Blockchain research report

Robbe Bryssinck (team Abracadabra)

Contents

Introduction

For our Q&A application, “Abracadabra”, we decided that trust is an essential part of our service. To achieve this, we thought about using a blockchain to decentralize the user data, making it near impossible to mutate the data. It also means that the data is stored securely, since there is no single point of failure. In this research document, we will explore what a blockchain is, analyze every type of blockchain to see which one would be the best fit for our application, what an implementation of the blockchain in the context of a Q&A site would look like, and finally, we will conclude whether it is feasible or not to implement. The main method for conducting this study will be a literature study.

What is a blockchain?[1]

To start out, I will give a high-level overview of what the blockchain is. The blockchain is a type of technology where every piece of information is stored in a linked list. The list is linked from one "block" (containing (meta)data) to the previous one in the chain. A block in the blockchain contains the following elements:

- The actual data to be stored on the blockchain.
- An index number specifying the index of the block within the blockchain.
- A hash built from the data, index, and the hash of the previous block.
- The hash from the previous block.

If someone were to change any data within a block in the blockchain, the new hash of that block would be different, and the next block would have an invalid hash of the previous block. Therefore, the entire blockchain would be invalidated.

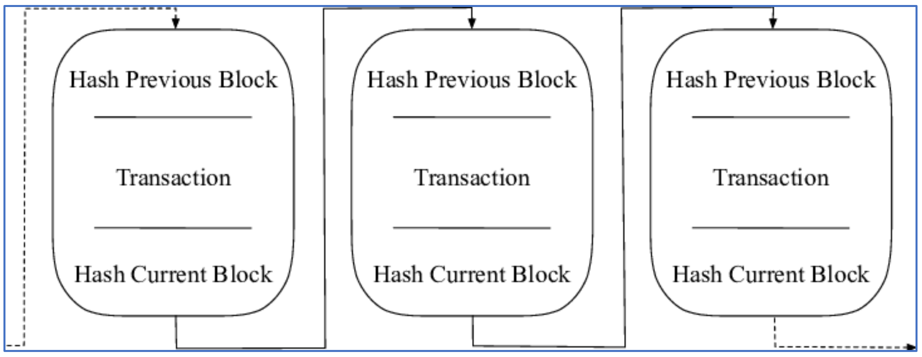

Figure 1: blockchain structure visualization

With the current system, it would be trivial to change the data within a block and recalculate all the hashes. In a real-world implementation of a blockchain, there are two main ways to prevent this:

1. The entire blockchain is distributed. This way, everyone who participates has a copy of the blockchain, and it would already be much harder to change the contents of a block.
2. Distribution of information is not enough. There needs to be a consensus mechanism to determine what the right version of the blockchain is. This is commonly achieved through either "proof of work" or "proof of stake", depending on the type of blockchain that is being used.

In short, proof of work delays the number of blocks possibly generated through computational puzzles, and proof of stake randomly chooses someone in the network to determine the next block favoring the nodes with the highest stakes. Proof of work is often used in a public, permissionless blockchain. Proof of stake is more often used in a permissioned blockchain.

What is a public, permissionless blockchain?

A public, permissionless blockchain is writeable to everyone with an internet connection. Any user can join the peer to peer network and their machine would become a node. The user receives blockchains from other people in the network. The main question here is: "which one is the correct blockchain?". There are a few systems in place to determine that. In the following section, this will be explained by the hand of an example of a public, permissionless blockchain using proof of work as its consensus algorithm.

The user joins the network and sends out a new entry to the ledger of information (aka the blockchain). This information still needs to be put in a block and that block needs to be added to everyone's blockchain. This is where proof of work comes in. Special nodes called "miners" will receive the information transmitted by the user. The first thing these miners will do is check whether the data is "valid". Validity of data in a blockchain is determined beforehand by setting certain rules (business logic). For example, cryptocurrencies use asymmetric encryption to determine ownership of coins. The miners check whether the signature signed with the private key matches with the public key.

Next, the miners will start working on creating a block by solving a puzzle. Often, the implementation of this puzzle is simply calculating a (SHA256) hash of the block where the first X number of characters are '0'. The higher the variable X, the more difficult the puzzle becomes. To make the SHA256 hash change every time, there is something included in the block called a "nonce" value, which is changed each iteration of the algorithm until the puzzle is solved.

When a miner successfully mines a block, it will transmit their version of the blockchain to the entire network. Every node in the network will perform two checks: a check to determine whether the data in the block is in accordance with the business logic of the application attached to the blockchain, and a check to determine whether the hash of the new block is correct. If these checks pass, the nodes will add the new block to their version of the blockchain.

It is possible that a malicious node/miner sends out a tampered blockchain with recalculated hashes so that their blockchain looks valid. The nodes receive this block, and at the same time receive legitimate blocks from miners. In this case, the blockchain with the longest string of blocks wins. It is very hard for a malicious miner to remain the longest since that miner must work against an entire network of miners to keep the blockchain the longest.

That is the essence of proof of work; the blockchain with the most work put into it wins, and the combined computational power of an entire network will be larger than a few malicious nodes. The only way for a bad blockchain to be distributed across the network is for a malicious actor to control most of the network (aka a "51% attack"). In a large, distributed network, this is extremely expensive.

This type of blockchain provides an unmatched level of immutability, decentralization and foregoing of a central authorization/a single point of failure. Regardless, there are a few downsides which make its uses niche:

- The blockchain is only effective when it is distributed among a significant number of nodes.
- There needs to be an incentive for miners to do all that computational work (in cryptocurrency for example, miners get coins for mining a block).

What is a permissioned blockchain?[2]

A permissioned blockchain restricts the write access to the blockchain, so that only a select number of users can write to the blockchain. This already defeats a lot of the decentralization, since a central authority must decide on who can join the network and who cannot. Regardless, other properties remain intact, namely the immutability and the fact that it is still distributed, albeit among a smaller, select network. One could also still make the blockchain public, so that people can still register as nodes and receive new blocks. They just cannot write to the blockchain.

The second thing that makes a permissioned blockchain different from a permissionless one is the fact that a consensus algorithm is often not implemented through proof of work, since permissioned blockchains do not deal with nearly as much data as permissionless blockchains. Sometimes, proof of stake is used instead. Proof of stake is a lot less computationally heavy, which is ideal for a smaller network. This also allows for the blockchain to be much faster, in terms of write speeds. As always, the data within the selected node still gets checked against a set of parameters through ordinary business logic, and the hashes can be checked to see whether the blockchain is still intact.

What would a blockchain look like for our application?[3]

There are a few issues with implementing a blockchain for our application. The first issue is the nature of our data. Take Bitcoin for example. The data they add to the ledger are always the same: a signature, the amounts of coins sent, a receiver etc. In other words, the data is not arbitrary. In our case, the data is in fact arbitrary: the questions and the answers are input from users. This presents a problem. Everything posted to the blockchain is stored permanently, and there is no way to delete or moderate illegal content. One solution to that problem would be to store the user input in a traditional database, and instead generate a hash of said input, and store that hash on the database. This way, when a user fetches an answer from the site, the user can generate a hash based on the content and check it against the blockchain. The moderators of the site can moderate the traditional database while maintaining trust.

The second major issue is the incentive for mining. Since our product is not a cryptocurrency, coming up with a good incentive for people to give up their computational power for block mining is a lot harder. Proof of stake is not a realistic option here either since it cannot be implemented in a decentralized way. In any case, a public, permissioned blockchain would be the best option here, where the selected experts are the only ones who's answers' hashes are submitted to the blockchain, since those answers are the most important ones to keep immutable. The experts' machines will serve as nodes determining the consensus on what the actual blockchain is. This can be achieved through the "practical Byzantine fault tolerance" algorithm. Since the blockchain is still public (which in this case means readable for everyone), the answers and the blocks are still verifiable by everyone.

The third problem that comes with any implementation of a blockchain where identity of users is important, is authentication/authorization. The only way to do that in a decentralized way is with asymmetric encryption; have every expert generate their own public/private key pair, and, with each answer post, the experts post their public key along with a signature crafted with their private key. This would only work if public/private keys are generated and stored separately from the site, and that each signature is crafted independently from our application, since there can be no risks of the expert sharing their private key with the application's owners.

Conclusion

With a public, permissioned blockchain developed specifically for our application, it is possible to decentralize the data, make it immutable, and gain the trust of the user, given that the application accounts for the following issues:

1. The nature of the data (arbitrary data).
2. The lack of incentive.
3. Custom authentication/authorization.

Recommendations

I do not recommend the use of a blockchain in this situation. While it is technically feasible to create a custom implementation of the blockchain that would still be able to get the trust between users and owners of the application, it would have to be implemented in such a way that the user experience would suffer from it tremendously. The application would also be needlessly complicated and hard to maintain. Given that this implementation would not even be 100% distributed and decentralized, which is one of the main features of the blockchain, I recommend going for a different approach.

References

[1]: Bitcoin paper: https://bitcoin.org/bitcoin.pdf
[2]: Permissioned blockchain: https://www.investopedia.com/terms/p/permissioned-blockchains.asp
[3]: “Do you need a blockchain?”: https://eprint.iacr.org/2017/375.pdf