```
6.824 2020 Lecture 19: Bitcoin

Bitcoin: A Peer-to-Peer Electronic Cash System, by Satoshi Nakamoto, 2008

why this paper?
  like Raft -- state, log of operations, agreement on log content and thus state
  unlike Raft -- many participants are certain to be malicious -- but which?
  more distributed than most systems -- decentralized, peer-to-peer
    identities of participants are not known, not even the number
  the agreement scheme is new and very interesting
  Bitcoin's success was a big surprise

why might people want a digital currency?
  might make online payments easier, faster, lower fees
  credit cards have worked well but aren't perfect
    insecure -> fraud -> fees, restrictions, reversals
    record of all your purchases
  might reduce trust required in various entities (banks, governments)

what are the technical challenges?
  outright forgery (easy to solve)
  double spending (hard, bitcoin does pretty well)
  theft (hard, bitcoin not particularly strong)

what's hard socially/economically?
  how to persuade people that bitcoin has value?
  how to make a currency that's useful for commerce, storing value?
  how to pay for infrastructure?
  monetary policy (stimulus, control inflation, &c)
  laws (taxes, laundering, drugs, terrorists)

idea: signed sequence of transactions
  (this is the straightforward part of Bitcoin)
  there are a bunch of coins, each owned by someone
  every coin has a sequence of transaction records
    one for each time this coin was transferred as payment
  a coin's latest transaction indicates who owns it now

what's in a transaction record?
  pub(user1): public key of new owner
  hash(prev): hash of this coin's previous transaction record
  sig(user2): signature over transaction by previous owner's private key
  (BitCoin is more complex: amount (fractional), multiple in/out, ...)

transaction example:
  Y owns a coin, previously given to it by X:
    T6: pub(X), ...
    T7: pub(Y), hash(T6), sig(X)
  Y buys a hamburger from Z and pays with this coin
    Z sends public key to Y
    Y creates a new transaction and signs it
    T8: pub(Z), hash(T7), sig(Y)
  Y sends transaction record to Z
  Z verifies:
    T8's sig(Y) corresponds to T7's pub(Y)
  Z gives hamburger to Y

only the transactions exist, not the coins themselves
  Z's "balance" is set of unspent transactions for which Z knows private key
  the "identity" of a coin is the (hash of) its most recent xaction

can anyone other than the owner spend a coin?
  current owner's private key needed to sign next transaction
  danger: attacker can steal Z's private key, e.g. from PC or smartphone or online exchange
```

```
    this is a serious problem in practice, and hard to solve well

can a coin's owner spend it twice in this scheme?
  Y creates two transactions for same coin: Y->Z, Y->Q
    both with hash(T7)
  Y shows different transactions to Z and Q
  both transactions look good, including signatures and hash
  now both Z and Q will give hamburgers to Y

why was double-spending possible?
  b/c Z and Q didn't know complete set and order of transactions

what do we need?
  publish a log of all transactions
  ensure everyone sees the same log (in same order!)
  ensure Y can't un-publish a transaction
  result:
    Z will see Y->Z came before Y->Q, and will accept Y->Z
    Q will see Y->Z came before Y->Q, and will reject Y->Q
    a "public ledger"

why not publish transactions like this:
  1000s of peers, run by anybody, no trust required in any one peer
  transactions are sent to all peers
  peers vote on which transaction to append next to the log; majority wins
    assumes a majority are honest, will agree and out-vote malicious minority
  how to count votes?
    how to even count peers so you know what a majority is?
    perhaps distinct IP addresses?
  problem: "sybil attack"
    IP addresses are not secure -- easy to forge, or botnets of real computers
    attacker pretends to have a vast number of computers -- majority
    when Z asks, attacker's majority says "Y->Z is in log before Y->Q"
    when Q asks, attacker's majority says "Y->Q is in log before Y->Z"
  voting is hard in "open" schemes!

the BitCoin block chain
  the goal: agreement on transaction log to prevent double-spending
  the block chain contains transactions on all coins
  many peers
    each with a complete copy of the whole chain
    each with TCP connections to a few other peers -- a "mesh overlay"
    new blocks flooded to all peers, by forwarding over TCP
    proposed transactions also flooded to all peers
  each block:
    hash(prevblock)
    set of transactions
    "nonce" (can be anything, as we'll see)
    current time (wall clock timestamp)
  new block every 10 minutes containing xactions since prev block
  payee doesn't accept transaction until it's in the block chain

who creates each new block?
  this is "mining" via "proof-of-work"
  requirement: hash(block) has N leading zeros
  each peer tries random nonce values until this works out
  trying one nonce is fast, but most nonces won't work
    it's like flipping a zillion-sided coin until it comes up heads
    each flip has an independent (small) chance of success
    mining a block *not* a specific fixed amount of work
  it would likely take one CPU months to create one block
  but thousands of peers are working on it
  such that expected time to first to find is about 10 minutes
    though the variance is high
  the winner floods the new block to all peers
```

```text
  proof-of-work solves the Sybil problem -- your CPU must be real to win

how does a Y->Z transaction work w/ block chain?
  start: all peers know ...<-B5
    and are mining block B6 (trying different nonces)
  Y sends Y->Z transaction to peers, which flood it
  peers buffer the transaction until B6 computed
  peers that heard Y->Z include it in next block
  so eventually ...<-B5<-B6<-B7, where B7 includes Y->Z

Q: could there be *two* different successors to B6?
A: yes:
   1) two peers find nonces at about the same time, or
   2) slow network, 2nd block found before 1st is known
  two simultaneous blocks will be different
    miners know about slightly different sets of new transactions, &c.
  if two successors, the blockchain temporarily forks
    peers work on whichever block they heard first
    but switch to longer chain if they become aware of one

how is a fork resolved?
  each peer initially believes whatever new block it sees first
  tries to create a successor
  if more saw Bx than By, more will mine for Bx,
    so Bx successor likely to be created first
  even if exactly half-and-half, one fork likely to be extended first
    since significant variance in mining time
  peers switch to mining the longest fork once they see it, re-inforcing agreement
  what about transactions in the abandoned fork?
    most will be in both forks
    but some may be in just the abandoned fork -- appear, then disappear!

what if Y sends out Y->Z and Y->Q at the same time?
  i.e. Y attempts to double-spend
  correct peers will accept first they see, ignore second
  thus next block will have one but not both

what happens if Y tells some peers about Y->Z, others about Y->Q?
  perhaps use network DoS to prevent full flooding of either
  perhaps there will be a fork: B6<-BZ and B6<-BQ

thus:
  temporary double spending is possible, due to forks
  but one side or the other of the fork highly likely to disappear soon
  thus if Z sees Y->Z with a few blocks after it,
    it's very unlikely that it could be overtaken by a
    different fork containing Y->Q
  if Z is selling a high-value item, Z should wait for a few
    blocks before shipping it
  if Z is selling something cheap, maybe OK to wait just for some peers
    to see Y->Z and validate it (but not in block)

can an attacker modify just an existing block in the middle of the block chain?
  and tell newly starting peers about the modified block?
  e.g. to delete the first spend of the attacker's coin?
  no: then "prev" hash in next block will be wrong, peers will detect

could attacker start a fork from an old block, with Y->Q instead of Y->Z?
  yes -- but fork must be longer in order for peers to accept it
  since attacker's fork starts behind main fork,
    attacker must mine blocks *faster* than total of other peers
  with just one CPU, will take months to create even a few blocks
    by that time the main chain will be much longer
    no peer will switch to the attacker's shorter chain
  if the attacker has more CPU power than all the honest
```

```
    bitcoin peers -- then the attacker can create the longest fork,
    everyone will switch to it, allowing the attacker to double-spend

why does the mining scheme work?
  random choice over all participants for who gets to choose which fork to extend
    weighted by CPU power
  if most participants are honest, they will re-inforce agreement on longest fork
  if attacker controls majority of CPU power, it can force honest
    peers to switch from real chain to one created by the attacker

validation checks:
  peer, new xaction:
    no other transaction spends the same previous transaction
    signature is by private key of pub key in previous transaction
    then will add transaction to txn list for next block to mine
  peer, new block:
    hash value has enough leading zeroes (i.e. nonce is right, proves work)
    previous block hash exists
    all transactions in block are valid
    peer switches to new chain if longer than current longest
  Z:
    (some clients rely on peers to do above checks, some don't)
    Y->Z is in a block
    Z's public key / address is in the transaction
    there's several more blocks in the chain
  (other stuff has to be checked as well, lots of details)

where does each bitcoin originally come from?
  each time a peer mines a block, it gets 12.5 bitcoins (currently)
  it puts its public key in a special transaction in the block
  this is incentive for people to operate bitcoin peers

Q: 10 minutes is annoying; could it be made much shorter?

Q: if lots of miners join, will blocks be created at a higher rate?

Q: why does Bitcoin extend the longest chain? why not some other rule?

Q: are transactions anonymous?

Q: if I steal bitcoins, is it safe to spend them?

Q: can bitcoins be forged, i.e. a totally fake coin created?

Q: what can adversary do with a majority of CPU power in the world?
   can double-spend and un-spend, by forking
   cannot steal others' bitcoins
   can prevent xaction from entering chain

Q: what if the block format needs to be changed?
   esp if new format wouldn't be acceptable to previous s/w version?
   "hard fork"

Q: how do peers find each other?

Q: what if a peer has been tricked into only talking to corrupt peers?
   how about if it talks to one good peer and many colluding bad peers?

Q: could a brand-new peer be tricked into using the wrong chain entirely?
   what if a peer rejoins after a few years disconnection?
   a few days of disconnection?

Q: how rich are you likely to get with one machine mining?

Q: why does it make sense for the mining reward to decrease with time?
```

```
Q: is it a problem that there will be a fixed number of coins?
   what if the real economy grows (or shrinks)?

Q: why do bitcoins have value?
   e.g. people seem willing to pay $8,935 per bitcoin (on may 5 2020).

Q: will bitcoin scale well?
   in terms of CPU time?
     apparently CPU limits to 4,000 tps (signature checks)
     more than Visa but less than cash
   in terms of storage?
     do you ever need to look at very old blocks?
     do you ever need to xfer the whole block chain?
     merkle tree: block headers vs txn data.
   in terms of network traffic?
      a few megabytes (one block) every ten minutes
   sadly, the maximum block size is limited to a few megabytes

Q: could Bitcoin have been just a ledger w/o a new currency?
   e.g. have dollars be the currency?
   since the currency part is pretty awkward.
   (settlement... mining incentive...)

weak points in the design?
  too bad it's a new currency as well as a payment system
  transaction confirmation takes at least 10 minutes, or 60 for high confidence
  flooding limits performance, may be a point of attack
  maximum block size plus 10 minutes limits max transactions per second
  vulnerable to majority attack
  proof-of-work wastes CPU time, power
  not very anonmyous
  too anonymous -- illegal uses may trigger legal response
  users have trouble securing private keys

key idea: block chain
  public agreed-on ledger is a great idea
  decentralization might be good
  mining is a clever way to avoid sybil attacks in an open system,
    and ensure most blocks are created by benign peers
```