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 also implement a MD5 hashing function to generate all necessary hashes in the blockchain. In the Research section, we discuss a general overview of the blockchain technology, its consensus protocols and the importance of digital signature in such systems.

1 Introduction

Although the first research describing a chain of blocks linked together using cryptography traces back to Haber et al.[1] in 1991, it was only practically implemented in 2008 by an unknown individual (or group of individuals) named Satoshi Nakamoto[2]. Nakamoto released the software on January 2009, effectively creating the first public cryptocurrency blockchain, the Bitcoin. The complete software is quite complex, as it comprises a consensus mechanism among a peer-to-peer network, but how can its core functionality be broken down to a simple Python implementation ?

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[3]. 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 and 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 (once added to the chain) 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 new transaction is added to the block. The root hash follows the properties of a Merkle Tree[4]. Once a user is done adding Transactions to a Block he can add the block to a Blockchain object, at which point the previous block hash is written to the added block.

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.

The root of the Merkle Tree is computed recursively in its `compute_root()` class method. This method hashes a list of transactions by grouping consecutive transactions into pairs, hashing their concatenation and appending the result to a new list. The function calls itself with this new list as a parameter until the list only contains one element, which is the root hash of the Merkle Tree.

If the `VERBOSE` variable is set to true in the `blockchain.py` file (it is, by default), every action is logged to the standard output. A graphical representation of the chain can be displayed directly by printing the chain (also done by default). To run a demo, simply use `make demo`. In the `demo.py` file, the `SLEEP_TIME` variable can be set to an integer number of seconds to run the demo at a slower pace.

3 Test Results

The following tests can be reproduced by running the `make tests` command.

Plaintext	Hash	Matches RFC test
	0xd41d8cd98f00b204e9800998ecf8427e	✓
a	0x0cc175b9c0f1b6a831c399e269772661	✓
abc	0x900150983cd24fb0d6963f7d28e17f72	✓
message digest	0xf96b697d7cb7938d525a2f31aaf161d0	✓
abcdefghijklmnopqrstuvwxyz	0xc3fcd3d76192e4007dfb496cca67e13b	✓
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789	0xd174ab98d277d9f5a5611c2c9f419d9f	✓
12345678901234567890123456789012345678901234567890123456789012345678901234567890	0x57edf4a22be3c955ac49da2e2107b67a	✓

Table 1: Comparing MD5 test suite to our implementation

This demo can be reproduced (beware, the number of blocks and transactions are random) using the `make demo` command.

```
New block (#2) created at 09:40:23.854783 on 10/29/19

New transaction
Timestamp : 09:40:23.854839 - 10/29/19
Sender    : Smith
Receiver  : Adams
Value     : 13
Hash      : 0xec8991975474eaf816f39c17e28fc3be

Transaction 0xec8991975474eaf816f39c17e28fc3be added to block 2

Block #2 added to the chain

Blockchain contains 2 blocks

                    Block 2 - 0xaa609b31d5c8374fe7015345c12f5919 -  1 transactions

 Timestamp                  : 09:40:23.854783 10/29/19
 Previous block hash        : 0x81a84f9febcdead2f3b49fd525b2e033
 Transaction root hash      : 0xe9146457514171fbff3c5824306093d5
 2019-10-29 09:40:23.854839 : 13 transferred from Smith to Adams

                                          |
                                          |
                                          V

                    Block 1 - 0x81a84f9febcdead2f3b49fd525b2e033 -  3 transactions

 Timestamp                  : 09:40:23.850715 10/29/19
 Previous block hash        : 0x0
 Transaction root hash      : 0x3fa07fe4847c53cac4c7056439a582d5
 2019-10-29 09:40:23.850904 : 81 transferred from Isaac to Lewis
 2019-10-29 09:40:23.851571 : 92 transferred from Lewis to Adams
 2019-10-29 09:40:23.852360 : 91 transferred from Smith to Alice
```

Figure 1: A demo run of the implementation

4 Research

According to Wikipedia[5], a blockchain is a growing list of records, called blocks, that are linked together 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 third-party 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 the policy of acceptance of new blocks within the chain. 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 [6] cryptographic puzzle. By solving such a puzzle, a user can prove that it has performed a computationally expensive task, which can be easily verified 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 a defined 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 requests. 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.

There exists some other consensus protocols such as Proof of Space, Proof of Authority or Proof of Burn. As of 2019, they are even less used than Proof of Stake because of some loopholes such protocols may contain.

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

Athough MD5 was considered a secure hash function in 1992, a 2013 cryptanalytic attack by Xie Tao et al.[7] allows to find collisions in 2^{18} MD5 compressions, which takes around one second on an average computer. This would make our blockchain implementation vulnerable as false blocks could be forged easily, and bruteforce would not be the most efficient PoW algorithm anymore. This is why actual public blockchains use more secure hash functions such as double SHA-256 in the case of Bitcoin.

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

6 Conclusion

References

[1] Stuart Haber and W. Scott Stornetta. How to time-stamp a digital document. *Journal of Cryptology*, 3(2): 99–111, Jan 1991. ISSN 1432-1378. doi: 10.1007/BF00196791. URL https://doi.org/10.1007/BF00196791.

[2] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2009. URL http://www.bitcoin.org/bitcoin.pdf.

[3] 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.

[4] 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].

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

[6] Adam Back et al. Hashcash-a denial of service counter-measure, 2002. URL http://www.hashcash.org/papers/hashcash.pdf. Accessed: 2019-10-23.

[7] Tao Xie, Fanbao Liu, and Dengguo Feng. Fast collision attack on md5. Cryptology ePrint Archive, Report 2013/170, 2013. https://eprint.iacr.org/2013/170.