Blockchain Based File Storage:

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 name 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)
2. Nonce += 1

From, the output of POW\_Comparison.py file, we can say for smaller difficulty level, the running time does not vary too much. However, for larger difficulty level, first algorithm can be faster, where nonce is generated randomly. One reason of that includes transactions being added concurrently when POW is still running. In second algorithm, where nonce is incremented by constant factor 1, if any new transaction is added after POW has started running, the entire equation to generate hash will change and it might require new nonce compared based on the new information in the block. However, the 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. 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 any given time. So, algorithm has higher probability of giving output in smaller amount of time for larger difficulty level.

Security:

The second algorithm is not quite secure because nonce value can be estimated based on the running time of the algorithm.

For example, for range between 0 -10000, if running time is smaller, nonce value is small value 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 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: