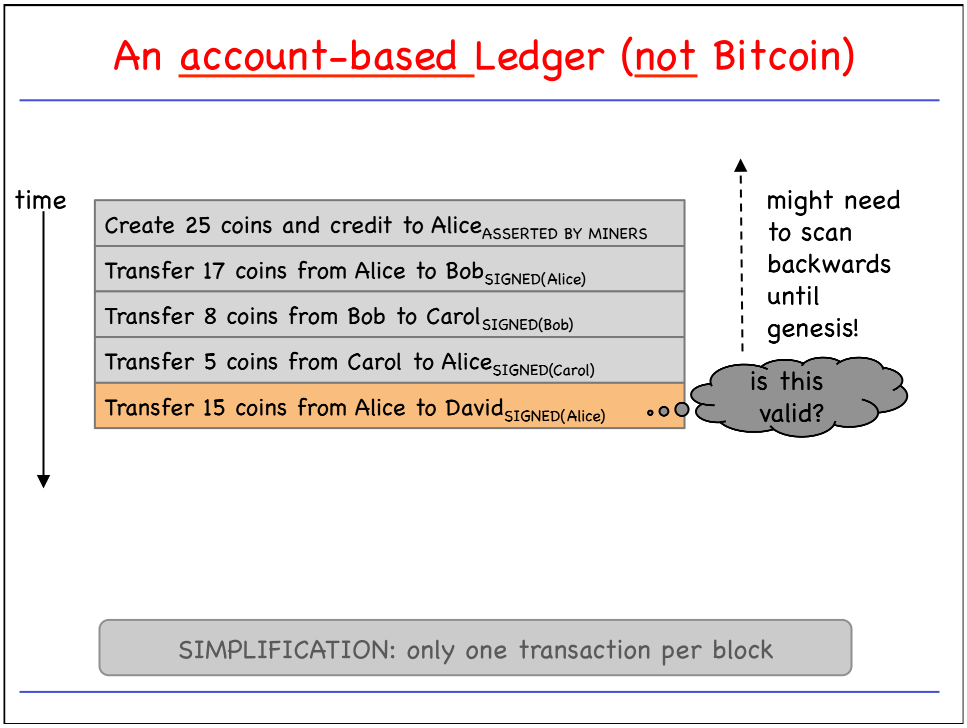
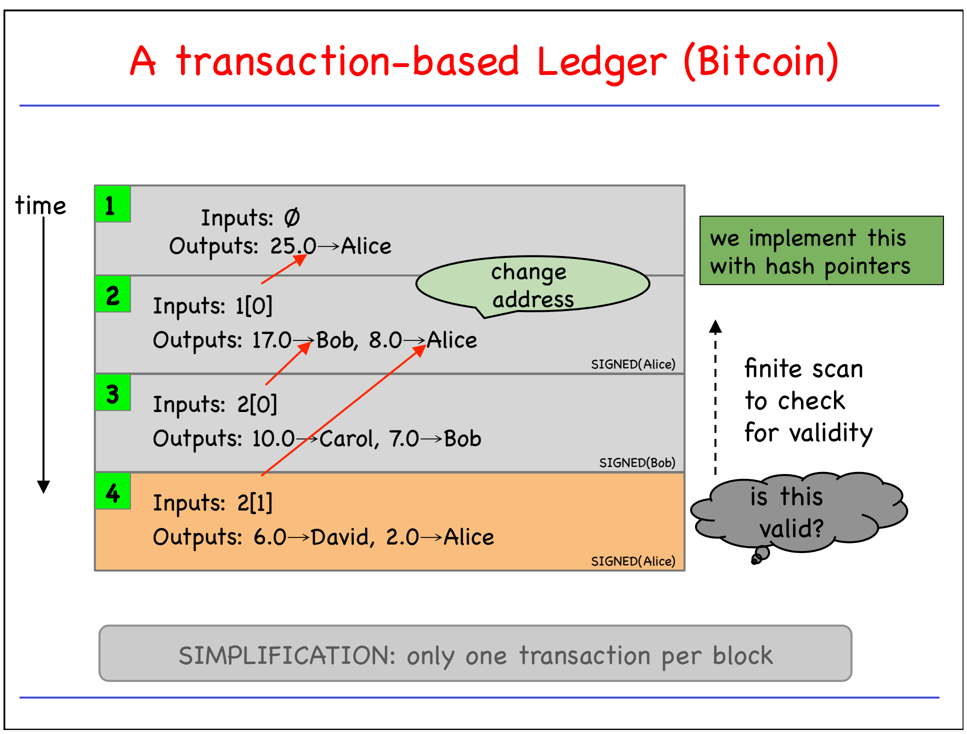
Mechanics of Bitcoin

- Bitcoin Transactions
- Bitcoin Scripts
- Applications of Bitcoin Scripts
- Bitcoin Blocks
- The Bitcoin Network
- Limitations and Improvements

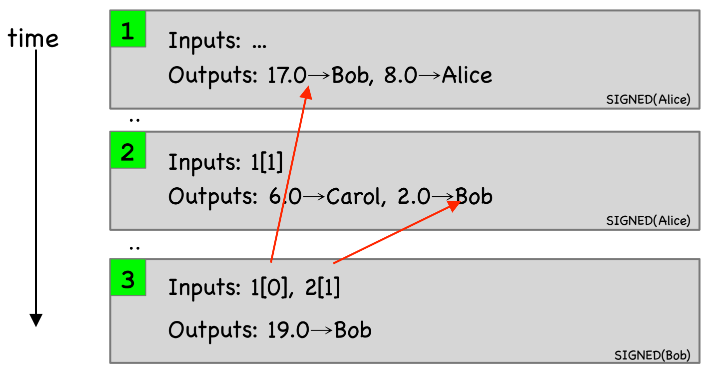
Mechanics of Bitcoin

- Bitcoin Transactions
- Bitcoin Scripts
- Applications of Bitcoin Scripts
- Bitcoin Blocks
- The Bitcoin Network
- Limitations and Improvements


An account-based Ledger (not Bitcoin)
time
Create 25 coins and credit to Alice ASSERTED BY MINERS
Transfer 17 coins from Alice to Bob SIGNED(Alice)
Transfer 8 coins from Bob to Carol SIGNED(Bob)
Transfer 5 coins from Carol to Alice SIGNED(Carol)
Transfer 15 coins from Alice to David SIGNED(Alice)
might need to scan backwards until genesis!
is this valid?
SIMPLIFICATION: only one transaction per block

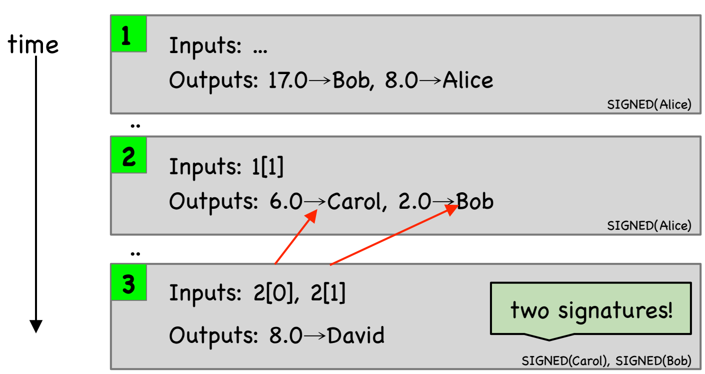

A transaction-based Ledger (Bitcoin)
time
1
Inputs: Ø
Outputs: 25.0→Alice
2
Inputs: 1[0]
Outputs: 17.0→Bob, 8.0→Alice
SIGNED(Alice)
change address
3
Inputs: 2[0]
Outputs: 10.0→Carol, 7.0→Bob
SIGNED(Bob)
4
Inputs: 2[1]
Outputs: 6.0→David, 2.0→Alice
SIGNED(Alice)
we implement this with hash pointers
finite scan to check for validity
is this valid?
SIMPLIFICATION: only one transaction per block

Merging Value



SIMPLIFICATION: only one transaction per block

Joint Payments



SIMPLIFICATION: only one transaction per block

The Real Deal: a Bitcoin Transaction

metadata

input(s)

output(s)

```
{
    "hash":"5a42590fbe0a90ee8e8747244d6c84f0db1a3a24e8f1b95b10c9e050990b8b6b",
     "ver":1,
     "vin_sz":2,
     "vout_sz":1,
     "lock_time":0,
     "size":404,
     "in":[
       {
         "prev_out":{
           "hash":"3be4ac9728a0823cf5e2deb2e86fc0bd2aa503a91d307b42ba76117d79280260",
           "n":0
         },
        "scriptSig":"30440..."
       },
       {
         "prev_out":{
           "hash":"7508e6ab259b4df0fd5147bab0c949d81473db4518f81afc5c3f52f91ff6b34e",
           "n":0
         },
         "scriptSig":"3f3a4ce81...."
       }
     ],
     "out":[
       {
         "value":"10.12287097",
         "scriptPubKey":"OP_DUP OP_HASH160 69e02e18b5705a05dd6b28ed517716c894b3d42e
OP_EQUALVERIFY OP_CHECKSIG"
       }
     ]
}
```

The Real Deal: Transaction Metadata

transaction hash	"hash":"5a42590...b8b6b",
housekeeping	"ver":1, "vin_sz":2, "vout_sz":1,
"not valid before"	"lock_time":0,
housekeeping	"size":404,

```
{
    "hash":"5a42590...b8b6b",
     "ver":1,
     "vin_sz":2,
     "vout_sz":1,
     "lock_time":0,
     "size":404,
...
}
```

more on lock_time later...

The Real Deal: Transaction Inputs

previous transaction

signature

(more inputs)

```
"in":[
    {
      "prev_out":{
        "hash":"3be4...80260",
        "n":0
      },
      "scriptSig":"30440....3f3a4ce81"
    },
   ...
   ],
```

The Real Deal: Transaction Outputs

output value

recipient address??

(more outputs)

```
"out":[
  {
    "value":"10.12287097",
    "scriptPubKey":"OP_DUP OP_HASH160 69e...3d42e
  OP_EQUALVERIFY OP_CHECKSIG"
  },
  ...
 ]
```

more on this soon...

Mechanics of Bitcoin

- Bitcoin Transactions
- **Bitcoin Scripts**
- Applications of Bitcoin Scripts
- Bitcoin Blocks
- The Bitcoin Network
- Limitations and Improvements

Output "Adresses" are really *Scripts*

```
OP_DUP
OP_HASH160
69e02e18...
OP_EQUALVERIFY OP_CHECKSIG
```

Input "Adresses" are *also* Scripts

Sig-script

```
30440220...
0467d2c9…
```

PubKey-script

```
OP_DUP
OP_HASH160
69e02e18...
OP_EQUALVERIFY OP_CHECKSIG
```

TO VERIFY: Concatenated script must execute completely with no errors

Why Scripts?!

Redeem previous transaction by signing with correct key

"This can be redeemed by a signature from the owner of address *X*"

Recall: address *X* is hash of public key

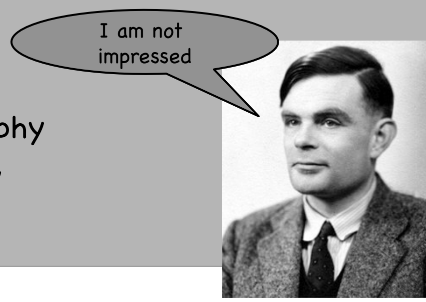
What is public key associated with *X*?!

"This can be redeemed by a public key that hashes to *X*, along with a signature from the owner of that public key"

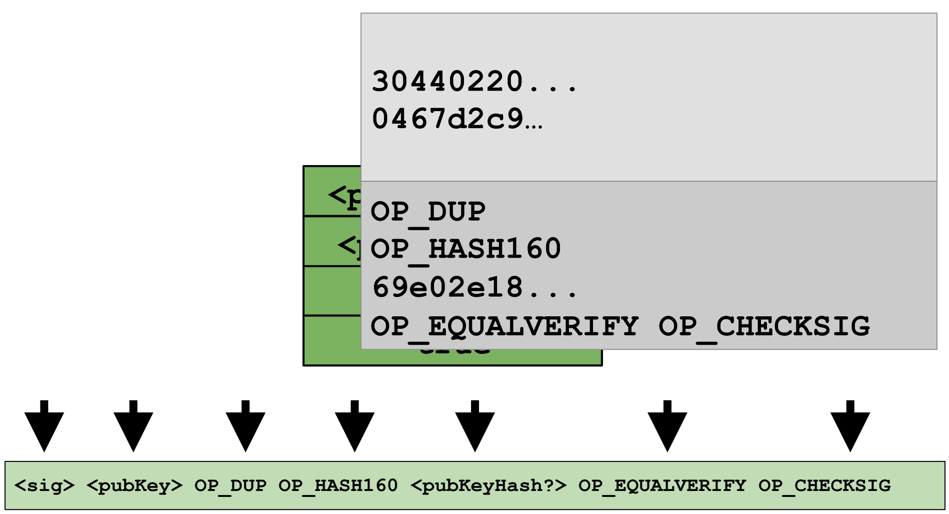
Bitcoin Scripting Language ("Script")

Design "goals":

- Built for Bitcoin (inspired by Forth)
- Simple, compact
- Stack-based
- No looping
- Support for cryptography
- Limits on time/memory
- Not Turing complete!



Bitcoin Script Execution Example



Bitcoin Script Instructions

256 opcodes total (15 disabled, 75 reserved)

- Arithmetic
- If/then
- Logic/data handling
- Crypto!

OP_DUP	Duplicates the top item on the stack
OP_HASH160	Hashes twice: first using SHA-256 and then RIPEMD-160
OP_EQUALVERIFY	Returns true if the inputs are equal. Returns false and marks the transaction as invalid if they are unequal
OP_CHECKSIG	Checks that the input signature is a valid signature using the input public key for the hash of the current transaction
OP_CHECKMULTISIG	Checks that the *k* signatures on the transaction are valid signatures from *k* of the specified public keys.

OP_CHECKMULTISIG

Built-in support for joint signatures

Specify *n* public keys

Specify *t*

Verification requires *t* signatures

Incidentally: There is a bug in the **multisig** implementation.
Extra data value popped from the stack and ignored

Scripts in Practice (as of 2015)

Theory: Scripts let us specify arbitrary conditions that must be satisfied to spend coins.

Q: Is any of this used in practice?

- 99.9% are simple signature checks
- ~0.01% are **MULTISIG**
- ~0.01% are Pay-to-Script-Hash
- Remainder are errors, proof-of-burn

More on this soon

Most nodes whitelist known scripts

Proof-of-Burn

this script can never be redeemed ☹

```
OP_RETURN
<arbitrary data>
```

Uses for Proof-of-Burn:
- Destroy coins and transfer them to alternative currency
- Add arbitrary data to block chain

Should Senders specify Scripts?

Pay-to-Script-Hash (P2SH) Workflow

Bob

- creates a redeem script with whatever script he wants
- hashes the redeem script
- sends redeem script hash to Alice.

Alice

- creates a P2SH-style output containing Bob's redeem script hash.

When Bob wants to spend the output

- provides his signature along with the redeem script in the signature script.

P2P network then

- ensures the redeem script hashes to the same value as the script hash Alice put in her output;
- it then processes the redeem script exactly as it would if it were the primary pubkey script,
- letting Bob spend the output if the redeem script does not return false.

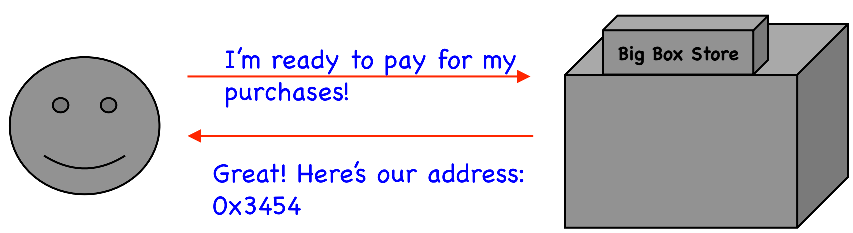
Solution: Use the Hash of Redemption Script

<signature>

<pubkey>
OP_CHECKSIG

"Pay to Script Hash"

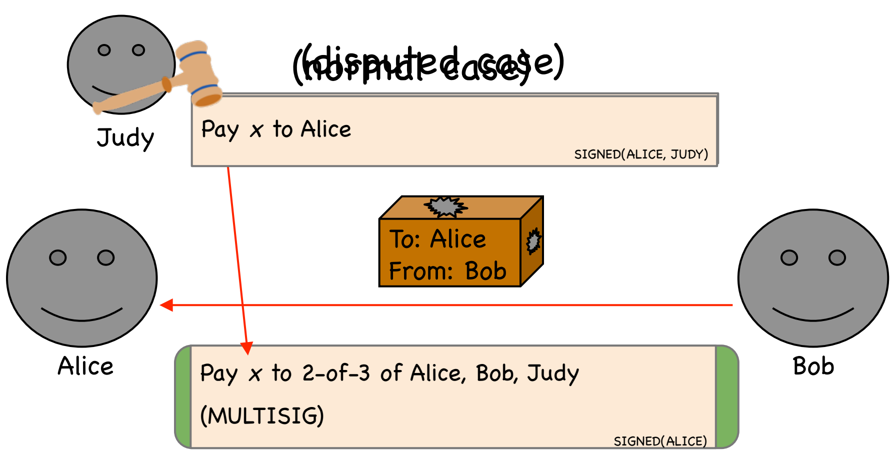
Pay-to-Script-Hash



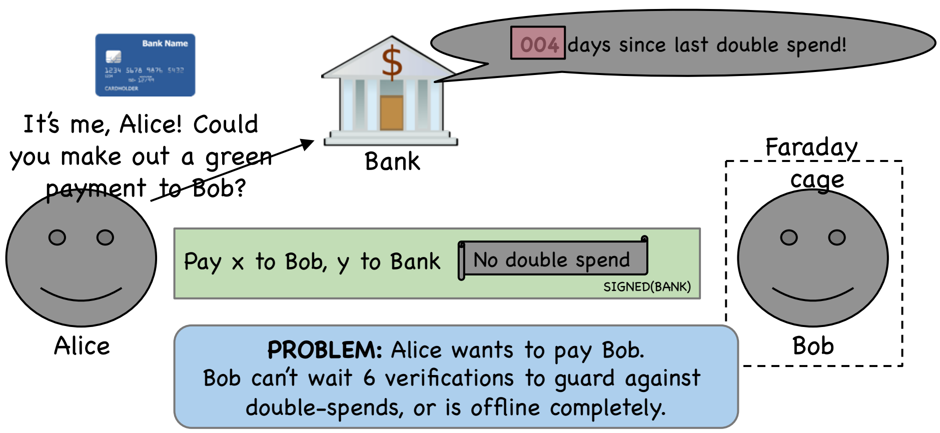
Mechanics of Bitcoin

- Bitcoin Transactions
- Bitcoin Scripts
- **Applications of Bitcoin Scripts**
- Bitcoin Blocks
- The Bitcoin Network
- Limitations and Improvements

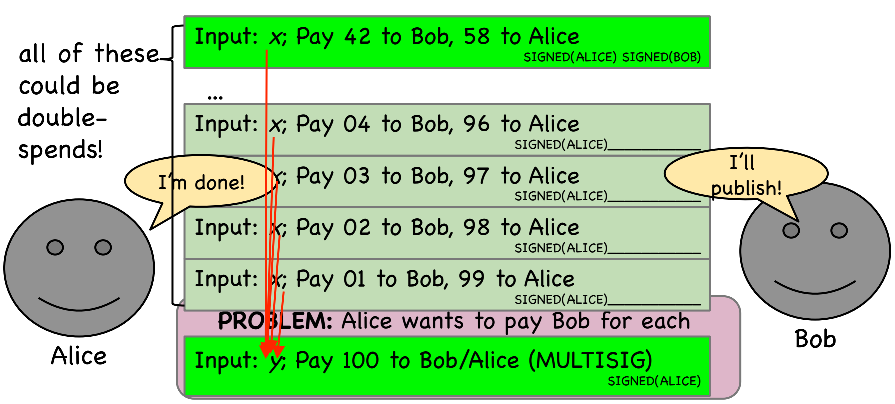
Example 1: Escrow Transactions



Example 2: Green Addresses



Example 3: Efficient Micro-Payments



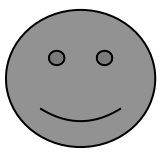
Example 3: Efficient Micro-Payments

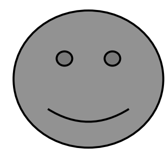
What if Bob never signs??

Input: *x*; Pay 42 to Bob, 58 to Alice
SIGNED(ALICE)___________

Alice demands a timed refund transaction before starting

Input: *x*; Pay 100 to Alice, LOCK until time *t*
SIGNED(ALICE) SIGNED(BOB)


Alice


Bob

Input: *y*; Pay 100 to Bob/Alice (MULTISIG)
SIGNED(ALICE)

lock_time

```
{
    "hash":"5a42590...b8b6b",
     "ver":1,
     "vin_sz":2,
     "vout_sz":1,
     "lock_time":315415,
     "size":404,
...
}
```

Block index or real-world timestamp before which this transaction can't be published

More advanced Scripts

Multiplayer Lotteries

Coin-swapping Protocols

"Smart Contracts"

Mechanics of Bitcoin

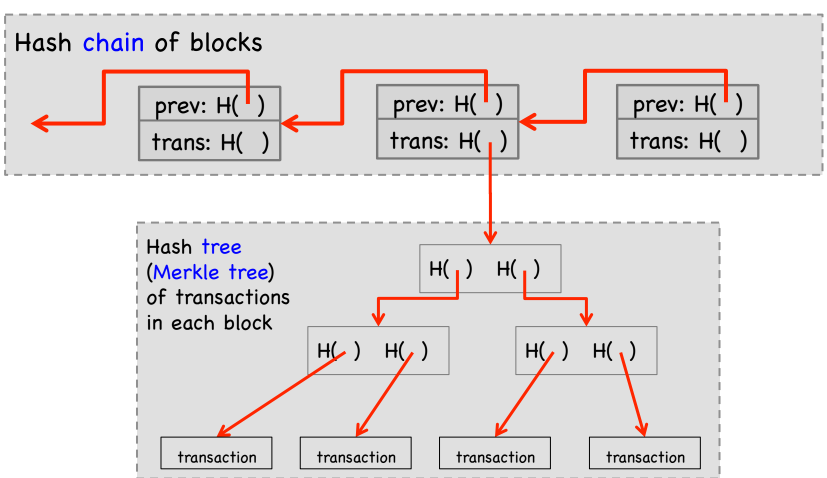
- Bitcoin Transactions
- Bitcoin Scripts
- Applications of Bitcoin Scripts
- Bitcoin Blocks
- The Bitcoin Network
- Limitations and Improvements

Bitcoin Blocks

Q: Why bundle transactions together?

1. Requiring consensus for each transaction separately would reduce transaction acceptance rate.
2. Hash-chain of blocks is much shorter.
3. Faster to verify history.

Bitcoin Block Structure



The Real Deal: a Bitcoin Block

block header

transaction data

```
{
  "hash":"00000000000000001aad2...",
  "ver":2,
  "prev_block":"00000000000000003043...",
  "time":1391279636,
  "bits":419558700,
  "nonce":459459841,
  "mrkl_root":"89776...",
  "n_tx":354,
  "size":181520,
  "tx":[
    ...
  ],
  "mrkl_tree":[
    "6bd5eb25...",
    ...
    "89776cdb..."
  ]
}
```

The Real Deal: a Bitcoin Block Header

```
{
  "hash":"00000000000000001aad2...",
  "ver":2,
  "prev_block":"00000000000000003043...",
  "time":1391279636,
  "bits":419558700,
  "nonce":459459841,
  "mrkl_root":"89776...",
  ...
}
```

hash
timestamp
indication of difficulty
chosen nonce
root of trans. tree

hashed during mining

not hashed

coinbase Transaction

New coins are created with **coinbase** transaction:

- Single input and single output
- Does not redeem previous output
  - Hash pointer is null
- Output value is miner's revenue from block:
  - *output value = mining reward + transaction fees*
  - transaction fees come from all transactions in block
- Special **coinbase** parameter
  - contains arbitrary value

The Real Deal: coinbase Transaction

```
"in":[
      {
        "prev_out":{
        "hash":"000000.....0000000",
        "n":4294967295
      },
      "coinbase":"..."
      }
   ]
"out":[
      {
        "value":"25.03371419",
        "scriptPubKey":"OPDUP OPHASH160 ... "
      }
   ]
```

redeeming nothing

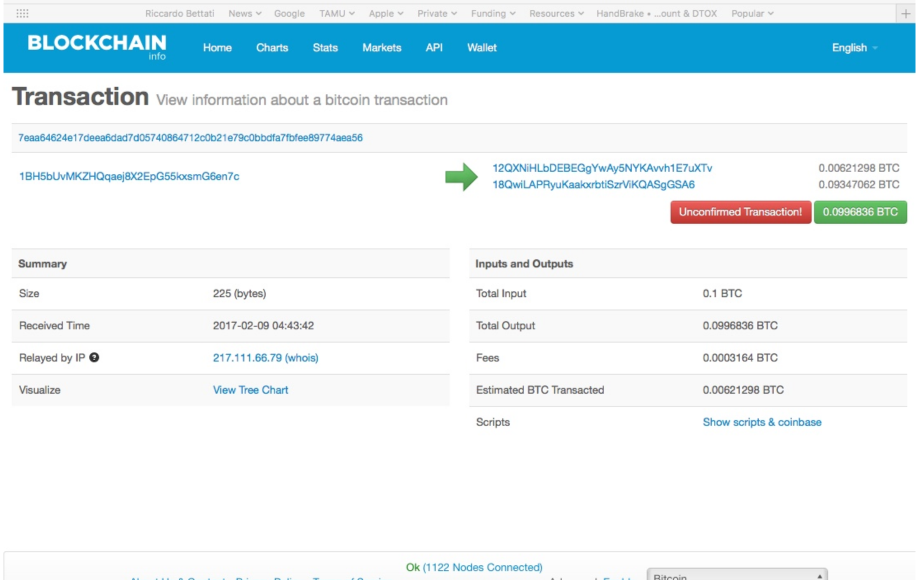
arbitrary

Null hash pointer

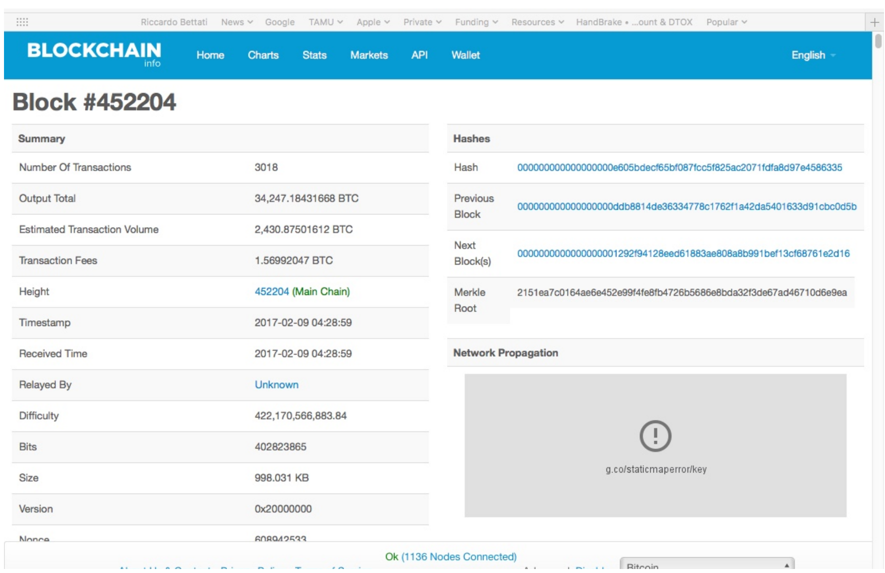
First ever coinbase parameter: "The Times 03/Jan/2009 Chancellor on brink of second bailout for banks"

block reward + transaction fees

See for yourself!



See for yourself!



Mechanics of Bitcoin

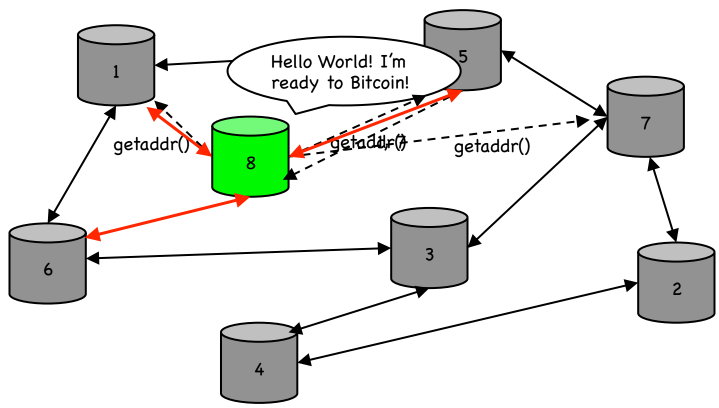
- Bitcoin Transactions
- Bitcoin Scripts
- Applications of Bitcoin Scripts
- Bitcoin Blocks
- The Bitcoin Network
- Limitations and Improvements

Bitcoin P2P Network

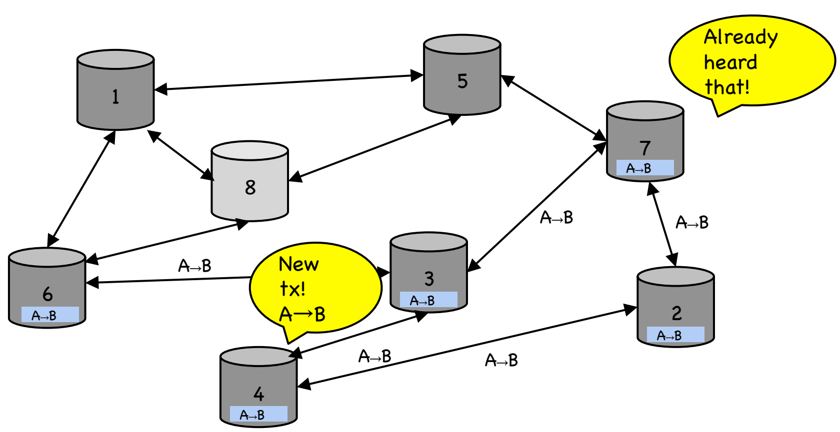
Participants can
- publish transactions
- insert transactions into block chain

The network:
- Ad-hoc protocol (runs on TCP port 8333)
- Ad-hoc network with random topology
- All nodes are equal
- New nodes can join at any time
- Forget non-responding nodes after 3 hr

Joining the Bitcoin P2P Network



Transaction Propagation (Flooding)

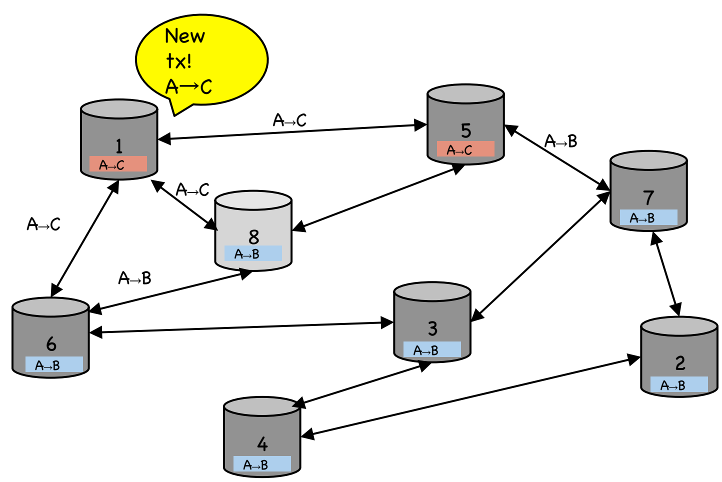


Should I relay a proposed Transaction?

- Transaction valid with current block chain
- (default) script matches a whitelist
  - Avoid unusual scripts
- Haven't seen before
  - Avoid infinite loops
- Doesn't conflict with others I've relayed
  - Avoid double-spends

Sanity checks only...
Some nodes may ignore them!

Nodes may differ on Transaction Pool

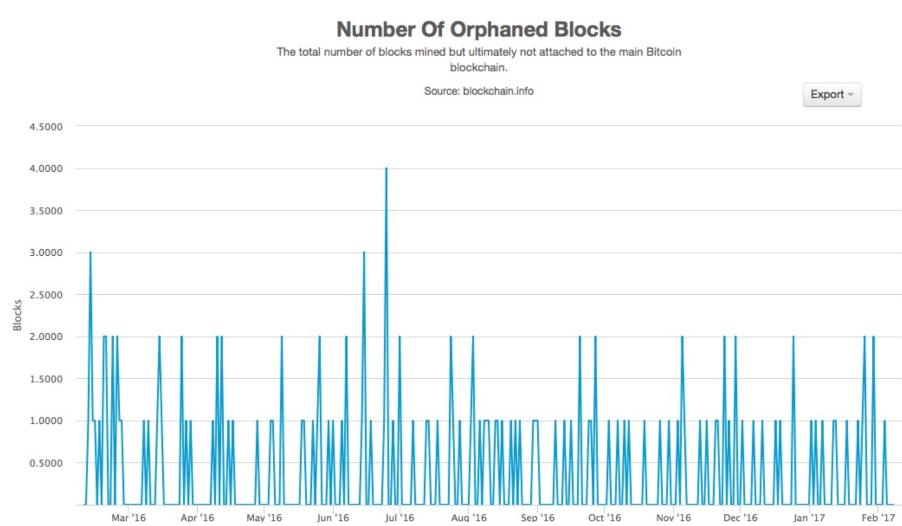


Race Conditions

Transactions or blocks may conflict

- This is called **"race condition"**
- Default behavior: accept what you hear first
- Tie broken by whoever mines next block
  - picks only one transaction/block
- Network position matters
- Miners may implement other logic!

Orphaned Blocks



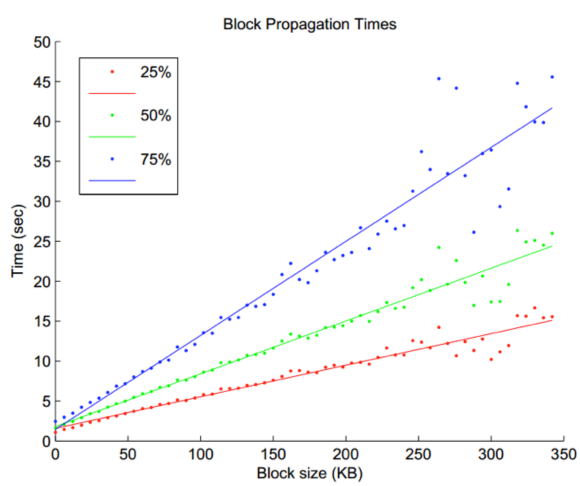
Block Propagation

Propagation of blocks is nearly identical:

Relay a new block when you hear it if:

1. Block meets the hash target
2. Block has all valid transactions
   - Run *all* scripts, even if you wouldn't relay
3. Block builds on current longest chain
   - Avoid forks

Latency of Flooding Algorithm



Source: Yonatan Sompolinsky and Aviv Zohar: "Accelerating Bitcoin's Transaction Processing" 2014

Size of the Network

Q: How big is the Network?

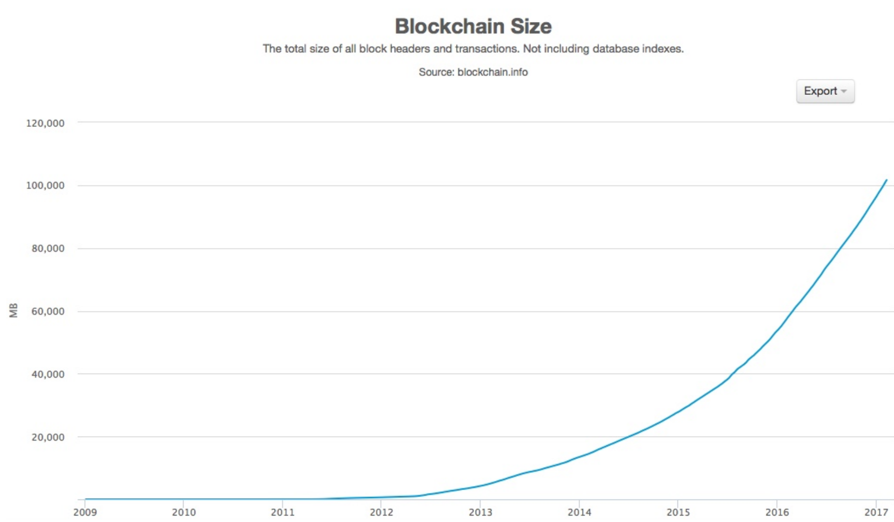
Impossible to measure exactly

- Estimates-up to 1M IP addresses/month
- Only about 5-10k "full nodes"
  - Permanently connected
  - Fully-validating
- This number may be dropping!

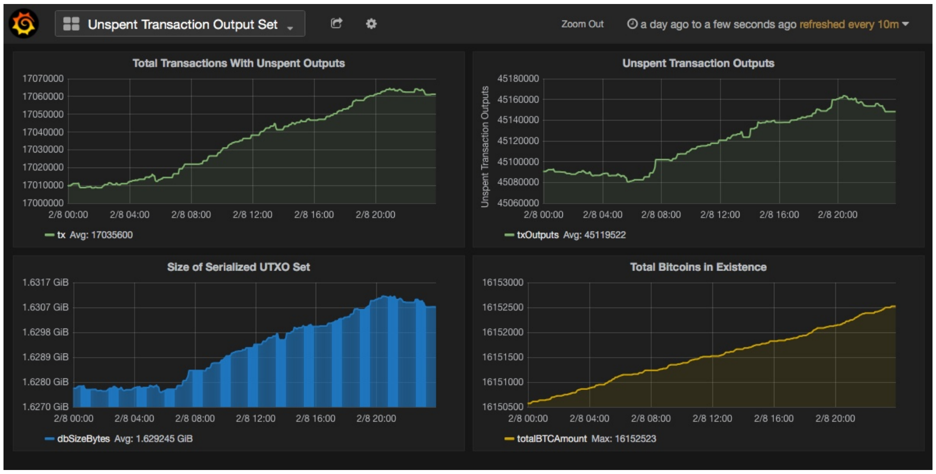
Fully-validating Nodes:

- Permanently connected
- Store entire block chain
- Hear and forward every node/ transaction

Storage Costs



Unspent Transaction Output fits in RAM



Thin/SPV Clients (not fully-validating)

Idea: don't store everything

– Store block headers only

Request transactions as needed

– To verify incoming payment

Trust fully-validating nodes

1000x cost savings!

Software Diversity

- About 90% of nodes run "Core Bitcoin" (C++)
  - Some are out of date versions
- Other implementations running successfully
  - BitcoinJ (Java)
  - Libbitcoin (C++)
  - btcd (Go)
- "Original Satoshi client"

Mechanics of Bitcoin

- Bitcoin Transactions
- Bitcoin Scripts
- Applications of Bitcoin Scripts
- Bitcoin Blocks
- The Bitcoin Network
- Limitations and Improvements

Hard-coded Limits in Bitcoin

- 10 min. average creation time per block
- 1 M bytes in a block
- 20,000 signature operations per block
- 100 M *satoshis* per bitcoin
- 23M total bitcoins maximum
- 50,25,12.5... bitcoin mining reward

These affect economic balance of power too much to change now

Throughput Limits in Bitcoin

Blocks are limited to 1 M bytes each (10 min)
With
at least 250 bytes/transaction
this gives about
7 transactions/sec!

Compare to:

- VISA: **2,000-10,000** transactions/sec
- PayPal: **50-100** transaction/sec

Cryptographic Limits in Bitcoin

1. Only 1 signature algorithm (ECDSA/P256)
2. Hard-coded hash functions

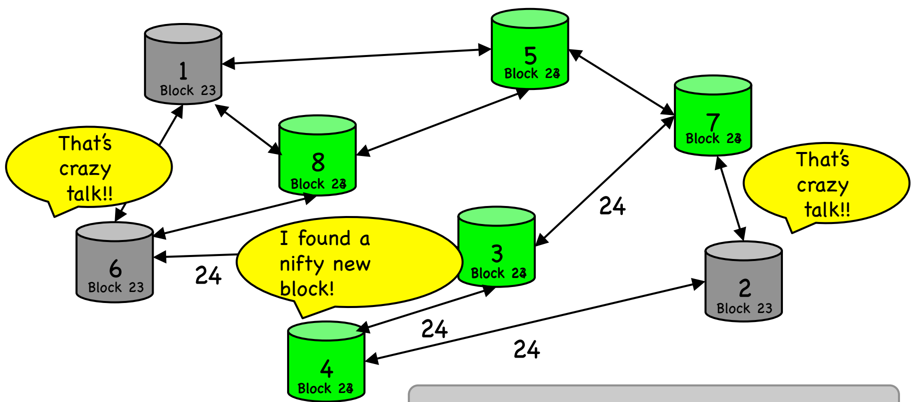
Crypto primitives might break by 2040…

Changing the Protocol: Hard vs. Soft Forks

Q: So, you want to change the protocol.
What to do about "old" nodes?

Hard Fork: Change introduces new features that were previously considered invalid.

"Hard-forking" Changes to Bitcoin



PROBLEM: Old nodes will never catch up

Changing the Protocol: Hard vs. Soft Forks

Q: So, you want to change the protocol.
What to do about "old" nodes?

Hard Fork: Change introduces new features that were previously considered invalid.

Soft Fork: Change introduces new features that make validation rules stricter.

Soft Forks

Observation: We can add new features that only *limit* the set of valid transactions.

- Need majority of nodes to enforce new rules.
- Old nodes will approve.

RISK: Old nodes might mine now-invalid blocks

Soft Fork Example: Pay-to-Script-Hash

<signature>
<<pubkey> OP_CHECKSIG>

OP_HASH160
<hash of redemption script>
OP_EQUAL

Old nodes will just approve the hash, not run the embedded script.

Soft Fork Possibilities

- New signature schemes
- Extra per-block metadata
  - Stricter formatting of the coinbase parameter
  - Add Merkle Tree of UTXOs (Unspent Transaction Outputs) in each block

Hard Forks

- New op codes
- Changes to size limits
- Changes to mining rate
- Many small bug fixes (e.g. MULTI-SIG)

Currently very unlikely to happen.

We will revisit this when we discuss altcoins.