Blockchain

Abstract

In this assignment, we implement a basic blockchain allowing users to create transactions, add transactions to a block, and add blocks to a chain. We implement a MD5 hashing function to generate all hashes in the blockchain. In the Research section, we discuss a general overview of blockchains, consensus protocols and the importance of digital signature in such systems.

1 Introduction

2 Design and Implementation

The implementation contains two main modules : `hash.py` and `blockchain.py`. The `hash.py` module contains an implementation of the MD5 function as specified in RFC 1321[1]. Its main function `md5()` is used as the hash function in the rest of the program. It takes a `bytes` object as input and outputs its 64 bits message digest in hexadecimal format. The `blockchain.py` module contains the classes used to model a blockchain : Blockchain, Block, Transaction, MerkleTree. The conceptual base class is the Transaction, which comprises a timestamp, a sender, a receiver and a value. Once instantiated, Transaction objects can be added to Block objects, which contains a block number, a creation timestamp, the previous block's hash and the transaction root. The transaction root is a MerkleTree object containing all transactions and their respective hashes, as well as the root hash which is computed each time a transaction is added to it. The root hash follows the properties of a Merkle Tree[2]. Once a user is done adding transactions to a block he can add the block to the chain, at which point the previous block hash is added to the final block.

If the `VERBOSE` variable is set to true in the `blockhain.py` file, every action is logged to the standard output. A graphical representation of the chain can be displayed directly by printing the chain. To run a demo, simply use `make demo`.

The strategy used to hash complex objects is to concatenate their relevant attributes into a single string and hash the string's utf-8 encoding representation.

3 Test Results

Plaintext	Plaintext equivalence	Ciphertext equivalence
128 bits	✓	✓
192 bits	✓	✓
256 bits	✓	✓

Table 1: Comparing NIST test data to our implementation

4 Research

According to Wikipedia[3], a blockchain is a growing list of records, called blocks, that are linked using cryptography. Each block contains a cryptographic hash of the previous block, a timestamp, and transaction data. The transaction data is generally represented as a Merkle tree, which allow efficient and secure verification of the contents of large data structures.

A blockchain can be used as a distributed ledger, as its design makes it resistant to data alteration. Each block being linked to the previous one by its cryptographic hash, modifying transaction data in one block would require a modification of every subsequent block, which requires consensus of the network majority. This design solves some of the long-standing problems of digital currencies, such as the double-spending problem, without the need of a trusted authority. The decentralized design of this system makes it secure by not having a central point of failure whereas traditional databases may be subject to exploits because all the data is held centrally.

Blockchains can be permissionless (open) which means anyone can add records to it, or permissioned (private), in which case the access to the chain is restricted to authorized users only. In open blockchains, access control is usually done by consensus mechanisms in order to limit the number of new blocks added to the chain. In the case of the Bitcoin blockchain, the consensus mechanism used is a Proof of Work. Private blockchains, on the other hand, do not benefit from the network advantages, and are closer in design to traditional databases.

To achieve overall reliability in blockchains, such systems rely on consensus protocols. These protocols dictate whether a new block is accepted in the chain or not. The most commonly used consensus protocols are Proof of Work (PoW) and Proof of Stake (PoS), which are described below.

The main idea behind Proof of Work is that the requester (of a service) must prove that it has performed a measurable effort in order to submit its request. In blockchains, this effort is to solve a Hashcash-like[4] cryptographic puzzle. By solving such a puzzle, a user can prove that it has performed a computationally expensive task, which can be verified easily by the service provider.
Let's take the Bitcoin blockchain as an example. In order to add a block to this chain, a network of users must "mine" it by discovering a nonce which, when hashed together with the current block, produces a hash value below the current target value. To do so, miners have no other choice than to bruteforce different nonces until the hash value begins with the necessary amount of zeros. Once this nonce is discovered the new block can be added to the chain, and other users can verify with little effort that the cryptographic puzzle has been correctly solved. In the Bitcoin blockchain, a double SHA-256 is used as hash function, and the numbers of leading zeros of the target values depends on the current difficulty of the blockchain, which regulates the average amount of time it takes to find a new block to around 10 minutes.

On the other hand, Proof of Stake does not rely on computationally intensive tasks in order to add blocks to the chain, but chooses the next block based on random selection combined with different factors such as wealth or age of the candidate blocks. In Peercoin's blockchain for example, the account allowed to forge the next block is chosen by a randomized combination of the amount of coins held and the amount of time these coins have been held (their age). A direct advantage of PoS over PoW is that it does not require the huge amount of processing power necessary to solve PoW cryptographic puzzles, and this processing power often comes from electricity generated from non-renewable sources of energy. However, PoS is much more complex to implement securely and reliably than PoW. It is less widely used in public blockchains, and only a few cryptocurrencies use it as their consensus protocol.

In order to function properly, blockchain systems rely heavily on asymmetric cryptography. Each transaction contained in a block is signed by the private key of the sender. This allows every user of the network to verify the authenticity of a transaction, and solves the double spending problem by preventing repudiation of the transaction. Asymmetric cryptography is also used to derive the address of users, which are a hash of their public key combined with other information such as checksums of the network properties when the key pair is generated. This demonstrates the usefulness of digital signatures in the context of blockchains.

5 Discussion

Do not use MD5, do not implement own hash library Sender, receiver -¿ public keys not names

6 Conclusion

References

[1] Ronald L. Rivest. The md5 message-digest algorithm. RFC 1321, RFC Editor, April 1992. URL http://www.rfc-editor.org/rfc/rfc1321.txt. http://www.rfc-editor.org/rfc/rfc1321.txt.

[2] Wikipedia contributors. Merkle tree — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Merkle_tree&oldid=911089346, 2019. [Online; accessed 28-October-2019].

[3] Wikipedia contributors. Blockchain — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Blockchain&oldid=921399625, 2019. [Online; accessed 23-October-2019].

[4] Adam Back et al. Hashcash-a denial of service counter-measure, 2002. URL http://www.hashcash.org/papers/hashcash.pdf. Accessed: 2019-10-23.
https://rosettacode.org/wiki/MD5/Implementation
https://www.ietf.org/rfc/rfc1321.txt
https://en.wikipedia.org/wiki/MD5