Blockchain

File-store

Name : B Mohith Raagesh

R.No : 21BCE1840

Project Description

We developed a web-based application for decentralized file storing using blockchain. In this application any user can upload as many files(one at a time) as he/she likes. All other peers and the user himself can download and access those file in their system. File can be of any type and any size. Refer to project demo link to see the detailed explanation.
We are using randomly generated nonce in proof of work concept to acheive the required difficulty (diff = 3). Once peer uploads the file, the file is stored in a block including username, filesize and file data. These block gets appended to the current blockchain, which makes it impossible to edit or delete the file/block.
The reason to implement file storing using blockchain is its abilitiy to avoid any modification or deletion. No one can delete or corrupt our files that are stored.

Importance of Blockchain:

Blockchain data structure is used to store digital information securely. Blockchain is an open ledger which can be accessed by multiple parties all at once. Blockchain holds various types of digital information such as transaction information, files, messages, etc. The successful implementation of blockchain holds different types of functionalities such as consensus algorithm, mining block, validation of block etc.

As part of this project, we have implemented a blockchain containing file information, which also allows users to upload/download all types of files from publicly available website. It uses SHA256 cryptographic algorithm to ensure that the block is kept secret. As a consensus algorithm, proof of work is implemented, which requires miners to solve any cryptographic puzzle before they get to announce new block on chain. In our application, we require to find a hash value that starts with three 0's as part of a puzzle.

Proof of Work Algorithm:

Blockchain-based applications are typically peer-to-peer networks that must be as decentralized as possible. To support and enhance network decentralization, all peers in the network should be able to add new blocks to the network. Proof of work algorithms simulate the idea of each peer being able to add a new block. The goal of proof of work algorithms is to solve various cryptographic puzzles. Peers who solve these puzzles more quickly can add a new block to the blockchain.

Two different proof of work algorithms are included in our blockchain implementation. Both are attempting to solve the same cryptographic puzzle, but in different ways. According to these proof of work algorithms, whoever finds a hash value for their block that contains a certain number of leading zeros will be able to announce it to the chain. For example, our blockchain only allows miners to add a block if the hash value of their block begins with three zeros, indicating that the difficulty of the blockchain is 3. The term miner refers to peers who attempt to add blocks to the blockchain.

Based on this idea, we have implemented two algorithms. Both calculate nonce differently. One calculates nonce randomly in each iteration and another one simply increments nonce by one in each iteration. We have analyzed running time and behavior of both algorithms in the file named POW_Comparison.py. Based on the output of the file, we have concluded some results for both the methods.

Comparison of proof of work algorithms:

Difference:

1. Nonce = random.randint(0,99999999) (first algorithm: p_o_w)
2. Nonce += 1 (second algorithm: p_o_w_2)

The running time for first algorithm:

	Attempt #1	**Attempt #2**	**Attempt #3**	**Attempt #4**
Difficulty #2	0.00018	0.00281	0.00102	0.00039
Difficulty #3	0.00069	0.03207	0.00485	0.00356
Difficulty #4	0.13479	0.22688	0.34565	0.19841
Difficulty #5	4.06034	2.08288	0.58391	0.2094

The running time of Second algorithm:

	Attempt #1	**Attempt #2**	**Attempt #3**	**Attempt #4**
Difficulty #2	0.00035	0.00080	0.00062	0.00108

	Attempt #1	Attempt #2	Attempt #3	Attempt #4
Difficulty #3	0.02190	0.02463	0.02104	0.01625
Difficulty #4	0.00366	0.03813	0.32095	0.02145
Difficulty #5	0.04403	3.10820	1.53688	1.50288

Why First Algorithm is better than Second one?

Probability of Valid output & running time

Based on the output of the POW_Comparison.py file, we can conclude that for lower difficulty levels, the running time does not vary significantly. However, for higher difficulty levels, the first algorithm, where the nonce is generated at random, can be faster. One reason for this is that transactions are added concurrently while POW is still running. If a new transaction is added after POW has started running, the entire equation to generate hash will change, and the ultimate nonce value will change compared to the previous value POW was searching for. In 2nd algorithm, previously tested nonce will not be repeated even though there is still a probability that previous nonce can be the solution.

For Example,

Assume that POW has started running.

Nonce has reached to 99978. So, probability for 0-99977 being a solution is 0.

At this point, new transaction is added to the block, which will change entire equation of calculating the hash value. Now, probability of 0-99977 being a solution is not 0. However, those values will not be tested again. This behavior may affect the run time and there can be such case where there is no solution or very less probability for any of the later nonce to be the solution. In that case, miner will fail to solve the puzzle or running time is considerably higher.

On the other hand, in first algorithm, where nonce is chosen randomly, each nonce is equally probable of getting picked at any given time. So, algorithm has higher probability of giving output in less time for larger difficulty level.

Security

Secondly, 2nd algorithm is not quite secure because nonce value can be estimated based on the running time of the algorithm. For example, Assume that Nonce value is between (0,10000). If running time is less, nonce will be small number such as between 0-1000. If running time is higher, then nonce can be close to 10000. Here, we might wonder why the security of nonce is important. If someone has information about one block and they can also figure out nonce for that block, then they can easily break the entire blockchain system because all the blocks are connected to each other with their hash values.

Some Issues with first algorithm

Calculating random values can be expensive. So, we might need to find random functions that have constant running time or quicker compared to other random functions. For example, python random.random() function is faster than random.randint().

Overall, any proof of work algorithm is computationally expensive and requires too many resources. There is an alternative to proof of work algorithm, that is proof of stack algorithms, which are also effective in terms of supporting decentralized network. In proof of stack algorithm, validators are randomly chosen. The probability of being chosen also depends on the value of stacks they hold for that blockchain. The chosen validator acquires a right to add a new block to the chain. Based on the validity of the block, the value of stack that the validator hold will increase or decrease. This method is not expensive yet effective.

Comparison for On-chain and Off-Chain Blockchain:

Blockchain applications can also be divided into two types: On-chain Blockchain & Off-chain Blockchain. On-chain Blockchain refers to storing information inside blocks and Off-chain Blockchain refers to storing actual data outside the block and only keep metadata in block.

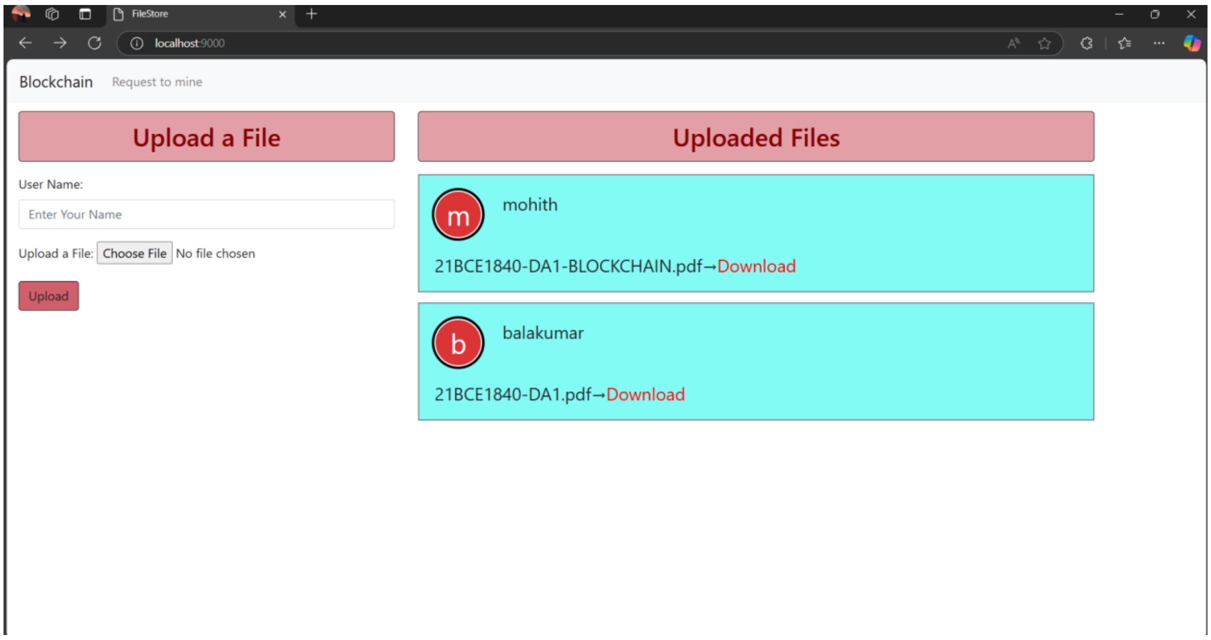
Advantages of On-chain blockchain:

On-chain blockchain can be more secure because information is capsulated in secure blocks. Infomation can be recovered easily in case of break in the system.

Disadvantages of On-chain blockchain:

Runnning time of the insertion and other block-operations can be slow because it holds too much data to process. It is expensive and requires more resources to maintain.

Here, issues with On-chain blockchain can be solved by using off-chain blockchain but On-chain blockchain is more effective where main concern is security and back-up of information. For this project, we have implemented On-chain blockchain which contains entire file data in block including file size and file name.



Upload a File

User Name:

Enter Your Name

Upload a File: Choose File No file chosen

Upload

Uploaded Files

m mohith

21BCE1840-DA1-BLOCKCHAIN.pdf→Download

b balakumar

21BCE1840-DA1.pdf→Download