the Bitcoin network
so far we've talked about:
signatures
mining and blocks
transactions and scripts
... now to put it all together
recap: signatures
public / private keys
private key can sign() a message
can verify(public key, message, sig)
useful for proving identity, ownership. Better than paper signatures!
recap: mining and blocks
cchange a nonce, hash a bunch of
times, get a low output. Proves work
Include the previous data as part of your input, and you make a chain of work -- a blockchain
## recap: txs and scripts

Transactions have inputs and outputs

<table>
<thead>
<tr>
<th>inputs</th>
<th>outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>txid:index (36B)</td>
<td>script (25B)</td>
</tr>
<tr>
<td>signature (100B)</td>
<td>amount (8B)</td>
</tr>
<tr>
<td>txid:index</td>
<td>script (pubkey)</td>
</tr>
<tr>
<td>signature</td>
<td>amount</td>
</tr>
</tbody>
</table>
**recap: txs and scripts**

Inputs point to old outputs and have signatures.

Outputs have scripts and coin amounts.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>txid:index (36B)</td>
<td>script (25B)</td>
</tr>
<tr>
<td>signature (100B)</td>
<td>amount (8B)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>txid:index</td>
<td>script (pubkey)</td>
</tr>
<tr>
<td>signature</td>
<td>amount</td>
</tr>
</tbody>
</table>
tx mining process
users make txs, sign, broadcast
someone takes all the txs, puts them in a block, and does work
those txs are now "confirmed", and the next block can be built
tx mining: header
the block header is the message which must satisfy the proof of work
Headers have a hash of the txs in the block
Really it's the headers that make a chain, not the blocks. Headerchain.
tx mining: header
headers are 80 bytes; similar to pset02 blocks
Main components are
prev hash, merkle root, nonce
merkle root recap
Hash in a binary tree
Same level of commitment as $h(0,1,2,3)$
## header fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>4B</td>
<td>indicates block version</td>
</tr>
<tr>
<td>prev hash</td>
<td>32B</td>
<td>version</td>
</tr>
<tr>
<td>merkle root</td>
<td>32B</td>
<td>Was used for fork signalling; future use</td>
</tr>
<tr>
<td>time</td>
<td>4B</td>
<td></td>
</tr>
<tr>
<td>diff</td>
<td>4B</td>
<td>future use</td>
</tr>
<tr>
<td>nonce</td>
<td>4B</td>
<td>unclear</td>
</tr>
</tbody>
</table>
header fields

version  4B  hash of previous block
prev hash  32B  block
merkle root  32B

time  4B

diff  4B

nonce  4B
header fields

version 4B  hash of all
prev hash 32B  transactions in
merkle root 32B  the block
time 4B
diff 4B
nonce 4B
<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>4B</td>
<td>Unix time</td>
</tr>
<tr>
<td>prev hash</td>
<td>32B</td>
<td>(seconds since 1970) of claimed block creation</td>
</tr>
<tr>
<td>merkle root</td>
<td>32B</td>
<td></td>
</tr>
<tr>
<td>time</td>
<td>4B</td>
<td></td>
</tr>
<tr>
<td>diff</td>
<td>4B</td>
<td></td>
</tr>
<tr>
<td>nonce</td>
<td>4B</td>
<td>(can be before previous block's time!)</td>
</tr>
<tr>
<td>header fields</td>
<td></td>
<td>description</td>
</tr>
<tr>
<td>------------------------</td>
<td>---</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>version</td>
<td>4B</td>
<td>PoW target in a weird floating point format</td>
</tr>
<tr>
<td>prev hash</td>
<td>32B</td>
<td>weird floating point format</td>
</tr>
<tr>
<td>merkle root</td>
<td>32B</td>
<td>point format</td>
</tr>
<tr>
<td>time</td>
<td>4B</td>
<td></td>
</tr>
<tr>
<td>diff</td>
<td>4B</td>
<td>pretty much useless as can be computed anyway</td>
</tr>
<tr>
<td>nonce</td>
<td>4B</td>
<td></td>
</tr>
<tr>
<td>Field</td>
<td>Size</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>version</td>
<td>4B</td>
<td>nonce - anything goes here</td>
</tr>
<tr>
<td>prev hash</td>
<td>32B</td>
<td>goes here</td>
</tr>
<tr>
<td>merkle root</td>
<td>32B</td>
<td>but there's a problem...</td>
</tr>
<tr>
<td>time</td>
<td>4B</td>
<td></td>
</tr>
<tr>
<td>diff</td>
<td>4B</td>
<td></td>
</tr>
<tr>
<td>nonce</td>
<td>4B</td>
<td></td>
</tr>
<tr>
<td>header fields</td>
<td>size</td>
<td>contents</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>version</td>
<td>4B</td>
<td>nonce - anything</td>
</tr>
<tr>
<td>prev hash</td>
<td>32B</td>
<td>goes here</td>
</tr>
<tr>
<td>merkle root</td>
<td>32B</td>
<td>but there's a problem...</td>
</tr>
<tr>
<td>time</td>
<td>4B</td>
<td></td>
</tr>
<tr>
<td>diff</td>
<td>4B</td>
<td></td>
</tr>
<tr>
<td>nonce</td>
<td>4B</td>
<td>too small!</td>
</tr>
</tbody>
</table>
# header fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Size</th>
<th>Possible Nonces</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>4B</td>
<td>$2^{32}$</td>
</tr>
<tr>
<td>prev hash</td>
<td>32B</td>
<td>nonces</td>
</tr>
<tr>
<td>merkle root</td>
<td>32B</td>
<td></td>
</tr>
<tr>
<td>time</td>
<td>4B</td>
<td>But current</td>
</tr>
<tr>
<td>diff</td>
<td>4B</td>
<td>blocks need $2^{70}$</td>
</tr>
<tr>
<td>nonce</td>
<td>4B</td>
<td>work!</td>
</tr>
</tbody>
</table>
header fields

version   4B   adjust time
prev hash  32B
merkle root 32B  modify merkle root
time       4B
diff       4B
nonce      4B
tx order in block

tx0 is the coinbase tx:
generates new coins, and takes fees from all other txs in block

all other txs can be in any order, but can only spend outputs from previous txs
tx order in block
if txB spends an output of txA, then txA must come first in block ordering
this ensures linear verification of transactions can proceed
intermission

256 second break

prove work by moving body mass against force of gravity

work = f*d = m*g*h
sync process
I just downloaded bitcoin!
What's been going on for the last 9 years?
sync process
Download binary / compile code
Verify GPG signatures somehow...
Hardcoded DNS seeds to find peers
connect, ask for headers
download & verify 500K headers
sync process
Get the header chain first - quick
takes under a minute with good
connections
verify all the work before any
signatures
sync IBD

After headers, Initial Block Download (IBD)

Request blocks from peers, match tx list to merkle root in header, process each tx in order
sync IBD
Delete all input txos
sync IBD
Delete all input txos
Add output txos
sync IBD
Do this ~300M times
Downloads 170GB
End result:
55M txos, ~3.2GB
pruning
By default, store all 500K blocks
Can serve to others who need to IBD
But can "prune" / delete blocks after IBD with no loss of security
Downside?
pruning
By default, store all 500M blocks
Can serve to others who need to IBD
But can "prune" / delete blocks after IBD with no loss of security
Downside? Not everyone can prune
blockchain data

What does it store?

banlist.dat
chainstate
peers.dat
wallet.dat
bitcoin.conf
blocks
database
debug.log
mempool.dat
## blockchain data

### What does it store?

<table>
<thead>
<tr>
<th>File</th>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>banlist.dat</td>
<td>1.8K</td>
<td>bad nodes</td>
</tr>
<tr>
<td>chainstate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>peers.dat</td>
<td>4.0M</td>
<td>good nodes</td>
</tr>
<tr>
<td>wallet.dat</td>
<td>1.4M</td>
<td>my precious</td>
</tr>
<tr>
<td>bitcoin.conf</td>
<td>144</td>
<td>config file</td>
</tr>
<tr>
<td>blocks</td>
<td></td>
<td></td>
</tr>
<tr>
<td>database</td>
<td></td>
<td></td>
</tr>
<tr>
<td>debug.log</td>
<td>11M</td>
<td>log file, rotates</td>
</tr>
<tr>
<td>mempool.dat</td>
<td>20M</td>
<td>more like diskpool</td>
</tr>
</tbody>
</table>
blockchain data

What does it store?

banlist.dat  1.8K
chainstate  3.0G  utxo set
peers.dat    4.0M
wallet.dat   1.4M
bitcoin.conf 144
blocks       183G  all the blocks
database     80K   ? nothing?
debug.log    11M
mempool.dat  20M
blockchain as database
186GB, but a really crummy database
remember tx 9e95c3c3c96f57527cdc649550bf8e92892f7651f718d846033798ae6333b0c3
from back in 2014?
blockchain as database
186GB, but a really crummy database

remember tx 9e95c3c3c96f57527cdc649550bf8e92892f7651f718d846033798ae333b0c3

from back in 2014?

No. It's somewhere in the blocks folder but I don't know where.

It's not in chainstate
blockchain as database
how about output

02b1500a0f3b059819dd923f1c78bacc0a3de303fc51836ce7f46a3206b29ba7:0

it's an op_return output, can you tell me what the data is?
Blockchain as database

How about output

02b1500a0f3b059819dd923f1c78bacc0a3de303fc51836ce7f46a3206b29ba7:0

It's an op_return output, can you tell me what the data is?

Nope! op_return outputs don't get stored in the chainstate.
blockchain as database

Hey I have a pubkey with hash
1d493f9536c692d096536ba9d1c081feabd7ccf3

how many coins do I have? How many outputs?
blockchain as database

Hey I have a pubkey with hash 1d493f9536c692d096536ba9d1c081feabd7ccf3

how many coins do I have? How many outputs?

No idea! Gotta search through all of chainstate. Doesn't index based on PkScript, only txid:index
blockchain as database
how many coins does output
have?

7434e09a302eaa4e2e0826aea08c2cca282a8bfc606cb680aa1f3f331a7e4f69:1
blockchain as database
how many coins does output
have?

Lots! 239.99913132. It's in the utxo set because it hasn't been spent yet.
Can quickly find based on txid:index
blockchains are bad databases

Only keeps track of utxos, which is hard enough

Can add further indexes, but they take lots of space. Most common is "address index" so people can ask if they have any money.
blockchains are bad databases
DB queries not given to network peers
Network peers are scary, ban them if they act funny
Provide headers, blocks, txs, other nodes IPs
bad DB but good consensus
Everyone's got the same utxo set
Even though they all really want more utxos. Or to break the system. It seems to work.
pset02 update
Bunch of blocks mined
Pls reduce server queries; an 18.
address is doing 5+ TCP connections per second. Also GCE? 35.
Could use blocks here to start a coin...
For information about citing these materials or our Terms of Use, visit: https://ocw.mit.edu/terms.