```
6.824 2015 Lecture 23: Bitcoin

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

why aren't credit cards the perfect digital money?
  +  work online
  +  hard to steal (a complex situation)
  +- can cancel transactions, call customer support
  -  relies on 3rd parties to verify (banks)
     b/c users cannot directly verify charges
  -  3% fee
  -  long settling time
  -  hard to become a merchant
  +- tied to currency controlled by government

bitcoin: e-money without a central trusted party
  a public ledger: anyone can verify transactions

what's hard technically?
  forgery
  double spending
  theft

what's hard socially/economically?
  why does it have value?
  how to pay for infrastructure?
  monetary policy (intentional inflation &c)
  laws (taxes, laundering, drugs, terrorists)

let's design OneBit, a simple e-money system
  to illustrate a public, verifiable ledger using transaction chain
  each user owns some coins
  single server -- OneBank -- everyone talks to it
  OneBank records all transactions

OneBit transactions
  every coin has a chain of transaction records
    one for each time this coin was transferred as payment
  OneBank maintains the complete chain for each coin
  chain helps ensure that only the current owner spends

what's in a OneBit transaction record?
  public key of new owner
  hash of this coin's previous transaction record
  signed by private key of previous owner
  (BitCoin is much more complex: amount (fractional), multiple in/out, ...)

OneBit example:
  Y owns a coin, previously given to it by X:
    T7: pub(Y), hash(T6), sig(X)
  Y buys a hamburger from Z and pays with this coin
    Z tells Y Z's public key ("address")
    Y creates a new transaction and signs it
    T8: pub(Z), hash(T7), sig(Y)
  OneBank verifies that:
    no other transaction mentions hash(T7),
    T8's sig() corresponds to T7's pub()
  Z waits until OneBank has seen/verified T8,
    verifies that T8's pub() is Z's public key,
```

```
    then Z gives hamburger to Y

  why include pub(Z)?

  why sign with sig(Y)?

  why include hash(T7)?
    hash(T7) identifies the exact coin to be spent
    a coin ID scheme might be ambiguous if Y owned this coin previously

  where is Z's resulting coin value "stored"?
    coin balance = unspent xaction
    the "identity" of a coin is the (hash of) its most recent xaction
    Z "owns" the coin: has private key that allows Z to make next xaction

  does OneBit's transaction chain prevent stealing?
    current owner's private key needed to sign next transaction
    danger: attacker can steal Z's private key
      Z uses private key a lot, so probably on his PC, easy to steal?
    a significant problem for BitCoin in practice

  what if OneBank is corrupt?
    it can't forge transactions (doesn't know private keys)
    but it can help people double-spend!

  double-spending with OneBit
    suppose OneBank is cooperating with Y to cheat Z or Q
    Y creates two transactions for same coin: Y->Z, Y->Q
      both have has pointing to same current end of chain
    OneBank shows chain ending in Y->Z to Z, and Y->Q to Q
      both transactions look good, including signatures and hash
    now both Z and Q will give hamburgers to Y

  why was double-spending possible?
    OneBank can *hide* some information,
    or selectively reveal it

  what's needed?
    many servers ("peers")
    send all transactions to all peers
    much harder for a few bad peers to hide transactions
    conventions to un-fork if problems do arise

  the BitCoin block chain
    single block chain contains transactions on all coins
    a copy stored in each peer
      so each peer can validate new transactions against old ones
    each block:
      hash(prevblock)
      set of transactions
      "nonce" (not quite a nonce in the usual cryptographic sense)
      current time (wall clock timestamp)
    a transaction isn't real unless it's in the block chain
    new block every 10 minutes containing xactions since prev block

  who creates each new block?
    all the peers try
    requirement: hash(block) < "target"
    peers try nonce values until this works out
    can't predict a winning nonce, since cryptographic hash
    trying one nonce is fast, but most nonces won't work
```

```
    it would 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
    the winner sends the new block to all peers
    (this is part of "mining")

  how do transactions work w/ block chain?
    start: all peers know ...<-B5
      and are working on B6 (trying different nonces)
    Z sends public key (address) to Y
    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

  what if bad person wants to double-spend?
    start with block chain ...<-B6
    Y gets Y->Z into block chain
      ...<-B6<-BZ (BZ contains Y->Z)
    Z will see Y->Z in chain and give Y a hamburger
    can Y create ...<-B6<-BQ
      and persuade peers to accept it in place of ...<-B6<-BZ?

  when will a peer accept chain CX it hears from another peer?
    suppose peer already knows of chain CY
    it will accept CX if len(CX) > len(CY)
    and if CX passes some validity tests
    will not accept if len(CX) = len(CY): first chain of same length wins

  so attacker needs a longer chain to double-spend
    e.g. ...<-B6<-BQ<-B8, which is longer than ...<-B6<-BZ
    and must create it in less than 10 minutes
      *before* main chain grows by another block
    10 minutes is the time it takes the 1000s of honest peers
      to find one block
    if the attacker has just one CPU, will take months to create the blocks
      by that time the main chain will be much longer
      no peer will accept the attacker's shorter chain
      attacker has no chance
    if the attacker has 1000s of CPUs -- more than all the honest
      bitcoin peers -- then the attacker can double spend

  summary:
    attacker can force honest peers to switch chains if attacker
      controls majority of peer CPU power

  how long should Z wait before giving Y the hamburger?
    until Z sees Y flood the transaction to many peers?
      no -- not in the chain, Y might flood conflicting xaction
      maybe OK for low-value transactions
    until Z sees chain ...<-BZ?
      maybe
      risky -- non-zero chance that some other chain will win
        i.e. some lucky machine discovered a few blocks in a row
          quickly, but its network msgs have so far been lost
        perhaps that chain won't have Y->Z
        probability goes down rapidly with number of blocks
    until Z sees chain with multiple blocks after BZ?
      yes -- slim chance attacker with few CPUs could catch up

  validation checks:
```

```
  much of burden is on (honest) peers, to check new xactions/blocks
    to avoid ever having to scan the whole block chain
    and so that clients don't have to maintain the whole block chain
  peer, new xaction:
    no other transaction refers to same previous transaction
    signature is by private key of previous transaction
    [ then will add transaction to txn list for new block being mined ]
  peer, new block:
    hash value < target (i.e. nonce is right, proof of work)
    previous block hash exists
    new chain longer than current chain
    all transactions in block are valid
  Z:
    Y->Z is in a recent block
    Z's public key / address is in the transaction
    multiple peers have accepted that block
    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 creates a block, it gets 25 bitcoins
    assuming it is the winner for that block
  it puts its public key in a special transaction in the block
  this is incentive for people to operate bitcoin peers
  but that number halves every 210,000 blocks (abt 4 years)
  the point: motivate people to run peers

Q: how do peers communicate / find each other?

Q: what prevents a bad peer from modifying an existing block?

Q: what if two nodes disagree on the validity of a transaction?
   (slight implementation differences between software versions)

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

Q: are transactions anonymous?  pseudonymous?  analogy: IP addresses.

Q: can bitcoins be stolen?

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

Q: what can adversary do with a majority of CPU power in the world?
   can double-spend
   cannot steal others' bitcoins
   can prevent xaction from entering chain
   can revert past xactions (by building longer chain from before that block)

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

Q: if more people (CPUs) mine, will that create new bitcoin faster?
   important use of block timestamps: control difficulty (hash target)

Q: why mine at all? why not start with a fixed number of coins?

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?

Q: what if the real economy grows (or shrinks)?
```

```
Q: why do bitcoins have value?
   e.g., 1 BTC appears to be around US$242 on may 12th 2015.

Q: will we still need banks, credit cards, &c?
   today, dollar bills are only a small fraction of total money
   same may be true of bitcoin
   so properties of bitcoin (anonymity &c) may not be very important

Q: what are the benefits of banks, credit cards, &c?
   disputes (Z takes payment and does not give hamburger to Y)
   loss / recovery (user cannot find their private key)

Q: will bitcoin scale well?
   as transaction rate increases?
     claim CPU limits to 4,000 tps (signature checks)
     more than Visa but less than cash
   as block chain length increases?
     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.

key idea: block chain
  public ledger is a great idea
  decentralization might be good
  mining seems imperfect, but does avoid centralized trust
  tieing ledger to new currency seems bad
```