Secure Blockchain Design

Alan Adame aadame4@csu.fullerton.edu
Douglas Galm douglasgalm@csu.fullerton.edu
Johnson Lien johnsonlien95@csu.fullerton.edu
Michael Lindwall michaellindwall@csu.fullerton.edu

Abstract

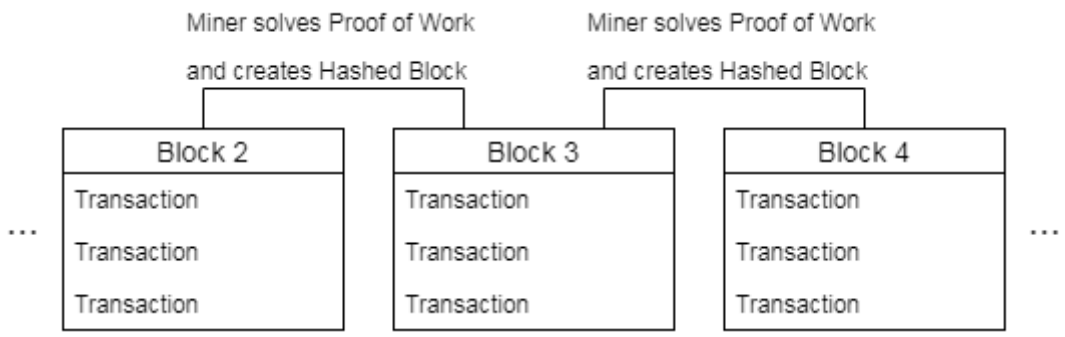
This project explores the inner workings of a blockchain written in python and adds an extra layer of security to the protocol by signing each transaction with the initiating user's private key. The basic features of this project include a micro-API to create a lightweight server for each node to broadcast their chain and transactions, a way to register nodes, a way to create transactions, a way to mine blocks, a way to get the longest chain, and a way to view the whole chain. The project is usable by a large number of nodes, which we tested virtually using pipenv (a virtual environment for python).

Introduction

In order to get started we had to learn about how blockchain works. We followed the guide for the blockchain implementation in python to get an overview of the requirements for a blockchain to work. The requirements for a blockchain are blocks, which are mined by solving a computational problem for a hash, transactions, which are added to blocks and contain a sender, recipient, and amount, and a proof-of-work algorithm for the miners to race and solve to create the next block. Other requirements for the blockchain to be useful is a way to register nodes with the chain, to simulate multiple users, and a way to resolve conflicting chains, by replacing shorter chains with the longest chain in the network of nodes, and a way to view the whole blockchain to see which transactions were added to which blocks, and to verify the hashes of the blocks and the signatures of the transactions.

Design

The basic idea of a blockchain involves a structure of blocks that contain transactions. Blocks are added to the chain only when a miner successfully solves the Proof of Work computational problem to find a specific hash. Then the block is created and the whole chain is hashed with the new block.



Security Protocols

This blockchain implementation makes extensive use of SHA hashing and RSA asymmetric encryption techniques. By using SHA hashing, the miners have a way to do work in order to solve for the next block. Miners race to find a number that when hashed with the previous block's hash produces a hash with 4 leading zeros. When a miner finds the number to create that hash, they are granted a coin for solving the problem and providing proof. By using the public-key scheme RSA, each user/node is required to generate a unique digital signature/fingerprint of the transaction by signing the message contents + a timestamp (to prevent duplicate transactions from creating the same signature). This way there is a way to verify that each transaction is authentic and was in fact initiated by the node who signed it.

Implementation

In order for this project to work correctly, one must be using python3.6.5. Once the newest version of python is running we need to get some libraries. The most important libraries used are Flask, which is a micro-framework for hosting lightweight web servers, RSA for python3, which we use to generate keys and sign transactions, Hashlib which we use to create SHA hashes of each block and to create digital signatures, and pipenv to create virtual environments to run multiple instances of the blockchain in a network. In order to handle the HTTP requests, we utilized an HTTP client called Postman

Conclusion

This project was very interesting and cleared up a lot about what blockchain is exactly and how it works. It is a very good idea that is not that difficult to implement and customize for a variety of applications. It was very cool to be able to apply two important ideas (hashing and digital signatures) to a real-world application.