Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

GANACHE: A Robust Framework for Efficient and Secure Storage of Data on Private Ethereum Blockchains

Garima Mathur (garimamathur@sistec.ac.in)
SISTec Gandhi Nagar

Research Article

Keywords: Ethereum, Gas limit, Blockchain, transaction, solidity, Metamask

Posted Date: October 30th, 2023

DOI: https://doi.org/10.21203/rs.3.rs-3495549/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Abstract

Blockchain technology has gained significant attention in recent years due to its decentralized and secure nature. Ethereum a popular blockchain platform supports the execution of smart contracts and enables the storage of data on the blockchain. However Ethereum's public nature may not be suitable for scenarios where sensitive or private data needs to be stored and accessed securely. In this paper we present GANACHE an efficient private Ethereum blockchain tool that addresses the confidentiality and security concerns associated with data storage. GANACHE leverages various cryptographic techniques such as encryption and zero-knowledge proofs to ensure the privacy of stored data while maintaining the benefits of blockchain technology.

I. Introduction

1.1 Overview

Blockchain technology has gained significant traction in recent years for its potential to revolutionize various industries [1]. One of the most well-known blockchain platforms is Ethereum which supports the development and execution of smart contracts. While Ethereum's public blockchain offers decentralized security and transparency there are situations where private blockchains are preferred.

Private blockchains provide organizations with more control over their data and network governance. However storing data securely on private blockchains can be a challenging task [2]. In many cases traditional storage solutions may not provide adequate security or efficiency when it comes to large-scale data storage. To address these challenges a robust framework called Ganache has been developed for efficient and secure storage of data on private Ethereum blockchains.

Ganache leverages various features and techniques to enhance the security and efficiency of data storage. One of its key components is the implementation of advanced encryption algorithms to ensure the confidentiality and integrity of stored data. By encrypting the data before it is stored on the blockchain Ganache protects sensitive information from unauthorized access..

This innovation can be employed in securing DNA data [3,4], prescriptions for pharmaceuticals with supply-chain management, pregnancy, and sensitive data that must be regulated, data sharing, and managing a survey trail of therapeutic activities. Additional healthcare services zones that can benefit from blockchain technology advancement include provider authorizations, clinical charging, contracting, medical record exchange, clinical basics, and so on [5,6].

Medical systems are currently undergoing transformations to accommodate a patient-centered approach. Personal data security may be established through the use of blockchain technology in the healthcare sector; patients can now own their medical records and preserve them without interference from any central body. Another difficulty in the healthcare system is the storage and movement of sensitive patient data through the internet, which is highly vulnerable to many forms of attacks.

The problem can be solved by implementing a blockchain architecture within the healthcare system, as blockchain can render data illegible. Patients can now safely access their data from any location at any time. Application of Blockchain Technology in Medicine and Healthcare, challenges faced in using them, and their future perspectives can be seen in figure 1.

1.2 Background Details

Ganache is a powerful and efficient private Ethereum blockchain tool that is specifically designed for implementing and securing the storage of data. It provides developers and organizations with a secure and customizable environment to build test and deploy smart contracts and decentralized applications (dApps) on a private blockchain network.One of the key advantages of using Ganache is its ability to work seamlessly with the Ethereum Virtual Machine (EVM allowing developers to leverage all the features and functionalities offered by the Ethereum platform. This includes utilizing the Solidity programming language Ethereum's native currency (Ether and the extensive range of Ethereum-based smart contracts and dApps [24].

Ganache offers various features that make it an ideal choice for implementing and securing the storage of data. Firstly it provides a local development network which allows developers to test their smart contracts and applications in an isolated environment. This eliminates the need for connecting to the main Ethereum network reducing the risk of exposing sensitive data during the development phase. Furthermore Ganache allows developers to simulate different network conditions such as latency and network congestion enabling them to accurately test the performance and scalability of their applications.

This is crucial for ensuring that the storage and retrieval of data remain efficient and secure even under real-world conditions. In terms of security Ganache offers several built-in features that protect the storage of data. It provides a secure key management system that allows developers to generate and manage private keys for accounts ensuring that only authorized parties have access to the data stored on the blockchain. Additionally Ganache supports the integration of various security protocols such as encryption and multi-factor authentication to further enhance the protection of stored data.

Ganache also offers comprehensive documentation and a user-friendly interface making it easy for developers to set up and manage their private Ethereum blockchain networks. It provides detailed logs and debugging tools allowing developers to quickly identify and resolve any issues related to data storage and retrieval.

Overall Ganache is an efficient and reliable tool for implementing and securing the storage of data on a private Ethereum blockchain network. Its features and capabilities empower developers to build robust and scalable decentralized applications while ensuring the confidentiality and integrity of stored data. By leveraging Ganache organizations can effectively harness the power of blockchain technology to transform their data storage processes and enhance the security of sensitive information.

II. Data management issues and solution in the healthcare industry

Blockchain technology has a wide variety of uses in the healthcare industry. It can be used to safely communicate patient medical records, manage the pharmaceutical supply chain, and might be very beneficial to genetic code decoding studies in the medical field. From dealing with the resurgence of dangerous diseases to securely encrypting patient records, blockchain technology is already gaining prominence [7].

Given that security is a major concern in the healthcare sector, the most well-known blockchain application for medical services today is to safeguard our important clinical data. Over 176 million patient records were found in information leaks between the years 2009 and 2017.

2.1 Problem:

Blockchain is a technology that is ripe with security applications since it can keep patient data unchangeable, decentralized, and transparent. Additionally, the blockchain has a contradictory quality in that it can be both transparent and private. Here, complex and secure codes that can guarantee the security of healthcare data are used to conceal the identity of each individual. Patients, specialists, and medical service providers may all access the same data quickly and securely because of this technology's decentralized design [8]. The following are a few concerns with data security in healthcare.

Issue 1: Assurance of Clinical Trials' Information Security

Clinical trials are used to evaluate the effectiveness of a specific medicine or to approve or reject the usage of a particular drug. Researchers are constantly interested in gathering and recording data regarding test findings, quality analyses, etc. Monitoring everyone is challenging due to the detailed study that each researcher is subjected to. The outcome of the research might then easily be changed by changing this information, or it could be concealed.

Issue 2: Management of Patient Data

The Health Insurance Portability and Accountability Act (HIPAA) strictly controls patient data privacy and demands that PHI be kept in the most secure possible manner. Nevertheless, there is a second problem with PHI that has been noted: individuals occasionally have to provide third parties access to their clinical records (for example with drug stores when they have to purchase explicit meds). So how does blockchain manage to preserve data while allowing limited access?

2.2 A possible solution for the problems with healthcare data security

Everyone may now verify the accuracy of any report that has been stored on a device thanks to this technology. It offers proof-of-existence by adding records to the transaction's form, and blockchain-based healthcare information exchange platforms for the security of healthcare data validate the data by all system nodes. Blockchain stores immutable data, as was already said [10,11]. Clinical trial data can be

rendered immutable by this feature, which also makes it challenging to alter. To understand how blockchain can provide clinical preliminary assessments with existing evidence, two experts from Cambridge University conducted research in 2016. They determined whether or not data had been modified by comparing the code created by the system with the original code. Because of this special property of SHA 256, each time the information is altered, a new hash is created. Blockchain creates a hash and patient ID for each PHI block. Every entity can obtain the crucial information they need by using an API without knowing a patient's identity. Additionally, a patient can decide to whom to grant access and whether this entrance would be fully or partially open. Additionally, if the patient was unsure about what he or she was doing, they might choose explicit third parties who would have to agree to the sharing of PHI. Along with healthcare, blockchain has excellent potential for use in a variety of other sectors.

III. Implementation

GANACHE implements a comprehensive data storage mechanism that maintains the confidentiality of the stored data. This mechanism includes encryption of data at rest and in transit ensuring that only authorized parties can decrypt and access the information [9]. GANACHE also supports access control policies enabling fine-grained permissions to regulate data access based on user roles and privileges. We conducted extensive performance evaluations to assess GANACHE's efficiency and scalability. The results demonstrate that GANACHE maintains a high level of performance even when dealing with large amounts of data and concurrent access requests. The overhead incurred by the privacy-preserving techniques is minimal making GANACHE suitable for practical deployment.

The software used to implement the architecture is described in this section. The Ethereum Blockchain is simulated using the tool named"Ganache." Due to its benefits, such as the accessibility of the blocks or transactions, a graphical user interface is implemented. Additionally, the implementation of writing, compiling, and migrating Smart Contracts in the programming language "Solidity" uses the "Truffle Suite Framework" plugin for "Node.js." In addition to "Solidity," the script that executes the component's primary functions is created using the "Python" programming language. Additionally, to communicate with an Ethereum blockchain, Python's "web3.py" package is necessary.

3.1 Installing Dependencies

It is necessary to install and set up a few dependencies before the application-building process can continue. The dataset is stored using the GANACHE private blockchain technology in this work. This private blockchain enables the quickest Ethereum-based application development. This tool aids in both the creation and the deployment of our dApps in a highly secure setting [12]. We get 100 free accounts on our personal Ethereum blockchain as a result.

The private blockchain is constructed with smart contracts using the solidity programming language. The following dependencies must be installed before developing smart contracts:

1. **NPM :** NPM, or node package management, is a component of Node.js and is installed by typing $ node -V.
2. **Truffle:** For creating Ethereum smart contracts [13,14], the Truffle Framework is a crucial tool. Smart contracts are created using the Solidity programming language. Truffle is the most important dependency used for developing decentralized ethereum blockchain applications. It can be installed using the command **$ npm install –g truffle.**
3. **GANACHE:** GANACHE is a personal blockchain, which is a local development blockchain that may be utilised as a public blockchain. It is used for smart contract deployment and testing [15]. It provides us with 100 free accounts on our private ethereum blockchain. It can be installed from the truffle website.
4. **Metamask:** The plugin used with Google Chrome, and Metamask is another dependency. Before using the Ethereum blockchain, we must install it from the Google Chrome Web Store, it is installable. It is important to add metamask extension to google chrome because current browsers doesn't support connecting to blockchain networks. The browser becomes a blockchain browser when the Metamask addon is added [16]. We may deploy our project by logging into our Ganache accounts through Metamask. As a means of connecting to the blockchain and for transactional purposes, MetaMask is also used to manage user accounts. We can access the distributed web in your browser by using the extension or plugin known as MetaMask. Without running a full Ethereum blockchain, it enables you to run Ethereum Decentralized applications directly in your browser [17]. A secure identity vault is a feature of MetaMask that offers a user interface for managing your online personas and signing blockchain transactions. The MetaMask add-on is available for Chrome, Firefox, and Opera.

3.2 Smart Contracts based on the Blockchain

This section of the paper covers the most important part, which includes what version of compiler to be used and it can be better identified from first line of code. Keeping track of the records of exchanged data is the Smart Contract's most crucial task [18]. As a result, this is accomplished. The two variables are "time" for the time stamp and "temp" for the temporary variable expected by the function. With the method "push," these are added at the end of the corresponding arrays "timeStamp" and "only temp." Before this, it is determined that the originator of the transaction must be the holder of the Smart Contract using the function "require." Before the values are transmitted, the following function from the Smart Contract that is defined must be invoked. In this case, the owner is set to be the transaction's sender. The Smart Contract's code 1 is where specific getter functions are defined. These can be used to query the specific data arrays, their overall length, or the number of readings that have been saved in the blockchain.

Code 1: Displays the use of “get” function in Smart Contract

```
Pragma Solidity>=0.4.18<0.6.0;
function getTimeData() public view returns (uint[] memory)
 { require(msg.sender == owner);
return timeStamp;
 }
function getTempData() public view returns (uint[] memory)
{
require(msg.sender == owner); return onlyTemp;
}
function getLength() public view returns (uint)
{
require(msg.sender == owner); return onlyTemp.length;
}
```

Code 2: Displays key function of Smart Contracts based on Solidity programming

```
Pragma Solidity>=0.4.18<0.6.0
Function addData(uint time, uint temp) public
{
require(msg.sender==owner); timestamp.push(time);
onlyTemp.push(temp);
}
Function addData(uint time, uint temp) public {
Require(msg.sender==owner); timestamp.push(time);
onlyTemp.push(temp);
}
```

Once the solidity smart contract is created using remix IDE. This contract can be deloyed using Ganache. Figure 2 illustrates how blocks are created in the GANACHE, here each block has its cryptographic hash value to refer to its predecessor. The produced hash value of the data, the block's source address, the block's creation time (i.e., the time it was mined), and other information are all fully disclosed in each block.

Each block contains the total number of ether used in its creation in terms of gas value used. Figure 3 shows specifics on the gas value used, the gas limit specified, the time taken to create a block, and the block 1 hash value.

3.3 Python-based Integration of an Ethereum Blockchain

The system required the installation of the web3.py module to connect with an Ethereum blockchain using Python. Numerous essential functionalities are offered by this library. The blockchain's IP address needs to be entered first. The Web3 HTTP Provider function Object() { [native code] } is used for this, and you must enter the address and port there before Ganache will utilize these in its settings. The first account in Ganache is then designated as the default account, and this is where all transactions are carried out. The Smart Contract needs to be initialized at the start. The Smart Contract cannot be called until that time. The function "transact()" must be expanded with the option "gas" if Ganache is not run with the option "automine."

The transaction is then checked to see if it was successful using the function "waitForTransactionReceipt()." Among other things, this is used to verify the evaluation tests. The following function, "waitForTransactionReceipt," is used to determine the time when a block was created. The variable "blocktime" contains information about the block addressed by "confirm.blockNumber." The "blockzeit.timestamp" command, among other things, provides access to the block's creation time.

3.4 Workflow

In the proposed approach, a user uploads data or any other file to the blockchain. In the beginning, Ganache's private local blockchain is configured, and it is integrated with a web browser via the Metamask plugin.

As was already mentioned, the user first needs a local blockchain network, which Ganache offers. For transactional purposes, Ganache's accounts are added to Metamask. A particular amount of Gas in the form of Ether is required for each transaction. The cryptocurrency token known as ether powers the blockchain network. Ten accounts, each with 100Eth that can be used for transactions, are provided by Ganache. On a private Ethereum blockchain, we may use these accounts to perform tests, send commands, inspect state, and manage chain operations. Any one of these 10 accounts may be used to log in to Metamask using the private key made available by that account in Ganache.

Now that our Web browser supports the blockchain network, we can upload the files using an interface we created ourselves (UI). When a file is chosen for uploading, Metamask creates a window to confirm the transaction. The ETH is taken from our account when we confirm the transaction, but we also have the option to refuse it. The previously acquired hash value, which is generated when the file is uploaded, is necessary for the process of getting the file back.

IV. Result Analysis

A secure peer-to-peer and decentralized system is advised for usage by all healthcare practitioners and patients for the maintenance and management of complicated healthcare data. The proposed system has a cryptographic POS consensus mechanism as its core[19]. The main argument in favor of Blockchain adoption is that it cannot be changed. With the help of Hyperledger, healthcare data can be

exchanged between parties in a way that protects data privacy by isolating transactions and consuming less power and processing power [20].

Medical data can be securely exchanged across several network nodes because of the network's decentralized architecture. The suggested framework frequently resists mistakes and data loss. The nodes that are included in the proposed framework construct a ledger of the previous transaction using the Blockchain hashing technique and the SHA 256 cryptographic hash function [21]. To keep the hashes of each transaction, both earlier and later, the nodes require appropriate disc space. The decentralized blockchain technology makes use of a secure hashing algorithm.

4.1 Algorithm:

A modular exponential method was used to calculate the hash values displayed in Figure 6. This approach finds the remainder of an integer base value and the exponent (e) of power nth by dividing the modulus of a non-negative integer (n) by its exponent (e) (b).

Equation (1) describes the idea of trapdoor one-way functions, that have been defined between two distinct sets like X and Y. The hash value is also produced using Equation (1).

F:ab, **(1)**

where F(a)=F(b)

Where Equation (2) specifies the operation of hashing and f is a trapdoor function that hides some concealed information.

h(m)=m with h:M0,1n **(2)**

Algorithm 1 transforms the message into binary codes using the ASCII code. The padding of the code with zeros yields the proper block size i.e 512 bits [22]. The words in each group are denoted by w, and the block is then split into 16 groups, starting with M_0-M_{15}.

1: Increase the number of zero 0 bits in the data input until it reaches 448 bits, then add a 64-bit length to the supplied data until it reaches 512 bits.

2: After the 512-bit data has been combined, divide it into 16 groups. Mo-M15

3. Configure the vectors Ko-K63 and ho-h7.

4: Assume that the initial values for A, B, C, D, E, F, G, and H are ho-h7.

Computation

5. After setting t to loop from 0 to 63, update as follows: Bt+1=At.

6: Ct+1 = Bt, Dt+1 = Ct, Ft+1 = Et, G t+1 = Ft, Ht+1 = Gt, At+1 = Ht+Σ1(E1)+ Ch(Et, Ft, Gt) + Kt + Wt + Σo(At) + Maj(At, Bt, Ct), Et+1+ Ht + Σ1(Et)+ Ch(Et, Ft, Gt) + Kt + Wt + D1,

7: Let ho= ho +A63, h1 = h1+B63, h2+C63, h3=h3+D63, h4 = h4+E63, h5 = h5+F63, h6=h6+ G63, h7=h7+ H63 ho-h7.

8: Σ1(Et), Σ0(At), Maj(At, Bt, Ct) and Ch(Et, Ft, Gt) are logical functions, and Wt, is updated according to:

$$W_t = \begin{cases} Mo, & 0 \le t \le 15, \\ \sigma 1\,(Wt-2) + Wt-7 + \sigma 0\,(Wt-15) & \\ \quad + Wt-16, & 16 \le t \le 63 \end{cases}$$

In order to combine recent transactions B with older transaction blocks P, the blockchain's mining algorithm 2 uses block hashing. Maintaining a record of previous transactions makes it easier to calculate and validate future transactions using this method.

1. P: The hash of the previously mined block
2. B: A block of transaction
3. H: A hash function
4. D: Difficult level
5. 0 Retrieve P
6. Modify B
7. IF H (P, B, Some Random Number) >= D End
8. GOTO 1

For hash verification, the elliptic curve technique, which is based on the Weierstrass Equation, is used (3).

$$y2 = x3 + ax + b \quad (3)$$

Blockchain makes use of ECDSA to validate user- or participant-entered ledger data. The cryptographic method ECDSA ensures that data is only transferred among authorized nodes because it depends on the hash value [23][24]. The hashes of the prior and subsequent blocks have been combined using Equation (4). The terms in the equations stand in for hash blocks containing private information.

$$b = (x2 - y2)/(x1 - y1)mod$$
$$Mt1 = b2 - y1 - x1modM$$
$$t2 = b(y1 - t1) - x2modM \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (4)$$

Metamask asks the user to pay when a request is made to create a transaction on the blockchain. Gas is required for transactions written to Ethereum, and the transaction cost (Tcost) is determined using the formula below:

$$Tcost = gasLimit \times gasPrice$$

where miners decide and control the gasLimit and gasPrice values, which change with each block. It is less expensive to store the compressed files on the blockchain than it is to store the actual. The amount of storage needed to store the hash values (Sh) is represented by the following equation:

$$S(h) = \frac{H + Hash\ size \times N}{H + \sum Ti} \ldots\ldots\ldots\ldots\ldots\ldots (6)$$

Where T is the native size of each transaction, HashSize is the size of the hash for each transaction, N represents the number of transactions or hashes kept in the blockchain, and I spans from 1 to N. The method for uploading and storing documents on the blockchain is described in Algorithm 1.

Document storing algorithm in Blockchain:

Function upload (File)

For each file **do**

Retrieve file

Buffer transforms text file to a binary stream

Send binary stream to SHA 256

SHA 256 assigns the hash value to it

Hash value is returned

Hash value is sent to the Blockchain for storage

Function Storage (Hash)

For each Hash **do**

Smart contracts are instantiated by Web3

Ganache displays transaction request

Transaction is approved, Hash is stored on Ethereum

4.2 Analysis:

Ganache first needs to be started and set up before running the general tests. Then, using the relevant console command, the Smart Contract may be moved. The incoming transactions and the subsequent blocks are both listed in the ganache at the same time. Following the test run, the first account's TX COUNT in Ganache displays the total number of transactions that were carried out. As was previously said, after calling the function and transferring the value to the blockchain, the Python script waits for a transaction hash to confirm the transfer. Since no hash is transmitted if the transaction fails, this is for function checks. Querying the data records saved in the blockchain after the test run is another option. The time required to execute a smart contract climbs linearly and can take up to 9500 seconds or 3,4 hours. The use of the PoW algorithm within the Ethereum blockchain provides a clear explanation for this growing duration.

V. Conclusion and future scope

GANACHE offers a comprehensive and efficient approach for implementing and securing the storage of data on private Ethereum blockchains. By incorporating cryptographic techniques and access control mechanisms GANACHE addresses the privacy concerns associated with public blockchain platforms. The results of our performance evaluations and use case scenarios confirm the efficacy and practicality of GANACHE as a powerful tool for private data storage on the Ethereum blockchain. The suggested system involves a user who wishes to upload data or any other file to the blockchain. Ganache provides private local blockchain configuration, and the Metamask plugin is used to integrate it into the web browser. Compressed files are kept on the blockchain to reduce the expenses associated with storing huge amounts of data there. The findings indicate that there is a cost difference between compressed and uncompressed files, and that cost will rise as more transactions are made on the blockchain.

Declarations

Future Scope: GANACHE has immense potential for further development and improvement. By addressing scalability interoperability privacy smart contract optimization governance integration with external systems user experience and security GANACHE can become a robust and efficient tool for implementing and securing the storage of data on private Ethereum blockchains.

Data availability: The datasets generated during and/or analyzed during the current study are not publicly available due to [security reasons] but are available from the corresponding author on reasonable request.

Funding

No funding received

Acknowledgements

No one to be acknowledge

Competing interests: The authors declare no competing interests.

References

1. Khan, M.A.; Salah, K. IoT security: Review, blockchain solutions, and open challenges.Future Gener.Comput. Syst. 2018, 82, 395–411.
2. Garima Mathur, Anjana Pandey and Sachin Goyal, " Blockchain based healthcare information exchange systems for security of healthcare data", TOJQI,2021,Vol.12, issue 8,pp. 4498-4507.
3. Aste, T.; Tasca, P.; Di Matteo, T. Blockchain Technologies: The Foreseeable Impact on Society and Industry. Computer 2017, 50, 18–28.
4. S. Shadab, M. T. Alam Khan, N. A. Neezi, S. Adilina, and S. Shatabda. (2020). DeepDBP: deep neural networks for the identification of DNA-binding proteins. Informatics in Medicine Unlocked. vol. 19, article 100318.
5. Sardaraz, M.; Tahir, M.; Ikram, A.A. Advances in high throughput DNA sequence data compression. J. Bioinf. Comput. Biol. 2016, 14, 1630002.
6. Zhu, Z.; Zhang, Y.; Ji, Z.; He, S.; Yang, X. High-throughput DNA sequence data compression. Briefings Bioinf. 2015, 16, 1–15.
7. Khan, M.A.; Salah, K. IoT security:Review, blockchain solutions, and open challenges.Future Gener.Comput. Syst. 2018, 82, 395–411.
8. Aumasson, J. Serious Cryptography: A Practical Introduction to Modern Encryption; No Starch Press: San Francisco, CA, USA, 2017.
9. G. Mathur, A. Pandey and S. Goyal, "Immutable DNA Sequence Data Transmission for Next Generation Bioinformatics Using Blockchain Technology," 2nd International Conference on Data, Engineering and Applications (IDEA), Bhopal, India, 2020, pp. 1-6, doi: 10.1109/IDEA49133.2020.9170715.
10. Sestrem Ochôa, I.; Reis Quietinho Leithard, V.; Calbusch, L.; De Paz Santana, J.F.; Delcio Parreira, W.; Oriel Seman, L.; Zeferino, C.A. Performance and Security Evaluation on a Blockchain Architecture for License Plate Recognition Systems. Appl. Sci. 2021, 11, 1255. https://doi.org/10.3390/app11031255.
11. Raval, Siraj (2016). "What Is a Decentralized Application?". Decentralized Applications: Harnessing Bitcoin's Blockchain Technology. O'Reilly Media, Inc. pp. 1–2. ISBN 978-1-4919-2452-5. OCLC 968277125. Retrieved 6 November 2016 – via Google Books.
12. Metamask. https://metamask.io/.
13. Rossi, L., Salviotti, G., & Abbatemarco N. (2019). Towards a Comprehensive Blockchain Architecture Continuum. Proceedings of the 52nd Hawaii International Conference on System Sciences, 4605-4614. Available at: doi: 10.24251/HICSS.2019.557.

14. Dolgui, Alexandre, Dmitry Ivanov, Semyon Potryasaev, Boris Sokolov, Marina Ivanova, and Frank Werner. 2019a. “Blockchain-Oriented Dynamic Modelling of Smart Contract Design and Execution in the Supply Chain.” International Journal of Production Research 57: 1–19.
15. Christidis, Konstantinos, and Michael Devetsikiotis. 2016. “Blockchains and Smart Contracts for the Internet of Things.” IEEE Access 4: 2292–2303.
16. Cong, Lin William, and Zhiguo He. 2019. “Blockchain Disruption and Smart Contracts.” The Review of Financial Studies 32 (5): 1754– 1797.
17. Goni, A.; Burgos, A.; Dranca, L.; Rodriguez, J.; Illarramendi, A.; Bermudez, J. Architecture, cost-model and customization of real-time monitoring systems based on mobile biological sensor data-streams. Comput. Methods Programs Biomed. 2009, 96, 141–157.
18. "Blockchains: The great chain of being sure about things". The Economist. 31 October 2015. Archived from the original on 3 July 2016. Retrieved 18 June 2016. M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.
19. Morris, David Z. (15 May 2016). "Leaderless, BlockchainBased Venture Capital Fund Raises $100 Million, And Counting". Fortune. Archived from the original on 21 May 2016. Retrieved 23 May 2016.K. Elissa, “Title of paper if known,” unpublished.
20. Popper, Nathan (21 May 2016). "A Venture Fund With Plenty of Virtual Capital, but No Capitalist". The New York Times. Archived from the original on 22 May 2016. Retrieved 23 May 2016.
21. Brito, Jerry; Castillo, Andrea (2013). Bitcoin: A Primer for Policymakers (PDF) (Report). Fairfax, VA: Mercatus Center, George Mason University. Archived (PDF) from the original on 21 September 2013. Retrieved 22 October 2013.
22. Trottier, Leo (18 June 2016). "original-bitcoin" (selfpublished code collection). github. Archived from the original on 17 April 2016. Retrieved 18 June 2016.” This is a historical repository of Satoshi Nakamoto's original bit coin sourcecode”
23. Narayanan, Arvind; Bonneau, Joseph; Felten, Edward; Miller, Andrew; Goldfeder, Steven (2016). Bitcoin and cryptocurrency technologies: a comprehensive introduction. Princeton: Princeton University Press. ISBN 978-0-691-17169-2.
24. K. Bhosale, Akbarabbas, Jadhav Deepak, Awani Sankhe (2019). “Blockchain based Secure Data Storage”. International Research Journal of Engineering and Technology. Volume 6, issue 3.

Figures

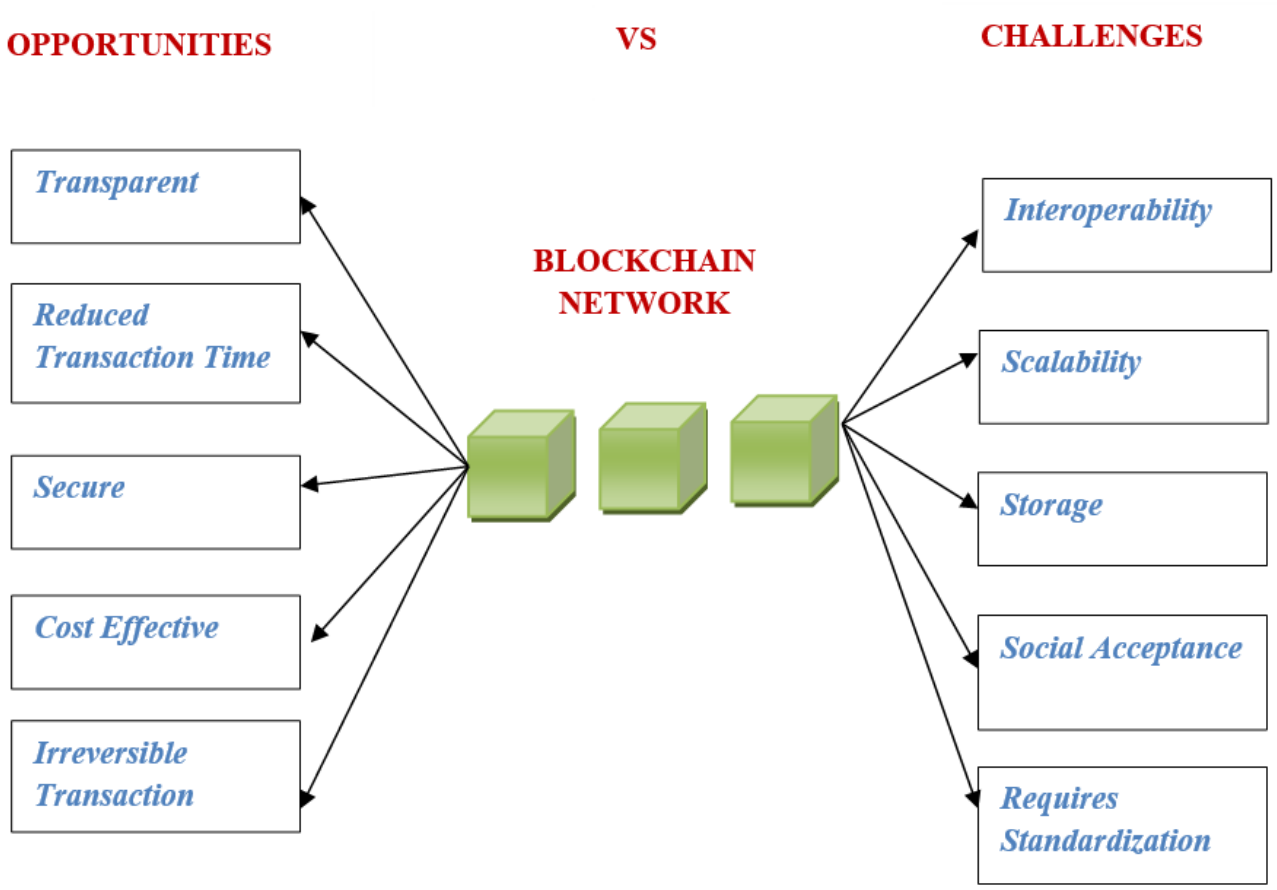


Figure 1

Blockchain Technology in Healthcare: Issues and Prospects

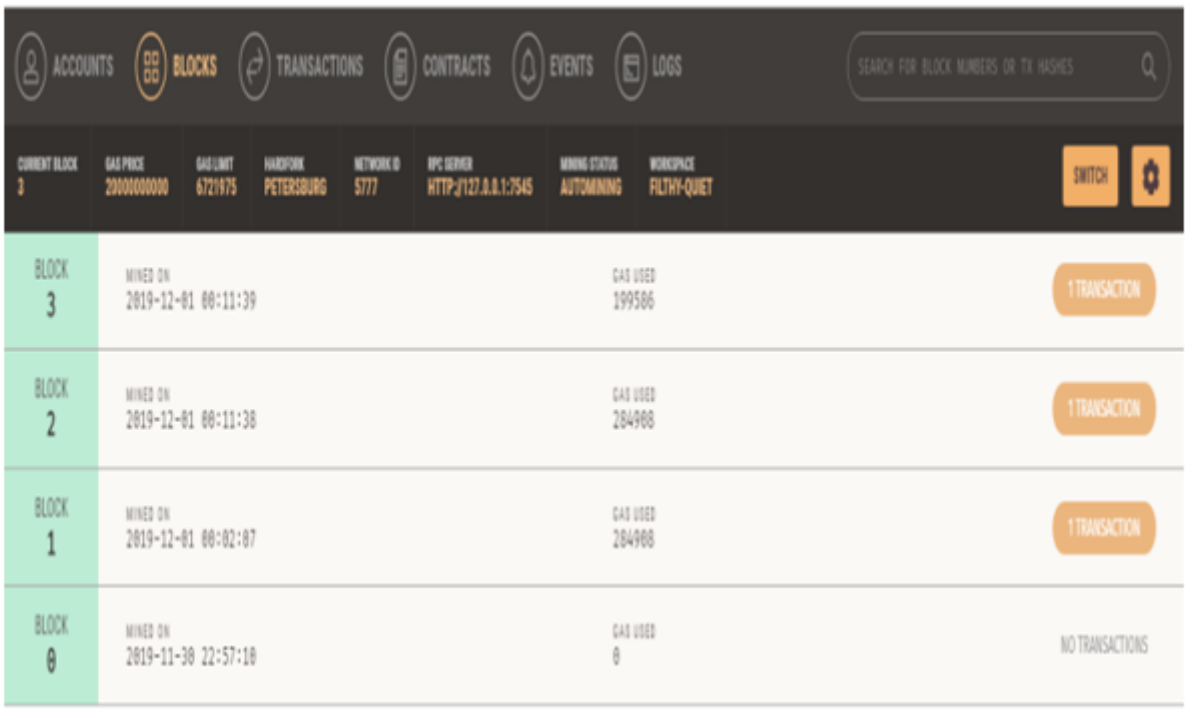


Figure 2

Preview of ethereum exchanges

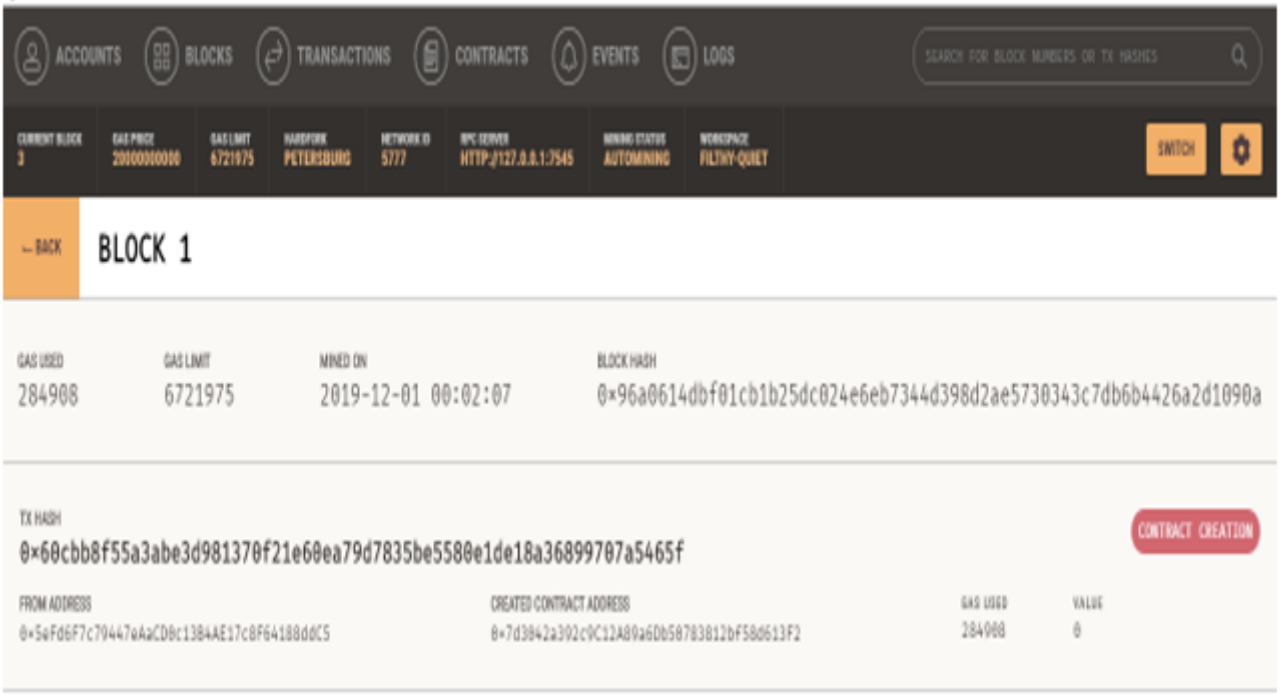


Figure 3

A detailed preview of the ethereum block.

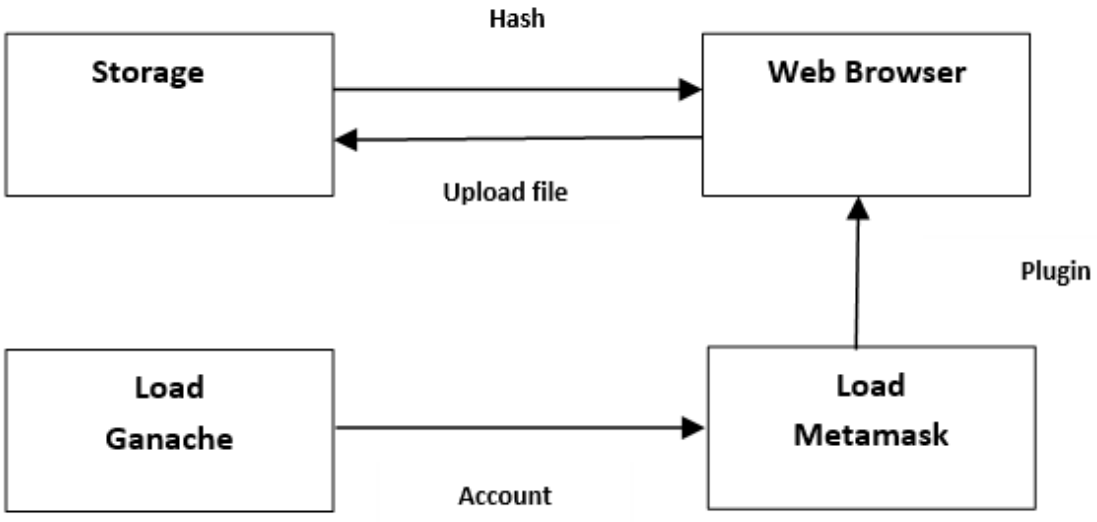


Figure 4

Figure 5. Uploading file

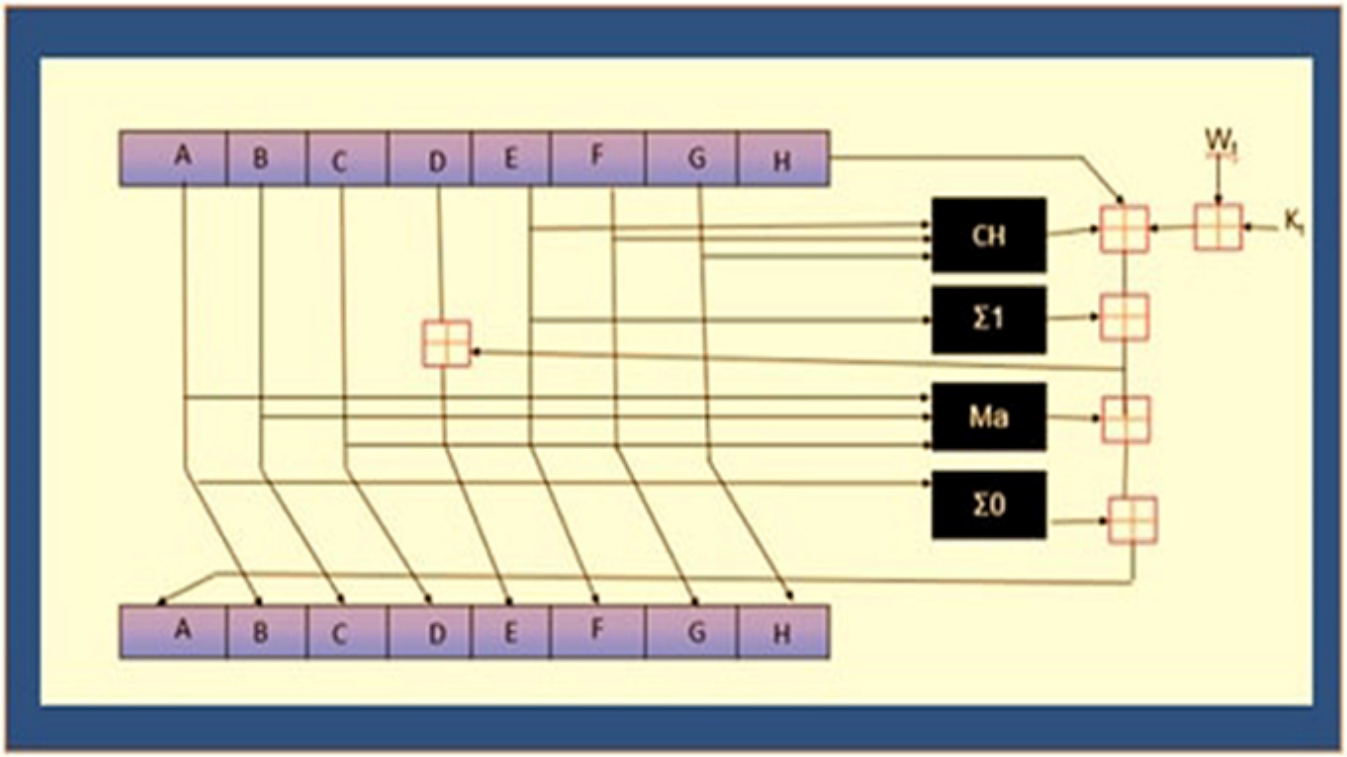


Figure 5

Figure 6. Hash computation