Implementation of Blockchain for Secure Bank Transactions

Akhilesh NS
Student, Dept of ISE
BMS College of Engineering
(Affiliated under VTU)
Bangalore, India
nsakhilesh02@gmail.com

Aniruddha MN
Student, Dept of ISE
BMS College of Engineering
(Affiliated under VTU)
Bangalore, India
aniruddha.murthy2@gmail.com

Sowmya K S
Assistant Professor, Dept of ISE
BMS College of Engineering
(Affiliated under VTU)
Bangalore, India
sowmyaks.ise@bmsce.ac.in

***Abstract*—Blockchain is a distributed database or a decentralized ledger which is most commonly used to exchange digital currency and perform transactions securely. Every participant of the network has access to the ledger which will be updated by every new transaction. The Blockchain ledger is a collection of all transactions executed in the past. The Blockchain ledger is a continuously growing tamper-proof data structure containing blocks that hold batches of individual transactions. The completed blocks are added in chronological order. Blockchain via Bitcoin has had a massive impact on the world in the past decade and it's safe to say that this will continue, especially with many people working tirelessly to remove the various limitations that are prohibiting blockchains from becoming mainstream. One such limitation is the high processing and electrical costs that come from the Proof-of-Work consensus protocol. In this paper, we propose an alternative proof-by-approval protocol which is a more advanced form of the proof-of-reputation protocol, that offers better security and is a more decentralized approach than the former at the cost of being less performant and harder to setup**

***Index Terms*—Block chain, Proof-by-Approval, Secure transaction**

I. INTRODUCTION

With nations moving toward digital transactions and the internet being as centralized as it is right now, many of our net-based activity often requires us to place a good deal of trust in various organizations. While this model of operation has served us well for many years, it does come with some flaws, the most obvious of which is that our data is usually at the mercy of the centralized authority to whom we conform with.

Blockchain solved this issue by providing a decentralized peer-to-peer network that allows its users to carry out transactions and interactions without having to place trust in each other or a centralized authority, the users of the network only need to place trust in the globally accepted mechanism of the system, Blockchain stores the users' data in a distributed immutable ledger. The mechanism of operation of the blockchain ensures that any data that is put in the ledger is permanently embedded into the ledger and cannot be changed. This is accomplished by making all the data in the ledger cryptographically linked to each other and by having the ledger distributed across a large peer-to-peer network where each node both validates and passes the ledger to fellow nodes, and where the network is able to quickly identify any illegitimate ledgers sent by an attacker node and quickly diffuse it, this procedure is done using a protocol called the consensus protocol.

There are many different consensus protocols developed for the Blockchain, the one that is most commonly used and accepted is the Proof-of-Work consensus protocol, this is also the protocol that is used by Bitcoin and Ethereum (although Ethereum is moving towards the Proof-of-Stake protocol). The protocol works by requiring anyone who wishes to add data to the ledger to solve a cryptographic puzzle to do so. The cryptographic puzzle, in turn, is designed to be very difficult to solve, meaning that anyone who wants to put a block on the blockchain would have invested in vast amount of resources to do so, thereby repelling any ideas of malicious activity. This process of solving a complex cryptographic puzzle to put data on the blockchain is referred to as mining. Bitcoin was the first to pioneer the PoW protocol and has used it effectively for more than a decade now.

But mining is an incredibly inefficient process which consumes an excessive amount of power. Studies have shown mining can consume as much as 75 TeraWatt Hours of power in a single year. For context, the country of Switzerland consumes about 58 TeraWatt

Proceedings of the International Conference on Mainstreaming Block Chain Implementation (ICOMBI) 2020
978-93-5406-901-7

Hours of power in a year. It is for this specific reason that modern Blockchains struggle to see adoption at a large scale. To tackle this disadvantage, many alternate consensus protocols have been proposed including the proof-by-stake protocol (which Ethereuem is planning on shifting to).

One other such example is the Proof-of-Reputation protocol. In this protocol, the act of adding a block to the Blockchain can only be done by a verified group of users, these users are usually large companies or the like who have a strong reputation to maintain and would not risk losing this reputation by adding a fraudulent block to the blockchain (which the entire world would then be able to see). An example of a PoR blockchain is the GoChain. The PoR protocol works especially well in private or permissioned networks, for example, a company could deploy a PoR blockchain to maintain a record of public information amongst all of its employees without risk of being compromised while authorizing specific users to verify and add blocks to the Blockchain. Though this approach does start to dwindle a bit towards the centralized system that Blockchain is ultimately trying to work against, as any data has to be validated and thus passed to a centralized group of users who may then again be managed by a central authority (Although this isn't always the case) before it can enter the blockchain, the primary rules of the Blockchain still haven't changed, once data has entered the chain, it cannot be changed by either the authority managing the permissioned users or the user who verified and put that block on the chain.

But centralized control over who has write privileges over the chain while useful in a private organization can be limiting in certain other cases, for example, what about a government office, which takes up the task of approving and managing various extremely important documents that concern a large number of citizens. If a citizen wishes to get a certain document from the office and the office wishes to make record of this, a Proof-of-Reputation based Chain could be used to accomplish this task but it also puts all authority in the hands of the employees of the Government office, which is not ideal as citizens should be allowed to have authority over the documents that pertain directly to them (i.e citizens should be able to validate that their document has been approved from their end as well). To accomplish this, this paper presents a variant of the PoR protocol, called the Proof-by-Approval protocol that is designed to handle this task.

II. LITERATURE REVIEW

A. Proof of Work and Bitcoin

The Proof of Work (PoW) consensus protocol is the most popular consensus protocol in the world as it powers Bitcoin, the biggest Blockchain application as of present. PoW was first introduced by Cynthia Dwork and Moni Naor in 1993. Although it did not gain popularity until almost 15 years later when Satoshi Nakamoto implemented it in his/her/their paper about Bitcoin. It wasn't even called "Proof-of-Work" until 1999 where it was used by Markus Jakobsson and Ari Juels in their own publication [3].

In the Bitcoin paper by Satoshi Nakamoto, it is clearly described how to achieve a practical Blockchain. A practical Blockchain, that is to say, a real-world production-ready Blockchain is one which fulfills a set of criteria: [3]

An Immutable Ledger: This can be particularly difficult to achieve in a network since there can be no way of knowing if a group of nodes alter data within the Blockchain. To circumvent this, the Bitcoin paper suggests cryptographically linking the blocks in the chain, by having each block contain a hash made up of all the values in that block and the hash of the previous block. This effectively creates a hash link which any node in the network can easily validate. And changing the value of any block in the chain breaks the chain at that block thus creating an immutable chain [3].

A Peer-to-Peer Network: This is a large scale peer-to-peer network that needs to be established over the internet. In the initial years of Bitcoin, IRC Seeding was used to establish such a network though these days it uses DNS Seeding. DNS Seeding works by simply having a few known clients that a new client can connect to. The known clients then connect the new client to a set of other clients who do the same with the clients they know and this repeats till the client becomes a part of the network [3].

Network Consensus: Perhaps the hardest part of the protocol is ensuring that there is a consensus throughout the network about the Blockchain's current state. In Bitcoin, when a new block is generated, it is sent throughout the network using the Gossip Protocol and each node only accepts a chain that is bigger than the one it already has. In the case of conflicting chains, the network waits for one of the chains to grow larger (typically this happens with the chain that is more spread throughout the network) and chooses

that chain. This ensures that so long as more than 50 percent of the network is genuine, the chain cannot be hacked. Another vital aspect of Bitcoin's consensus approach is mining [3].

Mining is a process by which nodes have to solve a complex computational puzzle in order to generate a block. This puzzle needs to be such that it is hard to solve but easy to validate. And with mining, Bitcoin accomplished three things [3]:

1) It controlled the rate at which new blocks can be generated by altering the difficulty of the puzzle. This means that in the time a new block is generated, the chain would have synchronized throughout the network [3].
2) By making mining public that is, by allowing any node in the network to mine new blocks, it created a competition between nodes to mine blocks before others. Since only one block can be added to the chain at a time [3].
3) The competition makes it harder for a fraudulent user to inject a malicious block into the network since he'd likely be competing with many other genuine nodes trying to mine a block to get the reward associated with mining. And the amount of resources the fraudulent user would need to invest to add a fraudulent block would be so vast that it deters them. (This is on top of the large number of validations that each node performs to the chain) [3]

This is also why the protocol is called Proof-of-work since each new block is essentially proof that a large amount of work was put into making it. The problem however is that the same work amounts to nothing if the node doesn't generate the block first. And this is where the large wastage of resources in Bitcoin comes from. Nevertheless, Bitcoin's design provided a base for many other Blockchain applications (such as Ethereum to rise) and some of its design principles have even been applied to the consensus protocol we are proposing in this paper [3].

B. Proof-of-Stake

The Proof-of-Stake protocol is an alternate consensus protocol to the Proof-of-work protocol created to tackle the issue of the massive power consumption that comes from mining. While the proof-of-stake protocol is not the only alternate to the proof-of-work protocol, it is nevertheless the most popular and perhaps the most effective [7].

In Proof of Stake, there is no mining. Infact, in this protocol, the term used for people who create blocks is minters. And this is because, unlike in proof-of-work where every miner is essentially in a computational battle with every other miner to add blocks to the chain, minters are people chosen by the network to add blocks to the chain. The basic principle behind this is, when a new block needs to be mined, a group of users volunteer to mine the block, amongst these volunteers, the network then chooses a few of them, based on certain criteria to then create a block, and finally rewards them for doing so. This means that at any given moment, each block is created using just a single user, and not the first user amongst a large amount of other users who are trying to mine the block and whose computational expenditure is effectively wasted [7].

The key aspect of the protocol is the criteria the network uses to choose a volunteer or volunteers. In proof-of-stake, every volunteer is selected based on three factors [7]:

1) A certain predefined amount of randomness [7]
2) A monetary stake put up by that specific node [7]
3) And the amount of time the node has put up that monetary stake [7]

It is good to think of PoS as a sort of auction where every node can increase its chances of being selected as a volunteer by staking as many coins of the network as possible. (Though coin age and randomness also filters in, ensuring that nodes which are rich with a large amount of coins don't dominate the network). Similar to the PoW protocol, the PoS protocol can [7]:

1) Control the rate at which new blocks are added to the network since it chooses who mints the block [7]
2) Creates a competitive environment that makes it extremely difficult for a fraudulent user to inject something malicious. (In this case, the fraudulent user would be competing with people who have higher stakes and better coin ages) [7]

All of these act as a deterrent to any fraudulent users. However the PoS protocol does come with certain issues, the first is that its very difficult to establish a PoS protocol from the very beginning, since coins of the network are generally used as stake, acquiring these coins at the early stages of a Blockchain app when they have little value is fairly simple. It becomes much harder when you have to deal with well-established chains however. Buying a majority of the coins in Bitcoin is not just expensive but outright difficult due in part to the number of sales you would need to make [7].

Another problem is that unlike PoW, a monetary stake is needed to participate in the protocol. Now keep in mind, that the stake only exists as a measurement criteria and is returned to the user as is, but stakes are made of coins whose value can change. So that means that one would need to buy into the network before being able to participate. Mining on the other hand has no barrier to entry and is so simple to do that one can do it on a phone (not to great effect but it is possible) [7]

Regardless, the PoS protocol does provide a reasonably effective alternative to the PoW protocol especially if you are a well established chain (which is why Ethereum is shifting towards it) [7]

C. Proof-of-Reputation

The Proof-of-Reputation consensus protocol (PoR) is the protocol on which this paper derives from. The PoR itself however is derived from the Proof-of-Authority protocol (PoA). In PoA, unlike PoW, users have different levels of authorities and not just any user can validate a block. Instead, specific users known as validators are allowed to validate blocks. This makes sense in a private network inside an organization where the organization manages who has access to the network and what kind of authority they have [12].

The PoR protocol builds on the PoA protocol by having the validators be companies instead of individuals. The motive behind this being that companies unlike individuals have less reason to do something malicious since they risk losing the trust and brand value that they have worked for. Thus one could say that a company is putting its reputation at stake every time it validates a block, this is what acts as the security of the network [12].

As such, in order for the protocol to be as secure as possible, the validators must be organizations with a great deal of reputation to put at stake. Large organizations such as Google, Microsoft etc become ideal candidates to be validators in such a protocol. Once a list of validators is established within the protocol, this list is maintained within the Blockchain [12].

We will go into more of the differences between this protocol and the Proof-by-Approval protocol being presented in this paper in the upcoming sections [12].

D. Hyperledger Fabric

Hyperledger Fabric is a special technology that was originally created by IBM, Digital Asset and Blockstream to create and managed private permissioned Blockchains. The technology is a part of a cluster of projects known as Hyperledger which is managed by the Linux Foundation. Today Hyperledger Fabric is an open-source distributed ledger technology which is seeing a great deal of adoption and traction compared to other similar projects. And it has a thriving developer community constantly striving to make the technology better and simpler to use. The current latest version of Hyperledger Fabric at the time of writing of this report is v1.4.2 [2].

Hyperledger Fabric uses an approach similar to something like Ethereum while still possessing a large number of vital differences. Like Ethereum, Fabric uses Smart Contracts (In Fabric, they are referred to as Chaincode) to provide a highly flexible way to add Blocks to the Blockchain. But the similarities end there and moving forward we present a number of key differences Fabric possesses compared to the former which played a vital role in why we chose this framework for the implementation of our protocol and application [2].

Fabric's smart contracts are special compared to other Distributed Ledger Technologies (DLTs) primarily because they can be written using general-purpose programming languages such as JavaScript, Golang, Java or Python (Currently these are the only four languages officially supported but more languages are being added on a regular basis). This was especially important to us since we came from a largely web development background and we had the luxury of being able to use JavaScript (A language which we were already familiar with) to work with Fabric [2].

Fabric is also primarily designed for private Blockchains. This stands in stark contrast to something like Ethereum where the Blockchain is public and anyone anywhere can participate in interacting the chain. In Fabric, the Blockchain can be setup to only accept interactions with authorized personnel and it is even possible to assign roles to individuals such that any interaction with the Blockchain can be controlled and regulated. This means that Fabric can be used by organizations such as banks, schools, colleges, government offices etc [2].

Another extremely important characteristic of Hyperledger Fabric is that it is customizable. Since

Fabric was built to accommodate for a myriad of different industry use cases, it is extremely modular. This means Fabric can be used with any consensus protocol whether it be proof-of-work or practical byzantine fault tolerance (both of which will be discussed later on in the literature survey) or in this case, the custom protocol we have made, proof-by-approval. And since Fabric is primarily private and all the users of the system are known and authenticated, there is no need for a cryptocurrency as is the case with Ethereum or Bitcoin [2].

Having a custom protocol means mining becomes optional which in turn allows for Hyperledger Fabric to be highly performant and efficient. The conditional requirement for mining or cryptocurrency also helps reduce security risks and eliminates potential attack points in the system. This also decreases the overall cost required to deploy the system bringing it more in line with normal distributed systems [2].

III. MECHANISM

Let us start by assuming all the elements of a basic blockchain setup are present:

- A Cryptographically linked Immutable Ledger (Similar to the one used by Bitcoin, see Literature Review section for details)
- A Peer-to-Peer Network (Again similar to the one used by Bitcoin which uses DNS Seeding)

There are two types of user entities in this protocol:

- Users: Read and Write-after-approval to the Blockchain
- Approvers: Reads and Approves writes to the blockchain. But cannot directly write to the Blockchain

Similar to the PoR protocol, becoming a user is relatively simple but becoming an approver follows a much more rigid process. An approver must be someone whose public information can be verified and traced back to the approver such that they should suffer consequences should they approve something fraudulent.

Only users are allowed to write to the blockchain for this particular protocol but they can only do so after having their data validated and approved by an approver.

P.S: Validation and Approval are two different things in the context of this protocol - validation is the process of ensuring that a block has all the properties and follows all the guidelines necessary to be a valid block whereas approval is the process of checking to make sure the information being stored inside of the block is not fraudulent.

A short overview of the working process can be described in three steps (See below for a more detailed explanation):

1) A User creates a block he/she wishes to add to the blockchain, signs it and sends it to an Approver via the P2P Network (The process of sending it can be done by randomly choosing an approver or in this case by having the user specifically choose an approver)
2) The Approver receives the block from the user, validates the block by checking its hash etc, checking the signature to ensure that the user is also a valid user and finally approves the information inside of the block after which the approver then signs the block (which acts as an approval) and sends it back to the user
3) The User receives the block from the approver, checks to make sure that the block has not been altered, checks the signature of the approver so as to ensure the approver is genuine and finally pushes the block to the blockchain

In this case, the blocks are passed between the user and approver through the Peer-to-Peer network, hashing is can be done using any standardized hashing algorithm (though we recommend SHA256 just because it is the most commonly used today but if you wish for a stronger hashing algorithm such as SHA512, it can be used as well, though keep in mind that it will be slower) and signing can done using something like RSA or ECDSA.

A more detailed explanation of the process is as follows:-

1) A User creates a block with the data that that the user wants to put in it and creates a 'user' property which in turn contains properties 'timestamp' which is used to indicate when the block was first created by the user, 'no' which is used to indicate the nth block created by this specific user in the blockchain (so 1 would indicate that this is the first block created by this user that is to be put into the blockchain) and 'name' which indicates the username of the User which can be used to trace back to the

user's account where the user's public key can be retrieved (Needless to say only the public key can be derived, all other user specific information is private and secure).

```
{
  "user": {
    "timestamp": "(time at which the
        block was created in ms)",
    "no": "(id of the block w.r.t the
        user's blocks on the blockchain)
        ", "name": "(username of the
        user)"
  },
  "data": "(SOME DATA)"
}
```

Note. The data property is flexible enough to follow strict guidelines if required. For example. you could have a data property which has properties 'from', 'to' and 'amount' if you were building a transaction based application like bitcoin or opt to store some other form of data, the choice is yours, since a format can be validated by an approver.

The User then creates a signature by first hashing the 'user.timestamp', 'user.name', 'user.no' and 'data' properties of the block and encrypting this hash using a private key (which the user must safeguard at all costs) while making the public key available globally so that anyone can use it to decrypt the signature and validate that it is in fact the User's signature. The User then adds a property to the 'user' property called 'signature' which contains the result of the encryption.

```
{
 "user": {
     "timestamp": "(time at which the
         block was created in ms)",
     "no": "(id of the block w.r.t the
         user's blocks on the blockchain
         )",
     "name": "(username of the user)",
     "signature": "(encrypted hash of [
         user.timestamp, user.no, user.
         name, data] using
         USER_PRIVATE_KEY)"
 },
 "data": "(SOME DATA)"
}
```

And this block is then finally sent to the Approver for approval

2) The approver upon receiving the block from the user performs a number of steps to validate the block

- Check if the 'user.no' is one greater than the 'user.no' of the last block the user put in the blockchain
- Check if 'user.timestamp' is valid, and if 'user.timestamp' is between the timestamp of the last block the user put in the blockchain if any and the present time.
- Check to see if the name of the user leads back to the account of a valid user where the user's public key can be derived
- Check to see if the public key can decrypt the signature thereby confirming that user's signature
- Check to see if the decrypted signature matches the hash of the 'user.timestamp', 'user.no', 'user.name' and 'data' properties thereby confirming the user has signed the data that the user has sent

After all these steps (and these are only validation steps), the approver then approves the information inside of the 'data' property of the block by checking if the information is not fraudulent and is in fact genuine (In a transaction app, this could mean that the user must have sufficient balance etc)

After approving the block, the approver can then attach an approval to the block by adding a property called 'approver' which in turn has properties 'timestamp' indicating the time at which the block was approved and 'name' which indicates the username of the Approver which can be used to trace back to the approver's account where the approver's public key can be retrieved.

```
{
 "user": {
     "timestamp": "(time at which the
         block was created in ms)",
     "no": "(id of the block w.r.t the
         user's blocks on the blockchain
         )",
     "name": "(username of the user)",
     "signature": "(encrypted hash of [
```

```
            user.timestamp, user.no, user.
            name, data] using
            USER_PRIVATE_KEY)"
    },
    "approver": {
        "timestamp": "(time at which the
            block was approved in ms)",
        "name": "(username of the approver
            )"
    },
    "data": "(SOME DATA)",
}
```

Then the approver creates a signature by hashing the 'approver.timestamp', 'approver.name', 'user' and 'data' properties of the block and encrypting this hash using a private key (which the user must safeguard at all costs) while making the public key available to anyone so that anyone can use it to decrypt the signature and validate that it is in fact the Approver's signature. The Approver then adds a property to the 'approver' property called 'signature' which contains the result of the encryption.

```
{
    "user": {
        "timestamp": "(time at which the
            block was created in ms)",
        "no": "(id of the block w.r.t the
            user's blocks on the blockchain
            )",
        "name": "(username of the user)",
        "signature": "(encrypted hash of [
            user.timestamp, user.no, user.
            name, data] using
            USER_PRIVATE_KEY)"
    },
    "approver": {
        "timestamp": "(time at which the
            block was approved in ms)",
        "name": "(username of the approver
            )",
        "signature": "(encrypted hash of [
            user, approver.timestamp,
            approver.name, data] using
            APPROVER_PRIVATE_KEY)"
    },
    "data": "(SOME DATA)",
}
```

This block is then finally sent back to the user by the approver

3) The user upon receiving the block from the approver performs a number of steps to validate the block

- Checks to see if the user's data has not been altered by comparing the hash of 'user.timestamp', 'user.no', 'user.name' and 'data' to the value obtained by decrypting 'user.signature' using the User's public key
- Checks to see if 'approver.timestamp' is between 'user.timestamp' and the present time
- Checks to see if 'approver.name' leads back to a valid Approver account from which a public key can be derived
- Checks to see if the public key can decrypt 'approver.signature' using the public key thereby confirming that it is in fact that approver's signature
- Checks to see if the hash of 'user', 'approver.timestamp', 'approver.name' and 'data' matches the value obtained by decrypting 'approver.signature' using the Approver's public key

After validating the block and its approval, the user than adds the properties 'previous_hash' which contains the hash of the previous block, 'id' which contains the hash of 'user', 'approver' and 'data' and acts like a unique id for the block and 'current_hash' which contains the hash of 'user', 'approver', 'data', 'id' and 'previous_hash'.

```
{
    "user": {
        "timestamp": "(time at which the
            block was created in ms)",
        "no": "(id of the block w.r.t the
            user's blocks on the blockchain
            )",
        "name": "(username of the user)",
        "signature": "(encrypted hash of [
            user.timestamp, user.no, user.
            name, data] using
            USER_PRIVATE_KEY)"
    },
    "approver": {
        "timestamp": "(time at which the
            block was approved in ms)",
        "name": "(username of the approver
            )",
```

```
        "signature": "(encrypted hash of [
            user, approver.timestamp,
            approver.name, data] using
            APPROVER_PRIVATE_KEY)"
    },
    "data": "(SOME DATA)",
    "id": "(hash of [user, approver, data]
        )",
    "previous_hash": "(hash of the
        previous block)",
    "current_hash": "(hash of [user,
        approver, data, id, previous_hash])
        "
    }
```

The user then finally adds this to the blockchain and synchronizes the chain

Synchronizing the chain can be done in a fashion similar to Bitcoin, by using the Gossip Protocol to spread the chain and have nodes accept chains larger than the ones they hold. This process does require that the rate at which new blocks are added should be fixed and controlled, in Bitcoin, this is achieved by controlling the difficulty of the cryptographic puzzle and in PoS, this is done by the network itself when it chooses new minters. This protocol can implement a fixed block rate by having approvers maintain a fixed timegap between each approval.

A. Block Validation by Nodes

Each node in the network upon receiving a blockchain with a new block will validate the new block with the following steps

- Check if the 'user.timestamp' and 'approver.timestamp' are valid timestamps, if 'approver.timestamp' is after 'user.timestamp' and if the 'user.timestamp' is after the 'user.timestamp' of the last block the user put in the chain
- Check if the 'user.no' is one greater than the 'user.no' of the last block the user put in the chain
- Check if 'user.name' is a valid username that can lead back to a User's account where the user's public key can derived
- Check if the public key can decrypt the 'user.signature' thereby confirming that it is in fact the user's signature
- Check if the hash of 'user.timestamp', 'user.no', 'user.name' and 'data' is equal to the value obtained by decrypting 'user.signature' using the user's public key
- Check if 'approver.name' is a valid username that can lead back to an Approver's account where the approver's public key can be derived
- Check if the public key can decrypt the 'approver.signature' thereby confirming that it is in fact the approver's signature
- Check if the hash of 'user', 'approver.timestamp', 'approver.name' and 'data' is equal to the value obtained by decrypting 'approver.signature' using the approver's public key
- Check if the hash of 'user','approver' and 'data' is equal to 'id'
- Check if the 'previous_hash' matches the 'current_hash' of the previous block
- Check if the hash of the 'user', 'approver', 'data', 'id' and 'previous_hash' is equal to the 'current_hash'

Note. It is also possible to include a data approval step for approving the 'data' property of the block on the node's end as well if necessary to enforce even greater security

IV. IMPLEMENTATION

This paper seeks to present an application similar to the Google Pay app (a Bank-Bank payments application by Google) but one which uses Blockchain. The main objective of the paper is to create an application where one could use a decentralized network to make transactions between centralized systems. So, the intention isn't to start the next race towards decentralization but rather to build a practical blockchain application that could be implemented in the present time.

In the application, two characters are considered, a user (the one who would use the app to make payments or transfer money to other people directly via their bank accounts) and a bank (who would approve of such transactions). The working of the app would go as:

Whenever a user wants to make a transaction, he/she would create a block with details of the transaction as described in the working of the protocol above, (Only the user's username would be visible to everyone who has a copy of the blockchain. The user would have his/her personal details separately shared to his/her bank via KYC for example) and send it to the bank where his/her account is present (or optionally it could also be setup in such a way that banks communicate limited information to each other via encryption for security in order to make it

possible for transactions to be made via any approver).

The bank would receive details of the transaction from the block. The bank then after validating, approves of the transaction (but doesn't go through with it yet) if it is not fraudulent and sends the block back to the user.

The user would then validate the block and add it to the blockchain where the block's data essentially acts like a receipt on a permanent unhackable ledger compelling the concerned banks to process and complete the transaction.

In a PoR version of the same, the banks would also put the block into the blockchain which for this context would probably work just as fine but our intention was to try and give users a bit more control over their transactions and this would also force banks to be a bit more more careful with the transactions they will be handling.

V. COMPARISON WITH POR

This protocol is very similar to the PoR protocol. Which might make one wonder how exactly does this compare to the PoR protocol?

The biggest difference that this protocol has to the PoR protocol is that it adds another layer of validation by having users validate approved blocks. In the PoR protocol, the validator (PoR equivalent of approver) can directly add blocks to the blockchain while putting up their reputation as stake. This protocol strives to try and take as much power away from approvers as possible and in the process try and decentralize as much as can be done. By letting the users validate blocks additionally, there is more pressure on approvers to not attempt anything malicious as compared to PoR. But there are situations where the PoR protocol makes more sense, this protocol does not seek to replace or stand as a better alternative to the PoR protocol but simply provides an additional option to anyone who might find it useful.

VI. EXPERIMENT RESULTS

A basic implementation of the Proof-by-Approval[1] is written in JavaScript and run on the NodeJS runtime.[2][3]

Additional libraries that are used in the program are:[4] -

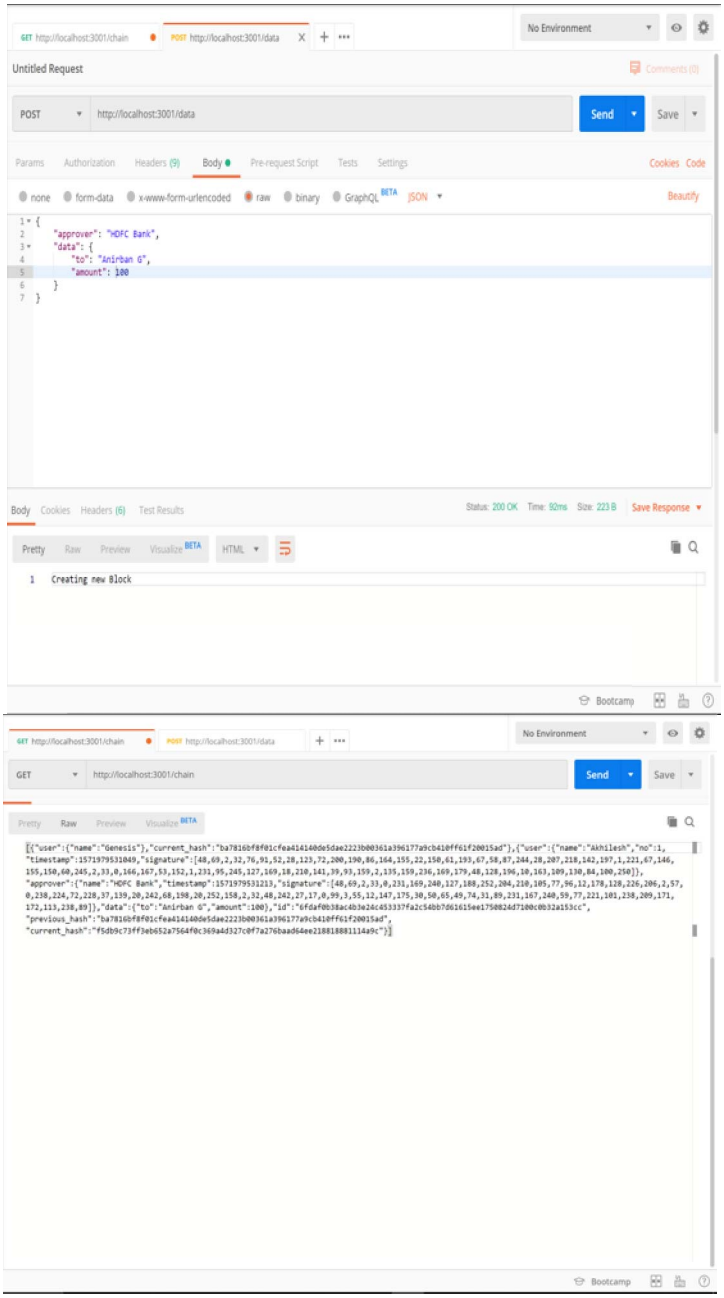
- ws: A library that allows for the creation and use of websockets in the program. Websockets will act as the primary way in which peer-to-peer networking is conducted in the program
- crypto-js: A library that provides various cryptographic functions. The program makes use of the SHA-256 hashing function provided by this library
- elliptic: A library that provides various encryption based functionality based on elliptic cryptography. The program uses the ECDSA encryption algorithm provided by the library. Note. This implementation of the algorithm uses the secp256k1 curve which is relatively safe but it is highly recommended to not use ECDSA at all in real-world applications (even though Bitcoin uses it) and to instead use EdDSA wherever possible. The library does also provide implementation for the same as well.
- express: A library to create http-servers or REST Apis. The program uses this to create a REST Api through which users can interact with the program

The program upon execution sets up 16 REST Apis and 16 websocket servers which all connect to each other and form a Peer-to-Peer network. Of these 16 servers, 13 belong to regular users whose information can be found.

To interact with Blockchain, one can simply use a REST Api client (such as curl or postman) to then send the following requests: -

- GET Req at localhost:3001/chain: Returns the current blockchain at localhost:3001
- POST Req at localhost:3001/data: with the req body, adds a new block to the blockchain

```
{
  "approver": "Kotak Mahindra Bank",
  "data": "ANY DATA YOU WANT"
}
```



VII. CONCLUSION

Blockchain has proven to be an extremely groundbreaking technology but its implementation mainstream has been hindered due to various limitations. In this paper, we seek to present a protocol that would address one of the major limitations of Blockchain, the high resource utilization, cost and maintenance of the predominant consensus protocol being used. Our protocol looks to allow for Blockchain's continued integration into everyday life with realistic results. In this paper, we show how Blockchain can be used alongside traditional banking systems to enhance their security and improve the transparency of their transactions

ACKNOWLEDGMENT

We would like to express our deepest appreciation to all those who provided us the support to complete this paper. A special gratitude to our guide, Dr.Sowmya K S, whose contribution in stimulating suggestions and encouragement helped us through every step of writing this paper.

We would also like to thank our respected Principal, Dr. B. V. Ravi Shankar, for giving us this opportunity to work on this paper

References

[1] S.Nakomoto, "Bitcoin: A Peer-to-Peer Electronic Cash System". Cryptography Mailing list at https://metzdowd.com. Oct 2008.

[2] Androulaki, Elli, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart et al. "Hyperledger fabric: a distributed operating system for permissioned blockchains." In Proceedings of the thirteenth EuroSys conference, pp. 1-15. 2018.

[3] P. Hooda, "Proof of Work (PoW) Consensus - GeeksforGeeks", GeeksforGeeks, 2019. [Online]. Available: https://www.geeksforgeeks.org/proof-of-work-pow-consensus/.

[4] P. Hooda, "practical Byzantine Fault Tolerance,", GeeksforGeeks, 2019. [Online]. Available: https://www.geeksforgeeks.org/practical-byzantine-fault-tolerancepbft/.

[5] P. Javeri, "Smart Contracts and Blockchains", Medium, 2019. [Online]. Available: https://medium.com/@prashunjaveri/smart-contracts-and-blockchains-c24538418bf6.

[6] W. Kenton, "Proof of Burn (Cryptocurrency) Definition", Investopedia, 2019. [Online]. Available: https://www.investopedia.com/terms/p/proof-burn-cryptocurrency.asp.

[7] P. Hooda, "Proof of Stake (PoS) in Blockchain,", GeeksforGeeks, 2019. [Online]. Available: https://www.geeksforgeeks.org/proof-of-stake-pos-in-blockchain/.

[8] T. Marler, "Hyperledger Fabric: Transaction," Medium, 2018. [Online]. Available: Hyperledger Fabric Medium

[9] P. B, "Architecting a Hyperledger Solution - Things to keep in mind," Hackernoon, 2018. [Online]. Available: Hyperledger Fabric Hackernoon

[10] J. H, "Proposed System - Google Docs 1,", Scribd, 2019 [Online]. Available: Proposed System

[11] Moindrot, Olivier, and Charles Bournhonesque. "Proof of Stake Made Simple with Casper." ICME, Stanford University (2017).

[12] Gai, Fangyu Wang, Baosheng Deng, Wenping Peng, Wei. (2018). Proof of Reputation: A Reputation-Based Consensus Protocol for Peer-to-Peer Network. 10.1007/978-3-319-91458-9_41.

[13] Buterin, Vitalik. "What is Ethereum?." Ethereum Official webpage. Available: http://www.ethdocs.org/en/latest/introduction/what-is-ethereum.html (2016).