Computer Security

13. Blockchain & Bitcoin

Paul Krzyzanowski

Rutgers University

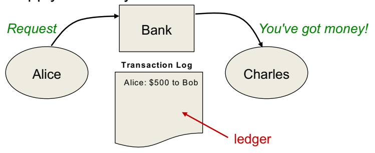
Spring 2019

Bitcoin & Blockchain

Bitcoin cryptocurrency system
- Introduced in 2009 – anonymously by Satoshi Nakamoto
- First blockchain
- Designed to be public – anyone can participate in the system & use it

Traditional Payments

- Suppose Alice wants to pay Charles
- Send a message to the bank:
  - Transfer $500 from Alice to Charles
- Bank is a trusted third party
  - Owns register of activity & account balances
  - Only the bank can manipulate the data
  - Also – banks control supply of money



Centralized systems

- Transactions are simply modifications to the bank's database
- We can simply
  - Subtract $500 from Alice's account
  - Add $500 to Charles' account
  - The log is just nice for auditing but not necessary

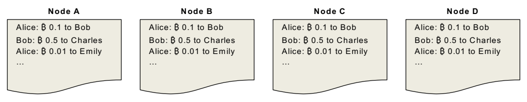
Problems?

This is a centralized system

- What if the bank disappears?
- What if the banker makes a mistake?
- What if the banker is corrupt?

Decentralized Solution – Bitcoin

- **Blockchain** = ledger = complete list of ALL transactions
  - Since Bitcoin was started in January 2009
- Complete copies of the ledger are replicated around the world
  - Thousands of nodes – all peers – running identical software
  - There is no master node or master version
- New systems can do a DNS query for well known peers
  - Names hardcoded in source (DNS seeds)
  - Return list of IP addresses of bitcoin nodes
  - Then use peer discovery process to find others



Identities

- Alice creates a {public, private} key pair that defines her wallet
  - 256-bit Elliptic Curve Digital Signature Algorithm (ECDSA) used
  - The wallet is just a place to store these keys
  - Wallets *may* store a transaction list but that's just for user records – the bitcoin network doesn't care
- **Bitcoins are associated with keys, not users**
  - Users are anonymous
  - A user's ID is the public key – anonymous – no association to name
  - The user's identity is called their **address**
  - Users may have multiple keys
- **Every transaction created by a user is signed with a private key**
  - Transaction identifies the user by the public key and can be verified
  - We know only the person with the corresponding private key could have created the request
- ***Nobody to call if you lose your private key!***

Transactions: Inputs

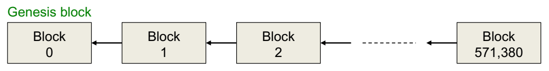
- If Alice wants to send some bitcoin to Bob
  - She creates a **transaction** and sends it to one or more bitcoin nodes
  - A node tells its peers about the transaction
  - Within ~5 sec. ever beer on the network has it
  - The transaction is currently **unconfirmed**
- **A blockchain is NOT a database**
  - There are no accounts to query – just lists of transactions
  - Alice needs to provide links to previous transactions that will add up to at least the required amount – these are **inputs**
- A node verifies inputs
  - Make sure they have not been used by another transaction (this would be **double spending**)
  - Make sure there is sufficient money in the inputs

Transactions: Outputs

- A transaction identifies
  - One or more inputs: transaction IDs & address where coin comes from
  - Output: who the money goes to
- Every input must be completely spent
  - Any excess change can be generated as another output
- Transaction contains:
  - One or more inputs: *identify transactions where coins come from*
  - Output: destination address & amount
  - Change: owner's address & amount
  - Transaction fee
- The amount of bitcoin you own is the set of transactions in the system that are outputs to your address (public key) but have NOT been used as inputs in any transaction

Blocks

- Transactions are grouped into blocks
  - Each block holds ~4,000 transactions and is ~1 MB in size
- Bitcoin ledger = linked list of chronologically-ordered blocks
- Approximately every 10 minutes, a new block of transactions is added to the blockchain

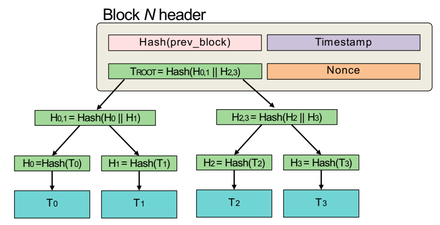


- Each block has
  - A link to the previous block
  - SHA-256 hash of the previous block

This creates the blockchain: **hash pointers** – a tamper-evident log

Transactions in blocks: Merkle trees

- Transactions within a block are stored in a Merkle tree
  - Binary tree of hash pointers
  - Using a tree makes it easy to find one of thousands of transactions
  - A Merkle tree makes it easy to check if the transaction is valid



Agreement & adding blocks

- Transactions can reach nodes in a different order
- Blockchains require all nodes to agree on the sequence of blocks
- Each node groups transactions into a block & can propose it as the next block in the blockchain
- To add a block to the chain, the hash of the block must meet a certain requirement

Let's make in challenging: create a puzzle

Suppose we want a hash output with a specific property?
- Example, starting with "0000"?
- No algorithmic way to do this
- Must try lots of variations of the input
- But once found – it is easy for anyone to verify that the data hashes to the result

Mining

- Solving this "puzzle" is called **mining**
  - Have a 32-bit field in the block where we can try different numbers
  - Try to get the block to hash to a desired output
  - The resulting number is called the **Proof of Work**

We demonstrate that work has been put into figuring out what the value should be to create the desired hash

- Everyone in the network can participate in this
  - The first system that finds it announces the block to everyone else in the network
  - Upon receiving an announcement
    - Each system validates the Proof of Work number against the block
    - A majority of systems must grant approval
    - If they do, the block (with the Proof of Work) is made part of the **blockchain**

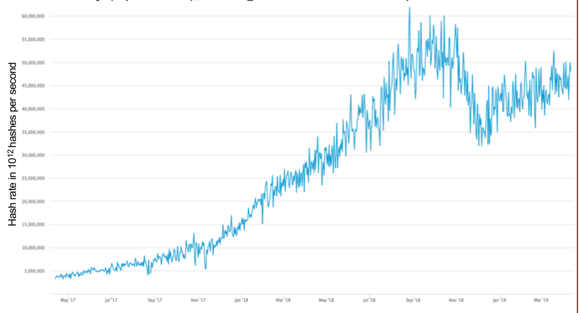
What's the puzzle?

- Bitcoin uses **hashcash** (created in 1997)
  - Hashcash searched for a *hash(message, random #, N)* where the leading k bits are 0
    - Random # – 128-bit starting value to make it unlikely that two systems start tat the same point
    - N – the nonce: the number we vary until we get the hash we need
    - Choice of *k* sets the difficulty of the problem
- Ensure that one node doesn't take credit for another's work
  - 256-bit SHA-1 hash of
    - ***B***, transaction block, which includes hash pointer to previous block
    - ***A***, recipient's reward address (public key of who gets credit)
    - ***N*** – nonce: the number we vary until we get the hash we need
- Bitcoin uses a floating-point *k* to scale the work more precisely

$hash(B, A, N) < 2^{n-k}$

How much work is going on?

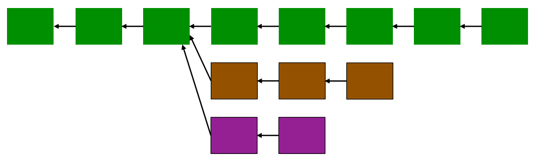
- Currently (April 2019), average 46.4×10^{18} hashes per second



See https://www.blockchain.com/charts/hash-rate

Competing chains

What if a malicious participants wants to modify an old transaction?
- Need to modify an old block
- Recompute the Proof of Work (which takes a lot of effort) for the block and each successive block (tons of work)
- This participant will be creating another chain in the blockchain



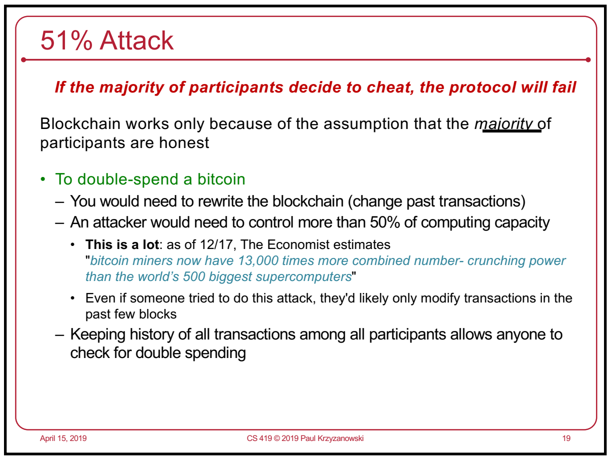
Competing chains

- BUT
  - One malicious participant will not be able to catch up with the cumulative work of all the others
  - It is expected that some nodes will occasionally have different versions
  - Length of chain = **score**
- If we observe two states of the blockchain, we select the one that was the hardest to generate (= longest chain)
  - Blockchain rules state that

    The longest chain in the network is the correct one
  - If a participant receives a higher-scoring version

    It overwrites its blockchain with the better data & transmits updates to peers

Producing a longer ledger than the current one requires computing power that competes with the rest of the entire network

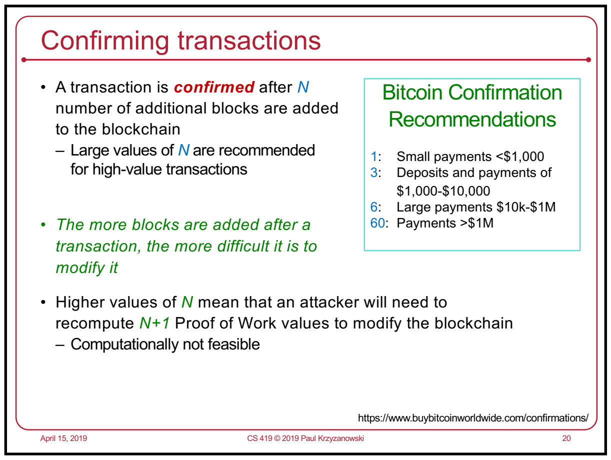


51% Attack

If the majority of participants decide to cheat, the protocol will fail

Blockchain works only because of the assumption that the *majority* of participants are honest

- To double-spend a bitcoin
  - You would need to rewrite the blockchain (change past transactions)
  - An attacker would need to control more than 50% of computing capacity
    - **This is a lot**: as of 12/17, The Economist estimates "*bitcoin miners now have 13,000 times more combined number- crunching power than the world's 500 biggest supercomputers*"
    - Even if someone tried to do this attack, they'd likely only modify transactions in the past few blocks
  - Keeping history of all transactions among all participants allows anyone to check for double spending



Confirming transactions

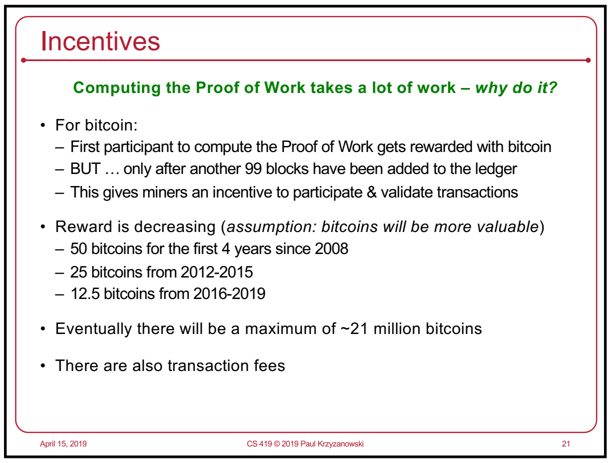
- A transaction is ***confirmed*** after *N* number of additional blocks are added to the blockchain
  - Large values of *N* are recommended for high-value transactions
- *The more blocks are added after a transaction, the more difficult it is to modify it*

Bitcoin Confirmation Recommendations

1:	Small payments <$1,000
3:	Deposits and payments of $1,000-$10,000
6:	Large payments $10k-$1M
60:	Payments >$1M

- Higher values of *N* mean that an attacker will need to recompute *N+1* Proof of Work values to modify the blockchain
  - Computationally not feasible

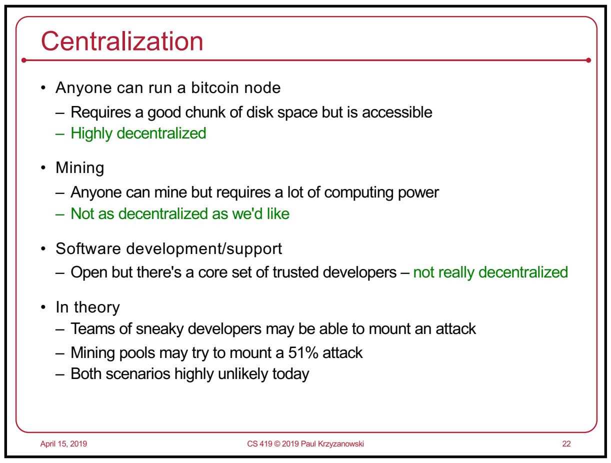
https://www.buybitcoinworldwide.com/confirmations/



Incentives

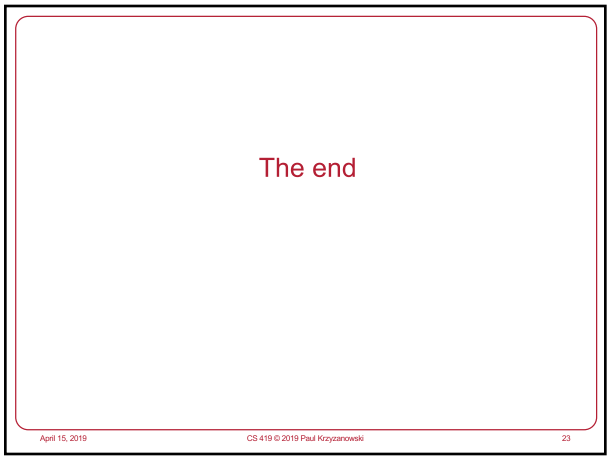
Computing the Proof of Work takes a lot of work – *why do it?*

- For bitcoin:
  - First participant to compute the Proof of Work gets rewarded with bitcoin
  - BUT … only after another 99 blocks have been added to the ledger
  - This gives miners an incentive to participate & validate transactions
- Reward is decreasing (*assumption: bitcoins will be more valuable*)
  - 50 bitcoins for the first 4 years since 2008
  - 25 bitcoins from 2012-2015
  - 12.5 bitcoins from 2016-2019
- Eventually there will be a maximum of ~21 million bitcoins
- There are also transaction fees



Centralization

- Anyone can run a bitcoin node
  - Requires a good chunk of disk space but is accessible
  - Highly decentralized
- Mining
  - Anyone can mine but requires a lot of computing power
  - Not as decentralized as we'd like
- Software development/support
  - Open but there's a core set of trusted developers – not really decentralized
- In theory
  - Teams of sneaky developers may be able to mount an attack
  - Mining pools may try to mount a 51% attack
  - Both scenarios highly unlikely today



The end