Bitcoin

CS 161 Fall 2021 - Lecture 17

Announcements

- Started recording
- Midterm Thursday, Oct 7, 7-9pm
  - All logistics are on the website
  - Piazza @479 for logistics questions
  - Converting discussion sections on Monday and Tuesday into topical review sections! The schedule is available at @532.

Bitcoin

textbook chapter 16

What is Bitcoin?

- Bitcoin is **a cryptocurrency**: a digital currency whose rules are enforced by cryptography and not by a trusted party (e.g., bank)
- **Core ideal:** avoid trust in institutions (e.g., banks, governments)
- Bitcoin is also a **ledger.** Its protocol is built on a technique called a blockchain, which has applications beyond Bitcoin.

Satoshi Nakamoto

- Created by Satoshi Nakamoto, an anonymous identity, in 2009
- Wrote beautiful white paper on Bitcoin, in the syllabus
- No one knows who he is, online presence only
- Name stands for clear/wise medium; most likely not Japanese, but pseudonym
- He is very rich! [But hasn’t sold yet]

Bitcoin technical design

Let’s work it out together!

Replacing banks

“IN BANKS WE DISTRUST”

Basic notions a bank provides:

- Identity management
- Transactions
- Prevents double spending

How can we enforce these properties cryptographically?

Two components

1. **Ledger:**
    1. publicly-visible,
    2. append-only, and
    3. immutable,

    log
2. **Cryptographic transactions**

Cryptographic transactions

- For now, assume the existence of a trusted ledger (append-only, immutable, everyone can see what is on it)

Identity

Q: How can we give a person a cryptographic identity?

- Each user has a PK and SK
- User referred to by PK

Transactions

Q: How can Alice transfer 10 ฿ (bitcoins) to Bob in a secure way?

- Idea: Alice signs transaction using her SK$_A$
- sign$_{SK_A}$(“PK$_A$ transfers *10 ฿ to PK*$_B$”)
- Anyone can check Alice intended the transaction

Q: Problems?

- Alice can spend more money than she has. She can sign as much as she wants.

Q: Ideas how to solve this still assuming a “trusted ledger owner”?

Include only correct transactions in the public ledger

- **For now only:** assume there is a trustworthy ledger owner, assume initial budgets for each PK

Q: how would you prevent double spending?

- Assume all signatures/transactions are sorted in order of creation; include previous transaction where money came from

TX = (PK_{sender}->$PK_{receiver}$;X ฿; PK_{sender}->PK_{sender};R ฿;
list of transactions L where money came from)

time →

Initial budgets: PK_A has 10 ฿	TX_1 = (PK_A->PK_B;10 ฿; from initial budgets) $sign_{SK_A}(TX_1)$	TX_2 = (PK_B->PK_C;5 ฿; PK_B->PK_B;5 ฿; from TX_1) $sign_{SK_B}(TX_2)$

How does the ledger owner check a transaction?

Verify TX = (PK_{sender}->$PK_{receiver}$;X ฿; PK_{sender}->PK_{sender};R ฿; list of transactions L where money came from); $sign_{sender}$ TX

1. The signature on TX verifies with the PK of the sender
2. The transactions in L have PK of sender as their recipient
   (that is, the sender receives Bitcoins in the transactions in L)
3. The transactions in L have not been spent before by sender
   (each transaction A->B can only be spent once by B, and once by A if there were remaining bitcoins in it)
4. Sender had X+R Bitcoins in L: the sum of the amounts received in the transactions in L total to X+R.

Two components

1. **Ledger:**
    1. publicly-visible,
    2. append-only, and
    3. immutable,

    log
2. **Cryptographic transactions**

Bitcoin’s ledger

1. Hash chain / blockchain
2. Consensus via proof of work

Blockchain

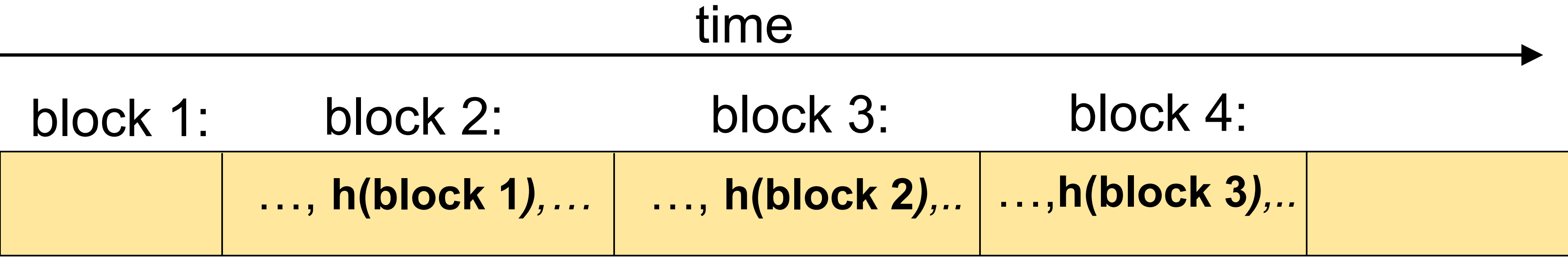
- Chain transactions using their hashes => hashchain
- Each transaction contains hash of previous transaction **(which contains the hash of its own previous transaction, and so on)**
- Recall that a cryptographic hash is collision resistant

time →

block 1:	block 2:	block 3:
Initial budgets: PK_A has 10 ฿	TX_1 = (PK_A->*PK_B;10 ฿;* from initial budgets; **h(block 1)** *)* $\text{sign}_{SK_A}(TX_1)$	TX_2 = (PK_B->*PK_C;5 ฿;* from TX_1; **h(block 2)** *)* $\text{sign}_{SK_B}(TX_2)$

block i refers to the entire block (transaction description and signature), so the hash is over all of this

Properties of the hashchain


time

block 1:

block 2: …, **h(block 1)**,…

block 3: …, **h(block 2)**,..

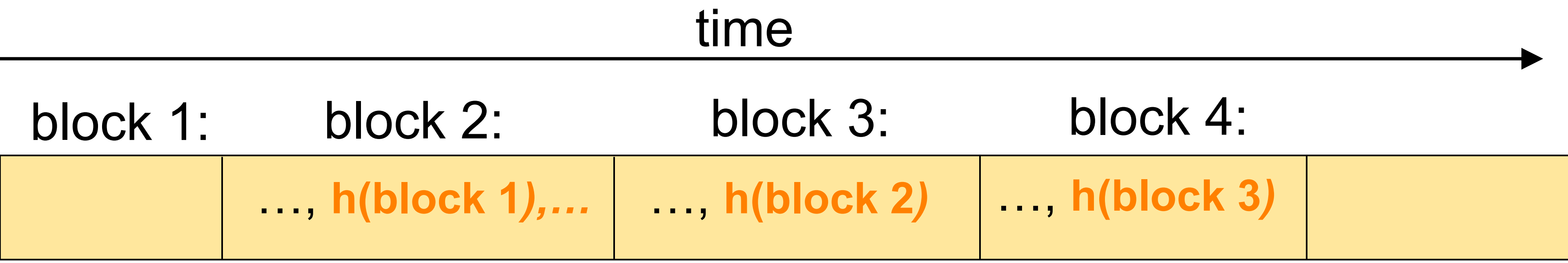
block 4: …,**h(block 3)**,..

Given h(block i) from a trusted source and all the blocks *1 … i* from an untrusted source, Alice can verify that blocks 1 … i are not compromised using h(block i)

Q: How?

A: Alice recomputes the hashes of each block, checks it matches the hash in the next block, and so on, until the last block, which she checks it matches the hash from the trusted source

Why can’t attacker cheat?



time

block 1: block 2: block 3: block 4:

…, **h(block 1),…** …, **h(block 2)** …, **h(block 3)**

Say Alice obtains h(block 4) from somewhere **trusted**

She fetches the entire blockchain from a compromised server.

Q: Why can’t the attacker give Alice an incorrect chain?

Say block 2 is incorrect.

block 1: block 2*: block 3: block 4:

…, **h(block 1),…** …, **h(block 2)** …, **h(block 3)**

A: because the hash is collision resistant

She fetches the entire blockchain from a compromised server.

Q: Why can't the attacker give Alice an incorrect chain?
Say block 2 is incorrect.

block 1:	block 2*:	block 3:	block 4:	
	…, **h(block 1),…**	…, **h(block 2)**	…, **h(block 3)**	

- If block 2* is incorrect, then hash(block 2*) ≠ hash(block 2)
- Then the third block is block 3* ≠ block 3 because it includes hash(block 2*)
- So hash(block 3*) ≠ hash(block 3)
- Then the fourth block is block 4* ≠ block 4 because it includes hash(block 3*)
- So hash(block 4*) ≠ hash(block 4)
- Hence, the hash of the block chain from the server will not match the trusted hash, detecting misbehavior
- If the hash does match, the attacker supplied the correct block chain

In Bitcoin:

- Every participant stores the blockchain
- There is no central party storing it
- When someone wants to create a new transaction, they broadcast the transaction to everyone
- Every node checks the transaction, and if it is correct, it creates a new block including this transaction and adds it to its local blockchain

- Some participants can be malicious
- The majority are assumed to be honest

Why is the hash chain not enough?

- People can choose to truncate blockchain or not include certain transactions
- So we need a way for everyone to agree on the content of the blockchain: consensus

Example

- Mallory can fork the hash chain
- Say she buys Bob's house from him for $500K in Bitcoins.   Then, she goes back in time and, starting from the block chain just before this transaction was added to it, she starts appending new entries from there. Can she get others to accept this forked chain, so she gets her $500K back? Yes.

Bitcoin’s ledger

1. Hash chain / blockchain

2. Consensus via proof of work

How do users agree on the same history?

Consensus via proof of work

Proof of work / Mining

- Not everyone is allowed to add blocks to the blockchain, but only certain people, called miners
- An honest miner will include all transactions it hears about after checking them
- All miners try to solve a proof of work: the hash of the new block (which includes the hash of the blocks so far) must start with N (e.g. 33) zero bits
  - Can include a random number in the block and increment that so the hash changes until the proof of work is solved
    - Eg: Hash(block || random_number) = 000…0000453a48b244
- Currently someone in the world solves the proof of work every 10-20mins

Propagating blocks

- Miners broadcast blocks with proof of work
- All (honest) Bitcoin nodes listen for such blocks, check the blocks for correctness, and accept the longest correct chain
- If a miner appends a block with some incorrect transaction, the block is ignored

Consensus: longest correct chain wins

- Everyone will always prefer the longer correct chain

Example

- An honest miner M1 stores current blockchain: b1->b2->b3
- M1 hears about transactions T
- M1 tries to mine for block b4 to include T
- Another miner M2 mines first b4 and broadcasts b4, with b3->b4
- M1 checks b4, accepts b4, and starts mining for block 5

Example (cont’d)

- M1 now has blockchain

  b1->b2->b3->b4
- M1 hears that some miners are broadcasting b1->b2->b3->b4’->b5’
- M1 checks this new chain, and then accepts this new chain, essentially discarding b4

Assumption

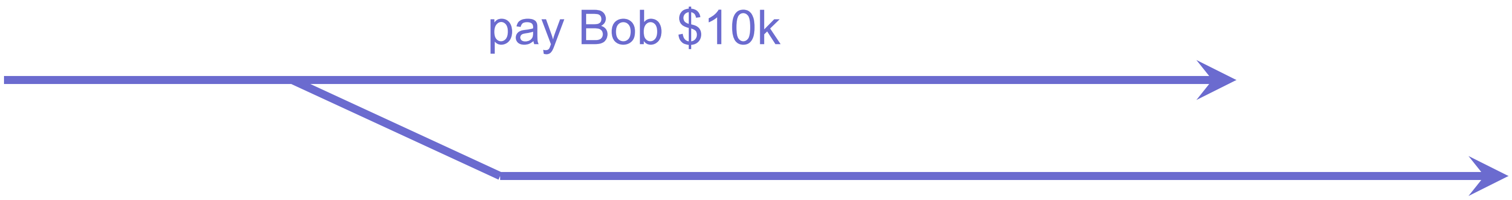
- Assumes more than half of the computing power is in the hands of honest miners
- So honest miners will always have an advantage to mine the longest chain

Consensus

- Can Mallory fork the block chain?
- Say she buys Bob’s house for $500K in Bitcoins. Then, she goes back in time and, starting from the block chain just before this transaction was added to it, she starts appending new entries from there. Can she get others to accept this forked chain, so she gets her $500,000 back?

Consensus

- Can Mallory fork the block chain?
- Answer: No, not unless she has ≥51% of the computing power in the world. Longest chain wins, and her forked one will be shorter (unless she can mine new entries faster than aggregate mining power of everyone else in the world).



“Longest chain” wins

- Problem: What if two different parts of network have different hash chains?

- Solution: Whichever is “longer” wins; the other is discarded

Proof of work can be adapted

- Mining frequency is ~15 mins
- If it takes too long to mine on average, make the proof of work easier (less zeros), else make it harder (more zeros)
- Q: what is the economic insight?
- A: if mining is rare, it means few machines in the network, give more incentives to join the network

How can we convince people to mine?

- A: Give a reward to anyone who successfully appends – they receive a free coin
  - Essentially they may include a transaction from no one to their PK having a coin

- Q: What happens to a miner's reward if his block was removed because an alternate longer chain appears?
- A: The miner lost their reward. Only the transactions and rewards on the longest chain "exist".

Let’s chew on consensus

- Q: What happens if Miner A and Miner B at the same time solve a proof of work and append two different blocks thus forking the network?
- A: The next miner that appends onto one of these chains, invalidates the other chain. Longest chain wins.

- Q: If a miner included your transaction in the latest block created, are you guaranteed that your transaction is forever in the blockchain?
- A: No, there could have been another miner appending a different block at the same time and that chain might be winning. So wait for a few blocks, e.g. 3 until your transaction is committed with high probability, though you can never be sure.

Let’s chew on consensus

- Q: What happens if a miner who just mined a block refuses to include my transaction?
- A: Hopefully the next miner will not refuse this. Each transaction also includes a fee which goes to the miner, so a miner would want to include as many transactions as possible

Watch the blockchain live

- https://blockchain.info/

Bitcoin

- Public, distributed, peer-to-peer, hash-chained audit log of all transactions (“block chain”).
- Mining: Each entry in block chain must come with a proof of work (its hash value starts with *N* zeros). Thus, appending takes computation.
- Lottery: First to successfully append to block chain gets a small reward (if append is accepted by others). This creates new money. Each block contains a list of transactions, and identity of miner (who receives the reward).
- Consensus: If there are multiple versions of the block chain, longest one wins.