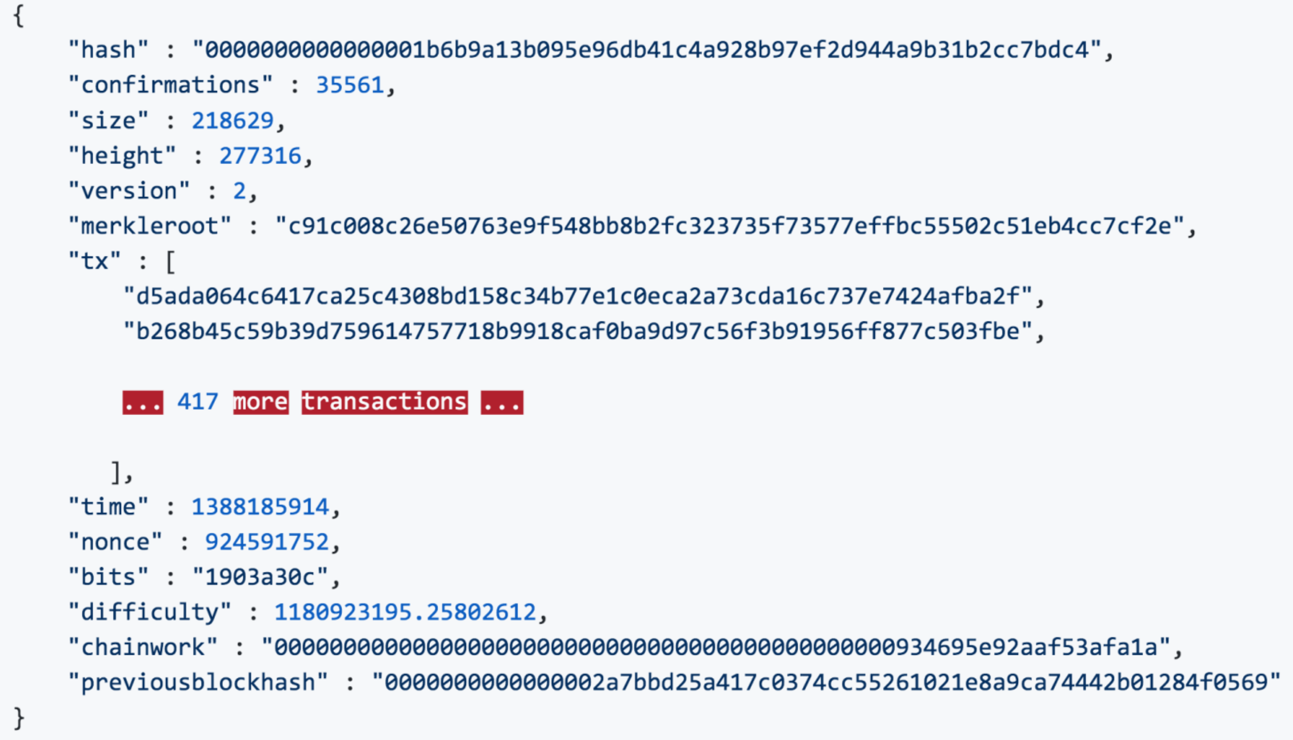
Mining and Consensus

•••

By Gerber

What Information does a Block Include ?

- Hash
- Confirmations
- Size
- Height
- Version
- Merkleroot
- Tx
- Time
- Nonce
- Bits
- Difficulty
- Chainwork
- Previous Block Hash



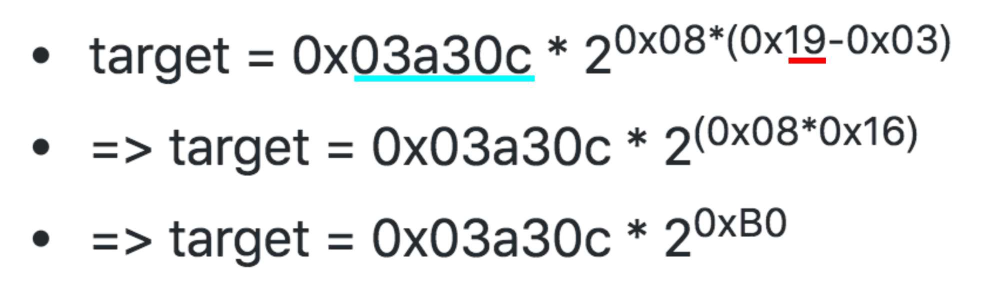
```
{
    "hash" : "0000000000000001b6b9a13b095e96db41c4a928b97ef2d944a9b31b2cc7bdc4",
    "confirmations" : 35561,
    "size" : 218629,
    "height" : 277316,
    "version" : 2,
    "merkleroot" : "c91c008c26e50763e9f548bb8b2fc323735f73577effbc55502c51eb4cc7cf2e",
    "tx" : [
        "d5ada064c6417ca25c4308bd158c34b77e1c0eca2a73cda16c737e7424afba2f",
        "b268b45c59b39d759614757718b9918caf0ba9d97c56f3b91956ff877c503fbe",

        ... 417 more transactions ...

       ],
    "time" : 1388185914,
    "nonce" : 924591752,
    "bits" : "1903a30c",
    "difficulty" : 1180923195.25802612,
    "chainwork" : "000000000000000000000000000000000000000000000934695e92aaf53afa1a",
    "previousblockhash" : "0000000000000002a7bbd25a417c0374cc55261021e8a9ca74442b01284f0569"
}
```

Target Representation

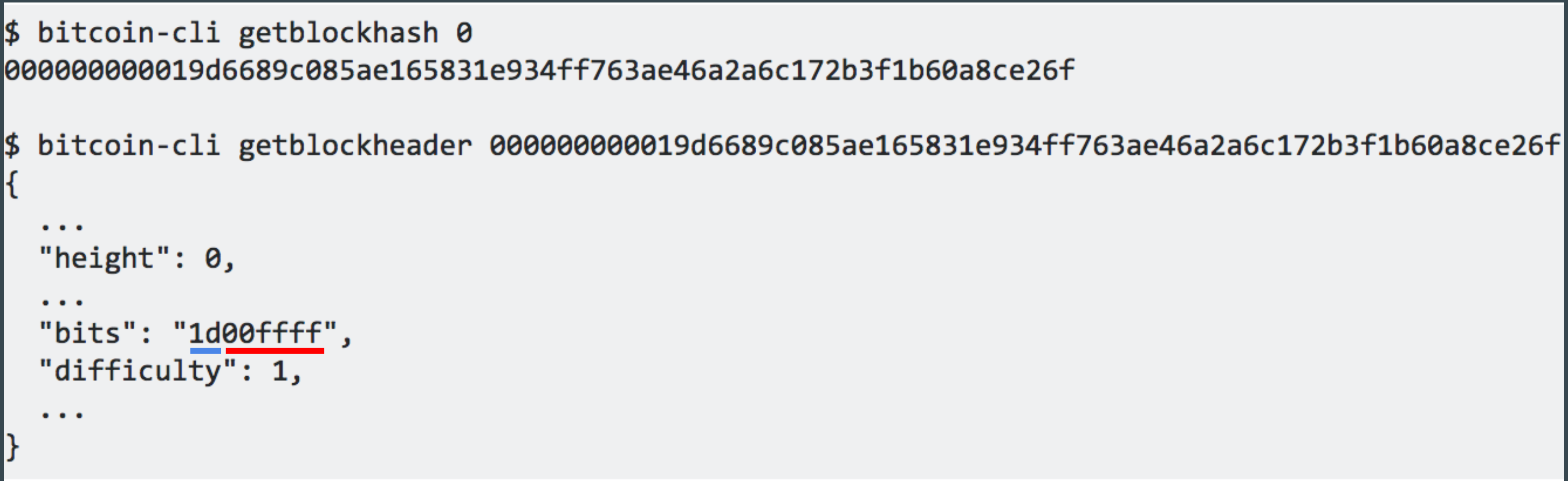
- 256 bits
- First two hexadecimal digits for the exponent and the next six hex digits as the coefficient.
- target = coefficient * 2^(8 * (exponent – 3))
- Ex : 0x1903a30c



- target = $0x\underline{03a30c} * 2^{0x08*(0x\underline{19}-0x03)}$
- $=>$ target = $0x03a30c * 2^{(0x08*0x16)}$
- $=>$ target = $0x03a30c * 2^{0xB0}$

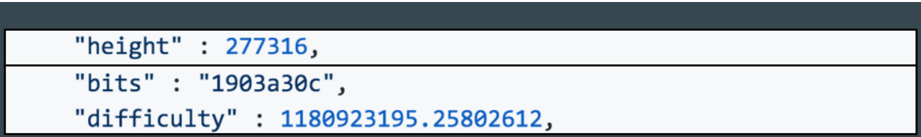
- **target = 0x0000000000000003A30C0000000000000000000000000000000000000000000000**

Difficulty



```
$ bitcoin-cli getblockhash 0
000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f

$ bitcoin-cli getblockheader 000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f
{
  ...
  "height": 0,
  ...
  "bits": "1d00ffff",
  "difficulty": 1,
  ...
}
```



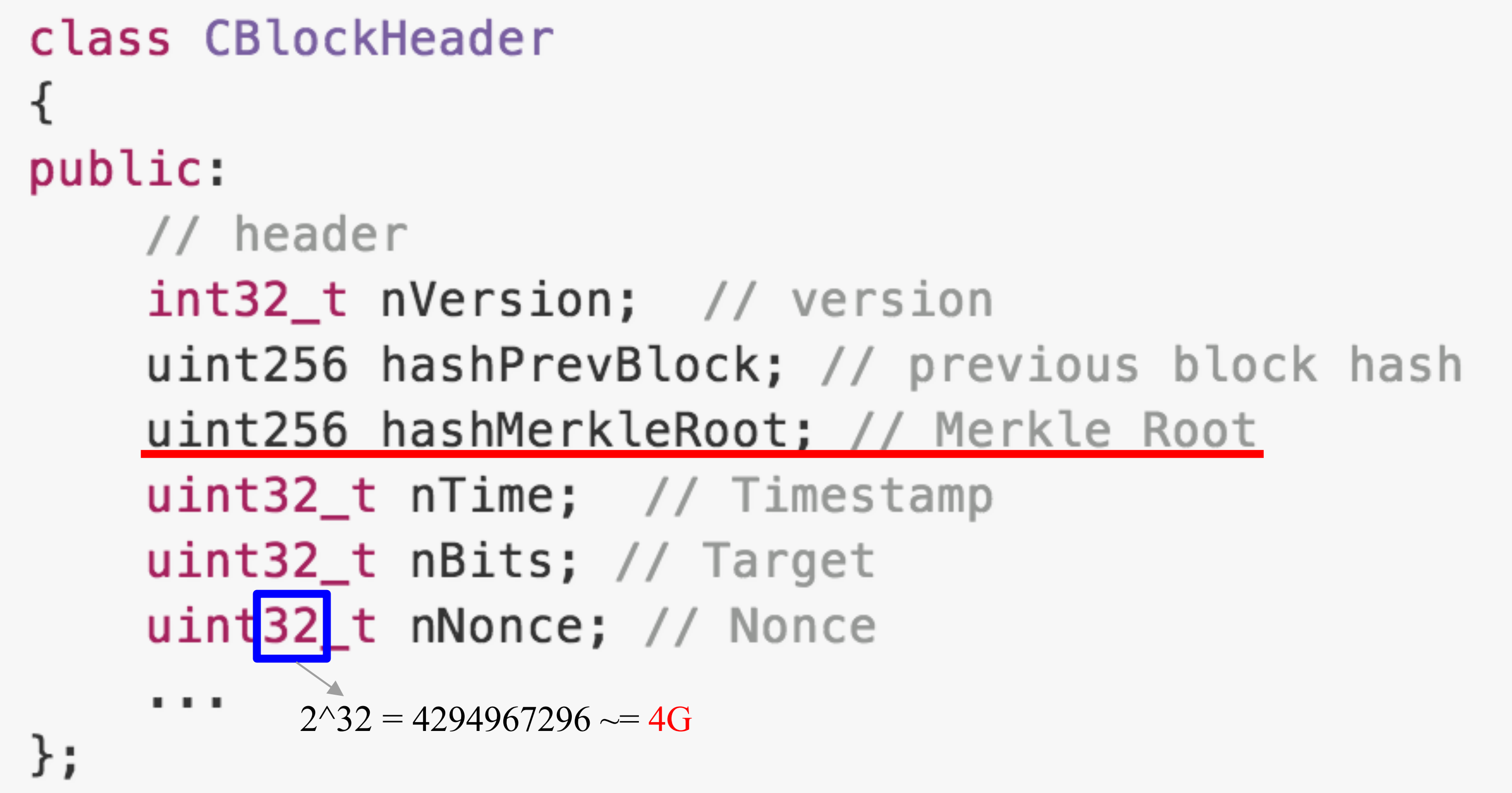
```
    "height" : 277316,
    "bits" : "1903a30c",
    "difficulty" : 1180923195.25802612,
```

- Difficulty of block 277316
  - (0x00ffff * 256 ^ (0x1d - 0x03)) / (0x03a30c * 256 ^ (0x19 - 0x03))
  - = 1180923195.25802612

What Information does a Block Header Include ?

- Version : A version number to track software/protocol upgrades
- Previous Block Hash : A reference to the hash of the previous block in the chain.
- Merkle Root : A hash of the root of the merkle tree of this block’s transaction.
- Timestamp : The approximate creation time of this block (seconds from Unix Epoch)
- Target : The Proof-of-Work algorithm target for this block.
- Nonce : A counter used for the Proof-of-Work algorithm.

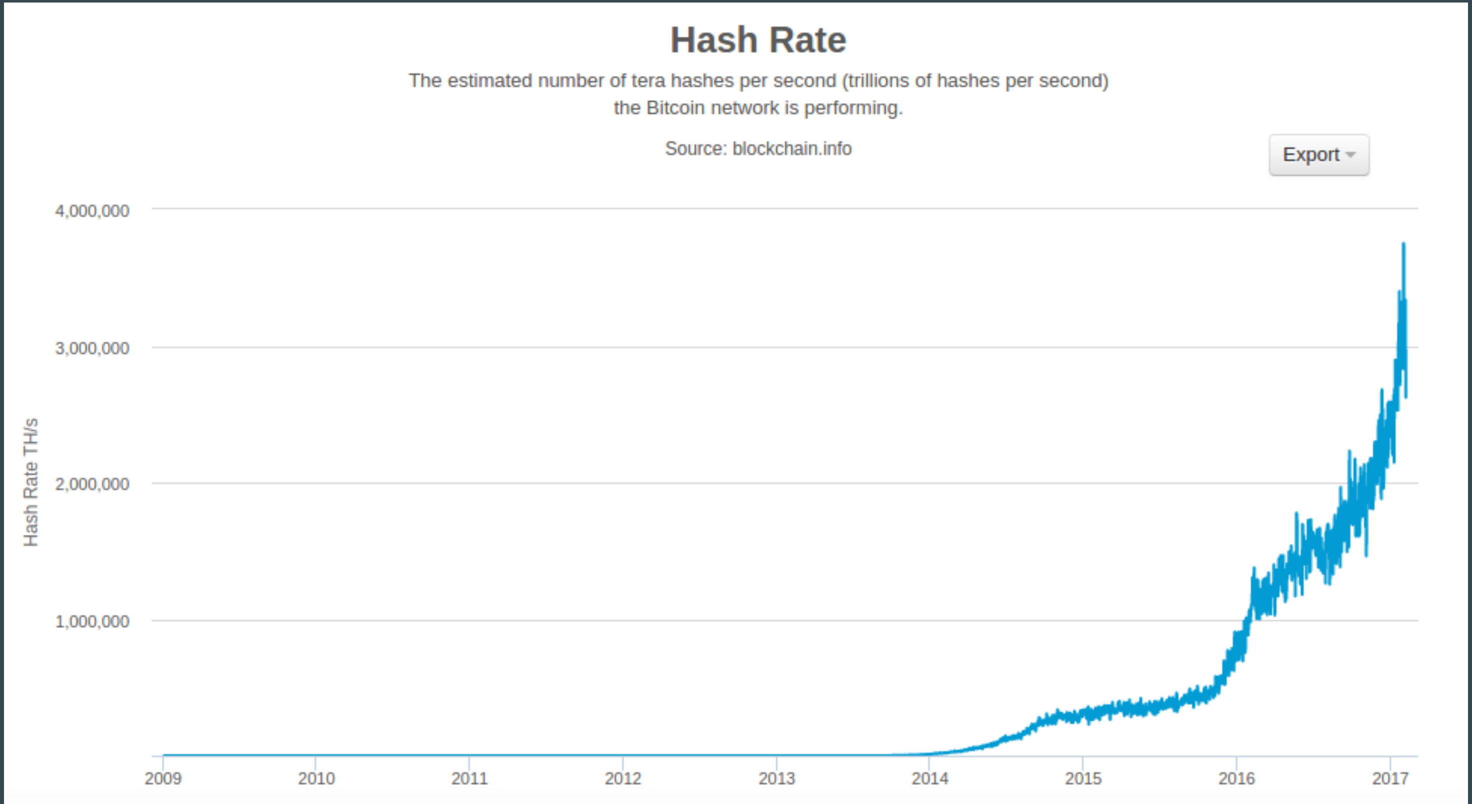
src/pow.cpp



```
class CBlockHeader
{
public:
    // header
    int32_t nVersion;  // version
    uint256 hashPrevBlock; // previous block hash
    uint256 hashMerkleRoot; // Merkle Root
    uint32_t nTime;  // Timestamp
    uint32_t nBits; // Target
    uint32_t nNonce; // Nonce
    ...
};
```

2^32 = 4294967296 ~= 4G

Hashing Power



Hash Rate

The estimated number of tera hashes per second (trillions of hashes per second) the Bitcoin network is performing.

Source: blockchain.info

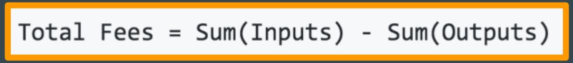
The Extra Nonce Solution

- Updating the block timestamp to account for the elapsed time.
  - Mining hardware > 4GH/sec, nonce values exhaust in less than a second.
  - Timestamp could be stretched a bit, but moving too far into the future => block invalid

- Use the coinbase transaction as a source of extra nonce values.
  - Coinbase script can store between 2 and 100 bytes of data. => extra nonce space
  - Coinbase transaction is in the merkle tree => merkle root changes

Coinbase Transaction

- The first transaction in the block is the coinbase transaction.
- Contains the reward for mining effort + transaction fees.
- Doesn’t consume UTXO as inputs. Creates bitcoin from nothing.
- The coinbase transaction has one output, payable to the miner’s own bitcoin address.
- Coinbase transactions do not have an unlocking script.
    - Replaced by coinbase data. (2-100 bytes)

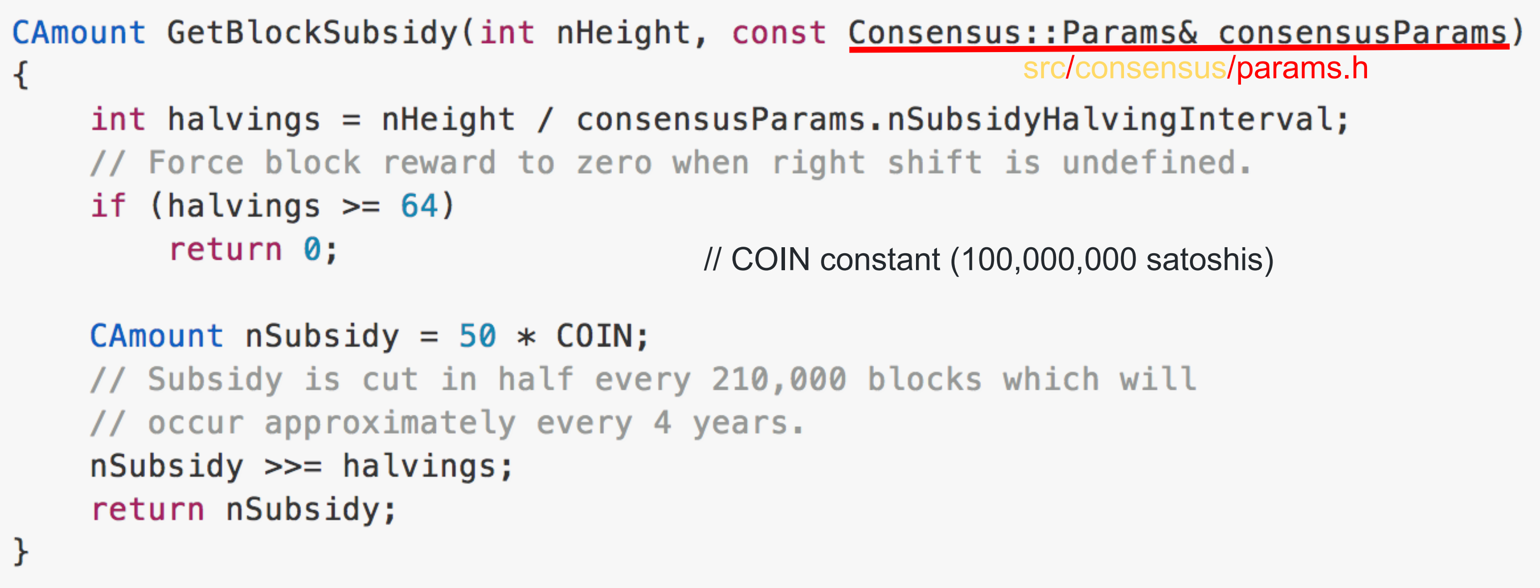


```
Total Fees = Sum(Inputs) - Sum(Outputs)
```

Coinbase data

- Ex : 03443b0403858402062f503253482f
    - Instructs the script execution engine to push the next three bytes onto the script stack.
    - Block height : in little-endian format. 0x043b44 = 277316
    - Extra nonce : random value
    - ASCII-encoded string “/P2SH/” : Indicates that the mining node that mined this block supports the P2SH improvement defined in BIP-16.

Block Reward



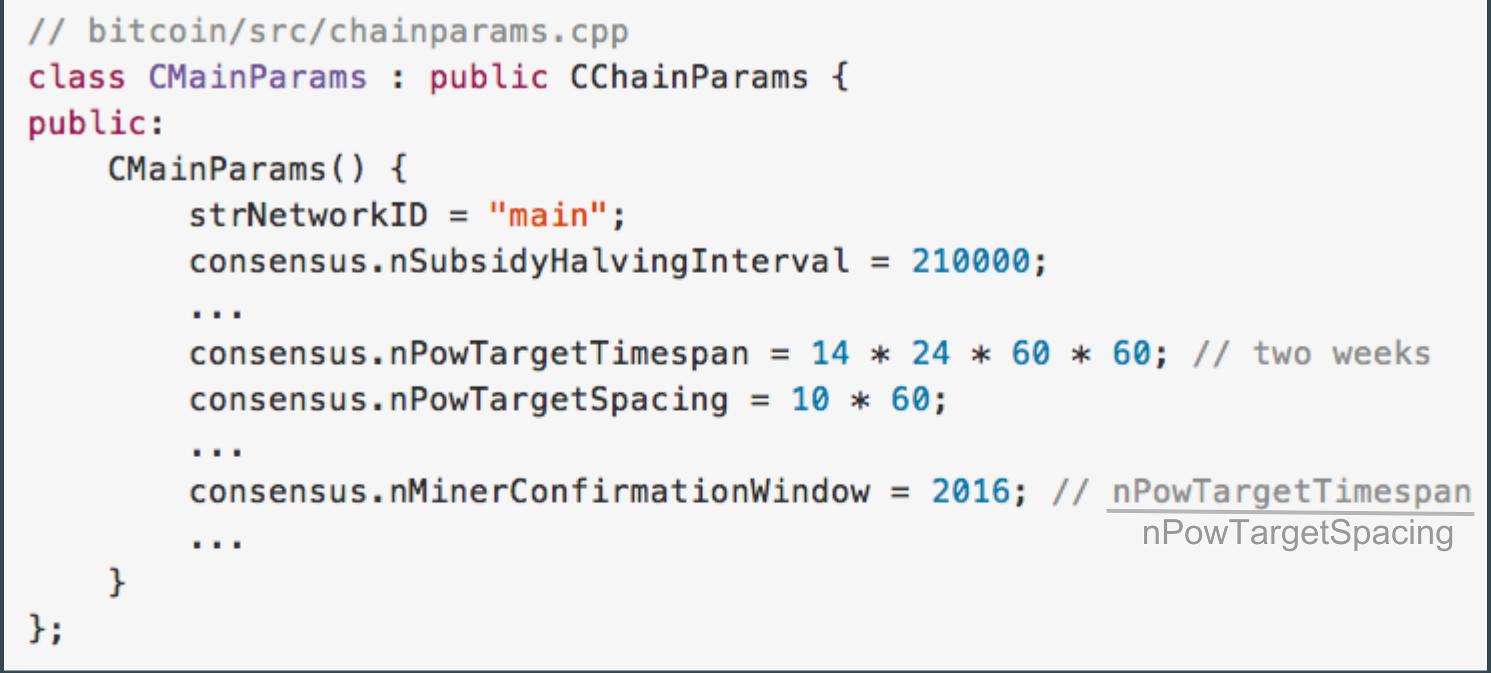
```
CAmount GetBlockSubsidy(int nHeight, const Consensus::Params& consensusParams)
{
    int halvings = nHeight / consensusParams.nSubsidyHalvingInterval;
    // Force block reward to zero when right shift is undefined.
    if (halvings >= 64)
        return 0;

    CAmount nSubsidy = 50 * COIN;
    // Subsidy is cut in half every 210,000 blocks which will
    // occur approximately every 4 years.
    nSubsidy >>= halvings;
    return nSubsidy;
}
```

src/consensus/params.h

// COIN constant (100,000,000 satoshis)

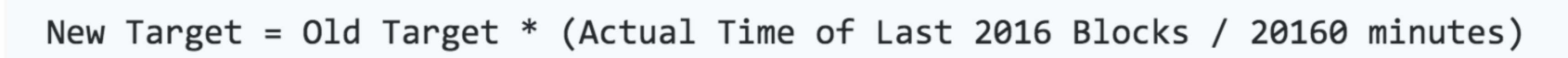
src/chainparams.cpp



```
// bitcoin/src/chainparams.cpp
class CMainParams : public CChainParams {
public:
    CMainParams() {
        strNetworkID = "main";
        consensus.nSubsidyHalvingInterval = 210000;
        ...
        consensus.nPowTargetTimespan = 14 * 24 * 60 * 60; // two weeks
        consensus.nPowTargetSpacing = 10 * 60;
        ...
        consensus.nMinerConfirmationWindow = 2016; // nPowTargetTimespan
                                                      nPowTargetSpacing
        ...
    }
};
```

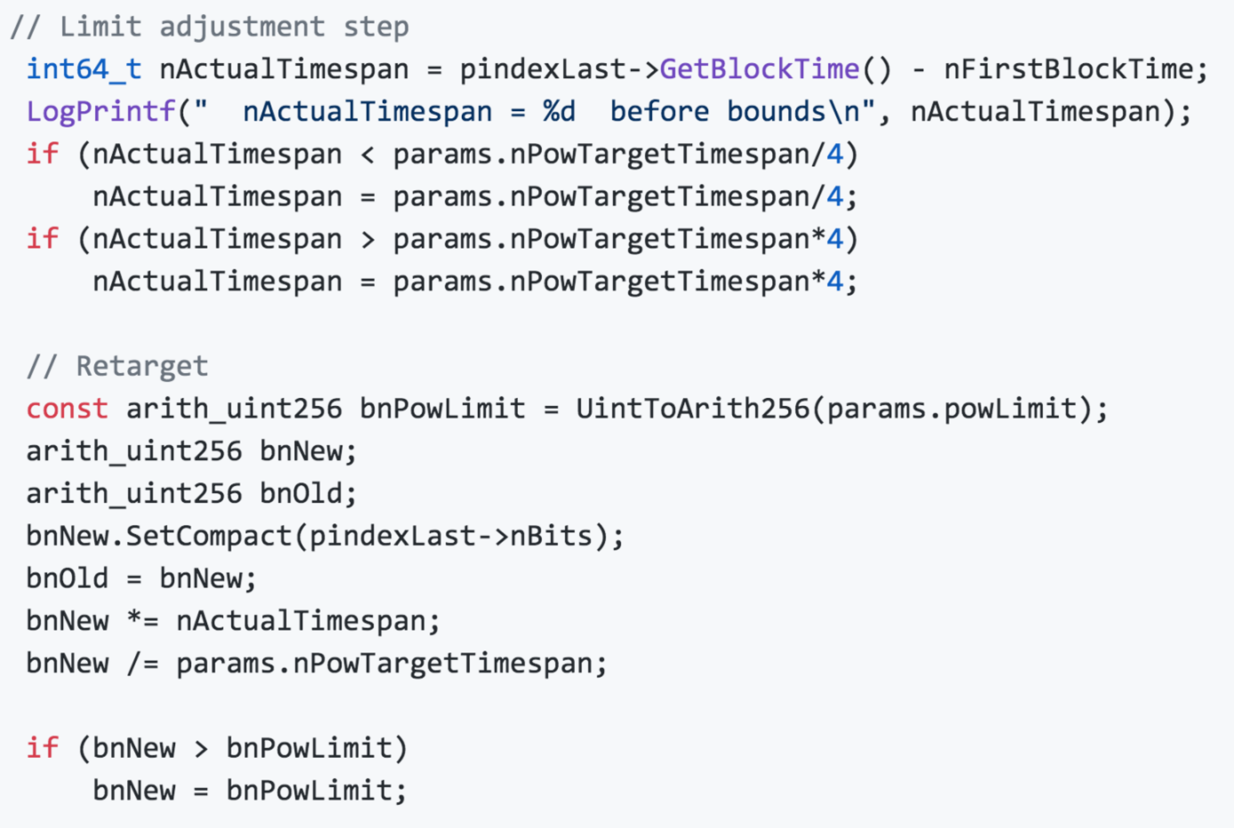
Retargeting to Adjust Difficulty

- Bitcoin’s blocks are generated every 10 minutes on average.
- Mining power increases over time.
- The target is set so that the current mining power will result in a 10-minute block interval.
- Every 2016 blocks, all nodes retarget the Proof-of-Work.



```
New Target = Old Target * (Actual Time of Last 2016 Blocks / 20160 minutes)
```

Retargeting to Adjust Difficulty



```
// Limit adjustment step
int64_t nActualTimespan = pindexLast->GetBlockTime() - nFirstBlockTime;
LogPrintf("  nActualTimespan = %d  before bounds\n", nActualTimespan);
if (nActualTimespan < params.nPowTargetTimespan/4)
    nActualTimespan = params.nPowTargetTimespan/4;
if (nActualTimespan > params.nPowTargetTimespan*4)
    nActualTimespan = params.nPowTargetTimespan*4;

// Retarget
const arith_uint256 bnPowLimit = UintToArith256(params.powLimit);
arith_uint256 bnNew;
arith_uint256 bnOld;
bnNew.SetCompact(pindexLast->nBits);
bnOld = bnNew;
bnNew *= nActualTimespan;
bnNew /= params.nPowTargetTimespan;

if (bnNew > bnPowLimit)
    bnNew = bnPowLimit;
```

Validating a New Block

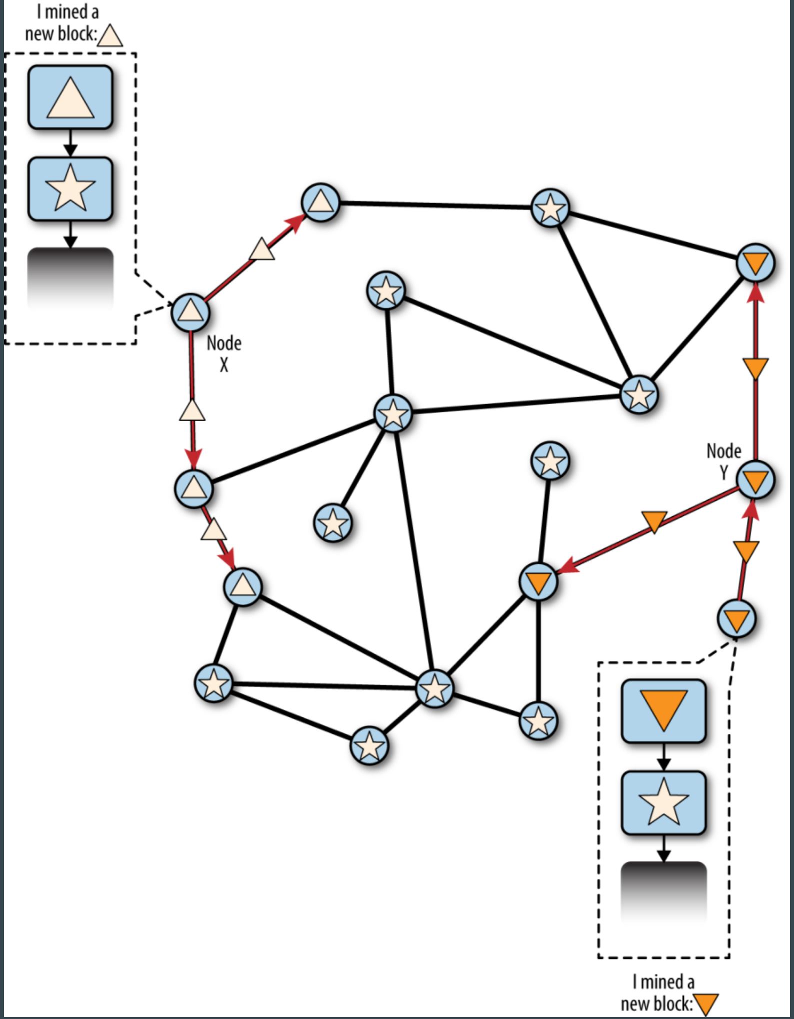
- The block data structure is syntactically valid.
- The block header hash is less than the target (enforces the Proof-of-Work).
- The block timestamp is less than two hours in the future (allowing for time errors)
- The block size is within acceptable limits.
- The first transaction (and only the first) is a coinbase transaction.
- ...

Assembling and Selecting Chains of Blocks

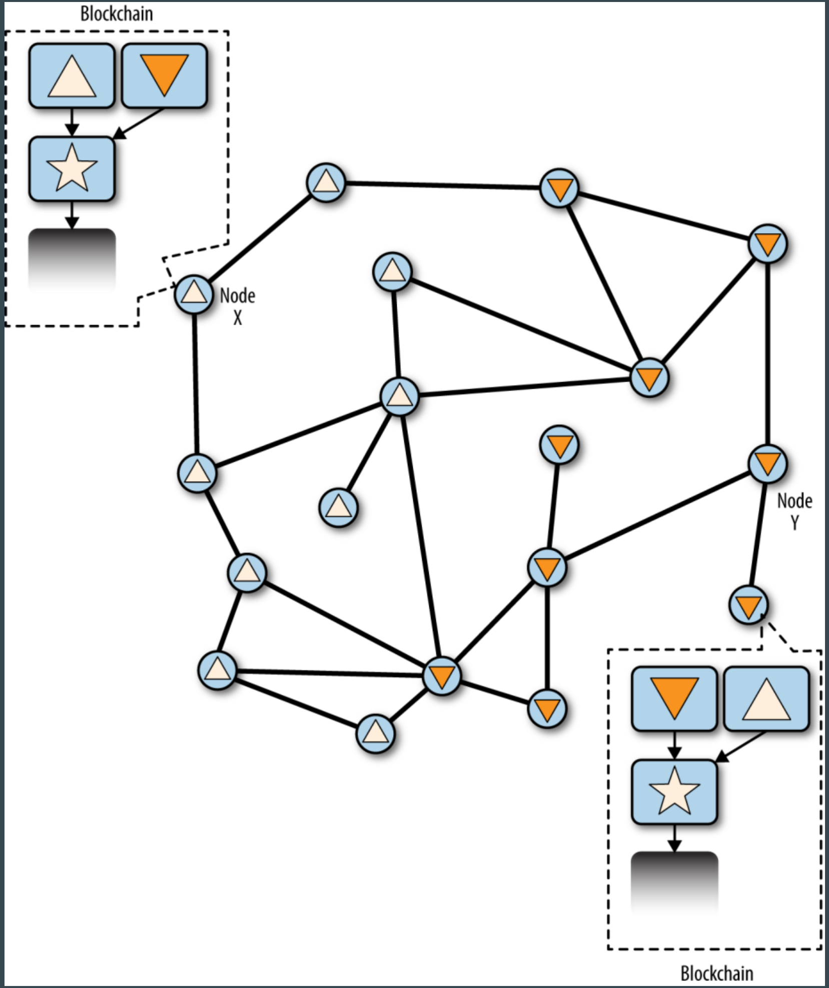
- Nodes maintain 3 sets of blocks :
    - Connected to the blockchain.
    - Form branches off the main blockchain (secondary chains).
    - Blocks that do not have a parent in the known chains (orphans).
        - If a valid block is received and no parent is found in the existing chains, that block is considered an “orphan.”
        - Two blocks that were mined in a short time are received in reverse order.
        - Orphan blocks are saved in the orphan block pool where they will stay until their parent is received.
        - Once the parent is received and linked into the existing chains, the orphan can be pulled out of the orphan pool and linked to the parent.
- The main chain at any time is whichever valid chain of blocks that has the most cumulative Proof-of-Work associated with it.

Block Forks

- A fork occurs whenever there are two candidate blocks competing to form the longest blockchain.
- Two miners solve the Proof-of-Work algorithm within a short period of time from each other.
- As both miners discover a solution for their respective candidate blocks, they immediately broadcast their own winning block to their immediate neighbors who begin propagating the block across the network.
- Each node that receives a valid block will incorporate it into its blockchain, extending the blockchain by one block. If that node later sees another candidate block extending the same parent, it connects the second candidate on a secondary chain.
- Some nodes will see one candidate block first, while other nodes will see the other candidate block and two competing versions of the blockchain will emerge.

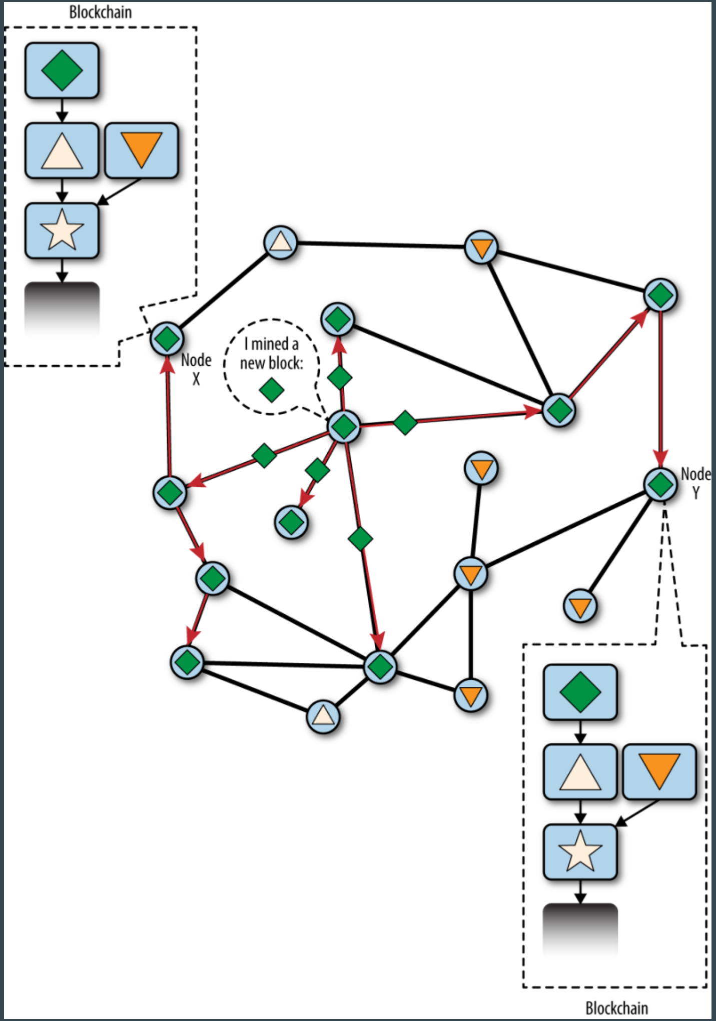


- Two miners (Node X and Node Y) who mine two different blocks almost simultaneously.
- Both of these blocks are children of the star block, and extend the chain by building on top of the star block.
- One is visualized as a triangle block originating from Node X, and the other is shown as an upside-down triangle block originating from Node Y.

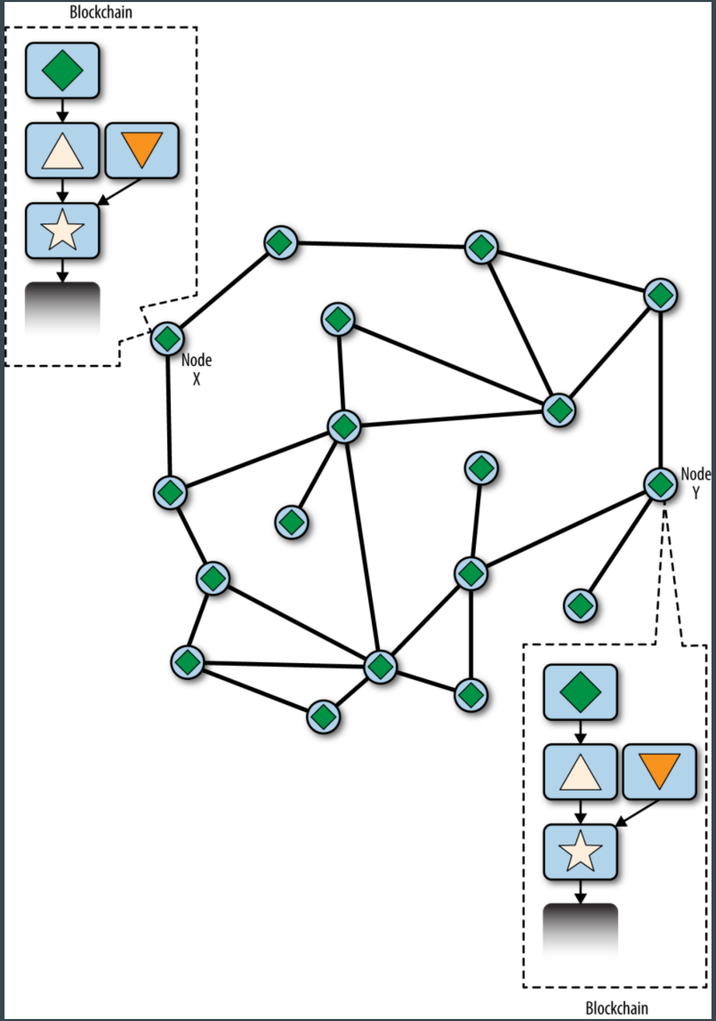


- Node X : △
- Node Y : ▼

- Node X and Y both construct a blockchain based on their own perspectives of the sequence of events.
- Neither side is “correct” or “incorrect”.
- Only in hindsight will one prevail, based on how these two competing chains are extended by additional work.



- It is likely that one set of miners will find a solution and propagate it before the other set of miners have found any solutions.
- Miners building on top of “triangle” find a new block “rhombus” that extends the chain. They immediately propagate this new block and the entire network sees it as a valid solution.
- The chain star-triangle-rhombus is now longer.

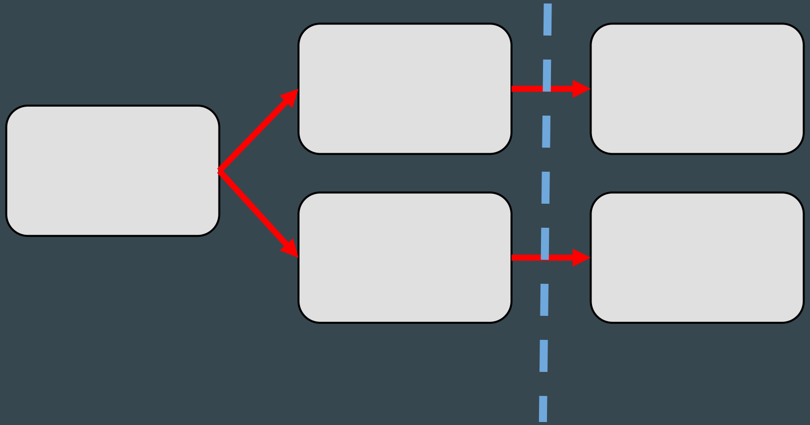


- Miner working on extending the chain “star-upside-down-triangle” will now stop that work because their candidate block is an “orphan”, as its parent “upside-down-triangle” is no longer on the longest chain.

- The transactions within “upside-down-triangle” that are not within “triangle” are re-inserted in the mempool for inclusion in the next block.

Block Forks

- It is theoretically possible for a fork to extend to two blocks, if two blocks are found almost simultaneously by miners on opposite “sides” of a previous fork.
- However, the chance of that happening is very low. Whereas a one-block fork might occur every day, a two-block fork occurs at most once every few weeks.



Mining Pools

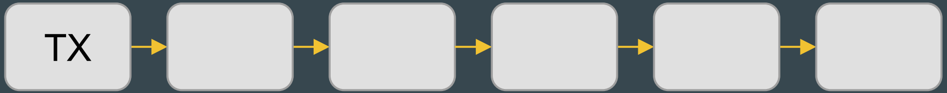
- Pooling hash power, sharing rewards
- Mining pool sets a higher target (lower difficulty) for earning a share.
- When someone in the pool mines a block, the reward is earned by the pool and then shared with all miners in proportion to the number of shares they contributed to the effort.

P2P Mining Pools

- Why need P2P ?
  - Pool operator might direct the pool effort to double-spend transactions or invalidate blocks.
  - single-point-of-failure : server down or slowed by a denial-of-service attack
  - P2Pool makes the mining ecosystem diversified. => makes bitcoin more robust overall
- P2Pool is a decentralized Bitcoin mining pool that works by creating a peer-to-peer network of miner nodes.
  - P2Pool creates a new blockchain in which the difficulty is adjusted so a new block is found every 30 seconds.
  - Blocks that get in to the P2Pool are the same blocks that would get into the Bitcoin blockchain.
  - P2Pool users must run a full Bitcoin node

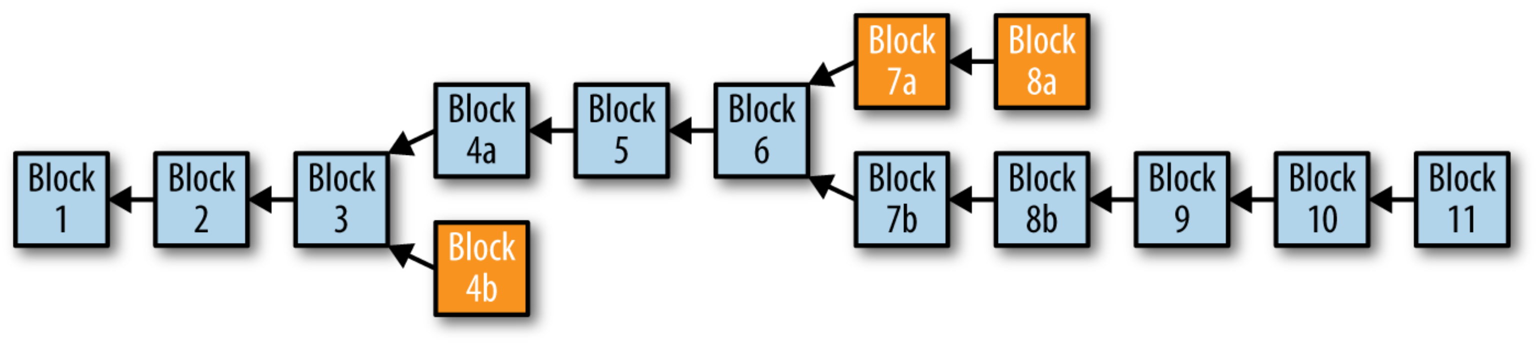
Consensus Attacks

- 51% attack
  - Majority of the total hashing power of the network.
  - Double-spend : can only be done on the attacker’s own transactions (need signature)
  - Denial-of-service
- What consensus attacks can’t do:
  - Consensus attacks also do not affect the security of the private keys and signing algorithm (ECDSA).
  - Steal or send bitcoin without signature.
- Six confirmations considered safe.



Hard Fork

- Block fork mentioned earlier isn't a hard fork.
- A change in the consensus rules. (not forward-compatible)
- Network doesn't reconverge onto a single chain => evolve independently



- 80% - 20% split (new, old)
  - Mining power 20% ↓ => every block 10 min → 12.5 min => target difficulty 20% ↓

Soft Fork

- In practice, a soft fork is not a fork at all.
- A soft fork is a forward-compatible change to the consensus rules that allows unupgraded clients to continue to operate in consensus with the new rules.
- Transactions and blocks created under the new rules must be valid under the old rules too.
- The new rules can only limit what is valid.