Lambda

Blockchain I - Structure

Blockchain is a data structure that's useful for sharing a ledger of transactions that is universally agreed upon, but without the need for a central authority. It forms the basis for cryptocurrency, and is useful in a variety of different fields, including voting, contracts, and many others.

The core principle of blockchain is to shift from the systems we use today that are based on trust, to one that is instead based on proof.

Trust means that we accept the validity of information based on our trust in the organization that is shepherding that information. The number in our bank account is trusted to be valid because we believe the institution of the bank will act according to the agreement that we have made with them. We trust that they will not arbitrarily change the number in our account to zero and take all of our money away, because if they did so, people would stop using them and they would go out of business. But as developers, we know that it is a trivial task to edit a database entry. Most of our institutions are based on this trust. Normally, this works fine, but there can be severe problems in times of unrest or corruption.

The concept of proof attempts to resolve this problem. Rather than trusting an authority, transactions, which can be monetary, contracts, or really any type of data, are posted in a public ledger that is readily available, and most importantly - immutable.

This sounds simple, but it's a form of a classic problem called the Byzantine Generals Problem.

As with most of human endeavor, the problem arises with bad actors seeking to cheat the system for their own benefit. Blockchain is an elegant solution to this problem.

The most famous, and most valuable, blockchain is called Bitcoin. It was first described in a whitepaper released by a person or entity called Satoshi Nakamoto.

A Technical Glossary of terms can be found here.

At the end of this module, you should be able to:

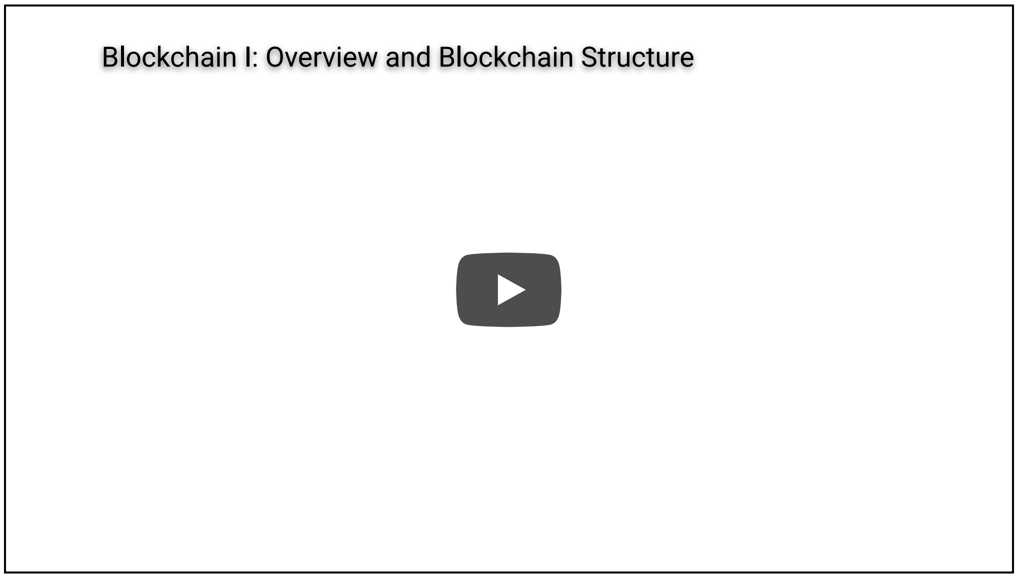
- diagram and code a simple blockchain, utilizing a cryptographic hash
- utilize a Proof of Work process to protect a blockchain from attack

Pro Tip

Remember that everyone around you is human and will make mistakes. Try to build up, not tear down.

Prepare

Review each preclass resource before class.



Learn

Learn to diagram and code a simple blockchain, utilizing a cryptographic hash

The structure of the blockchain itself is one of ways in which integrity is maintained. They are linked by hashed values that make it simple to spot if the integrity of the data has been comprimised

Overview

The first leg of the tripod that makes blockchain work is the chain itself. The chain is the structure that contains all of the transactions recorded. It is publicly available. For example, you can explore bitcoin here.

How it actually works is quite clever. Each link in the chain is called a block. It has the following components:

- Index - the number of the block in the chain, starting at 0 or 1, depending on the chain.
- Timestamp - the time at which the block was created. This is not required, but is often useful.
- Transactions - the monetary transactions, or any type of data, that is proofed by the block.
- Proof - the proof for this block. We'll get into this later.
- Previous Hash - the hash of the previous block.

The ultimate key to preserving integrity of the chain is the inclusion if the hash of the previous block within each block. This means that if a bad actor attempts to change a block in the middle of the chain, they must recreate all of the following blocks in the entire chain as well. Otherwise, this change can be identified very easily by iterating through the chain and looking for any blocks where the expected and actual previous hashes don't match.

We use a cryptographic hash to ensure that it is not meaningfully possible to reverse-engineer the previous hashes in the blockchain.

Our example uses the SHA-256 hashing algorithm.

Follow Along

Follow along as we build a simple blockchain.

We will use the SHA-256 hash algorithm to create this important part of our blockchain. Because each block has the hash of the previous block, it is impossible to change data in the chain without redoing all of the hashes.

We also use the SHA-256 hash algorithm for our Proof of Work. Cryptographic hashes are commonly used for this task because:

- The outcomes are unpredictable but deterministic
- A proposed solution using a specific hash can be quickly and easily verified by hashing that solution and seeing if it passes

Challenge

In the example we built, the server does the calculations for mining a new block. This won't do at all! We don't want to spend the electricity to mine blocks, we want other people to do it for us.

Build a client miner using the hash function to implement Proof of Work in your mining program.

Learn to utilize a Proof of Work process to protect a blockchain from attack

Proof of work secures the chain by making it nearly computationally impossible to cheat, because the cheater would have to do a greater amount of work than everyone else.

Overview

Proof of Work is an arbitrarily difficult problem to solve. By difficult, we mean that it would take the average computer about 100 years to mine a bitcoin block on its own.

Of course, it doesn't actually take 100 years for each block to be mined. By design, it takes about 10 minutes, and in Bitcoin, this is accomplished by tuning the difficulty of the problem to adjust for the number of current miners searching for the next block. All of the many hundreds of computers attempting to mine are essentially guessing over and over again, as fast as possible. It's like trying to find a needle in a haystack, or win the lottery. It's rare, unlikely, and there's only so much much you can do to improve your odds (usually with a cost), but someone will get lucky. Eventually, a solution will be discovered, a block added, and the process will start over again.

When a solution is discovered, a block is mined and it is added to the chain by the node that discovered it, or the first one it was reported to - assuming the node finds the solution to be valid. If it is a valid solution, the node will hash the previous block and add it to the new block, and add the rest of the properties, including a timestamp, index, and the list of pending transactions, which are now confirmed. It then shares the new block with the nodes in its network, which check the new block to make sure that it has an index one higher than the last block, a previous hash that matches the previous block, and a valid solution. If these checks pass, then the new block is added and spreads through the network, bearing in mind that `consensus` is determined by the longest valid chain.

This distribution means that the chain itself is nearly impossible to assault. We've learned that to change the chain itself, we need to redo the `previous hash` stored in all subsequent blocks. But as described above, it would take a vast amount of computing power to do so.

Let's say the chain is on block 1000, and we want to make a change to a transaction stored in block 98 to change a record in the ledger to give money to ourselves, instead of the intended recipient. Making this change is trivial, we simply open up our copy of the blockchain and edit the transaction.

However, we now have a problem. The `previous hash` stored in block 99 now no longer matches the hash of block 98. If you change even one character or digit, the hash is completely different. So if we try to share our changed blockchain as the valid one, everyone will reject it because it has a bad hash.

We could just chop off the the bad hash, but then our chain gets rejected because it's not the longest. The only recourse would be to mine new blocks and make a longer chain.

Just one problem here. If it takes the average computer 100 years to compute a solution to this problem, then we're in trouble. We have to get a large network together with a vast amount of computing power to create a solution and mine a new valid block in a reasonable amount of time. So, let's say we do that. We are stealing a great deal of bitcoin here, so investing a bit of resources makes sense.

We could buy 100 computers and solve the problem in a year. Even better, we could spend about the same amount of money per machine and get something called an ASIC - essentially a dedicated piece of hardware that is very fast and efficient at mining.

This sounds promising. With an investment of a little over 100k dollars, depending on the exact hardware we are comparing, we might be able to get the time to find a solution down to a pretty reasonable amount of time.

This still doesn't help. The biggest problem is that we are not alone. Everyone else mining is adding to the end of the chain. So not only do we need the hardware to solve this problem, but we need to outrun everyone else, because every 10 minutes, they add another block to the chain and move the goalpost farther away.

Follow Along

We will use cryptographic hashing to create a problem that takes a large amount of computation to solve, but is very easy to check if a solution is valid.

Challenge

Use this knowledge to create your own mining program.

Project

Blockchain

In this project, we'll modify existing blockchain code for a cryptocurrency (that's like a simple Bitcoin). The code currently mines on the server side, but we want to mine on the client. This unloads the server so that we don't have to pay for all that electricity and the work can be distributed to many clients.

Additionally, you'll write some client code to interface with the server API endpoints. The client will take over mining and will tell the server when it successfully mines a block.

Review

Class Recordings

You can use class recordings to help you master the material.

Blockchain I - Structure for CSPT4 with Tim Roy

CSPT4 Blockchain I - Structure with Tim Roy

All previous recordings

Demonstrate Mastery

To demonstrate mastery of this module, you need to complete and pass a code review on each of the following:

- Objective challenge:

  In the example we built, the server does the calculations for mining a new block. This won't do at all! We don't want to spend the electricity to mine blocks, we want other people to do it for us.

  Build a client miner using the hash function to implement Proof of Work in your mining program.

- Objective challenge:

  Use this knowledge to create your own mining program.

- Guided Project: Blockchain
- Project: Blockchain