Study
on
Bitcoin

Final Project Report
August 6, 2018

Compiled By:

Pushpendra
(1639812)

Guided By:
Prof. Dr. Weidong (Larry) Shi
Lei Xu
Lin Chen

University of Houston, Main Campus
Summer Semester, 2018

Table of Contents

1. Introduction

This report is compiled as part of Special Project COSC 6398. As we wanted to understand the fundamental principles and implementation in cryptocurrency, so we took Bitcoin as to study and fulfil the aspirations. We were given several problems and implementations to research on in reference to Bitcoin. This report consists of stating the problem followed by the analysis or finding along with standard references as part of proof. We have also tried to include the snippets or references from Bitcoin source code and further investigation wherever required.

2. Problem Analysis

a) The block header structure of Bitcoin block

Problem Statement: Find out the block header structure of Bitcoin in source code.

In Bitcoin the block header has following structure which consist of 80 bytes:

Field	Purpose	Size (Bytes)
Version	Block version number	4
HashPrevBlock	256-bit hash of the previous block header	32
HashMerkleRoot	256-bit hash based on all of the transactions in the block	32
Time	Current timestamp as seconds since 1970-01-01T00:00 UTC	4
Bits	Current target in compact format	4
Nonce	32-bit number (starts at 0)	4

This is one of the most basic concept we need to understand in Bitcoin. It is 80 byte long header. As we know that the header has to be hashed using SHA256 algorithm, which needs 64 byte of input, so its definitely raised a question as to how Bitcoin address this in hashing implementation. We are also interested in the optimization of the hashing function. Below is the reference of one such recommendation.

http://www.nicolascourtois.com/bitcoin/Optimising%20the%20SHA256%20Hashing%20Algorithm%20for%20Faster%20and%20More%20Efficient%20Bitcoin%20Mining_Rahul_Naik.pdf

Source code reference:

File name: block.h
Class name: CBlockHeader
Path: src/block.h

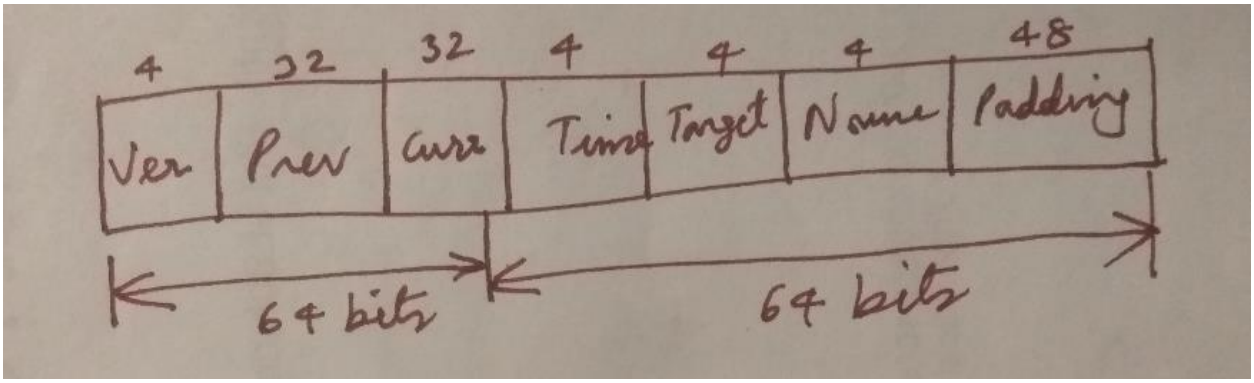
Code:

```
class CBlockHeader
{
public:
    // header
    int32_t nVersion;
    uint256 hashPrevBlock;
    uint256 hashMerkleRoot;
    uint32_t nTime;
    uint32_t nBits;
    uint32_t nNonce;
}
```

b) Find out padding bits done in block header in Bitcoin during hashing.

Problem Statement: Find out padding bits done in block header of Bitcoin during hashing.

The Bitcoin mining process repeatedly hashes an 80-byte block header by incrementing a 32-bit nonce which is at the end of the header data. The hashing process of the header involves two runs of the SHA256 hashing function or the compression function. The first time it consumes the first 64 bytes of the header and a second time it processes the remaining 16 bytes along with padding. It is important to note that first part is constant. The padding consists of 6 bytes of data. It is generally filled with 1 followed by all zeros.



Field	Size	Description
Version	32 bits	Block version information that is based on the Bitcoin software version creating this block
hashPrevBlock	256 bits	The hash of the previous block accepted by the Bitcoin network
hashMerkleRoot	256 bits	Bitcoin transactions are hashed indirectly through the Merkle Root
Timestamp	32 bits	The current timestamp in seconds since 1970-01-01 T00:00 UTC
Target	32 bits	The current Target represented in a 32 bit compact format
Nonce	32 bits	Goes from 0x00000000 to 0xFFFFFFFF and is incremented after a hash has been tried
Padding + Length	384 bits	Standard SHA256 padding that is appended to the data above

Table 3: Bitcoin Block Header Fields Along With Their Brief Description

Source code reference:

File name: sha256.cpp Path: crypto/sha256.cpp

https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2017-April/014077.html

http://www.nicolascourtois.com/bitcoin/Optimising%20the%20SHA256%20Hashing%20Algorithm%20for%20Faster%20and%20More%20Efficient%20Bitcoin%20Mining_Rahul_Naik.pdf

https://www.researchgate.net/publication/258144528_The_Unreasonable_Fundamental_Incertitudes_Behind_Bitcoin_Mining

c) Why Bitcoin does hashing twice?

Problem Statement: Provide the reasoning why Bitcoin does hashing twice.

Bitcoin uses two hash iterations (commonly referred as $SHA256^2$ i.e. SHA256 function squared). It is also called hashcash. The primary reason for double hashing is security. In past there has been partial attack on the smaller but related SHA1 hash. SHA1's resistance to birthday attacks has been partially broken as of 2005 in O(264) vs the design O(280), with practical attacks having been used successfully in early 2017.

While the Bitcoin hashcash relies on pre-image resistance and so is not vulnerable to birthday attacks, a generic method of hardening SHA1 against the birthday collision attack is to iterate it twice. A comparable attack on SHA256 does not exist so far, however as the design of SHA256 is similar to SHA1 it is probably defensive for applications to use double SHA256. So this is the fundamental reason for double hashing. It is not necessary given hashcash reliance on preimage security, but it is a defensive step against future cryptanalytic developments. The attack on SHA1 and in principle other hashes of similar design like SHA256, was also the motivation for the NIST SHA3 design competition which is still ongoing.

In design SHA-3 is very different from SHA-2 which is the next-generation tool for securing the integrity of electronic information. SHA-2 has not shown up any problem as of now and so SHA-3 is developed as a backup plan. Bitcoin being the cryptocurrency and so security is a critical concern, so we expect to accept SHA-3 very soon by its development community.

https://en.bitcoin.it/wiki/Hashcash#Double_Hash

d) Is it possible to steal the proof of work done by a miner?

Problem Statement: In Bitcoin mining, a miner solves a problem of POW and announce it to network. Is it possible to steal the proof of work done by a miner and announce it to the network by another miner and also end up stealing the reward of hard work done.

Let assume that the actual miner who mined the block is miner A and the one who is willing to stole the mining result is miner B.

The architecture of Bitcoin is designed in a way that it highly impossible to have the same hash value as mining result of a block by two different miners. Block mining is basically the process of guessing the hash of the block which starts with specific number of zero's. The hash is created by running the SHA-256 algorithm on six pieces of data.

i) The Bitcoin version number.
ii) The previous block hash.
iii) The Merkle Root of all the transactions selected to be in that block.
iv) The timestamp.
v) The difficulty target.
vi) The Nonce.

The Bitcoin system takes care of protected the miners in following ways:

a) **The Merkle Root**: Merkle root is calculated with the hashing of all transactions received by the miner from Bitcoin system. This collection of transactions has a very special transaction called **Coinbase transaction** which is separately inserted by the miner. Using this Coinbase transaction the miner can choose to have the block reward to himself which contains the miner's unique Bitcoin addresses. Due to this it is extremely unlikely for two people to have the same Merkle root because the first transaction in the block is a generation "sent" to one of miner's unique Bitcoin addresses. Since the block is different from everyone else's blocks, it is guaranteed to produce different hashes.

The block that A mined includes the mining rewards going to A's address. If B alters the block data to output the rewards to his own receiving address, then the nonce (and other variable values, including "extranonce" and timestamp) that A used to solve the block will almost certainly no longer solve the block.

So, A is protected by adding coinbase transaction with his own Bitcoin address.

Source code reference:

Filename: miner.cpp
Path: src/miner.cpp
Method Signature:
```
std::unique_ptr<CBlockTemplate>
BlockAssembler::CreateNewBlock(const CScript& scriptPubKeyIn, bool
fMineWitnessTx)
```

https://en.bitcoin.it/wiki/Block_hashing_algorithm

https://en.bitcoin.it/wiki/Protocol_documentation

https://chrispacia.wordpress.com/2013/09/02/bitcoin-mining-explained-like-youre-five-part-2-mechanics/

b) **Transaction List**: There is no requirement that all miners be working with the same set of transactions. We know that the miner can select their own choice of transactions based on transaction fees. Hence this can also help in protecting the miner's hard work.

Source code reference:

File name: miner.cpp Path: src/miner.cpp

In Bitcoin even if the result of mining is stolen by another miner B, it will be of no use to him.

e) Find out how Bitcoin calculates the transaction fees.

Problem Statement: Find out how Bitcoin calculates the transaction fees.

In Bitcoin a user sends funds or coins to another user and is called transaction. The miners add it to block using the process called mining. The user needs to pay some transaction fees as reward for miners who do the hard work of mining and make making transaction successful. Transactions compete regarding the fee they offer to the miner divided by the amount of block space they require to be included in a block. A user doing the transaction and miner have different interest as part of transaction fee.

- Users care about the total fee. This is the total fee that user is paying in a transaction. While miners care about the fee per byte (or kilobyte). This is the total fee divided by the number of bytes in a transaction. This is the most important measurement for miners. The reason is that they use this to decide whether to include the transaction in the blocks they attempt to produce since they can only include maximum of 1 MB of transactions in their blocks. As such, they prefer to include transactions that pay more fee per byte.
- It's important to note that the total Bitcoin amount of the transaction doesn't matter for fee calculation.

Transaction Size: In Bitcoin there are two most popular transaction type P2SH and P2PKH. Considering all the complexity involved we can formulate the size of these transaction using below formula:

```
Size of P2SH/P2PKH Transaction = in * 146 + out * 33 + 10
```

However, computing the size of P2SH/P2PKH transaction that is being funded with complex is inherently difficult and is dependent on the complexity of the redeemScript used to produce the scriptHash of the prior P2SH transaction.hvk. Details can be found at the below link:

https://bitcoin.org/en/developer-guide#standard-transactions

Priority transactions: Transaction priority is calculated as a value-weighted sum of input age, divided by transaction size in bytes:

```
priority = sum(input_value_in_base_units * input_age)/size_in_bytes
```

For example, a transaction that has 2 inputs, one of 5 btc with 10 confirmations, and one of 2 btc with 3 confirmations, and has a size of 500bytes, will have a priority of:

```
(500000000 * 10 + 200000000 * 3) / 500 = 11,200,000
```

Higher priority transactions will be processed first.

Network State: The Bitcoin is a bit like a highway in that it can get congested at peak times. If there is no need to hurry, one should wait till the number of unconfirmed transactions in the memory pool drops, taking the average transaction cost down with it.

It uses an algorithm which is conceptually very simple and does not attempt to have any predictive power over future conditions. It only looks at some recent history of transactions and returns the lowest fee rate such that in that recent history a very high fraction of transactions with that fee rate were confirmed in the block chain in less than the target number of blocks.

https://gist.github.com/morcos/d3637f015bc4e607e1fd10d8351e9f41

No of TxIn and TxOut: Lager the number of In-Out transactions involved means higher the transaction fees.

So if your transaction has **in** inputs and **out** outputs, the transaction size in bytes will be:

> ***in*** *x 180 +* ***out*** *x 34 + 10 plus or minus* ***in***

For this transaction, it has one input and two outputs. Its transaction size should be 258 bytes based on this formula.

> ***1*** *x 180 +* ***2*** *x 34 + 10 + − 1*

Change Output: Ignoring Coinbase transaction if the value of a transaction's outputs exceed its inputs, the transaction will be rejected. On the other hand, if the inputs exceed the value of the outputs, any difference in value may be claimed as a transaction fee by the Bitcoin miner who creates the block containing that transaction. Since each transaction spends Unspent Transaction Outputs (UTXOs) and because a UTXO can only be spent once, the full value of the included UTXOs must be spent or given to a miner as a transaction fee.

Free Transactions: Bitcoin even supports free transactions. A transaction is safe to send without fees if these conditions are met:

- It is smaller than 1,000 bytes.
- All outputs are 0.01 BTC or larger.
- Its priority is large enough

It may seem frustrating that there isn't a simpler way of determining fees, but due to the way Bitcoin works, the price users pay depends on a number of factors as we discussed above.

https://en.bitcoin.it/wiki/Techniques_to_reduce_transaction_fees

https://gist.github.com/morcos/d3637f015bc4e607e1fd10d8351e9f41

Source code reference:

File path: src/txmempool.cpp
Method: `CFeeRate CTxMemPool::GetMinFee(size_t sizelimit) const`

https://en.bitcoin.it/wiki/Transaction

https://en.bitcoin.it/wiki/Transaction_fees

Further work to be done: As per the official blockchain site, it states regarding Bitcoin transaction fees that:

Smaller transactions are easier to validate; larger transactions take more work, and take up more space in the block. For this reason, miners prefer to include smaller transactions. A larger transaction will require a larger fee to be included in the next block.

Upon investigation I found from the source code comment that extremely large transactions with lots of inputs can cost the network almost as much to process as they cost the sender in fees, because computing signature hashes is O(ninputs*txsize). Limiting transactions to MAX_STANDARD_TX_WEIGHT mitigates CPU exhaustion attacks. This section is still not clear and it requires further work.

f) Miner can select the transaction of his own choice for mining. If it is true, a miner will always select transaction with huge amount which can create issue of ageing.

Problem Statement: In Bitcoin, a miner can select the transaction of his own choice for mining. If it is true, a miner will always select transaction with huge amount which can create issue of ageing. But even with this Bitcoin works normally. Provide all finding on this.

Every Bitcoin transaction must be added to the blockchain before they can be considered successfully completed or valid in its official public ledger. Miners work to validate transactions and add it to the blockchain. For the hard work done by miner they are rewarded 12.5 BTC for every successful block mining. There is limit that the block being added to blockchain can have maximum size of 1 MB. For this reason, miners have a financial incentive to prioritize the validation of transactions that include a higher fee. This is the reason if someone wants to send funds and get a quick confirmation, the appropriate fee to include can vary greatly, depending on a number of factors. While the fee does not depend on the amount being sent, it does depend on network conditions at the time and the data size of the transaction. We discuss below the factors:

- **Transaction size:** Due to the fact that a block in the Bitcoin blockchain can contain no more than 1 MB of information, transaction size is an important consideration for miners. Smaller transactions are easier to validate; larger transactions take more work and take up more space in the block. For this reason, miners prefer to include smaller transactions. A larger transaction will require a larger fee to be included in the next block. In short, extremely large transactions with lots of inputs can cost the network almost as much to process as they cost the sender in fees, because computing signature hashes is O(ninputs*txsize).
  https://support.blockchain.com/hc/en-us/articles/360000939883-Explaining-bitcoin-transaction-fees
  http://diyhpl.us/wiki/transcripts/scalingbitcoin/hong-kong/validation-cost-metric/

- **Network conditions:** Because a block on the Bitcoin blockchain can only contain up to 1 MB of information, there is a limited number of transactions that can be included in any given block. During times of congestion, when a large number of users are sending funds, there can be more transactions awaiting confirmation than there is space in a block. When a user initiates to send funds and the transaction is broadcast, it initially goes into what is called the memory pool before being included into a block for mining. It is from this memory pool that miners choose which transactions to include, prioritizing the ones with higher fees. If the memory pool is full, the fee market may turn into a competition and users compete to get their transactions into the next block by including higher and higher fees. Eventually, the market will reach a maximum equilibrium fee that users are willing to pay and the miners will work through the entire memory pool in order. At this point, once traffic has decreased, the equilibrium fee will go back down.
  https://bitcoin.stackexchange.com/questions/42126/do-larger-blocks-make-it-harder-for-smaller-miners-to-compete-why

- **Priority fee and a Regular fee:** A user can choose between a Priority fee and a Regular fee. The Priority fee is calculated in a way so that the transaction gets included in a block within the hour. The Regular

fee is lower and is for users who can afford to be a bit more patient. For Regular fee a confirmation for a transaction will typically take a bit more than an hour. There is also a provision for advanced users who can set custom fees.

- **Transaction rejection:** It is possible and happens that a transaction is never confirmed by Bitcoin and gets rejected because of low transaction fees. In this case the user needs to increase the transaction fees and initiates the transaction again.
  https://www.safaribooksonline.com/library/view/mastering-bitcoin/9781491902639/ch08.html