Lecture 31

Digital Signature & Double Spending Problem

ECE 422: Reliable and Secure Systems Design

Instructor: An Ran Chen
Term: 2024 Winter

Schedule for today

- Key concepts from last class
  - Proof-of-Work (PoW)
  - Role of miners
  - Difficulty adjustment + The Longest Chain Rule
- Transactions and digital signatures
- Consensus for recording transactions
  - Balance checking
  - The Double Spending Problem
- Selenium brief introduction & IDE demo

Difficulty adjustment

With more participants and more computing power, the difficulty of the hash problem increases accordingly.

- Bitcoin automatically adjusts the difficulty after every 2,016 new blocks (in other words, every two weeks)
- New difficulty based on the number of participants in the mining network and their combined computational power

Week 1

Requirement: First 72 bits must all be 0s

Week 3

Requirement: First 73 bits must all be 0s

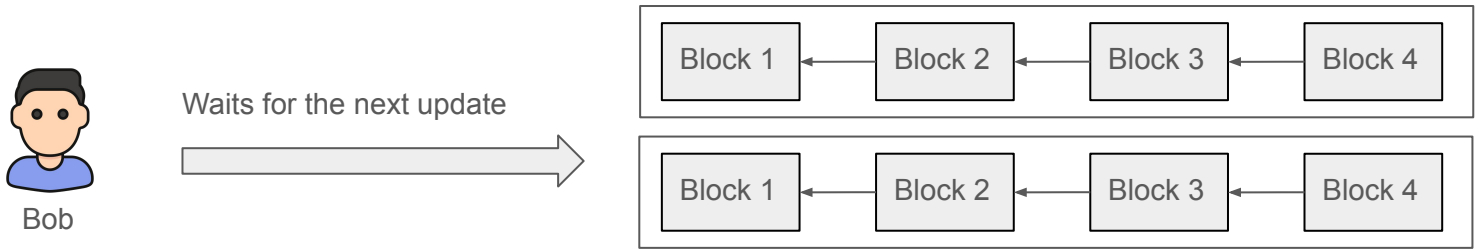
The Longest Chain Rule

The Longest Chain Rule allows every node on the network to agree on the same blockchain (e.g., same transaction history).

- Solving a consensus problem
- Protecting the immutability of the blockchain

Example of application:

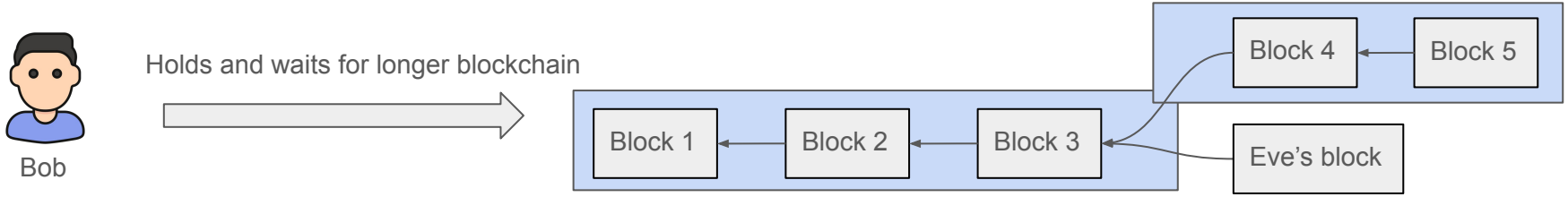
- For blockchains with the same length, Bob waits for the next block that makes one of blockchain longer



Why Proof-of-Work works?

Suppose Eve tries to send a block with fraudulent transactions:

- Eve first needs to find the special number based on the fraudulent transactions before everyone else, and broadcasts the blockchain
- Bob verifies the blockchain and copies it over
- However, Bob continues to listen to the broadcast
  - Any longer blockchain will replace the current one
  - For Bob to keep Eve's blockchain, Eve needs to keep extending the blockchain



Recap on mining principle

Mining is about creating a new block and verifying the transactions:

- First miner to find the special number gets to create the new block
  - Each block is represented by SHA-256(string + special number)
- Transactions are considered verified once the miner solves the hash problem
  - Proof-of-Work
- Difficulty of the hash problem is based on the total computational power in the network
  - Difficulty adjustment

What happens when a malicious node (pretending to be a Bitcoin user) broadcasts a fake transaction?

Schedule for today

- Key concepts from last class
  - Proof-of-Work (PoW)
  - Role of miners
  - Difficulty adjustment + The Longest Chain Rule
- Transactions and digital signatures
- Consensus for recording transactions
  - Balance checking
  - The Double Spending Problem
- Selenium brief introduction & IDE demo

Digital signature in blockchains

Digital signatures are used in blockchains to verify the authenticity of transactions.

- Objective: Every node must prove that they are authorized to spend the funds (i.e., having enough balance)
- At the same time, they must prevent other nodes from spending their funds

In Bitcoin, Bitcoin miners verify the authenticity of transactions:

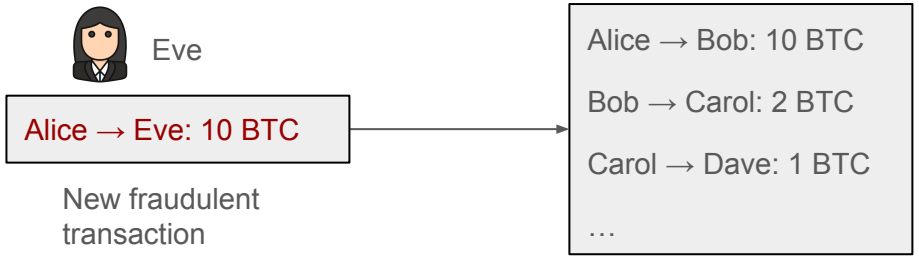
- Strategy: Verify the digital signature
- Check all other miners' work to agree on a correct state

The authenticity is based on three properties: Ownership + Integrity + Association

Fraudulent transaction

What happens when a malicious node broadcasts a fake transaction?

- In a decentralized ledger system, anyone can add a new transaction
  - E.g., Eve, a malicious user, can broadcast a new transaction: Alice → Eve: 10 BTC
  - What prevents Eve writing a new transaction without Alice's approval?



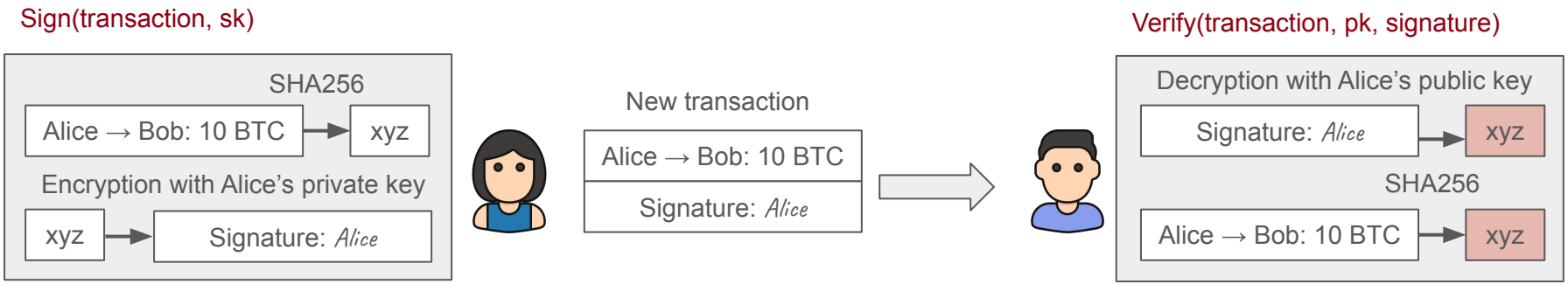
Each transaction is approved by a digital signature

Digital signature

Each transaction is signed by the owner (or owners) of the source funds

Different transactions generate unique signatures:

- Signed using a secret/private key (sk)
  - Sign(transaction, sk) → signature
- Verified using a public key (pk)
  - Verify(transaction, pk, signature) → accept | reject



Digital signature in blockchains

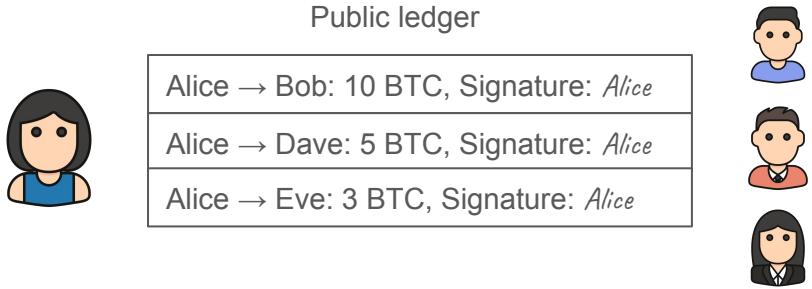
Digital signature verifies the following properties:

- Ownership: Digital signature belongs to Alice
  - Public and private keys for encryption and decryption
- Integrity: Transaction has not been modified
  - Decryption(Signature) = SHA256(Transaction)

Potential threat to the current protocol

Suppose the following protocol:

- For each transaction, Alice adds her signature
- Alice broadcasts each transaction to everyone else
- What can go wrong?
  - Assume that the network is secure (no transaction interception)
  - Hint: Public ledger (anyone can add new transaction) + Session fixation attack



Another fraudulent transaction

Problem: Malicious users may broadcast fraudulent transactions

- By replicating a valid transaction + signature
- Analogous to Session Fixation attack
  - Malicious user can copy over the signature + session ID

Public ledger

Alice → Eve: 3 BTC, Signature: *Alice*
Alice → Eve: 3 BTC, Signature: *Alice*
Alice → Eve: 3 BTC, Signature: *Alice*

Solution: Having each transaction generate a unique signature

- By adding transaction IDs as part of the transaction (generate a different hash value)

Public ledger

1. Alice → Eve: 3 BTC, Signature: *Alice1*
2. Alice → Eve: 3 BTC, Signature: *Alice2*
3. Alice → Eve: 3 BTC, Signature: *Alice3*

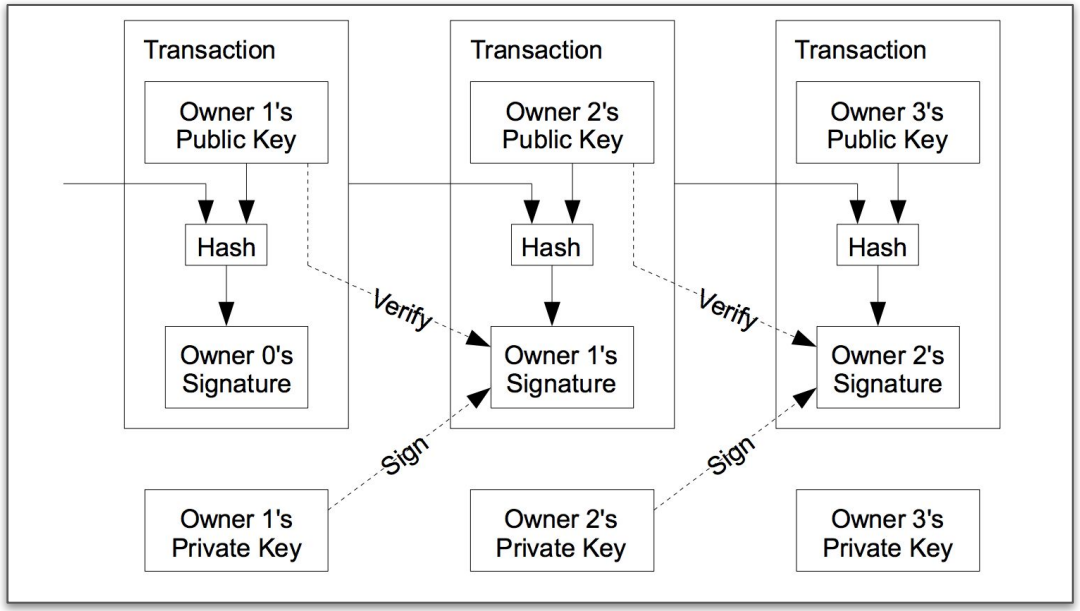
Digital signature in blockchains

Digital signature verifies the following properties:

- Ownership: Digital signature belongs to Alice
  - Public and private keys for encryption and decryption
- Integrity: Transaction has not been modified
  - Decryption(Signature) = SHA256(Transaction)
- Association: Digital signature is associated with a particular transaction
  - Unique transaction ID

Bitcoin Script

There are multiple implementations of digital signature for Bitcoin in the real world. It depends on the particular scripting system used for transactions.



Another common implementation of digital signature from the white paper

For more information:

- Bitcoin Script
- White paper
- Schnorr Signature
- Elliptic Curve Digital Signature

Schedule for today

- Key concepts from last class
  - Proof-of-Work (PoW)
  - Role of miners
  - Difficulty adjustment + The Longest Chain Rule
- Transactions and digital signatures
- Consensus for recording transactions
  - Balance checking
  - The Double Spending Problem
- Selenium brief introduction & IDE demo

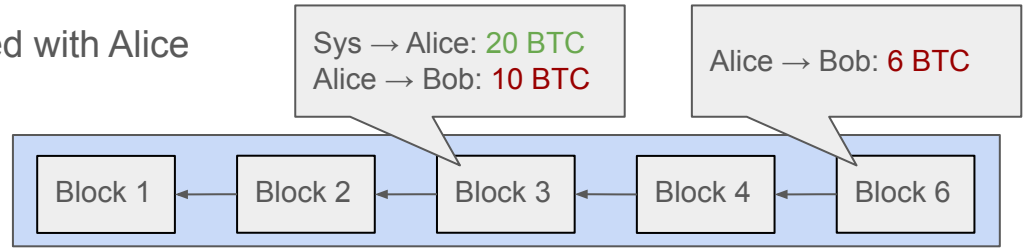
Balance checking

Bitcoin uses blockchain as a means to transparently record the transactions as well as the balance.

- Blockchain holds the complete transaction history
- Everyone can compute someone else's balance

Example: Alice → Bob: 5 BTC

- Alice (Bitcoin user) broadcasts the transaction
- Bitcoin miners verify the transaction
  - Identify every transactions associated with Alice
  - Calculate the remaining balance
  - Accept or reject

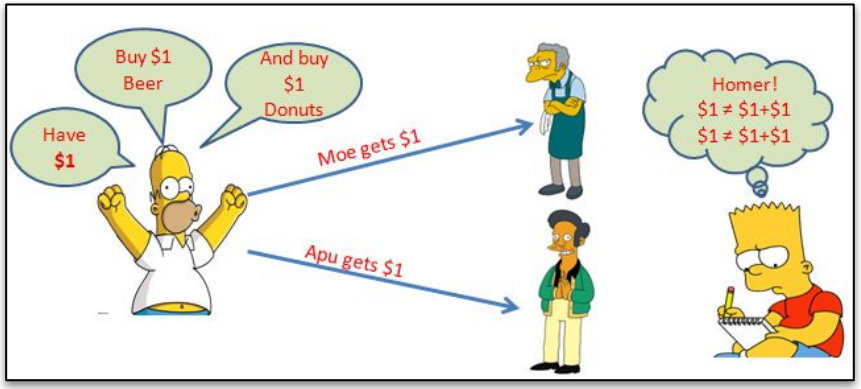


The Double Spending Problem

The Double Spending Problem describes a problem in transactions where the same BTC are spent more than once.

- Public ledger: Anyone can broadcast transactions
- Analogous to The Byzantine Generals problem

- Challenge 1: Agreeing on a single truth with decentralized systems
- Challenge 2: Reaching consensus even if some of them are faulty or malicious (Byzantine Fault Tolerance)



The Double Spending Problem

Suppose that Eve has 5 BTC

- Eve broadcasts two transactions at the same time
  - Eve → Bob: 5 BTC; Eve → Alice: 5 BTC;
- Bitcoin miners record the first transaction that they receive (network delays)
  - Carol (miner) records: Eve → Bob: 5 BTC
  - Dave (miner) records: Eve → Alice: 5 BTC
- However, both miners will wait until a block is created to confirm the transaction
  - Carol finds the special number and broadcasts the blockchain
  - Dave gives up his block and records Carol's block, following the Longest Chain Rule

The Double Spending Problem

Satoshi Nakamoto did not solve the Byzantine Generals Problem, but proposed a workaround using Proof-of-Work.

- Building trust based on The Longest Chain Rule

Challenge 1: Agreeing on a single truth with decentralized systems

- Trust the blockchain with the most work done (longest blockchain)

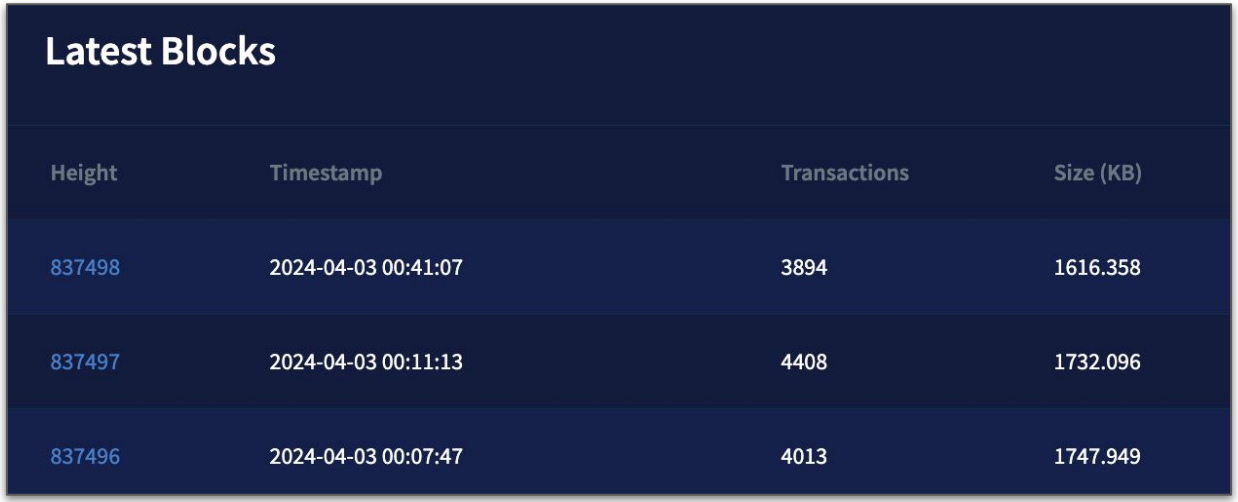
Challenge 2: Reaching consensus even if some of them are faulty or malicious

- Malicious nodes must done more work than the rest of the network to make the honest nodes trust their blockchain

Schedule for today

- Key concepts from last class
  - Proof-of-Work (PoW)
  - Role of miners
  - Difficulty adjustment + The Longest Chain Rule
- Transactions and digital signatures
- Consensus for recording transactions
  - Balance checking
  - The Double Spending Problem
  - BTC Explorer
- Selenium brief introduction & IDE demo

BTC Explorer



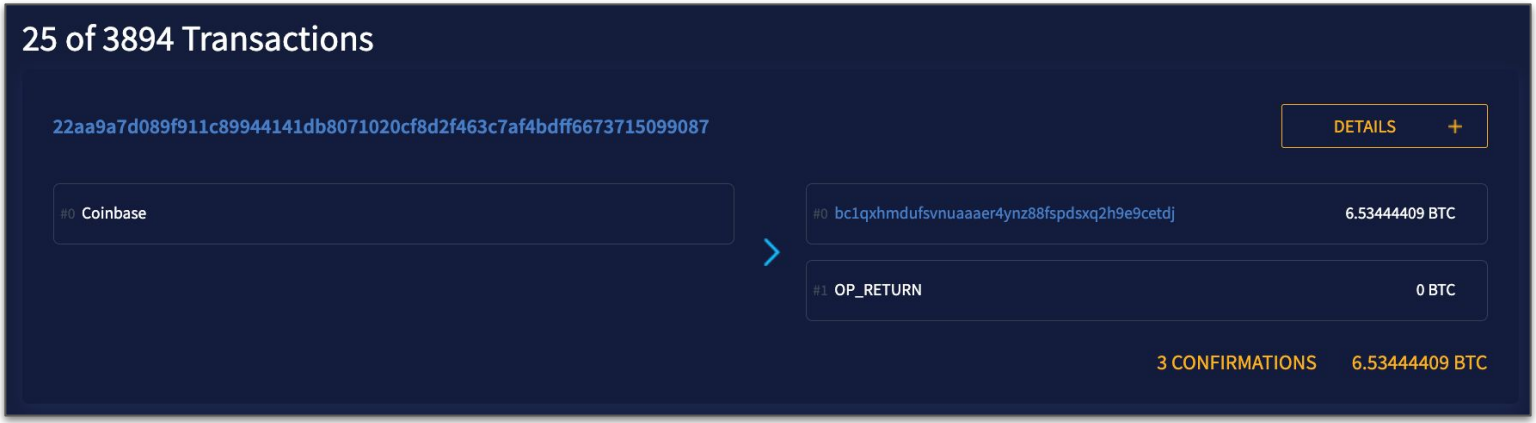
Latest Blocks

Height	Timestamp	Transactions	Size (KB)
837498	2024-04-03 00:41:07	3894	1616.358
837497	2024-04-03 00:11:13	4408	1732.096
837496	2024-04-03 00:07:47	4013	1747.949

BTC Explorer

- Height: Block number
- Timestamp: Time that the block was mined
- Transaction: Number of transactions in the block
- Size: Size of the block

Coinbase transaction



A coinbase transaction is the first transaction in a block.

- Transaction created by a miner
- Used to collect the block reward for their work and transaction fees
- Current block reward: 6.25 BTC
- Next Bitcoin halving on April 19, 2024: 3.125 BTC

Schedule for today

- Key concepts from last class
  - Proof-of-Work (PoW)
  - Role of miners
  - Difficulty adjustment + The Longest Chain Rule
- Transactions and digital signatures
- Consensus for recording transactions
  - Balance checking
  - The Double Spending Problem
- Selenium brief introduction & IDE demo

What is Selenium?

Selenium is an open-source, automated testing tool used to test web applications across different browsers.

- Goal: Automate testing process in software development life cycle (SDLC)

Selenium test suite comprises of four tools:

- Selenium IDE: plugin for recording user interaction and playing back
- Selenium Remote Control (RC) (deprecated): server that allows users to write application tests
- Selenium WebDriver: remote control interface for writing application tests
- Selenium Grid (mostly for Continuous Integration): server for parallel execution of tests on different browsers and different operating systems

How can Selenium be useful?

- Essential tool for QA analyst
  - Regression Testing
  - Integration with Continuous Integration (CI) Pipelines
- Project proposal for web developer
  - Functional bugs in UI
  - Cross-Browser Testing
- Assistance for UI tester
  - Complex Use Case / Application Flows
  - UI/UX Testing
- (Legal) Web scraping tool for freelancer
  - Data mining and extraction

Selenium IDE: The Basics

A Simple Demo: Assert text “Territorial Acknowledgement” element exists

Step 1 - Launch your Chrome menu and open the Selenium IDE plugin.

Step 2 - Select “Record a test in a new project.” Provide the name for your test.

Step 3 - Provide a link to the UAlberta webpage. The IDE starts recording by navigating to this web page.

Step 4 - Once the website opens, click on “OUR STORY” in the header of the webpage.

Step 5 - Scroll to the bottom of the page, select “Territorial Acknowledgement”

Step 6 - Now, modify the recording and change “Command” to “assert text”, enter “Value” as “Territorial Acknowledgement”