Announcements

ITIS 6200 / 8200

- Homework 2 due Sep.26th (Tuesday)
- Midterm Oct.3rd (Tuesday)

Cryptography Roadmap

	Symmetric-key	Asymmetric-key
Confidentiality	● One-time pads ● Block ciphers (e.g. AES-CBC) ● Stream ciphers	● RSA encryption
Integrity, Authentication	● MACs (e.g. HMAC)	● Digital signatures (e.g. RSA signatures)

- Hash functions
- Pseudorandom number generators
- Public key exchange (e.g. Diffie-Hellman)

- Key management (certificates)
- Password management

Bitcoin

Today: Bitcoin

ITIS 6200 / 8200

- Bitcoin: Sending and receiving money without trusting a central authority
- How does Bitcoin work?
  - How do we manage identity, transactions, and balances on a ledger?
  - How do we construct a decentralized, trusted ledger?
- The trouble with Bitcoin
- Bitcoin in practice

Bitcoin: Identity and Transactions

Problem Statement: The Decentralized Bank

- Alice, Bob, Carol, and Dave each have a sum of currency
- Anyone can send money to anyone else
  - Dave pays Alice 10 coins
  - Dave’s balance decreases by 10
  - Alice’s balance increases by 10
- No party can spend more currency than they currently have
  - Dave has 10 coins
  - Dave can send Alice 10 coins
  - Dave cannot send Alice 15 coins
- No party trusts any other party, and there is no central authority
  - Usually, a centralized authority (e.g. a bank) tracks balances and enforces spending rules
  - Without a central authority, we must use cryptography to enforce correctness!

Identity Management

- Idea: Use certificates to verify real-world identity?
  - But this needs central root CAs, which we don’t want
- Instead, just define each identity as a public key
  - Circumvents the identity problem since we completely ignore real-world identities
  - Instead of transactions between “people,” we use transactions between public keys

Alice

PK_A

Bob

PK_B

Carol

PK_C

Dave

PK_D

Transactions

- *For now*, assume the existence of a trusted, public ledger
  - Accessible to all parties: Everyone can view the ledger
  - Append-only: Everyone can add data to the ledger
  - Immutable: Nobody can change or delete existing data
  - This is a central authority for now, but we'll deal with it later
- How do we record transactions in a way that everyone can see them?
  - Idea: Put "PK_D paid PK_A 10 coins" on the ledger?
    - Problem: Mallory can forge a transaction (e.g. "PK_A paid PK_M 10,000 coins")
  - Solution: Use digital signatures: Put $\{$"PK_D paid PK_A 10 coins"$\}_{SK_D^{-1}}$ on the ledger
    - Problem: How do we check how much currency Dave (PK_D) has to spend?

Transactions

- Each party’s balance exists only as (total received) - (total spent) on the ledger
  - Real-time balances values aren’t listed directly on the ledger!
- How much time does this take?
  - Too long: We have to scan through the entire ledger to determine someone’s balance
  - The ledger grows indefinitely as more and more transactions occur

The Ledger v1

1. PK_A, PK_B, PK_C, and PK_D magically start with 10 coins.
2. {“PK_A paid PK_C 4 coins.”}$_{SK_A^{-1}}$
3. {“PK_B paid PK_D 6 coins.”}$_{SK_B^{-1}}$
4. {“PK_B paid PK_A 2 coins.”}$_{SK_B^{-1}}$

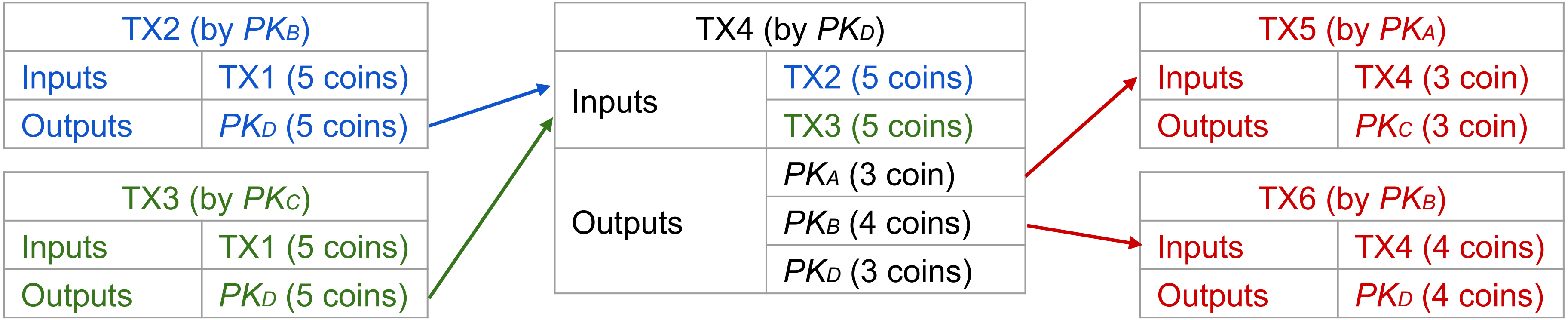
How many coins does PK_A have right now?

8 coins! 10 - 4 + 2 = **8**, for transactions 1, 2, and 4 on the ledger.

Transactions

- Construction: Each transaction has **inputs** (sources, where the money came from) and **outputs** (destinations, who currency is going to)
    - Now, each party only needs to keep track of transactions where they received money (they were the output)
    - Example: {"Using currency from TX 2 and TX 4, give 3 coins to PK_A and 4 coins to PK_B"}$_{SK_D^{-1}}$
        - TX 2 and TX 4 are now "spent" (used as an input), so we don't need to keep track of it anymore
        - Alice and Bob now have an additional unspent transaction
    - To validate a transaction *T* is unspent, check that no transaction after *T* uses *T* as an input
    - Pay whatever change you have back to yourself
        - Example: From TX 2 and TX 4, Dave has received 10 coins in total
        - {"Using TX 2 and TX 4, give 3 coins to PK_A, 4 coins to PK_B, and 3 coins to PK_D"}$_{SK_D^{-1}}$

Transactions



TX2 (by PK_B)	
Inputs	TX1 (5 coins)
Outputs	PK_D (5 coins)

TX3 (by PK_C)	
Inputs	TX1 (5 coins)
Outputs	PK_D (5 coins)

TX4 (by PK_D)	
Inputs	TX2 (5 coins)
	TX3 (5 coins)
Outputs	PK_A (3 coin)
	PK_B (4 coins)
	PK_D (3 coins)

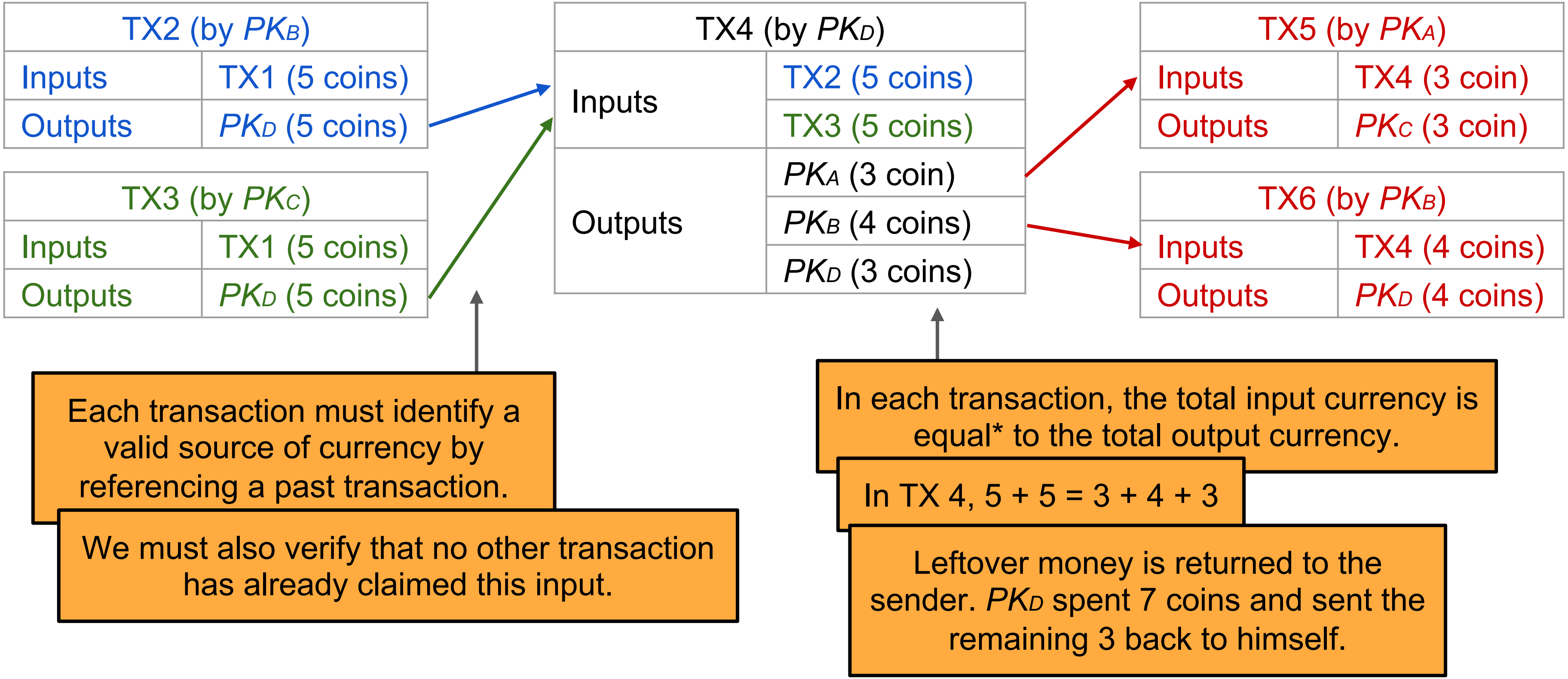
TX5 (by PK_A)	
Inputs	TX4 (3 coin)
Outputs	PK_C (3 coin)

TX6 (by PK_B)	
Inputs	TX4 (4 coins)
Outputs	PK_D (4 coins)

The Ledger v2

1. PK_B and PK_C magically start with 5 coins.
2. {"Using currency from TX 1, give 5 coins to PK_D."}$_{SK_B^{-1}}$
3. {"Using currency from TX 1, give 5 coins to PK_D."}$_{SK_C^{-1}}$
4. {"Using currency from TX 2 and TX 3, give 3 coin to PK_A and 4 coins to PK_B, and 3 coins to PK_D"}$_{SK_D^{-1}}$
5. {"Using currency from TX 4, give 3 coin to PK_C."}$_{SK_A^{-1}}$
6. {"Using currency from TX 4, give 4 coins to PK_D."}$_{SK_B^{-1}}$

Transactions



TX2 (by PK_B)	
Inputs	TX1 (5 coins)
Outputs	PK_D (5 coins)

TX3 (by PK_C)	
Inputs	TX1 (5 coins)
Outputs	PK_D (5 coins)

TX4 (by PK_D)	
Inputs	TX2 (5 coins)
	TX3 (5 coins)
Outputs	PK_A (3 coin)
	PK_B (4 coins)
	PK_D (3 coins)

TX5 (by PK_A)	
Inputs	TX4 (3 coin)
Outputs	PK_C (3 coin)

TX6 (by PK_B)	
Inputs	TX4 (4 coins)
Outputs	PK_D (4 coins)

Each transaction must identify a valid source of currency by referencing a past transaction.

We must also verify that no other transaction has already claimed this input.

In each transaction, the total input currency is equal* to the total output currency.

In TX 4, 5 + 5 = 3 + 4 + 3

Leftover money is returned to the sender. PK_D spent 7 coins and sent the remaining 3 back to himself.

Bitcoin: The Public Ledger

Recall: Hash Functions

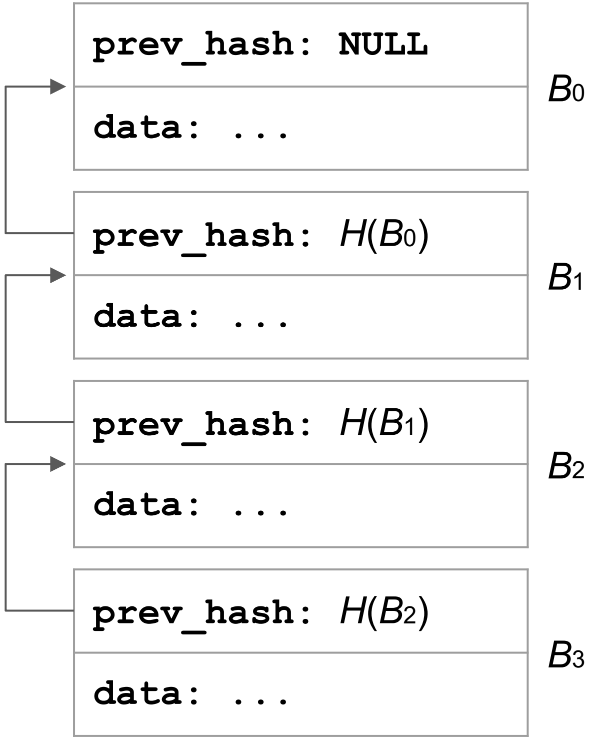
- Hash functions produce a fixed-length “fingerprint” over an arbitrary length of data
  - **Preimage resistant**: Given an output, difficult to find an input that hashes to the output
  - **Collision resistant**: Difficult to find two inputs that hash to the same output
- In practice, hash functions “look” random
  - Changing the input causes the output to change unpredictably
  - Each bit in the output has a 50% chance of flipping

Hash Chains (Blockchain)

ITIS 6200 / 8200

- What we want: A data structure where we can append a node and then compute a hash over all nodes efficiently

- Idea: To validate the previous block, include a hash of the previous block
  - The previous block validates the block before, etc.
- To append:
  - Compute the hash of the current block
  - Construct a new block containing the previous hash and the data
  - Set the head of the chain to the new block



`prev_hash: NULL`
`data: ...`
B_0

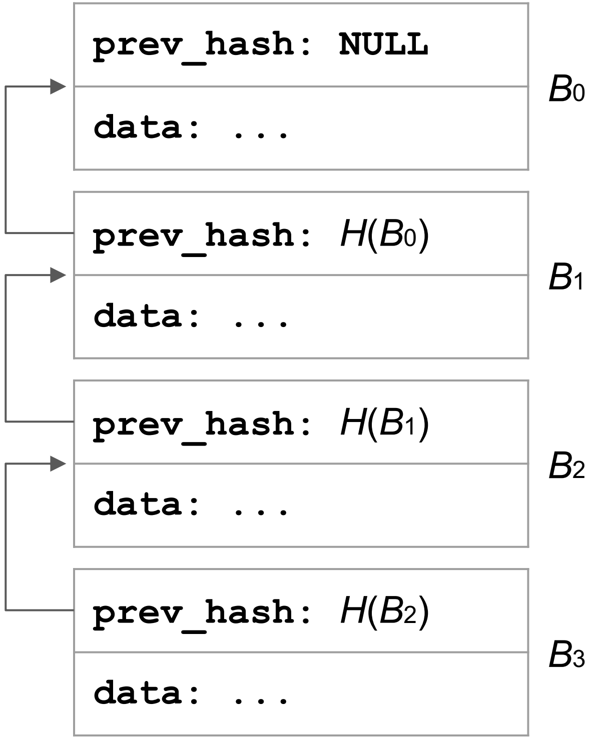
`prev_hash:` $H(B_0)$
`data: ...`
B_1

`prev_hash:` $H(B_1)$
`data: ...`
B_2

`prev_hash:` $H(B_2)$
`data: ...`
B_3

Hash Chains (Blockchain)

- Result
  - The latest hash represents a hash over all previous nodes
  - Changing data changes the block's hash, which changes the next block's hash, etc.
- This is really just an append-only linked list
  - Git uses this: Each commit contains a hash of the previous commit



`prev_hash: NULL`
`data: ...`
B_0

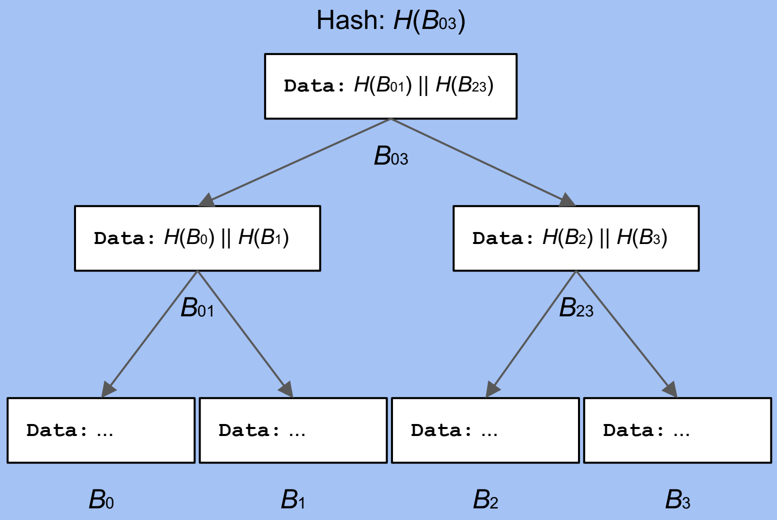
`prev_hash:` $H(B_0)$
`data: ...`
B_1

`prev_hash:` $H(B_1)$
`data: ...`
B_2

`prev_hash:` $H(B_2)$
`data: ...`
B_3

Merkle Trees

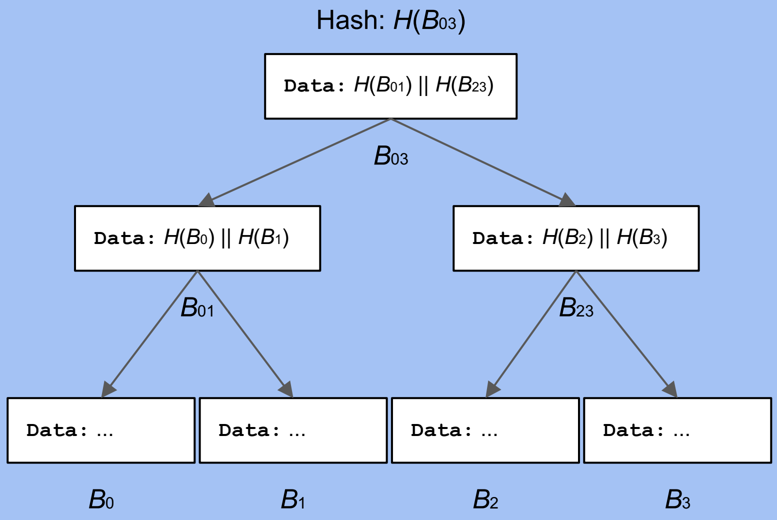
- What we want: A data structure where we can modify a node and then compute a hash over all nodes efficiently
  - Appending a node of size *n* to a structure with *m* nodes should take $O(n + \log m)$ time
- Idea: Instead of hashing all nodes at once, combine hashes as a binary tree
  - Each node is a hash of the two child node's hashes
- To modify:
  - Modify and re-hash the block
  - Re-hash the parent nodes until you reach the root



Merkle Trees

ITIS 6200 / 8200

- Result
  - The top hash represents a hash over all nodes
  - Can easily modify any node and compute $O(\log m)$ hashes in order to compute a new top hash
- Bitcoin uses a linear chain instead of a tree, but Merkle trees appear in many other applications



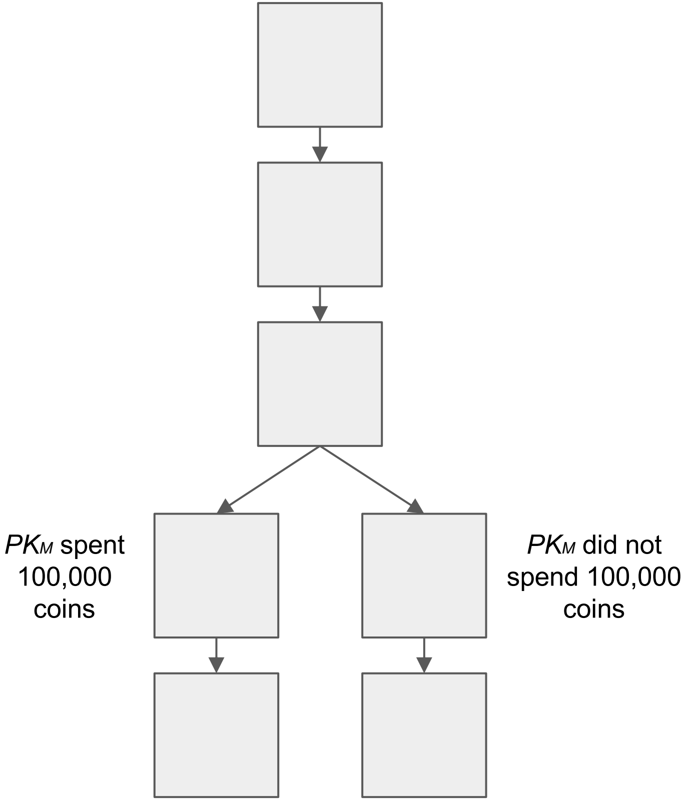
The Public Ledger

- Our previous scheme needed a public, trusted ledger
    - Append-only, immutable
    - Accessible to all parties
    - We can’t trust any individual to maintain the ledger for us
- What about Bitcoin?
    - Distributed ledger
    - **Consensus**: A way to reach agreement on something as a group, without placing trust in any individual

Forking Attacks

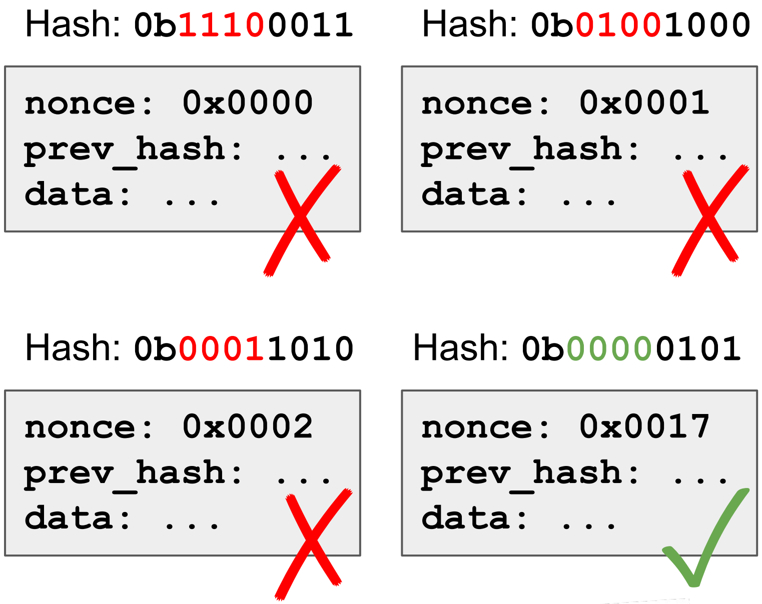
- Do hash chains allow a group of individuals to reach a consensus?
    - **No.** If Mallory creates an alternate history where she didn’t spend 100,000 coins, this is still a valid hash chain!
    - Need a way to agree on the head of the blockchain

- Proof of work


PK$_M$ spent 100,000 coins
PK$_M$ did not spend 100,000 coins

Proof of Work

- In each block, include a random number (“nonce”)
- For a block to be valid, the hash of the block must start with some number of 0’s
  - If you want 4 0’s, each nonce has a $1/2^4 = 1/16$ chance of creating a valid block
- Each user keeps track of their own blockchain



Hash: 0b11100011

```
nonce: 0x0000
prev_hash: ...
data: ...
```

✗

Hash: 0b01001000

```
nonce: 0x0001
prev_hash: ...
data: ...
```

✗

Hash: 0b00011010

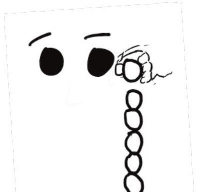
```
nonce: 0x0002
prev_hash: ...
data: ...
```

✗

Hash: 0b00000101

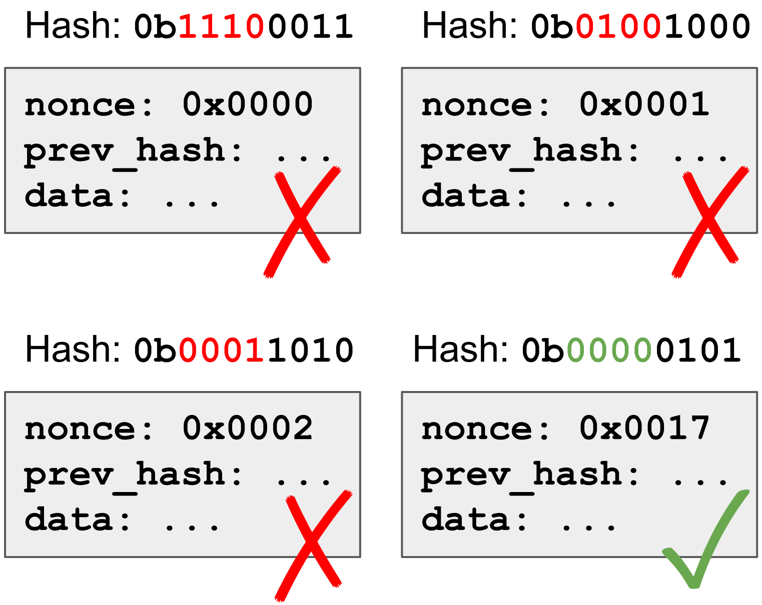
```
nonce: 0x0017
prev_hash: ...
data: ...
```

✓



Proof of Work

- Users “mine” a block by trying nonce values until a valid block is found
  - This requires trying many nonces to find one to produce the correct hash value for the block
  - The valid block is broadcasted to everyone else on the network
- When a valid block is received, users add it to their blockchain and start mining the next block



Hash: 0b11100011

```
nonce: 0x0000
prev_hash: ...
data: ...
```

✗

Hash: 0b01001000

```
nonce: 0x0001
prev_hash: ...
data: ...
```

✗

Hash: 0b00011010

```
nonce: 0x0002
prev_hash: ...
data: ...
```

✗

Hash: 0b00000101

```
nonce: 0x0017
prev_hash: ...
data: ...
```

✓

Proof of Work

- Consensus: The longest blockchain is the “true” blockchain
    - Everyone mines on the longest chain they know of
        - If we hear about a longer chain, we discard our current mining block and begin mining the longer chain
    - The more people agree to a chain, the more mining is done on that chain (more hashes per second)
    - The more mining is done on a chain, the faster the chain grows
- Defend against the forking attack
    - For Mallory to fork the blockchain, she must mine her new chain faster than the current “true” chain
    - To mine a chain faster than the current “true” chain, she must mine faster than every other (honest) node combined
    - This is infeasible!

Proof of Work: Examples

- What if two miners mine two blocks at the same time and append it to the blockchain, causing a fork?
  - The next miner that appends onto one of these chains invalidates the other chain. Longest chain wins.
- If a miner included your transaction in the latest block created, are you guaranteed that your transaction is forever in the blockchain?
  - No. There could have been another miner appending a different block at the same time, and that chain might be winning.
  - To confirm a transaction, wait for a couple more blocks to be appended to the chain with your transaction to ensure that it will probably win in the long term
- What happens if a miner who just mined a block refuses to include my transaction?
  - Hopefully, the next miner to mine a block is willing to include your transaction.

Incentivizing Mining

- Miners are incentivized to mine blocks in exchange for currency
    - Each signed transaction includes a small fee given to the miner of a block
        - Example: "... and pay 0.0001 coins to the miner of this transaction"
    - Each block may also give an agreed-upon number of free coins for the miner of the block
        - Example: Block *B* has 3 transactions and an additional transaction that reads, "PK_A receives 25 free coins"
- As more miners join the pool, the algorithm adjusts the number of 0's needed to mine a block
    - Generally, the time per block is targeted at a certain amount of time, so more global hash power ⇒ more 0's to make mining harder

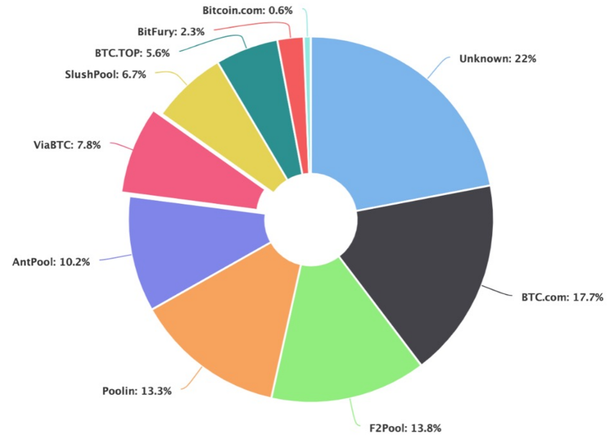
Attacks on Proof-of-Work

- **51% attack**: An attacker who controls 51% of mining power can effectively rewrite history and perform a forking attack
  - Creates a centralized authority: An entity is given control over the currency network
  - Proof-of-work makes this difficult: The attacker must control 51% of the world's computing power
- Bitcoin's security relies on the assumption that no attacker controls 51% of the world's computing power

The Trouble with Bitcoin

The Trouble with Bitcoin: Centralization of Power

- Bitcoin is often not decentralized in practice
    - Recall: In Bitcoin’s design, there is no centralized authority controlling the system
    - In practice, a few small groups have a lot of control over the Bitcoin system
- **Mining pools**: A team of users mining together
    - When one user receives a mining reward, everyone in the team shares the reward together
    - A user mining alone must get lucky to receive a reward
    - A mining pool gives users steady, smaller rewards
- A few large mining pools control most of the computing power in Bitcoin
    - If large pools team up, the 51% attack is possible!



The four largest mining pools combined control 55% of all Bitcoin hash power

The Trouble with Bitcoin: Centralization of Power

- Codebases are developed and maintained by a few groups
  - These groups can rewrite the code to affect the entire system
  - Example: When Ethereum was hacked, the developers changed the code to retrieve their money
- Some cryptocurrencies are not decentralized by design
  - **Private blockchain**: Only blocks signed by trusted private keys are valid
  - Only central authorities can sign blocks, but anyone can validate
  - Defeats the point of decentralized consensus!

The Trouble with Bitcoin: Pseudonymity

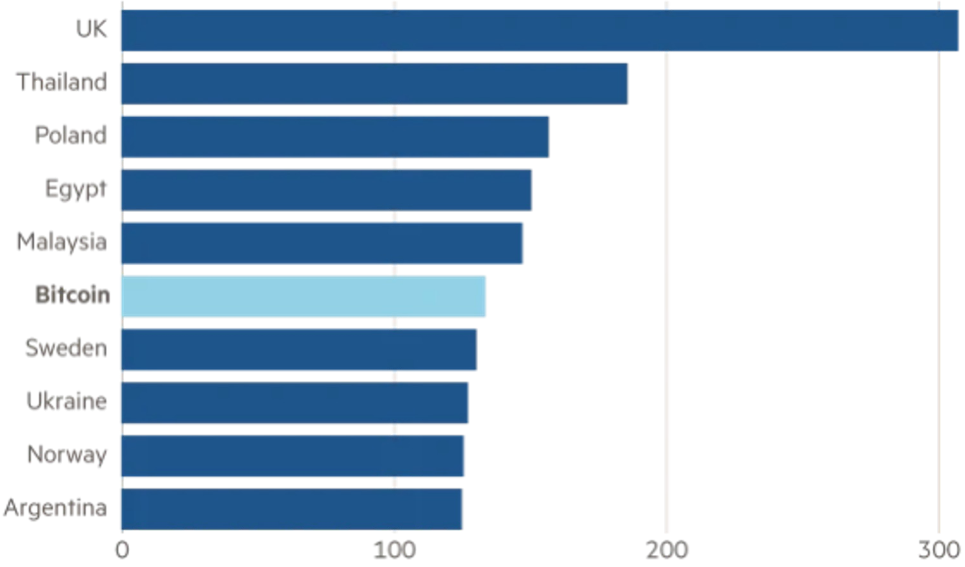
- Bitcoin is *pseudonymous*
  - **Pseudonymity**: Multiple actions can be linked to a single identity which is not your real identity, a pseudonym
    - The pseudonym is the public key
    - Your transactions can be linked to your public key
- Bitcoin transactions are not necessarily *anonymous*
  - **Anonymity**: Actions cannot be linked to your real identity
  - In theory, if you only ever use Bitcoin in an unpredictable manner, your pseudonym cannot be linked to your identity (anonymous)
  - In practice, your pseudonym can be linked to your real identity
    - If your shopping habits are predictable, it will be linked to you (the ledger is public)
    - If you exchange Bitcoin with real currency (dollars, euros, etc.), it will be linked to you

The Trouble with Bitcoin: Inefficiency

- Runtime inefficiency
  - Proof-of-work requires a huge amount of useless computational work
  - Bitcoin: Hashing random values until a hash starts with many zeros
- Storage inefficiency
  - Each Bitcoin user must store the entire transaction history to validate transactions
  - Alternative: Don’t store the entire transaction history
    - Problem: Now you have to ask a trusted source to send you the history
    - Defeats the point of decentralization!
- Result: Low processing capacity
  - Original design limited each block to 1 MB in size to defend against spam
  - Bitcoin can only process 3–7 transactions per second

The Trouble with Bitcoin: Power Consumption

- Bitcoin requires computers to waste energy on proof-of-work computations
- Bitcoin mining incentivizes energy waste
  - People seek mining rewards
  - If mining becomes more efficient, Bitcoin adjusts the mining problem to be harder, so efficiency gains are not realized



Today, Bitcoin consumes more electricity than entire countries (measured in terawatt-hours)

The Trouble with Bitcoin: Irreversibility

- Modern banking transactions are designed to be reversible
  - Protects against fraud: If your money is stolen, the bank can recover the money for you
- Bitcoin is not reversible
  - Once Bitcoin is sent, the transaction cannot be undone (unless the sender wants to return the money to you)
- You only need your private key to access your Bitcoin
  - If someone steals your private key, they can access all your money
  - If your Bitcoin is stolen, there is no way to recover it
  - Result: It is dangerous to store Bitcoin on any computer connected to the Internet

ITIS 6200 / 8200

Bitcoin in Practice

Breaking Proof-of-Work

- Proof-of-work viewed from an economic perspective
    - The system wastes $x per hour to defend against attacks
        - Each honest user contributes a small amount of the $x
        - Inefficiency: Money is constantly wasted, even if the service is not under attack
    - An attacker must spend more than $x per hour to control the system
        - Assumption: An attacker will not spend more than $x per hour
        - If an attacker can make more than $x per hour for an attack, they will spend $x per hour to execute the attack!
        - There are hashing services the attacker can pay to execute the attack

Proof-of-Work Alternatives

- Main problem: Proof-of-work is inefficient and wastes electricity
  - Proof-of-work is necessary to prevent an attacker controlling the system (makes it expensive to control the system)
- Alternative: Proof-of-stake
  - Proof-of-stake: Users with more coins have more mining power
  - People coins *stake* them in the system
  - If they act honestly, they gain more coins
  - If they act maliciously, their stake is slashed
  - Benefit: Uses much less electricity than proof-of-work, and the attacker must burn their own coins to attack the system!
  - Problem: Gives more power to wealthier users

Blockchain: Marketing and Buzzwords

- “Blockchain” is often marketed as brand-new technology, but it is mostly existing technology
  - Hash chains are over 20 years old: Linked timestamping services used hash chains and were proposed in 1990
  - Merkle trees were patented in 1979
  - Private blockchains are not new technology: Many existing applications already use append-only database structures
- “Blockchain” is a buzzword often applied to completely unrelated problems
  - “Use blockchain for electronic voting”
  - “Use blockchain to store medical records”
  - “Use blockchain to deliver vaccines”

Blockchain: Marketing and Buzzwords

ITIS 6200 / 8200

- Example of blockchain marketing: “Smart Contracts”
    - Smart contracts: Write an agreement in code, and execute the code to automatically follow the procedure
        - The code is stored on a blockchain so it cannot be modified
    - Problems with smart contracts
        - Unclear if they actually solve any real problem
        - Most “smart contracts” are actually standard finance bots: small programs to perform money transfers
        - If the code is vulnerable, you can violate the contract, with no way to reverse the effects
            - Contrast with legal contracts: Issues are handled by court systems

Bitcoin Enables Censorship Resistance

- Bitcoin has no central authority to block transactions
  - Bitcoin can be used for electronic payments that standard platforms would block
- Wikileaks: An organization that publishes leaked classified information
  - Used by whistleblowers to expose government corruption and corporate wrongdoing
  - Opposed and censored by many governments
- Bitcoin is used to support Wikileaks
  - Many major platforms refused any donations to Wikileaks
  - Bitcoin has no central authority, so nobody can stop the Wikileaks donations
- This is, generally, a good thing! But…

Bitcoin Enables Crime

ITIS 6200 / 8200

- Bitcoin is used for illegal transactions
  - Drug dealing
  - Money laundering
  - Illegal gambling
  - Hiring hitmen
  - Ransomware and extortion
- Bitcoin has no central authority to block illegal transactions
- Bitcoin is the most effective way to make illegal transactions

Economics of Bitcoin: Volatility

- Volatile currency: The value changes quickly
    - The value of Bitcoin changes far more often than standard currency

- Bitcoin is vulnerable to price shocks
    - Price shock: An extremely sudden change in value
    - When there are more transactions than the block capacity, prices increase
        - Users are competing for a limited number of transactions
    - Unknown attacks have also caused price shocks

- Result: Bitcoin behaves more like stock than currency
    - Users keep Bitcoin to try and grow their investment when the value of Bitcoin increases

Economics of Bitcoin: Currency Exchange

- Companies and people prefer to keep money in a more stable currency
  - To buy a product in Bitcoin, the buyer converts their standard currency to Bitcoin
  - The seller receives the Bitcoin and immediately converts it back to standard currency
- Users should be able to exchange Bitcoin for other currency (e.g. dollars)
- Buying and selling Bitcoin is difficult
  - Recall: Bitcoin transactions are irreversible
  - The buyer must trust that the seller will transfer the Bitcoin
  - The seller must trust that the buyer will pay when the Bitcoin is transferred
  - Ways to buy Bitcoin
    - Use another irreversible payment (e.g. cash)
    - Have a trusted relationship with the seller
    - Send a deposit first

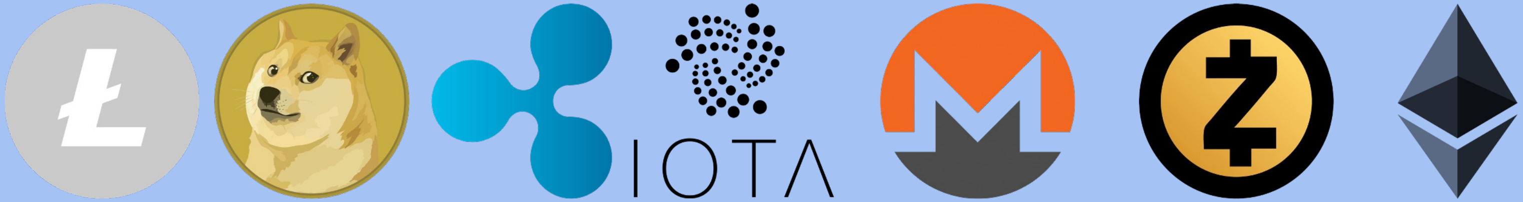
Economics of Bitcoin: Volatility

ITIS 6200 / 8200

- Stable currencies require reliable conversion to other currency
    - Someone must support easy conversion between Bitcoin and other currency
    - This is usually a centralized entity (e.g. a bank or a government)
    - The centralized entity violates Bitcoin’s main purpose
- Options for operating a conversion service
    - Follow government regulations (essentially becoming a regular currency)
    - Operate independently as a “wildcat bank” (similar to banks in the 1800s, before the US had a national currency)
    - Ignore federal regulations (e.g. Liberty Reserve, which was shut down by the government)
- Some cryptocurrency designs claim to be “algorithmic stablecoins”
    - Designed to be stable in the market
    - Most of these are snake oil that don’t work

Other Cryptocurrencies

- Most cryptocurrencies are based on the same principles
  - Public, append-only ledger structure
  - Designed with decentralization in mind
  - Some are software “forks” of the original Bitcoin blockchain
    - The fork ignores new Bitcoin blocks, and Bitcoin ignores fork blocks
- New cryptocurrencies are marketed with a distinguishing feature
  - Litecoin: Adds a catchy slogan
  - Dogecoin: Adds an Internet meme
  - Ripple: Centralized cryptocurrency with an additional settlement structure
  - IOTA: Designed its own brand-new cryptography (using trinary math)
  - Monero: Improves pseudonymity
  - Zcash: Adds real anonymity
  - Etherium: Adds million-dollar rewards for catching bugs



Cryptocurrency Scams

ITIS 6200 / 8200

- Public interest in cryptocurrency as a “get rich quick” scheme leads to fraud
- Many cryptocurrency frauds are old frauds with new technological branding
    - Ponzi schemes: Trick uninformed consumers to invest in a nonexistent product
        - Some “smart contracts” are actually modern Ponzi schemes
    - Wildcat banks: Independent, unregulated banks
        - If the bank shuts down, your money in the bank is gone
        - Physical wildcat banks stopped existing in the 1800s, but some cryptocurrencies essentially operate as wildcat banks
    - Unregulated securities: Stocks that are not regulated by the government
        - Often scams: The unregulated stock may be completely worthless
        - Initial coin offerings: Pay money now in exchange for some coins when the cryptocurrency launches later
        - If the cryptocurrency never launches, your money is gone

Bitcoin: Summary

- Goal: Create a currency system that does not rely on any central authority
- Identity: Each user is identified by their public key
- Transactions
  - Users sign transactions with their private key and add them to the ledger
  - Each transaction must reference a previous transaction to identify a source of money
- Public ledger
  - Hash chain: A linked list where each node contains the hash of the previous node
  - Append-only structure: Changing a node causes the hashes in all future nodes to change
  - Vulnerable to forking attacks: The attacker creates their own branch of the chain
- Proof-of-work
  - The blockchain only accepts blocks whose hash starts with a sequence of *n* `0`s
  - Finding valid blocks requires trying 2^n hashes. A reward is given to incentivize mining blocks
  - The longest hash chain is accepted as the true blockchain
  - An attacker must control 51% of the world’s computing power to create their own hash chain

The Trouble with Bitcoin: Summary

- Centralization of power: In practice, Bitcoin is controlled by a few groups
  - Mining pools: Teams of users mining blocks together
  - Codebase developers: Can change the code to alter the system
  - Private blockchains: Only trusted parties can append to the blockchain
- Pseudonymity
  - In theory, your transactions are only linked to your public key, not your true identity
  - With predictable transactions, your public key can be linked to your identity too
- Inefficiency
  - Proof-of-work requires a huge amount of hashing
  - Each user must store the entire blockchain
  - Bitcoin can only process a few transactions per second
- Power consumption: Hashing wastes electricity
- Irreversibility: Transactions are not reversible
  - If your Bitcoin is stolen, there is no way to recover it
- Reference
  - https://www.oreilly.com/library/view/mastering-bitcoin/9781491902639/ch08.html