talk about the design of the database tables and the data storage mechanism in the smart contract. Finally, we examine the code for our smart contract.

Deployment is covered in Chapter 4. A MySQL database, an Ethereum client, and a MetaMask Ethereum wallet are all part of this project. All of them require a network, port, and other setups.

Test and evaluation are covered in Chapter 5. In this chapter, we will put the created system to the test. We put a scenario where an authority address is cancelled via voting and a credential storage case. We also compare the benefits and drawbacks of this storage option to those of other alternatives.

Conclusion and Future Work is in Chapter 6. This section summarizes the whole study and proposes a few system specifics that may be improved in the future.

2 LITERATURE REVIEW

2.1 Blockchain

2.1.1 What is blockchain

Nakamoto (2008) originally proposed his idea "Bitcoin: A Peer-to-Peer Electronic Cash System", published on the Bitcoin Forum. He suggested that a ledger might be established by hashing a collection of data in the form of a block and broadcasting it, with each block carrying the preceding block's hash, to address the security issue of electronic money. Finally, the blocks would be linked to create a "blockchain." This technology may be used in a variety of areas, including financial services and social interactions. Bitcoin is a classic example of the first large-scale use of blockchain technology.

2.1.1.1 Block

Firstly, a block is a data structure containing a block header and block content used to record Bitcoin transactions. Secondly, blocks are linked by hashing the block header data. Thirdly, blockchain utilizes the block header data as a unique identifier for all blocks, and the hash of the parent block may be used to locate the uniquely connected blocks in the blockchain. This generates a chain of data structures that use hashes to track the most recent block back to the first block. Figure 1 depicts the structure of a block.

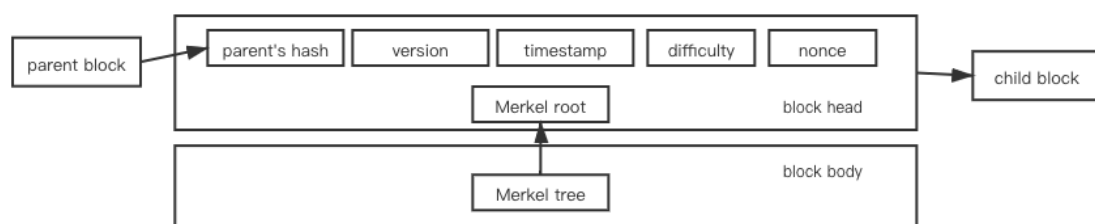


Figure 1 The structure of block

Satoshi Nakamoto created the Bitcoin system with a block-out time of about 10 minutes. If the preceding block's calculation time was relatively quick, the system would make the

following block more complex to calculate. In contrast, if the calculation time is excessively lengthy, the following one will be easier. Finally, the average time will be guaranteed to be 10 minutes. Satoshi's design, in my opinion, is not very logical. The maximum transaction verification time from issuance to all nodes adding this block to the blockchain is maybe 10 minutes, which is uncomfortable daily. Therefore, different block-out durations in a credential storage system must be selected to meet the system's requirements (Kraft 2016).

2.1.1.2 Address

Satoshi created the idea of address, which is similar to a blockchain account for holding assets. Firstly, we create a 256-bit private key using a random seed and then calculate the matching public key. Secondly, by using a hashing technique to the public key, we may get the address. Figure 2 depicts the calculation of the public key to the address as an irreversible one-way function. Satoshi created the address rather than utilizing the public key to identify the account. As a result, the receiver just has to publish his address rather than the public key. The first reason is that the public key is 256 bits long, whereas the address is just 160 bits long; thus, utilizing the address is more succinct and saves space. Calculating the private key from the public key, on the other hand, is theoretically possible but computationally impossible, and the Elliptic Curve method is unsure whether there is a backdoor (Latifa et al. 2017). Suppose a weakness in the algorithm is discovered after utilizing the address. In that case, those with the key still benefit from altering the algorithm since the user only exposes the address, not the public key.



Figure 2 How to generate an address

2.1.1.3 Merkle Tree

A Merkle Tree is a Bitcoin data structure that generates a Merkle root value for the whole transaction using a hashing method. This data structure may be used to securely store and verify the accuracy of transactions. What's more, Merkle Tree is a bottom-up, hash-based binary tree paradigm. Firstly, each leaf node in the tree represents a transaction that corresponds to a particular hash. Secondly, two neighbouring hashes are combined again to get both nodes' parents (Hutchison and Mitchell 1973). Layer after layer until the tree has just one hash. As illustrated in Figure 3, there are four transactions in the block data, beginning with the hash of the data in cells A to D to produce HashA, HashB, HashC, and HashD. After that, the hashes of two neighbouring nodes, HashAB and HashCD, were computed and stacked until the hash of the root node, HashABCD, was determined. Finally, 32 bytes are generated and placed in the block header. A most important benefit of this tree is that if the underlying data is tampered with maliciously, the associated leaf node hashes will likewise change, resulting in their Merkle root values. When determining if a transaction is included in the transaction list, a node may be validated by simply calculating the hash of $\log_2 N$ nodes (Szydło 2004). Using Merkle Tree to store transaction data lowers data transmission and calculation costs during transaction validation.

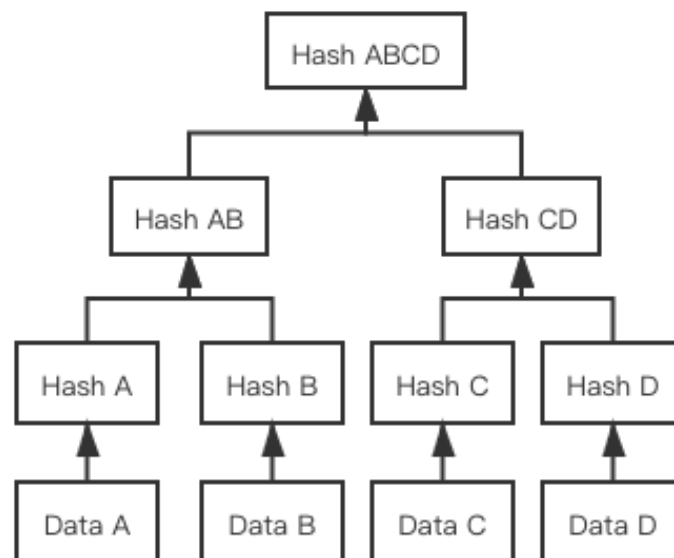


Figure 3 The structure of the Merkel tree

2.1.1.4 Signature

To encrypt a message, asymmetric encryption employs a pair of public and private keys. Firstly, the public key is used to encrypt the communication, while the private key is decryption. Secondly, the information encrypted by the private key may be verified using the public key. As a result, a digital signature guarantees that messages are securely sent. The sender hashes the message and then encrypts it using the private key. When the message is received, the text is decrypted using the public key and compared to the digital signature. If they are the same, it means that the text has not been tampered with (Merkle 1988).

For example, firstly, Alice creates and publishes her own private and public keys. Secondly, Alice conducts a transaction, obtains the hash of the message offline, signs the hash with her private key, and then places the text and signature on the blockchain. When Bob wants to verify, he must first decode the hash value using Alice's public key and then get the plaintext hash value to check if it is consistent. This architecture enables both parties to validate the message's validity without revealing their private keys. Transactions in the Bitcoin system are signed using ECDSA (Johnson et al. 2001).

2.1.1.5 Peer to Peer

Figure 4 depicts how the peer-to-peer protocol is primarily utilized as a network protocol for blockchain decentralized systems. Firstly, there is no centralized server in a P2P network, and nodes are functionally identical, either a client or a server. Secondly, each node may offer data services to other nodes and consume the data services of other nodes, which no longer needs to request data from the centralized server (Schollmeier 2001). When some network nodes fail or are attacked, the network data remains unaffected. Obviously, it ensures the data's security and integrity.

What's more, there are structured and unstructured peer-to-peer networks. Structured P2P networks utilize consistent hash tables to build routing tables for each node. As for an unstructured P2P network, it broadcasts routes between nodes and transmits data to its neighbouring nodes.

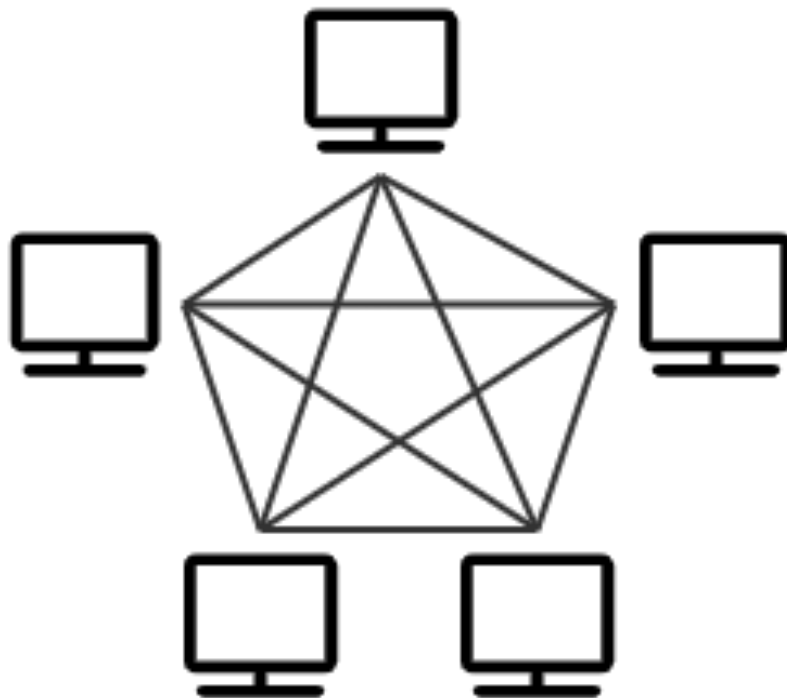


Figure 4 The Peer to Peer network

The P2P network system is a decentralized, de-trusted, data-reliable, and collaborative collaboration network system (Irestig et al. 2005), and the Ethereum blockchain has the following properties.

Decentralization. P2P networks are dispersed networks with many nodes monitoring each other to prevent over-centralization of power and enhance data security.

De-trusting. There is no central node in Ethereum, and the nodes create a trust connection with each other through digital signature to guarantee the dependability of transactions, resulting in high-reliability data transmission between nodes in the distributed network.

Data is trustworthy. All nodes in the distributed network constantly update and backup blockchain data, and if all nodes are damaged or attacked at the same time, the system

data is lost. The integrity of the whole system data is not jeopardized when a single node is attacked.

Cooperation. To guarantee that all network nodes may participate in the data identification process, Ethereum employs an incentive system, with a consensus method used to choose a node to add a block to the blockchain.

2.1.1.6 Timestamp

The timestamp represents the seconds from Jan 1, 1970, to now. Therefore, a timestamp is similar to a time signature (Dyreson and Snodgrass 1993). Similarly, timestamps in blockchain technology provide a temporal component to the blockchain, making data more traceable. Simultaneously, timestamps may serve as critical criteria for proof of existence, confirming the presence of certain data at a given point in time. This makes blockchain databases tamper-evident and unfalsifiable.

2.1.1.7 Hash

The hashing method can map an arbitrary length piece of information into a fixed-length string. First, if two pieces of information are the same, their hash values are the same, too. Second, even if two pieces of information are extremely similar, as long as they are distinct, the hash values between these two strings are unlikely to be the same (Carter and Wegman 1979). What's more, we cannot reverse the input data from the hash value, and the hash value may completely reflect the compressed data content. Although two different files can have the same hash value, this is uncommon. Therefore, the hashing algorithm's goal is to extract data characteristics. Similarly, we may consider the hash value to be the data's fingerprint.

2.1.1.8 Consensus Algorithm

The authentication concept of achieving agreement among all nodes is called the consensus algorithm (Cao et al. 2019). Through this approach, blockchain applications do not need to depend on a central authority to identify and validate items or transactions. What's more, consensus algorithms can decrease the incidence of fraudulent transactions. For example,

transactions may occur only when more than 51 percent of node members achieve consensus, favouring information consistency. Also, synchronizing all computational nodes is critical in distributed computers.

PoW

Bitcoin, the blockchain technology's forefather, employs a proof-of-work consensus method that is extensively utilised in other blockchain systems. Firstly, each node competing for processing power in the PoW algorithm is referred to as a miner, and the act of solving for random numbers is referred to as mining. When a transaction is received, the miner begins a new mining round. This mining process is a solution to a mathematical problem. At the same time, miners conduct hashing operations while producing block header information. Since the block header includes particular information to represent the block, the block's hash value must be distinct in each mining operation. Before each mining, the system computes a target value based on the difficulty parameter. Each time, the miner produces a random number and compares it to the goal. As a result, the mine is deemed successful if it is smaller than the desired value. Otherwise, the miner will recycle the random number until it falls below the desired value. Because SHA3 is an irreversible hashing method, obtaining a hash smaller than the goal value requires continual calculation solely linked to computer capacity (Gervais et al. 2016). Therefore, computing power may reflect the speed with which the issue is solved. Although the mining process takes time, the verification procedure is simple. It requires just hashing the random number with the block header to verify whether it is less than the goal value. As we know, the whole network would compete in computing the random number through the PoW generalisation process. If certain miners control more than 50% of the network's computational pool to mine, they may control the entire blockchain. However, this is almost difficult in a public chain because the PoW consensus algorithm can maintain network node consistency. To encourage miners to mine, Satoshi mixes the PoW consensus method with an incentive algorithm. In Bitcoin, the total processing power of the Bitcoin network surpassed the sum of the world's top 500 supercomputers in 2016. Simultaneously, the system is progressively revealing its unreliability, as more and more organisations have gained more computer

capacity in recent years. When one organisation gains 50% of the processing power in the whole network, this blockchain will become untrustworthy.

PoS

In 2010, Satoshi proposed the idea of money age, which resulted from a computation of the quantity of cash and the time spent holding it. The Proof of Stake method is founded on the idea of currency age. It also guarantees the blockchain's decentralized and trustworthy operation (Kiayias et al. 2017). Unlike the PoW consensus process, producing a block under PoS needs the input of a specific quantity of money, which is required to verify the node's ownership of the currency. When producing blocks, these currencies influence the difficulty parameter of the mining process. What's more, Proof of Interest defines a new kind of transaction. Unlike conventional transactions, interest coin transactions use the trader's coin age to acquire the ability to create blocks in the network and manufacture coins via the PoS process. These Interest coins are required to enter a certain Kernel, and the procedure is comparable to mining in the PoW method. However, PoS lowers the space needed to get the random number, thus lowering energy usage. As more interest coins are spent, the hash calculation area shrinks, making it simpler to calculate random values. For example, the goal value of all nodes in the PoW method is the same. In contrast, the target value of each node in the PoS mechanism is different, making mining simpler for nodes with older coins. In PoS, every transaction in the block costs currency age; ultimately, the blockchain will choose the chain that consumes the greatest coin age as the master chain. This design also addresses 51 percent forced assaults in the PoW consensus process, in which nodes must first possess a significant quantity of bitcoin old enough to fabricate blocks for an attack. It is a procedure that costs much more than the entire computing cost of a 51% network. At the same time, while assaulting the main chain, the attacker consumes coin age, which is also a loss for the attacker. Therefore, a mining process in PoS is linked to the processing power of nodes and the holding amount of nodes. As a result, mining will be simpler for nodes with high stakes than for nodes with small stakes, increasing mining efficiency and decreasing energy waste. Obviously, the development of the PoS mechanism demonstrates that people are increasingly becoming aware of Bitcoin's wastefulness. Although the PoW algorithm

provides power, the PoS approach is not without mining limitations. As nodes must still compute, there is still a lot of computational waste in PoS.

DPoS

The DPoS consensus algorithm is comparable to a board vote in several ways. In a decentralized structure, decision-making authority is distributed to all shareholders. When a proposal receives more than 51 percent of the vote, the result is final and irreversible. In DPoS, delegates, or nodes that produce blocks, play an important part in this process. When a node wants to become a delegate, a deposit is required to demonstrate trustworthiness. Also, each user may vote for a trustworthy delegate. As a result, the first n delegates with the most votes in the network are eligible to participate in the mining (Hu et al. 2021). The amount of votes they have is the same as the number of votes the nodes have. Regularly, the n delegators will take turns generating blocks, and blocks with a share of support more than 51 percent will be deemed successful. At the same time, delegates will be paid a transaction fee for each produced block, and these advantages will also serve as an incentive to keep participating. When choosing a delegate, shareholders may see the likelihood of this delegate generating the incorrect block, which is important in the DPoS consensus process. When a delegate provides the incorrect block, another delegate produces the correct one the following time. Obviously, the DPoS consensus mechanism outperforms the PoS consensus method. DPoS is not mined but based on the votes of all participants. However, when participation is low, representatives tend to be centralized. As a result, this blockchain will lose its decentralized characteristic.

Byzantine Consensus Mechanism

A consensus method that accepts malicious behaviour is known as Byzantine Fault Tolerance (BFT). It is extremely comparable to the historical issue of Byzantine generals telling other generals to attack Rome jointly through messengers. When a process comes to a halt, it cannot be restarted. Byzantine mistakes, on the other hand, can be arbitrary. Suppose a process does not transmit false signals to other processes. In that case, the system simply has to accept the mistake and take more than half of the votes to make a decision. If mistakes occur in f processes, the system must have at least $2f+1$ processes. Byzantine

mistakes, on the other hand, are more complicated. In a Byzantine system, the number of error processes allowed is fewer than the number of processes in a non-Byzantine system. This system's f limit is $f < N/3$, where N is the total number of processes in the system. This is known as the Byzantine mechanism (Lamport et al. 1982). What's more, the Practical Byzantine Fault Tolerance (PBFT) mechanism was suggested by Castro and Liskov (1999). One of the first to offer a realistic Byzantine fault tolerance method, it enables TPS of over 100,000 in networks. This method is utilized by the Hyperledger blockchain project, although the project is still under development and is primarily focused on maintaining the global ledger. Conversely, the PBFT consensus method may address the consensus issue in distributed systems, particularly when the consortium chain's failure tolerance is not too great. However, when there are Byzantine faults, the system will send many messages, which is costly for the network.

2.1.2 Classification of blockchain

The public chain, private chain, and consortium chain are the three types of blockchains, and the following is a short overview of each.

Firstly, the public chain is an open and fully decentralized blockchain, which means that anybody may connect to any node in the blockchain network (Guegan 2017). In a public chain, each node may take part in transaction execution and verification. Also, the consensus mechanism determines whether blocks may join the main chain and records the state of existing network nodes. As a derivation of decentralized trust, the public chain employs cryptographic methods to guarantee security, proof-of-work, and incentive systems to encourage additional nodes to mine. For example, mining nodes are paid with a specific amount of money whenever a block is approved.

Management nodes in the private chain have the write rights such as transaction and authentication permissions. Recently, many individuals have argued that the presence of management nodes contradicts the original purpose of blockchain technology. However, the most important benefit of private chains is encryption and auditing. Thus they are frequently employed for database administration inside institutions and businesses (Pongnumkul et al. 2017).

Consortium chains are decentralized blockchains. During the consensus process, certain nodes are pre-selected to control the blocks to achieve agreement. For example, in a blockchain system comprised of N financial institutions, each financial institution acts as a node in the validation process. It will pass only if more than two-thirds of the N nodes pass. Also, the read/write privileges and accounting rights in the consortium chain must be established by chosen nodes with a restricted number of nodes. Thus the consensus protocol cannot utilize the proof-of-work mining process but instead employs the quick and efficient PBFT algorithm for voting choices. Meanwhile, consortium chains have greater security and performance requirements in terms of time.

Table 1 Three types of blockchain

	Public Chain	Private Chain	Consortium Chain
Definition	A blockchain where anyone can send a transaction and anyone can participate in the consensus process	Write access to a blockchain controlled only by the organisation	Under federation consensus, pre-selected nodes control access
Participator	Anyone can enter and exit freely	Individual or within an organisation	Specific populations, agreement groups
How to trust	PoW	trust yourself	trust the group
Incentives	Yes	No	Optional
The degree of centralisation	Decentralisation	Centralisation	Multi-centralization
Strength	Self-establishment of trust	Transparent and traceable	High efficiency, cost optimisation

2.1.3 Blockchain Features

Decentralisation

The majority of today's popular databases are private and based on reliable servers. However, blockchain is a P2P structure comprised of several nodes that lacks a centralized administration node or organization. Each node in this network has equal privileges. Through distributed propagation, all nodes may record, verify, and transmit the computational results of the data to each other. Damage to individual nodes has little effect on the overall functioning of the system, and each participating node serves as the network's hub (Wang et al. 2018).

High level of security

A blockchain is a non-falsifiable, tamper-evident database. In a blockchain, a timestamp is created by combining blocks and chains. This timestamp records the history of network transactions, allowing you to access and locate transaction data. Besides, each node produces a block and a timestamp and informs all other nodes through the network, enabling each participating node to receive a complete copy of the transaction data. When information is validated and uploaded to the blockchain, it is permanently preserved. If someone wishes to tamper with past data, he must control more than half of the system's nodes. As a result, blockchain technology is a highly dependable data technology. The more nodes engaged in a transaction and computing power it has, the more safe its data will be. It is the tremendous computational power offered by PoW that ensures the security of a blockchain system. Because every transaction is signed, if someone wishes to fake a transaction, he must first create the block matching to the transaction and any blocks linked after that block. Suppose the counterfeiter creates blocks quicker than the blockchain can process them. In that case, the forged blocks will be detected and destroyed instantly.

De-crediting

The development of a new credit formation mechanism is the most crucial disruption of blockchain. In the traditional Internet, network users rely on third-party entities to create credit and conduct transactions. However, blockchain significantly alters the centralized

method of credit generation by using a set of consensus-based mathematical procedures to establish a network of "trust" amongst computers. Thus it constructs credit via technology rather than a centralized third-party platform (Nofer et al. 2017). Therefore, participants do not need to know who the necessary counterparties are, let alone via third-party endorsements and assurances. In total, the blockchain's consensus process produces credit for participants and establishes confidence and consensus.

2.2 Ethereum

2.2.1 What is Ethereum

Ethereum is a revolutionary global blockchain platform built on a public chain that enables anybody to create and utilize decentralized apps. The uniqueness of Ethereum is that it gives rise to a new kind of software program for computer networks, one that is genuinely decentralized. Unlike Bitcoin, Ethereum provides a new cryptographic technological foundation that enables users to create applications more simply and flexibly while sharing a sustainable economic environment (GAVIN WOOD 2014). The integration of several functional modules into an integrated platform for decentralized apps is a significant Ethereum innovation. Ethereum is also the ideal mix of blockchain and smart contracts, providing a complete smart contract solution. Besides, Ethereum offers a comprehensive and scalable collection of tools for implementing a blockchain comparable to Bitcoin, which is backed by technologies such as P2P networks, encryption, and HttpClient. What's more, Ethereum achieves consensus through a proof-of-work algorithm. Unlike Bitcoin, smart contracts on Ethereum may be created, allowing decentralized applications through the power of integration.

2.2.1.1 EVM

Ethereum Virtual Machine (EVM) provides an environment in which smart contracts may be executed. The selection of a Turing-complete computer environment is one of the Ethereum project's major breakthroughs. It is made up of several linked computers that enable users to develop and upload sophisticated programs while automating these processes. It also guarantees that the current and prior status of all programs is always

publicly accessible. These blockchain-based applications continue to execute as specified by the EVM, including endless loops. Furthermore, one contract may call other contracts, laying the groundwork for possibly recursive calls. Because each node in the Ethereum network runs the EVM, Ethereum functions as a parallel "world computer," recording the status of accounts on all nodes at the same time and achieving network-level agreement (Hirai 2017). This P2P method is not the most efficient, but it is the most secure. When a computer runs an Ethereum virtual machine, it becomes a node in the network and processes transactions the same way as other nodes. However, the present EVM is flawed, with issues such as no support for floating-point integers, a lack of standard libraries, and no support for contract code optimization and upgrading. Since Ethereum is a relatively young public chain, many blockchain solutions are still developed and deployed using EVM enhancements. This reintroduces Ethereum's basic functionalities while reducing development effort and using the current Ethereum ecosystem for fast development revisions. In brief, the Ethereum Virtual Machine is a critical component of Ethereum, without which smart contracts cannot be guaranteed. The heart is essential to the human body, and in certain ways, the Ethereum Virtual Machine may be considered the heart of Ethereum.

2.2.1.2 Account

In Ethereum, there are two kinds of accounts that share the same address space.

- External accounts, which are managed via public and private keys.
- Contract accounts, which are managed by the account's code.

An Ethereum account is made up of four components.

- A calculator determines that each transaction can only be completed once.
- The account's current Ether balance.
- The contract code for the account.
- Account storage.

2.2.1.3 Transaction

In Ethereum, a "transaction" is a signed packet of data storing a message sent from an external account. A transaction contains several pieces of information, including the signature used by the recipient of the message. Besides, each transaction requires a limit on the number of computation steps triggered by the execution code to prevent infinite loops in the code. START GAS is the fuel that needs to be paid for the limit on the computation step. GAS PRICE is the price that needs to be paid for the miner's fuel for each computation step.

2.2.1.4 Gas

Gas is a charge that the sender of a transaction on Ethereum must pay for each operation. The goal of Gas is to protect the Ethereum network against harmful assaults and misuse. Each transaction includes a Gas Used and a Gas Price, which are computed when EVM performs the transaction according to specified criteria. Suppose the transaction performs a command in the order indicated in the smart contract. In that case, each operation will use a certain amount of Gas. Each transaction must contain a Gas limit as well as an optional Gas fee. The gas limit is the maximum amount of Gas used during the transaction, which can be seen as a service charge for the transaction service itself, while the gas price can be interpreted as a tip. The miner has the option of selecting which transaction to pack first. The more you spend on a transaction, the quicker it will be verified. The highest transaction charge you may pay for a transaction is known as the Gas Limit. This amount will not be exceeded by the transaction charge. If you execute the transaction without exhausting your Gas limit, the excess Gas will be refunded to you in the form of ETH (Grech et al. 2018). If you want to pack a deal straight away, you must tip the miner. When you include a tip, the total amount of Gas used may surpass the Gas limit.

2.2.1.5 Message call

A smart contract may be triggered by the user through a message call or a transaction with a non-contract account. Both techniques have a source address, a destination address, a packet, Gas, and return data in common. Furthermore, each transaction may be seen as a

message call, which produces additional message calls. The contract that was invoked receives a new main memory and has access to the data area. The returned data is stored in a pre-allocated memory region at the end of the call.

2.2.2 Ethereum hierarchy

Ethereum is a distributed computing platform built on a public chain that offers decentralized virtual machines to run the Turing-complete programming language. Figure 5 depicts Ethereum's system architecture. Firstly, there is a data layer, which is the basic layer of Ethereum. Secondly, a network layer primarily contains each node's data transmission verification process. In the consensus layer, Ethereum employs the proof-of-work consensus mechanism. Besides, the incentive layer employs the incentive mechanism, primarily used to incentivize nodes to mine and maintain the Ethereum operation. Finally, a smart contract layer is unusual. It contains a virtual machine that can execute the Turing-complete scripting language and deploy smart contracts to Ethereum.

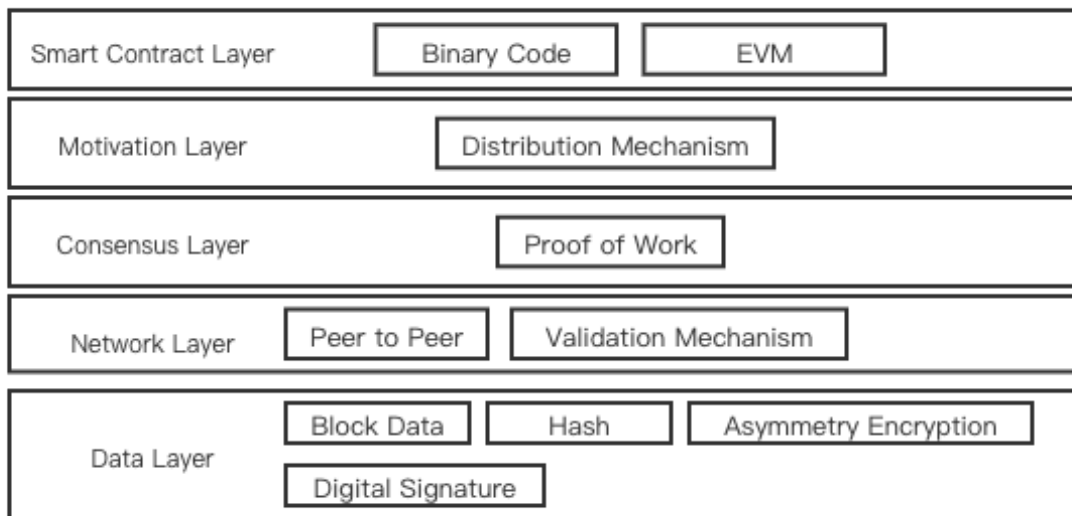


Figure 5 Five layers model of Ethereum

2.2.2.1 Data Layer

The data layer includes the block's data structure as well as data encryption content. The timestamp is the block's creation time; the random number is the mathematical problem's solution; the difficulty factor may dynamically change the mathematical problem's complexity; the Merkle tree is the unique identification of the transaction in the block body. By using a particular hashing method, these components create the block's hash. Then the block is linked from start to finish by recording the parent hash and computing its own hash, resulting in a chain structure.

Due to network delay, several new blocks are often added to the main chain concurrently while adding blocks to the blockchain one by one. Here, you must compare the difficulty factor, choose the chain with the total cumulative processing power as the primary chain, and keep adding blocks.

Besides, three Merkle trees are included in Ethereum blocks, which record the transaction, account status, and reception in that order. By combining transactions into recursive hashes, the Merkle tree inserts hash values and gets the Merkle root hash. A change in the data on any Merkle tree node changes the hash values of all nodes in the node route. As a result, putting the Merkle tree root hash in the blockchain header may validate all transactions.

The Merkle tree is also used in Ethereum transaction and receipt trees. In contrast, the Merkle Patricia tree is used for account state storage. The state tree must be queried or updated each time a transaction is performed, or an account is altered. In the event of frequent add or delete operations, the status tree must query and compute the root hash rapidly. Moreover, the Merkle tree coupled with the dictionary tree has a good search efficiency and can rapidly calculate the root hash. By the tree's structure, the root hash may be produced fast following an insert or modification action without rebuilding the tree.

At the same time, the tree's depth is limited. Even if an attacker were to generate some transactions on purpose, the tree would not grow indefinitely. Otherwise, an attacker may control the depth of the tree to launch a denial-of-service (DoS) attack, causing updates to be very sluggish. The tree's root hash solely refers to data in the leaf nodes, independent of

the order in which the data is updated. Therefore, updating in a different sequence has no effect on the root hash.

The mining process requires hash calculation, and data storage on the blockchain is also required. Ethash and SHA3 (Luo et al. 2016) hash algorithms are employed in Ethereum. Ethash is primarily used for solving random numbers in the mining process, and SHA3 is comparable to the SHA256 (Gilbert and Handschuh 2004) hash algorithm, which is used to store blocks in the process of safe encryption. In reality, data of any length is encoded, and SHA3 generates a 256-bit binary number for storage. Besides, the SHA3 hash algorithm is highly collision-resistant, and hash conflicts are uncommon. At the same time, the length of the data after hashing remains constant, and the unidirectional nature of hashing provides excellent protection for the original data.

Ethereum is a worldwide public ledger, and its account security is based on digital signatures and an asymmetric encryption mechanism. There are public and private keys to encrypt and decode data in Ethereum. An account is first encrypted with a private key, and only the matching public key may verify it. As a result, Ethereum establishes mutual trust between nodes via this method.

2.2.2.2 Network Layer

The network layer is responsible for implementing the P2P network, the message propagation mechanism, and data validation. First, the network layer is the network guarantee that allows Ethereum nodes to communicate with one another. To accomplish decentralization, Ethereum employs a peer-to-peer network. This P2P network has a flat topology, which means that each node has the same state. In the Ethereum system, there is no centralized server, and each node performs routing, authentication, and broadcasting tasks. Besides, the P2P network design serves as the foundation for Ethereum's decentralized network. In Ethereum, data is distributed in two ways: actively transmitting data and requesting data. When a node gets a transaction or generates a block, the message is broadcast to other networks. Since P2P network design cannot ensure that all nodes receive a message, nodes may send data requests or data to neighbouring nodes to synchronize. What's more, a data validation mechanism is also included in the network

layer, which validates data when a node receives a message from an adjacent node. In order to accomplish network message synchronization, the legitimate information will be disseminated to neighbouring nodes. Besides, suppose we want to avoid malicious attacks on the Ethereum network for incorrect information. In that case, some nodes should not perform data forwarding activities. If a node provides incorrect data, it will quickly lose its connection. In general, Ethereum's network layer uses distributed network technologies and a decentralized P2P network protocol to ensure high availability.

2.2.2.3 Consensus and motivation layers

An unavoidable issue in distributed systems is how to persuade nodes to agree in a fully autonomous decentralized system. In the Ethereum network, each node solves a mathematical problem and pays the first node, encouraging competition for real computing power throughout the network. Besides, Ethereum's PoW hash function is Ethash, and the hash algorithm has no practical method of operating in reverse. Each mining process generates a target value based on the difficulty factor in the block header before performing a hash operation between the information in the block header and a random integer. Suppose the resultant hash value is smaller than the goal value. In that case, the mining is deemed successful, and the random number represents the operation's solution. Because of the nature of the hashing method, even if the input is just slightly changed, the hash value varies significantly. Although the solution procedure takes a significant amount of computing resources, verification of the solution is simple, which can be accomplished by simply comparing it to the goal number. To fabricate block information and add it to the blockchain, an attacker must control 51 percent of the network's computing power. As a result, PoW must depend on incentive mechanisms to promote network-wide mining. Thus maintaining network-wide computing power big enough to guarantee Ethereum's decentralization remains safe and trustworthy. Ether, like Bitcoin, is virtual money that is used to incentivize miners to mine.

2.2.2.4 Contract layer

The contract layer is primarily a scripting language and a virtual machine that runs the scripting language. Because Ethereum can execute Turing-complete scripting languages,

sophisticated smart contracts may be used to build decentralized applications. Therefore, the contract layer is required for blockchain to be decentralized. To some extent, a smart contract is a piece of code that includes business logic and is executed automatically by an Ethereum virtual computer.

The code for smart contracts in Ethereum is written in stack-based bytecode, which means that the code is made up of bytes, each of which represents a distinct action. Typically, code execution is an endless loop in which the counter is increased by one for each iteration of the program until the code completes or an error occurs. For code execution, there are three kinds of storage data spaces:

- Stack: a data structure with a first-in, first-out (FIFO) ordering.
- Memory: infinite byte array
- Long term storage of contracts: unlike stacks and memory, it may be kept for a long period.

2.2.3 Smart Contract

Smart contract describes the mechanism through which the blockchain's owner of the currency changes. The value in the message may also be provided when a transaction is sent or approved. It is analogous to an electronic envelope containing electronic money and being sent across the network. It may also transmit certain additional information for other purposes.

A basic transaction may be thought of as a simple "script," which is a collection of instructions that enables nodes in the blockchain to carry out a transaction. However, when the script changes, users may utilise it to create complicated transactions. The idea of a smart contract that is both easy and verifiable is not new. Blockchain enables decentralisation; in other words, smart contracts are built on secure contracts that are not subject to central oversight (Buterin and Vitalik 2014).

Szabo (1997), a computer scientist working on digital contracts and digital currencies, invented "smart contract." An Ethereum smart contract is a piece of code that can be

executed by an Ethereum virtual computer. The Ethereum-specific binary form is used to store smart contracts on the blockchain. An Ethereum virtual computer interprets the code, which includes an Ethereum contract address. Smart contracts function as a trusted organisation that may store assets for users and is always handled in accordance with predefined criteria. It can react to approved messages, receive and store messages, and send messages.

Application Binary Interface is specified as a strong type in Ethereum smart contracts, and ABI is a string generated when the smart contract is built. Accounts use the ABI and address of the smart contract to call functions in the contract.

2.3 Blockchain-based Store

Many researchers have created blockchain-based degree systems in the past. Many difficulties exist in these blockchain-based storage systems, such as how to store user privacy. Where should the data be saved? How can the data be verified? How to minimize the storage load on blockchain nodes. How can we allocate electricity to nodes? However, it's becoming better all the time.

Ocheja et al. (2018) showed how to monitor and manage student learning data, such as credit and degree certificates, while keeping the information safe. He also explains how to upload data to the blockchain and retrieve previously contributed data. The nodes on the chain in this system reflect the resource suppliers and learners. Providers store the index of the learning resource on the blockchain, which is linked to the external resource database. The hash value of the learning material is then recorded on the blockchain. The request will be denied if the hash value linked to the subsequent request does not match the one on the chain. This article is based on PoW, and it prevents data from being changed when someone wishes to alter data in a single place. The student is not required to maintain a node, while the provider is required to do so. The paper seeks a viable method for storing all learning data on the blockchain.

Ocheja et al. (2019) created a credential tracking system that uses smart contracts to restrict access. The blockchain securely stores and verifies credentials. It is a long-term, continuous

information access system that stores one's academic accomplishments. However, when a student transfers schools, this system has difficulties linking data from the prior institution to the present one. This also makes analyzing academic performance more challenging. The writers subsequently explored using third-party centralized administration to link the data, but the necessity for a decentralized blockchain was removed.

Arenas and Fernandez (2018) suggested a distributed credential verification method that uses the blockchain to store compressed electronic degree certificates. Schools photograph the degree certificate, compress it, and ultimately upload it to the blockchain. This is, in my view, a really acceptable design concept. The storage space on the blockchain is very valuable, and compressed images may reduce the storage strain on a single node. However, I believe that even if the data is compressed, the picture data is still big, and the long-term stacking of photos necessitates each node constantly expanding its storage capacity, which is not an ideal design option. This technology also saves picture data directly in the blockchain without regard for user privacy, which is a serious issue.

Kuvshinov et al. (2018) created a framework for storing educational records distributedly. It is a system that includes a data privacy design mechanism as well as a full execution procedure. Students must first choose a course and then attend courses. Students must complete assignments, which are graded by instructors, before receiving a graduation certificate from their institutions. All of this information is stored on the blockchain, which is a highly developed technology. However, this is not an obvious option in my view. The blockchain will record all of a student's data operations, resulting in an overabundance of data on the blockchain. Furthermore, contacting smart contracts will incur high gas costs.

Rooksby and Dimitrov (2020) created a blockchain-based score storage system with a token mechanism. Only individuals who own the token have the ability to alter the blockchain. The data under this system, on the other hand, is visible and lacks user privacy. The authority node is unbounded. Institutions, on the other hand, only want to control their own students' papers and do not want other universities to change their students' information. How to ensure a single node's operating privileges became an issue for this system.

Farah et al. (2018) created a system that can sign students' learning tracks and then store them in a distributed system, ensuring the data's validity and tamper-evident nature. Ethereum smart contracts serve as the foundation for the system. Furthermore, this system only stores the data needed for retrieval and verification, not the original data. Those in possession of the original data search it up on the blockchain and compare it. This design preserves the data's privacy. This method, however, does not go into detail about the authority node's transfer.

Turkanović et al. (2018) offer a better platform for accumulating education credits based on the Credit Transfer and Accumulation System. The author presented a distributed architectural approach for storing higher education credits, allowing for globalized credential unification. This approach, however, is not based on smart contracts. Furthermore, no rules govern the autonomy of nodes at each level.

This thesis will build a decentralized education storage centre based on Ethereum and some technologies mentioned above. In terms of data storage, this approach would ensure the privacy and validity of user data while alleviating storage strain on each blockchain node. Addresses in this system will be split into three types based on their role: authority addresses, university addresses, and regular addresses. In the system, an ideal voting mechanism is created, via which authority addresses may elect and cancel other addresses without depending on a centralized system.

3 METHODOLOGY

3.1 Considerations

We defined the criteria required for a distributed Ethereum degree storage centre after analyzing the aforementioned system. To begin with, data privacy must be ensured since data saved on the blockchain is accessible by anybody, and exposing students' privacy may result in legal problems. Second, all nodes in the blockchain back up the data. A single node may not have enough disk space if the stored data is too big. Furthermore, as time passes, the amount of data on the blockchain grows. As a result, this system should think about how to keep less data on the blockchain. Furthermore, various kinds of addresses should have varying levels of authority in this system. In addition, in a decentralized system, permits must be handed along. The authority can authorise, but how can this be accomplished in a decentralized system? A voting system is an excellent option. The address on the blockchain may invoke the smart contract to realize the transfer of power by putting the voting rules into a smart contract. In this approach, the smart contract will check the address to see whether it has the authority to change data on the blockchain. Addresses are controlled by private keys. If an address has authorization, but the private key is compromised, the saboteur has the ability to alter the blockchain, which may have serious repercussions. As a result, we must explore a method to mitigate or eliminate the risk when the privileged owner discloses his private key.

Table 2 Some factors need to consider

Factor	Reason
Privacy	Disclosure of user privacy may pose legal problems.
Node Storage Burden	All nodes in the block are backing up the duplicate copy of data. If a student stores too much data, it will lead to insufficient disk space for each node and a long synchronisation time for new nodes.
Role division	Different accounts in the blockchain should have different permissions to manage and maintain the blockchain.
Voting mechanism	There should be a voting mechanism in the blockchain to select and discard the management nodes.
Preventing malicious nodes	If the private key of some nodes is stolen, there should be a mechanism to limit the scope of impact.

Role division

The system's nodes will be assigned one of three types of addresses: authority address, university address, and general address. National institutions have authority addresses, and in most cases, a nation has just one authority address. The authority address serves as the blockchain's administrator, and the address that establishes the smart contract will be the system's initial authority address by default. An authority address's function is to maintain university addresses and approve and reject university addresses. University addresses are used by universities. The authority address holder will verify the institution; the university will give its address, and the authority address will designate this address as the university address on the blockchain. University addresses are in charge of registering a student's degree credentials on the blockchain and may also cancel a student's degree credentials. Normal addresses have no authority and are simply used to query the blockchain's information.

Voting mechanism

The authority address has the most power in this system and has the ability to alter the university's authority. If the authority address's private key is exposed, a saboteur may take control of it and add unauthenticated universities while removing those that are authenticated. This is very hazardous and will interrupt the system's regular functioning. We include a voting mechanism into the smart contract in this system. More than 51% of the authority addresses on the blockchain must agree for an address to be authorized as an authority address. Similarly, if a private key of an authority address is revealed and someone uses it to alter data on the blockchain, any authority address in the network may start a vote to cancel this address through the smart contract. If more than 51% of the authority addresses agree, the system will execute the logic in the smart contract to revoke this authority address, thus preserving the data on the blockchain.

Privacy and storage burden

The information included in a student's data is critical, and the data on the blockchain is open and transparent. It becomes a challenge to consider how to guarantee the security of the data. We may immediately think of two major methods for preventing information from leaking. The first is encrypted storage, which encrypts the information using the key, places the encrypted cyphertext on the blockchain, and removes the key for decryption operations as required. The usual approach is to save the hash of the information, as illustrated in Figure 6, and the second is to store partial data on the blockchain. Given the storage requirements of nodes, it is clear that keeping the hash of degree credentials on the blockchain in our system would be a better option. This may decrease the storage load on each blockchain node while also protecting users' privacy.

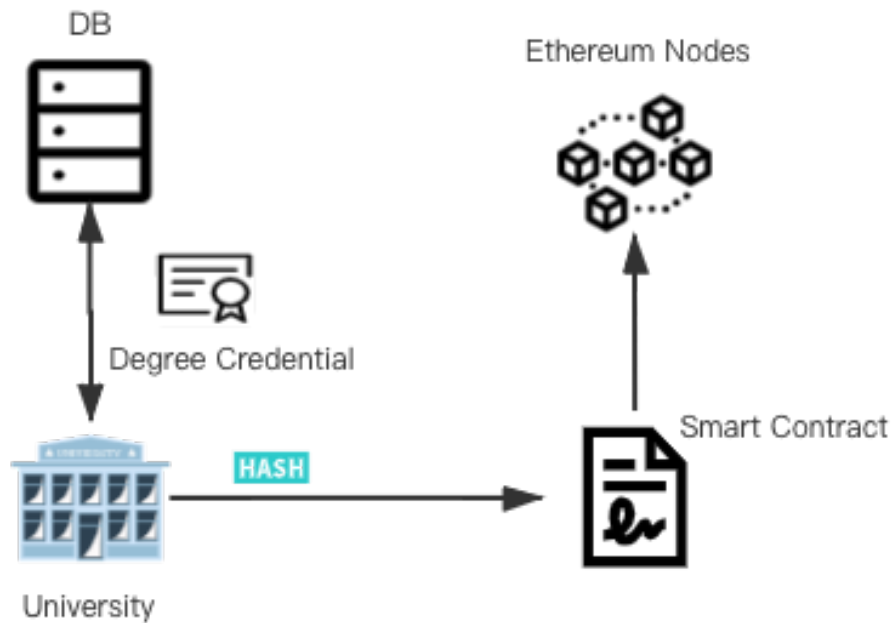


Figure 6 The process of issuing a degree certificate

Preventing malicious nodes

Our approach limits the ability of the malicious authority address to start a vote by other authority addresses. What happens if a rogue college address adds or removes a student's degree credentials at any time? We are issuing an ERC20 token. When the authority address authenticates the college address, the authority address may create a restricted amount of tokens through the smart contract interface and transfer them to the college address. Tokens must be used each time the university address uses the interface of granting degree credentials or withdrawing degree credentials on the smart contract. The smart contract cannot be performed if there are insufficient tokens at the university address. In this scenario, the holder of the university address should request tokens from the holder of the authority address. After reviewing the university's recent activities, the authority address will issue the token to the university address. If the most recent operation is anomalous, the authority address will contact the interface and request that the university address is revoked. When a university's private key is released, this method mitigates the effect of a saboteur.

3.2 Data Design

3.2.1 Data in the database

The university's database requires many fields to record the student's degree credentials. The first is the credential's number, which is generated according to particular criteria that will be explained later. The graduating student's name, ID card, and major are also important. Furthermore, the study's start and finish times should be documented in the system. Finally, a marker bit may be left to indicate whether the degree was acquired unlawfully.

Table 3 The table of degree certificate

Attributes	Value
certificate_id	String
name	String
id_card	String
major	String
start_time	Date
end_time	Date
is_valid	Boolean

3.2.2 Data on Ethereum

As previously discussed, in a blockchain system, we aim to minimize the amount of data on the blockchain as much as possible in order to prevent a storage load on all nodes. So,

what kind of data should we choose to keep on the blockchain? Figure 7 shows some critical facts.

certificate_id

The certificate ID is provided by the school, and the institution should have its own unique method to guarantee the ID's uniqueness. The following alternatives exist for ensuring the uniqueness of the ID. Initials of the school + first six digits of the address + incremental serial number. If it's the University of Manchester (UoM) with the address 0x41Eb82045e1FbB1b4B28f77dfc956B00D14949a3, and the current number has been raised to 115321, the id of the most recent graduation certificate should be UOM41Eb82115321. Of course, this design concept allows for two institutions with the same abbreviation, and the first six digits of the address are the same, so the certificates issued by the two universities will be confused. However, the world's university population is tiny, and the number of institutions having the same acronym is much fewer. Furthermore, the chance that the first six digits of the address are the same is $1/16^6$, which is very low. Of course, if we want to be cautious, we may take the first 8 or 10 digits of the address.

certificate_hash

The hash value of a degree certificate is produced using md5 digest after all information from this degree certificate is assembled into a string in accordance. In this case, we must examine an issue. In general, the MD5 has 128 bits. When considering the solid tamper-evident, we must preserve all the bits. If we want to conserve node storage capacity, we may consider just saving the first 64 bits. It saves storage space while also ensuring that the data is tamper-evident to some degree.

issuer_address

Many individuals believe that issuer address is unnecessary. In reality, once the degree credentials are published on the blockchain for the first time, the issuer address is recorded, and no future modifications to this record are permitted. The benefit of this is that when the smart contract calling it wishes to revoke the credential, it will compare the caller's

address with the issuer address to see whether they are consistent. The operation is refused if it does not match. If both A and B have university addresses and the right to store data on the blockchain, it is feasible that A will award a degree to a student studying at B. If the private key of a university address is revealed and then someone changes the credentials provided by other institutions, this is a highly hazardous activity. By determining if the operating address and issuer address are the same, the credentials may be isolated and the effect of key leakage reduced.

Table 4 The structure of a certificate on the Ethereum

Attributes	Description	Value
certificate_id	a unique id that represents the certificate	String
certificate_hash	the hash value of the degree certificate	String
issuer_address	the address that issue this certificate	Address

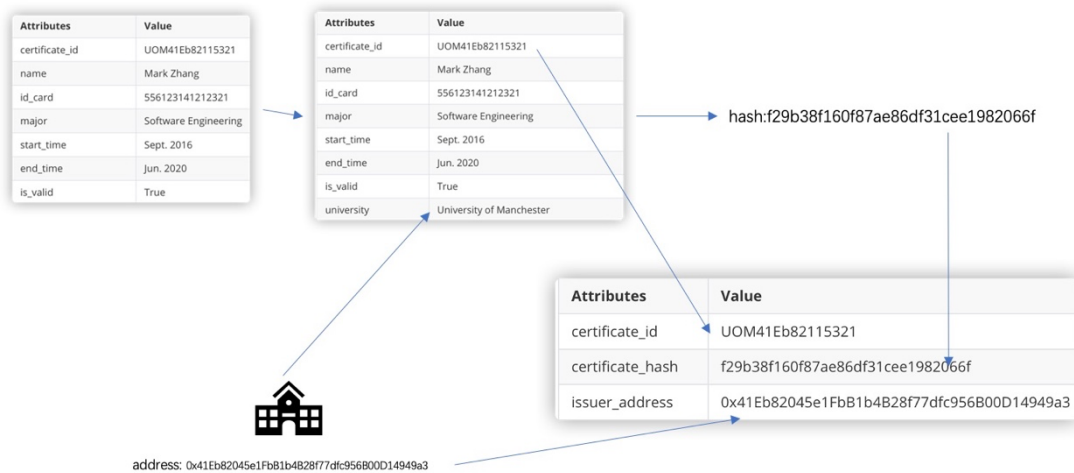


Figure 7 How the data come

3.3 System Design

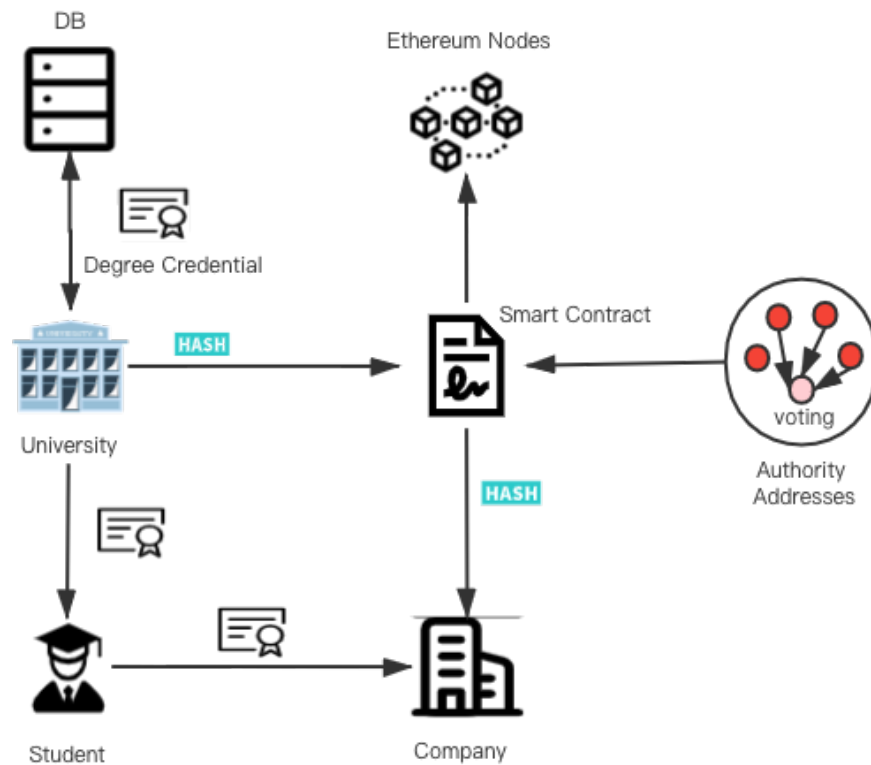


Figure 8 An overview of this system

Based on the design concepts presented above, this article suggests the system shown in Figure 8. To begin, the smart contract designer sends a transaction to the zero address to establish a smart contract. The creator then becomes the system's initial authoritative address. The authority address will be involved in the election of new authority addresses, as well as the vote to reject an authority address. We consider a nation to have just an authority address in our system, and authority addresses designate addresses as universities only after confirming the university in their own country. There is a variable in this system. TotalAuthorityAddressNum maintains track of the current total number of authority addresses, and if the current vote exceeds 50% of the total number determines whether future votes succeed or fail. If an address is designated as a university address, that address may use the smart contract to modify the data on the blockchain. When a student graduates,

the university creates the student's degree certificate and stores all the degree certificate's information in the university database. At the same moment, the institution computes the hash value of the degree credential and instructs the smart contract to store the hash value of the credential on the blockchain. The university maintains the certificate ID, which is an index of the hash value, in addition to the hash value. If a student's degree hash value is stored on the blockchain, but he breaches the university's regulations, and the institution wishes to revoke his degree credentials, this university address will invoke the smart contract's revoke method. It should be noticed here that the university address instructs the smart contract to modify the data on Ethereum, which is needed to use the gas and token provided by the authority address. When students graduate and have an interview, the university may provide them with their degree credentials. After they hand over the credentials to the employer, the latter analyses the data of the degree credentials in a certain format and produces the hash value. Then, using the certificate ID, he may discover the hash value of this credential on Ethereum. The credential is valid if the computed hash value matches the one recorded on the blockchain. If it is not the same, the supplied credential is invalid. Using this method, the business may then evaluate if the job applicant's degree is legitimate or not. Because all data on the blockchain is visible, there is no need to spend tokens or gas to get the hash value. We effectively link authorities, schools, and employers using this technology, ensuring tamper-evident and real-time data.

3.4 Smart Contract Design

The design of smart contracts is crucial in this system. This section will select a few methods from our smart contract to illustrate and basically outline the smart contract's design concepts. The programming language used here is solidity, and the code may be built and executed once the local EVM node is started.

3.4.1 Voting

The authority node voting method allows the system to be completely decentralized. We present the design concept of voting by cancelling the authorization of an address. The caller's identity must be checked in this function, and only the authority address has the

authorization to call it. In addition, we must identify whether the address used in the voting process is an authoritative node. If not, this isn't going to work. After confirming, we go through all the addresses that were voted on in this voting proposal and compare them one by one to see whether they match the caller. If they are, this caller has already voted and cannot vote again. Following that, determine whether this caller is the first to start the vote and, if so, designate the initiator of this proposition as the caller's address. This initiator has the right to withdraw this proposal at any time. The caller's address is then put to the voting queue, and it is determined if the existing votes are more than 50% of the total vote holders. If this is the case, the proposition becomes effective, and the authority address being voted on becomes a regular address. Finally, we must decrease the overall authority address counter by one.

```
1. function voteToCancelOneAuthority(address cancelAddr) public returns
   (bool success){
2.     require(authoritiesMap[msg.sender]);
3.     require(authoritiesMap[cancelAddr]);
4.     CancelAuthorityProposal storage prop = cancelAuthorityAuthorityM
       ap[cancelAddr];
5.     for(uint i=0;i<prop.votedAddrs.length;i++){
6.         require(prop.votedAddrs[i]!=msg.sender);
7.     }
8.     if(prop.votedAddrs.length==0){
9.         prop.issuer = msg.sender;
10.    }
11.    prop.votedAddrs.push(msg.sender);
12.    // finish vote, and check if the vote succeed, should more than
       50%
13.    if(prop.votedAddrs.length>currentAuthorityCount/2){
14.        authoritiesMap[cancelAddr] = false;
15.        currentAuthorityCount--;
16.        cancelAuthorityAuthorityMap[cancelAddr].votedAddrs = new add
           ress[](0);
17.    }
18.    success = true;
19. }
```

3.4.2 Add token

An authority address in this system has the power to manufacture coins and may call the appropriate smart contract function to transfer tokens to a university address. In this

approach, we must determine the identity of the caller, and only the authority address has the ability to transmit tokens. The smart contract will add the token to the university address after confirming the identification.

```
1. function giveTokenToUniversity(address addr, uint token_amount) public  
    ic{  
2.     require(authoritiesMap[msg.sender]);  
3.     universityMap[addr].token += token_amount;  
4. }
```

3.4.3 Issue Certificate

After the university address calls the smart contract, the hash value of the degree certificate will be published on the blockchain. To begin, we must determine if the amount of tokens at the current university address is higher than zero. Why is it not required to confirm if the address is a university address? Because only the university address may be added tokens to, there is no need to check if it is the university address and can simply evaluate the token quantity. This system may immediately deduct the number of tokens after determining that there are sufficient tokens. The blockchain then stores the degree certificate ID, hash value, and university address. The student's degree credentials have now been successfully registered on the blockchain. This URL is used to add and change credentials. If the degree certificate in the database is changed, the hash value on the blockchain may be updated in real-time. When updating, the code will check to see whether the issuer and the caller are the same. If they vary, no changes will be permitted. This makes it impossible for school B to modify the certificate issued by school A. Reduces the opportunity for a vandal to disrupt the whole network if a school's private key is compromised.

```
1. function issueDegree(string storage degreeID, string storage hashVal)  
    public returns (bool success){  
2.     address sender = msg.sender;  
3.     require(universityMap[sender].token-1>=0);  
4.     require(degreeHashMap[degreeID].issue_uni==sender || degreeHashMap  
        [degreeID].issue_uni == address(0));  
5.     universityMap[sender].token--;  
6.     degreeHashMap[degreeID].hashVal = hashVal;  
7.     degreeHashMap[degreeID].issue_uni = sender;  
8.     return true;  
9. }
```

3.4.4 Revoke Certificate

Some issues may be discovered after a credential has been given to a student, and the institution must withdraw the student's degree certificate. In this function, we must determine if the caller's address is designated as a university. Second, after we have obtained the credential based on the ID of the degree, we must determine if the issuer of the degree and the caller are the same address, and only if they are will the operation be permitted. This will prohibit University B from revoking the certificate granted by University A. Then, beneath the credential ID, we must clear the issuer and hash value. After that, a third party will be unable to find the hash of this credential on the blockchain.

```
1. function revokeDegree(string memory degreeID) public returns(bool success){
2.     require(degreeHashMap[degreeID].issue_uni==msg.sender||authoritiesMap[msg.sender]);
3.     degreeHashMap[degreeID].issue_uni = address(0);
4.     degreeHashMap[degreeID].hashVal = "";
5.     return true;
6. }
```

3.4.5 Request Hash

The recruiter must use the credential ID to get the hash value of the credential from the blockchain. This technique does not need the visitor's authentication since it simply requires a search operation with no change. Furthermore, the data on the blockchain is open and accessible, and anybody may query it.

```
1. function viewDegreeHash(string memory degreeID) public view returns (DegreeHash memory){
2.     return degreeHashMap[degreeID];
3. }
```


4 DEPLOYMENT

We focused on the data, code, and general design in the preceding part. This section will walk you through the process of installing the system, including the Mysql database, the Geth client, and the MetaMask wallet.

4.1 Mysql

4.1.1 Introduction

MYSQL is a database that is safe, cross-platform, and efficient. Many businesses now utilize MYSQL as their database since it takes up minimal disk space, is quick, and is open source. Mysql must have the following features: the ability to store data in a particular operation language, query data, and add, edit, and remove huge amounts of data. MYSQL provides a wide range of data operations (Rodriguez, J and Guardo 2005). MySQL also enables large amounts of data storage. To preserve data consistency in the case of a failure error, the database must be recovered. Individual users' data may be viewed concurrently, but they cannot interact with one other and cannot operate on incomplete data.

4.1.2 setting

Because the university saves students' graduation degree certificates in MySQL in this project, we must build a credential storage table in MySQL. Figure 9's table primarily contains the student's degree certificate ID, name, ID card number, major, start time, finish time, and whether it is legitimate. MySQL's deployment is complete after this table is created. If we want to improve MySQL's performance, we may build distinct indexes for name and ID cards here to speed up data retrieval. Meanwhile, MySQL's default port is 3389. We must determine if the local firewall permits external access to port 3389. If it is not permitted, we must open it.








Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G
 certificate_id	VARCHAR(100) ▾	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 name	VARCHAR(45) ▾	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 id_card	VARCHAR(45) ▾	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 major	VARCHAR(45) ▾	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 start_time	VARCHAR(45) ▾	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 end_time	VARCHAR(45) ▾	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 is_valid	TINYINT ▾	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Figure 9 The table of the degree certificate in MySQL

4.2 Geth

4.2.1 Introduction

A blockchain client is required to interact with the blockchain. A client is a piece of software that can connect to other clients through peer-to-peer communication channels, sign and publish transactions, and interact with smart contracts (Kim et al. 2018). In Ethereum, clients are often referred to as nodes. The Ethereum Yellow Book defines the functions that an Ethereum node must do. The Yellow Book specifies the tasks that an Ethereum node must do, as well as the mining algorithm and the ECDSA settings for the private/public key. It specifies all the characteristics that ensure the node's compatibility with the Ethereum client. Anyone may build their own Ethereum node in the language of their choice using the Ethereum Yellow Book. Geth and Parity are the most typical clients. The primary difference between their implementations is the programming language used, with Geth using Golang and Parity employing Rust. Because Geth is the most well-known customer, we will now concentrate on him. Geth is a GitHub repository that has an Ethereum client.

4.2.2 genesis.json

At the time of Geth starting, the Genesis.json file must be provided for initialization. Parameters like as token pre-allocation, blockchain ID, and mining difficulty are specified in the Genesis.json file.

```

1. {
2.   "config": {
3.     "chainId": 9,
4.     "homesteadBlock": 0,
5.     "eip150Block": 0,
6.     "eip155Block": 0,
7.     "eip158Block": 0,
8.     "byzantiumBlock": 0,
9.     "constantinopleBlock": 0,
10.    "petersburgBlock": 0,
11.    "istanbulBlock": 0,
12.    "berlinBlock": 0
13.  },
14.  "alloc": {
15.    "0x585d205ce1b41bb4bb98bd50667c0f04c5500a57": { "balance": "5000
    0000000000000000" }
16.  },
17.  "coinbase": "0x0000000000000000000000000000000000000000",
18.  "difficulty": "0x5f0000",
19.  "extraData": "",
20.  "gasLimit": "0x2fef8",
21.  "nonce": "0x0000000000000042",
22.  "mixhash": "0x0000000000000000000000000000000000000000000000000000000000000000",
23.  "parentHash": "0x0000000000000000000000000000000000000000000000000000000000000000",
24.  "timestamp": "0x00"
25. }

```

4.2.3 Initialisation

Following the configuration of the genesis.json file, we initialize it using Geth's init command and then access the Geth terminal, as shown in Figure 10. Here, we will expose Geth's RPC port so that other computers may contact Geth's methods directly through RPC to complete remote calls.

The image shows a terminal window titled 'geth -datadir my_project --networkid 9 --rpc --allow-insecure-unlock console'. The terminal output displays the Geth client's startup sequence, including loading the genesis file, allocating memory, writing the genesis state, and initializing the Ethereum protocol. The console also shows the RPC interface being started and the node's status. The output is split across two panes in the terminal window.

Figure 10 The console of Geth client

We are now in the Ethereum client's console, where we may call the methods wrapped in the client and conduct a number of activities. Account-related, tool-related, node-related, mining activities, and so on are the main operations here.

4.3 MetaMask

MetaMask is a Google Chrome Ethereum wallet. It is not necessary to download it. Add the necessary extension to Google Chrome, and you're ready to go. This wallet is required to rapidly see the balance of an address on the private chain as well as the transactions on an address. It may also start a transfer fast. Furthermore, the wallet can dynamically predict how much gas will be used to invoke the current smart contract function, making it more convenient for consumers. We wish to use MetaMask to query the data in the local Ethereum node that is running, and we allowed RPC access when we launched the local Ethereum node. The standard RPC port is 8545. We have finished the MetaMask installation process. Figure 11's left picture depicts the arrangement.

To test if we can access local data, we configured the address 0x585d205ce1b41bb4bb98bd50667c0f04c5500a57 in the Genesis.json file to have 5000000000000000000 wei by default in the Ethereum system. As a result, we can use

MetaMask to determine if the configuration is successful by checking the balance on address 0x585d205ce1b41bb4bb98bd50667c0f04c5500a57. Figure 11 shows that there are 5 ETH in this address.

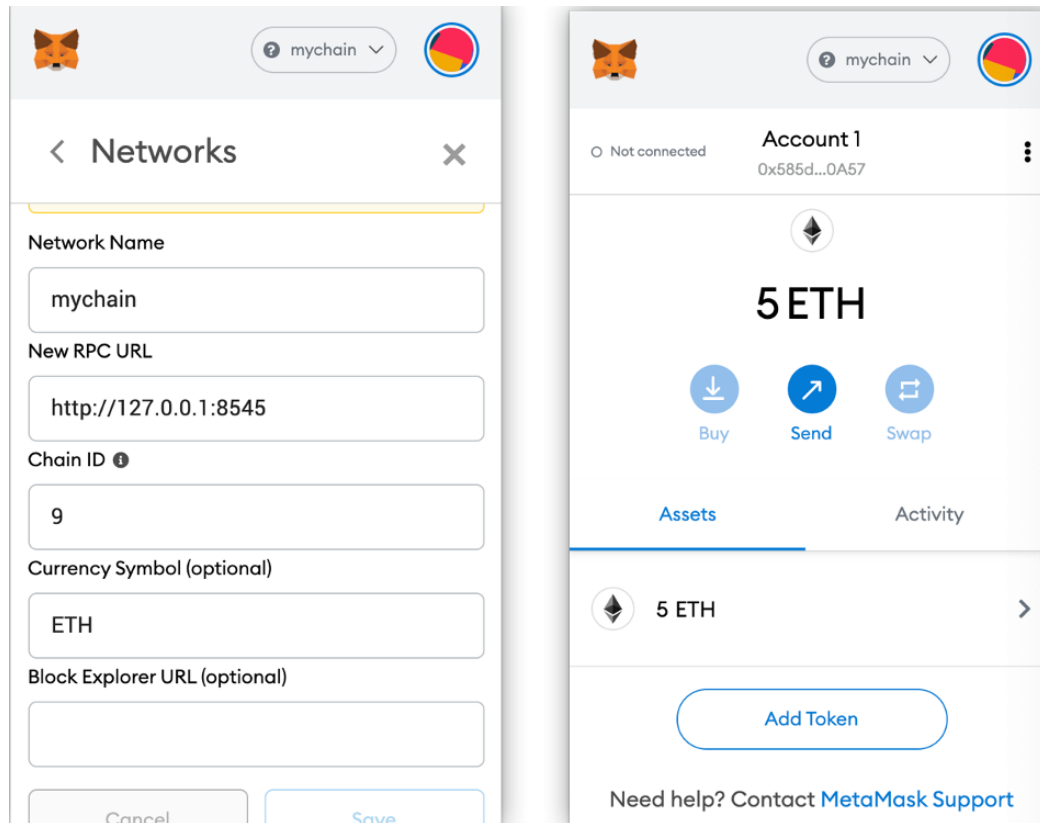


Figure 11 The Metamask wallet

5 TEST & EVALUATION

5.1 Test

Following the prior preparations, we will primarily test operations such as authority address voting, authorizing university addresses, recording degree credentials on the blockchain, and cancelling degree credentials. First and foremost, we must create five private keys before proceeding with the testing. The five keys are then imported into the wallet. These five addresses are, at first seem, regular addresses with no distinctions. They will have various responsibilities in the smart contract we design and will take on different duties after a further set of authorization procedures.

Table 5 The role of five addresses (idx.1)

name	address	role
account 1	0x600733c73c7bBC11BaD98ADE667fF94479d6BDf1	normal
account 2	0x41Eb82045e1FbB1b4B28f77dfc956B00D14949a3	normal
account 3	0xda16cDaaBA1c7C4054Ed8F3d2acADbCE9e1a0975	normal
account 4	0x3238eCc988880633A7d45213fB7773247B1271cb	normal
account 5	0x7e919EF790BD06976457e8c1C7315c7F36c77bc9	normal

These five addresses are devoid of ETH. ETH stands for gas in the Ethereum system. There is no method to call the EVM virtual machine for computation without gas, and there is no way to change any data on the blockchain without gas. To make the next work easier, we will mine using these five addresses in turn, and the ETH acquired via mining will be utilized as gas to invoke the smart contracts. Of course, you may mine with just account 1 and then transfer ETH to the other four accounts. Following this procedure, all five accounts will have sufficient ETH for contract release and contract call later.

5.1.1 Contract Deployment

We have written the code for the smart contract, but how does that code communicate with the Ethereum blockchain? When the code is called, the EVM will load it into memory and run it. In this section, we will show how the code in Figure 12 is built and saved. The code may be found in the remix network compiler. The remix environment is set to the local Ethereum network, the publisher's address is set to account 1, and the maximum authority address is set to 5. The code may then be saved on the blockchain.

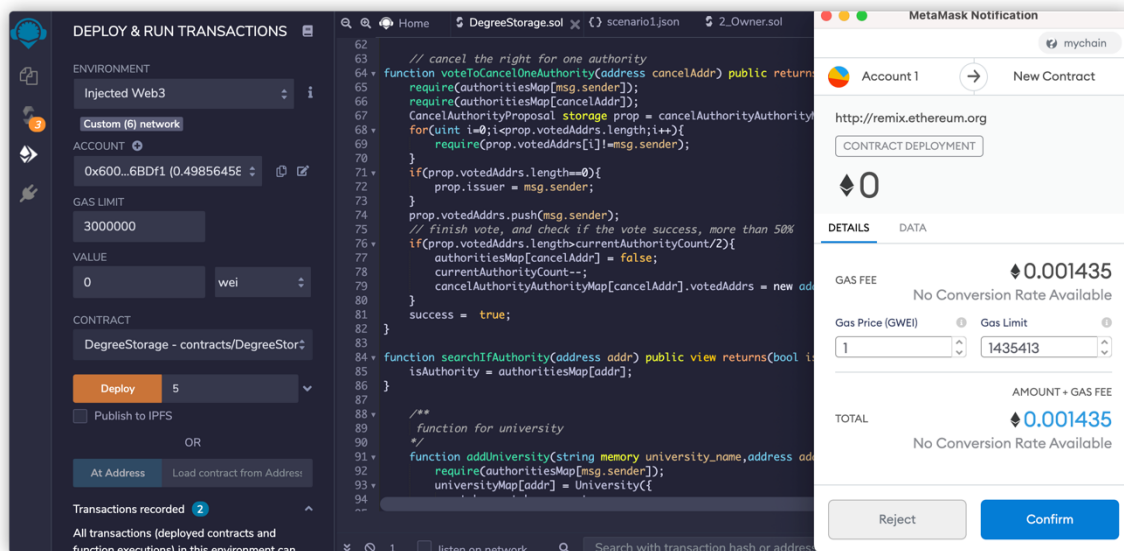


Figure 12 Deploy the smart contract

Deploying the smart contract uses a lot of gas since the code is built and saved on each node, which takes up a lot of storage space. MetaMask will determine how much gas to use depending on the amount of code. The gas is then converted into ETH via MetaMask. This smart contract is issued by the address 0x600733c73c7bBC11BaD98ADE667fF94479d6BDf1; according to our design principles, the publisher will become the first authority address in the system. As a result, the present status of these five accounts has changed.

Table 6 The role of five addresses (idx.2)

name	address	role
account 1	0x600733c73c7bBC11BaD98ADE667fF94479d6BDf1	authority
account 2	0x41Eb82045e1FbB1b4B28f77dfc956B00D14949a3	normal
account 3	0xda16cDaaBA1c7C4054Ed8F3d2acADbCE9e1a0975	normal
account 4	0x3238eCc988880633A7d45213fB7773247B1271cb	normal
account 5	0x7e919EF790BD06976457e8c1C7315c7F36c77bc9	normal

5.1.2 Voting

Account 1 is already the authority address after contract deployment. Assume that the present Ethereum network's five addresses are all authority addresses. We presume that account 2's key is accessible and that this address will tamper with the data on the blockchain. Accounts 1, 3, 4, and 5 start a vote to reject account 2 in order to preserve the health of a completely decentralized system. The rule of thumb in our system is that it will succeed if more than half of the people vote in favour of it. There are now five authority addresses on the network, and account 2 may be removed if three addresses (excluding account 2) vote for it.

Table 7 The role of five addresses (idx.3)

name	address	role
account 1	0x600733c73c7bBC11BaD98ADE667fF94479d6BDf1	authority
account 2	0x41Eb82045e1FbB1b4B28f77dfc956B00D14949a3	authority
account 3	0xda16cDaaBA1c7C4054Ed8F3d2acADbCE9e1a0975	authority
account 4	0x3238eCc988880633A7d45213fB7773247B1271cb	authority
account 5	0x7e919EF790BD06976457e8c1C7315c7F36c77bc9	authority

In Figure 13, we utilize accounts 1, 3, and 5 to make requests to the smart contract's interface for revoking an authority address, with the arguments being account 2's address. When the smart contract gets a request to invoke the voting function, it will determine if the current votes exceed 50% of the total voting addresses. If it is larger, the authority address of account 2 is instantly revoked.

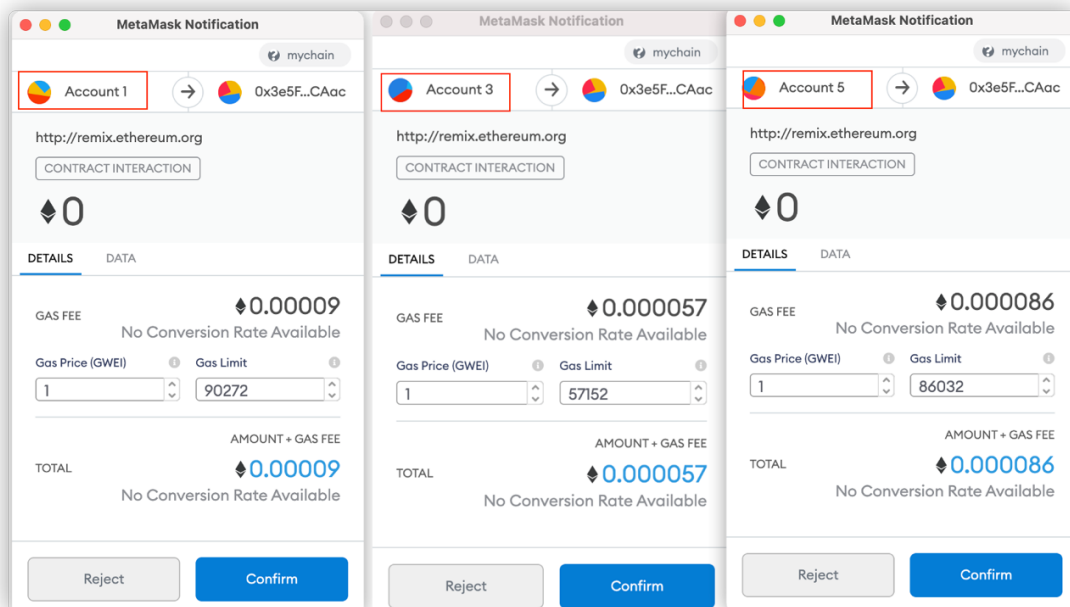


Figure 13 Sending transactions to call the smart contract

We call the smart contract interface to check if an address is an authority address, and the returned response indicates that this address is no longer an authority address. Figure 14 shows that the votes started by accounts 1, 3, and 5 are valid, and account 2's power has been removed. We used the example of revoking an authority address, but new authority addresses are elected using the same voting process. We won't go into detail here.

```

CAL [call] from: 0xda16cDaaBA1c7C4054Ed8F3d2acADbCE9e1a0975
L  to: DegreeStorage.searchIfAuthority(address) data: 0xcbc...949a3

transaction hash      call0xda16cDaaBA1c7C4054Ed8F3d2acADbCE9e1a09750x3e5F0bFB756626Bc02D5
                        0B25d6B06102Bdd6CAac0xcbed18f1000000000000000000000000041eb82045elfbb
                        1b4b28f77dfc956b00d14949a3

from                  0xda16cDaaBA1c7C4054Ed8F3d2acADbCE9e1a0975

to                    DegreeStorage.searchIfAuthority(address)
                        0x3e5F0bFB756626Bc02D50B25d6B06102Bdd6CAac

hash                  call0xda16cDaaBA1c7C4054Ed8F3d2acADbCE9e1a09750x3e5F0bFB756626Bc02D5
                        0B25d6B06102Bdd6CAac0xcbed18f1000000000000000000000000041eb82045elfbb
                        1b4b28f77dfc956b00d14949a3

input                  0xcbc...949a3

decoded input          { "address addr": "0x41Eb82045e1FbB1b4B28f77dfc956B00D14949a3" }

decoded output          { "0": "bool: isAuthority false" }

logs                  []

```

Figure 14 The result of searchIfAuthority function

Table 8 The role of five addresses (idx.4)

name	address	role
account 1	0x600733c73c7bBC11BaD98ADE667fF94479d6BDf1	authority
account 2	0x41Eb82045e1FbB1b4B28f77dfc956B00D14949a3	normal
account 3	0xda16cDaaBA1c7C4054Ed8F3d2acADbCE9e1a0975	authority
account 4	0x3238eCc988880633A7d45213fB7773247B1271cb	authority
account 5	0x7e919EF790BD06976457e8c1C7315c7F36c77bc9	authority

5.1.3 Add University

If account 2 is held by a university, the institution must contact one of the authority addresses to confirm its identification. Following that, the authorities will designate this location as a university. Account 5 will do the verification for account 2 since it is the authority address. If the authentication is successful, account 5 will invoke the smart contract's interface for adding a university. Furthermore, it will designate account 2 as a university address, describe the name of the institution, and deliver the token to account 2's address. Any call requiring EVM to participate in the computation spends ETH in Ethereum, although the token is different. It is a currency used to prevent unauthorized data change when a single node's private key leaks. Figure 15 shows the university name of account 2, as well as the number of tokens remaining in this address.

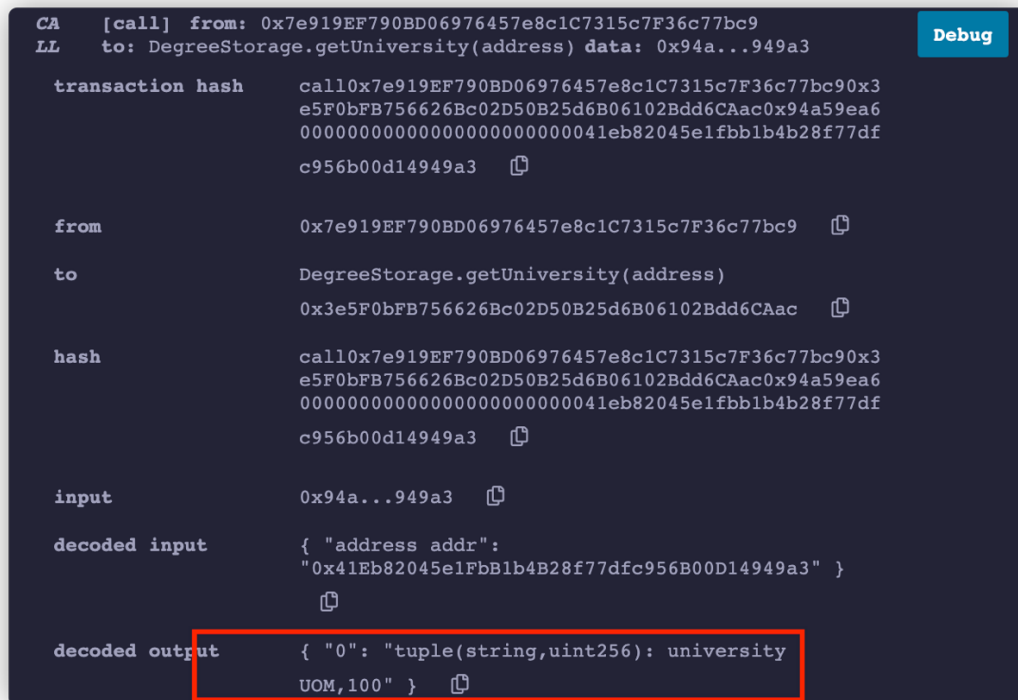


Figure 15 The result of getUniversity function

Table 9 The role of five addresses (idx.5)

name	address	role
account 1	0x600733c73c7bBC11BaD98ADE667fF94479d6BDf1	authority
account 2	0x41Eb82045e1FbB1b4B28f77dfc956B00D14949a3	university
account 3	0xda16cDaaBA1c7C4054Ed8F3d2acADbCE9e1a0975	authority
account 4	0x3238eCc988880633A7d45213fB7773247B1271cb	authority
account 5	0x7e919EF790BD06976457e8c1C7315c7F36c77bc9	authority

5.1.4 Degree Storage

Account 2 is the university address, which has the authority to grant and revoke degrees on the blockchain, and we will test this procedure in the next steps. If Mark is going to graduate and his institution will provide him with a diploma. To begin, the university must enter Mark's information into its own MySQL database, as illustrated in Figure 16. Following this stage, the data in MySQL is saved in a particular format. After we add the school's name to the data, we can compute the hash value of the whole dataset, as shown in Figure 17. Account 2 will commit the hash value, certificate ID, and address to the blockchain, completing the process of creating a degree certificate.

certificate_id	name	id_card	major	start_time	end_ti...	is_valid
UOM41Eb82115321	Mark Zhang	556123141212321	Software Engineering	Sept 2016	Jun 2020	1

Figure 16 Degree certificate in MySQL

Assume a university wishes to revoke a certificate. In that instance, the issuing address is all that is required to invoke the smart contract's revocation interface. It will remove the credential's hash from the blockchain, making it impossible for a third party to verify the degree's validity, thus accomplishing the goal of revoking the credential. Figure 19 depicts the input.













status	true Transaction mined and execution succeed
transaction hash	0xbe894a7011459b7b9018e2c62dc1a55e7b0199bcb8b131c94f57c076c3a95f61 
from	0x41Eb82045e1FbB1b4B28f77dfc956B00D14949a3 
to	DegreeStorage.revokeDegree(string) 0x16D06D748aCe210C42A9F302d20a38037e0f3371 
gas	47444 gas 
transaction cost	23722 gas 
hash	0xbe894a7011459b7b9018e2c62dc1a55e7b0199bcb8b131c94f57c076c3a95f61 
input	0x658...00000 
decoded input	{ "string degreeID": "UOM41Eb82115321" } 
decoded output	- 
logs	[]  
value	0 wei 

Figure 19 The result of revokeDegree function

5.2 Evaluation

We have thoroughly tested our system's whole procedure. How does this system perform in comparison to other systems? We will assess data security, storage strain on nodes, and the necessity for a centralized database.

- Plan A: This is the system proposed in this thesis. The actual data of the credentials are kept on the university's server, and the hash value of the credentials is calculated and stored

on the Ethereum blockchain.

- Plan B: Using Ethereum, save the plaintext of the credentials.
- Plan C: Universities use a key to encrypt the plaintext of the credential and get the ciphertext. The certificate's ciphertext and hash value are then stored on Ethereum.

The benefit of Plan A over the other two options is that it reduces the storage burden on each node in the Ethereum network. It just needs to hold the certificate's hash value. At the same time, since the hash is a one-way function, the plaintext of the credential cannot be deduced, ensuring a high level of anonymity. However, the original credentials must be kept on the university's server, and the failure tolerance is poor since the original data is only stored on the university's server.

The Plan B approach is the most basic, keeping the plaintext of credentials directly in Ethereum for anybody to view. The actual data, however, consume more storage space than the hash; thus, there is no privacy. The benefit is that all Ethereum nodes are backed up, resulting in great fault tolerance and the elimination of the need for extra storage servers.

Plan C involves colleges encrypting student information and storing it in Ethereum along with hash values. The benefit is that user privacy is quite strong, and institutions only need to access the blockchain to obtain the ciphertext when they require the credentials, then decode it. To validate a certificate, a third party may get the hash from Ethereum. This technique is extremely fault-tolerant and does not need the use of extra servers for storage. The cypher text, however, is kept on the blockchain, and information may still be broken. The fatal flaw is that Ethereum nodes both keep the hash and the credentials' ciphertext. There is an excessive storage load on nodes.

Plan A, which we suggested in this article, is a workable compromise. We can enhance information privacy and decrease the storage burden while compromising fault tolerance.

Table 10 Performance of three plans

	Node Storage Pressure	Privacy	Fault Tolerance	Additional Servers
Plan A	Low	Good	Bad	Need
Plan B	Medium	Bad	Good	No need
Plan C	High	Medium	Good	No need

6 CONCLUSION & FUTURE WORK

6.1 Conclusion

The goal of this project is to create an Ethereum-based system for storing degree certificates. We created the system's most important smart contract in solidity. Given the burden of node storage, the maintenance of authority addresses, and the privacy of credentials, the main task of this system is to design a voting mechanism in our smart contract, achieve flexible authority handover, and find a way to reduce the storage burden on blockchain nodes. In this system, the plaintext of credentials is kept on the university's server, while the hash values of the credentials are stored in Ethereum, substantially reducing node storage space. The test results indicate that this system effectively mimics the abrogation voting process in a completely decentralized system among five authority addresses, causing the voted node to lose its management power. Simultaneously, the system mimics the storage of credentials and the process of finding the hash value of credentials on the blockchain, significantly reducing the system's storage space and protecting users' privacy. There are also many difficulties in the creation and design of this system. For example, which solution in the voting process is the best? The voting method used will have a significant impact on the system's stability and durability. Furthermore, it is important to keep in mind the need of preserving the storage capacity while ensuring the security of degree certificates.

6.2 Future Work

This article builds an Ethereum-based credential storage system with a voting mechanism and minimal storage needs. However, there are many potential possibilities to enhance this system's storage space and voting process, and I provide some ideas below.

- Improve the authority node voting process. The system's present voting method utilizes 51 percent voting. However, in the real application situation, 51 percent voting may not be appropriate. We will need to modify it in the future based on the various companies. Furthermore, different authority addresses have the same weight in the present voting

method, and we may require a ballot with weight in certain situations, which is also an issue that has to be addressed in the future.

- To reduce reliance on university databases. When constructing, we have been attempting to minimize the storage load on Ethereum and transfer the burden to the university database. University servers, on the other hand, are often housed in the same room and are not physically separated. When an event happens, such as a fire, the original data of the credentials may be destroyed; thus, the original credentials must be regarded to be directly saved on Ethereum in the future. However, when saved on the blockchain, the credentials will take up much too much storage space. As a result, the safest and practical method is to discover a good data compression technique that compresses and encrypts the credentials before storing them on Ethereum.

- Embed credential verification calculations into our smart contract. The present approach requires the organization that needs to verify to visit Ethereum in order to get the hash and compare it to its own computed hash, which is inconvenient. In the future, we aim to include the verification function within the smart contract. Employers simply need to submit their credentials, and the smart contract will compute the hash value and determine if it matches the hash on the blockchain. Users must, however, pay a specific amount of ETH as gas since the smart contract requires some computation. As a result, computations on the Ethereum network are costly. The decision to include the calculating process must be made based on the real requirements.

REFERENCES

- Arenas, R. and Fernandez, P. (2018). CredenceLedger: A Permissioned Blockchain for Verifiable Academic Credentials. *2018 IEEE International Conference on Engineering, Technology and Innovation, ICE/ITMC 2018 - Proceedings*.
- Buterin and Vitalik. (2014). Ethereum White Paper: A Next Generation Smart Contract & Decentralized Application Platform. *Etherum*, (January), pp.1–36. [online]. Available from: <https://github.com/ethereum/wiki/wiki/White-Paper>.
- Cao, B. et al. (2019). When internet of things meets blockchain: Challenges in distributed consensus. *IEEE Network*, 33(6), pp.133–139.
- Carter, J.L.J. and Wegman, M.M.N.M. (1979). Classes of Hash Functions. *Journal of computer and system sciences*, pp.143–154. [online]. Available from: <http://www.sciencedirect.com/science/article/pii/0022000079900448>.
- Castro, M. and Liskov, B. (1999). Practical Byzantine Fault Tolerance. *Juvenile Delinquency in Europe and Beyond: Results of the Second International Self-Report Delinquency Study*, (February), pp.359–368.
- Dyreson, C.E. and Snodgrass, R.T. (1993). Timestamp semantics and representation. *Information Systems*, 18(3), pp.143–166.
- Farah, J.C. et al. (2018). A blueprint for a blockchain-based architecture to power a distributed network of tamper-evident learning trace repositories. *Proceedings - IEEE 18th International Conference on Advanced Learning Technologies, ICALT 2018*, pp.218–222.
- GAVIN WOOD. (2014). Ethereum: a secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151, pp.1–32.
- Gervais, A. et al. (2016). On the security and performance of Proof of Work blockchains. *Proceedings of the ACM Conference on Computer and Communications Security*, 24-28-Octo, pp.3–16.
- Gilbert, H. and Handschuh, H. (2004). Security analysis of SHA-256 and sisters. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3006, pp.175–193.
- Grech, N. et al. (2018). MadMax: surviving out-of-gas conditions in Ethereum smart contracts. *Proceedings of the ACM on Programming Languages*, 2(OOPSLA), pp.1–27.
- Guegan, D. (2017). Public Blockchain versus Private blockchain. *Documents de Travail du Centre d'Economie de la Sorbonne*.
- Hirai, Y. (2017). Defining the ethereum virtual machine for interactive theorem provers.

Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 10323 LNCS, pp.520–535.

Hu, Q. et al. (2021). An Improved Delegated Proof of Stake Consensus Algorithm. *Procedia Computer Science*, 187, pp.341–346. [online]. Available from: <https://doi.org/10.1016/j.procs.2021.04.109>.

Hutchison, D. and Mitchell, J.C. (1973). *Post-Quantum Cryptography*.

Irestig, M. et al. (2005). Peer-to-peer computing in health-promoting voluntary organizations: A system design analysis. *Journal of Medical Systems*, 29(5), pp.425–440.

Johnson, D., Menezes, A. and Vanstone, S. (2001). The Elliptic Curve Digital Signature Algorithm (ECDSA). *International Journal of Information Security*, 1(1), pp.36–63.

Kiayias, A. et al. (2017). PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security - CCS'16*, 1919(January), pp.1–27. [online]. Available from: <http://peerco.in/assets/paper/peercoin-paper.pdf>http://fc17.ifca.ai/preproceedings/paper_73.pdf<http://arxiv.org/abs/1606.06530>https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2977811<http://dl.acm.org/citation.cfm?doid=2976749.2978389><http://>

Kim, S.K. et al. (2018). Measuring ethereum network peers. *Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC*, pp.91–104.

Kraft, D. (2016). Difficulty control for blockchain-based consensus systems. *Peer-to-Peer Networking and Applications*, 9(2), pp.397–413.

Kuvshinov, K. et al. (2018). Disciplina : Blockchain for Education. *Semantic Scholar*, pp.1–17. [online]. Available from: <https://www.semanticscholar.org/paper/Disciplina-%3A-Blockchain-for-Education-Kuvshinov-Nikiforov/958c1a5760b3f41f65482593eb8e365dfc4ccf3f>.

Lamport, L., Shostak, R. and Pease, M. (1982). The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3), pp.382–401.

Latifa, E.-R. et al. (2017). Blockchain: Bitcoin Wallet Cryptography Security, Challenges and Countermeasures. *Journal of Internet Banking and Commerce*, 22(3), pp.1–29. [online]. Available from: <http://www.icommercecentral.com/open-access/blockchain-bitcoin-wallet-cryptography-security-challenges-and-countermeasures.pdf><https://search.proquest.com/docview/1992203656?accountid=29104>https://openurl.wu.ac.at/resolve?url_ver=Z39.88-2004&rft_val_

Luo, P. et al. (2016). Differential Fault Analysis of SHA3-224 and SHA3-256.

Merkle, R.C. (1988). A digital signature based on a conventional encryption function. *Advances*, pp.369–378.

- Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. *Artificial Life*, 23(4), pp.552–557.
- Nofer, M. et al. (2017). Blockchain. *Business and Information Systems Engineering*, 59(3), pp.183–187.
- Ocheja, P. et al. (2019). Managing lifelong learning records through blockchain. *Research and Practice in Technology Enhanced Learning*, 14(1).
- Ocheja, P., Flanagan, B. and Ogata, H. (2018). Connecting decentralized learning records: A blockchain based learning analytics platform. *ACM International Conference Proceeding Series*, pp.265–269.
- Pongnumkul, S., Siripanpornchana, C. and Thajchayapong, S. (2017). Performance analysis of private blockchain platforms in varying workloads. *2017 26th International Conference on Computer Communications and Networks, ICCCN 2017*.
- Rodriguez, J and Guardo, G. (2005). *MySQL Bible*.
- Rooksby, J. and Dimitrov, K. (2020). Trustless education? A blockchain system for university grades 1 . *Ubiquity: The Journal of Pervasive Media*, 6(1), pp.83–88.
- Schollmeier, R. (2001). A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. *Proceedings - 1st International Conference on Peer-to-Peer Computing, P2P 2001*, pp.101–102.
- Szabo, N. (1997). View of Formalizing and Securing Relationships on Public Networks | First Monday. *First Monday*, pp.1–21. [online]. Available from: <https://firstmonday.org/ojs/index.php/fm/article/view/548/469>.
- Szydło, M. (2004). Merkle tree traversal in log space and time. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3027, pp.541–554.
- Turkanović, M. et al. (2018). EduCTX: A blockchain-based higher education credit platform. *IEEE Access*, 6, pp.5112–5127.
- Wang, H. et al. (2018). Blockchain challenges and opportunities: a survey. *International Journal of Web and Grid Services*, 14(4), p.352.