LearnThings.Online

Bitcoin and Cryptocurrencies

This Course is from edX, scroll down and click “read more” for original link.

Developed by Blockchain at Berkeley and faculty from UC Berkeley’s premier Computer Science department, this course presents Bitcoin and cryptocurrencies as the motivation for blockchain technologies, and provides a comprehensive and in-depth overview of the fundamental concepts of the crypto space with a particular emphasis on Bitcoin.

The course covers basic properties of bitcoin, the mechanics behind it (e.g. including cryptographic hash functions, Bitcoin Script, privacy, and hash commitment schemes) and its roots in the Cypherpunk movement and Libertarian ideals. You’ll learn about practical applications of Bitcoin such as wallets and mining, as well as how to destroy bitcoins, including network attacks and malicious mining strategies. We will also take a brief look at Ethereum and how blockchain can be used outside of cryptocurrencies.

This course is open to anyone with any background. Whether you are planning your next career move as a blockchain developer, crypto trader, data analyst, researcher, or consultant, or are just looking for an introduction to the Bitcoin technology. This course will help you to begin developing the critical skills needed to future-proof your career.

This course is part of the Blockchain Fundamentals Professional Certificate program. If you are planning to enroll in the entire series, we suggest starting with this course and then progressing on to CS198.2x Blockchain Technology.

Bitcoin Protocol & Consensus: A High Level Overview

We begin with some fundamental concepts such as the basic properties and intent of centralized/decentralized currency. We then build an in-depth understanding of Bitcoin from the ground up, divided into four stages: Identity, Transactions, Record Keeping, and Consensus.

Blockchain History: From the Cypherpunk Movement to JP Morgan Chase

This module delves into the origins and historical significance of Bitcoin. We look into the roots of Bitcoin in the Cypherpunk movement and Libertarian ideals, and examine the revolutionary significance of Bitcoin as opposed to some of its early predecessors. We then move onto exploring the history of the crypto space as a whole.

Bitcoin Mechanics & Optimizations: A Technical Overview

We examine the in-depth mechanics behind Bitcoin, such as the Bitcoin network, cryptography and cryptographic hash functions, Bitcoin Script, privacy, and hash commitment schemes.

Bitcoin In Real Life: Wallets, Mining, and More

We examine the most frequently used real world aspects of Bitcoin, such as wallets, wallet mechanics, mining, transactions, and Bitcoin governance. We explain the various ways one can interface with the Bitcoin network, depending on the specific software they run.

Game Theory & Network Attacks: How to Destroy Bitcoin

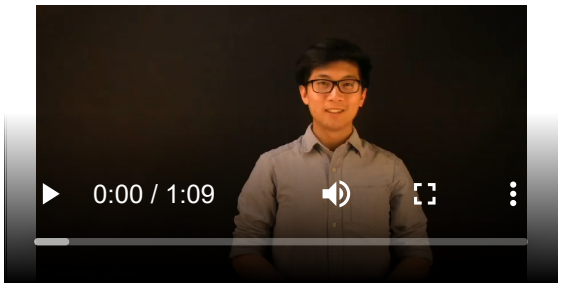
We look into how to destroy Bitcoin, including various network attacks. Specifically, we look into vulnerabilities such as pool cannibalization, double spending and forking attacks, network attacks, the Goldfinger attack, malicious mining profit strategies, and 51% attacks.

Ethereum & Smart Contracts: Enabling a Decentralized Future
This module focuses on the properties behind the second largest blockchain platform, Ethereum. We introduce the Ethereum Virtual Machine and the idea of Turing completeness, and examine some of the key protocol differences between Bitcoin and Ethereum, such as the UTXO vs. accounts model and functionality. We then look into some of the use cases of Ethereum, and conclude with an overview of smart contracts and building decentralized

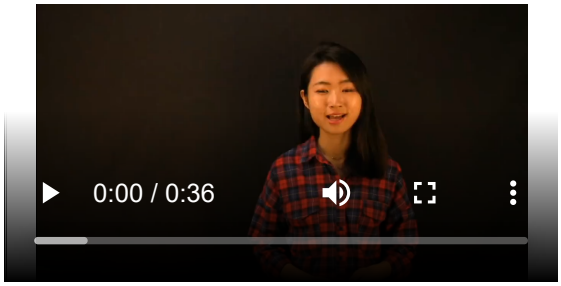
applications. Having spent the last modules primarily on cryptocurrencies, this module encourages students to think about blockchain use cases outside of cryptocurrency.

Introduction: Welcome to the Course

Welcome to Week 1



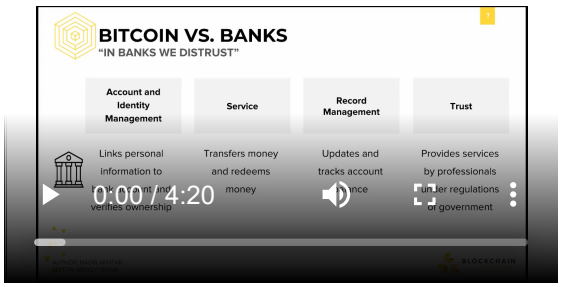
Intro: What is Bitcoin?



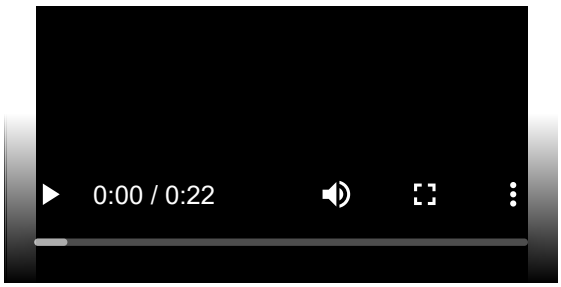
Subscribe

What is Bitcoin?

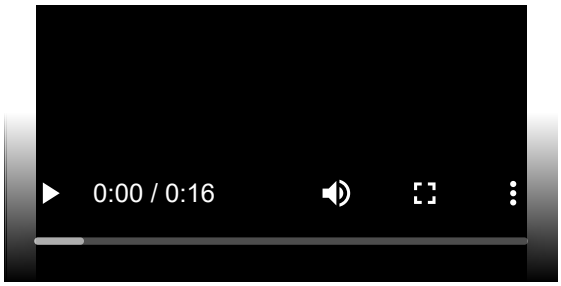
Bitcoin vs. Banks



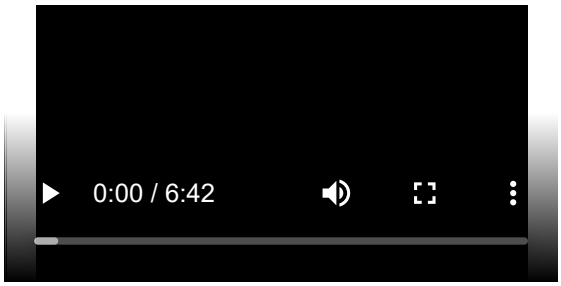
Intro: Bitcoin From the Ground Up



Intro: Identity (Stage 1)

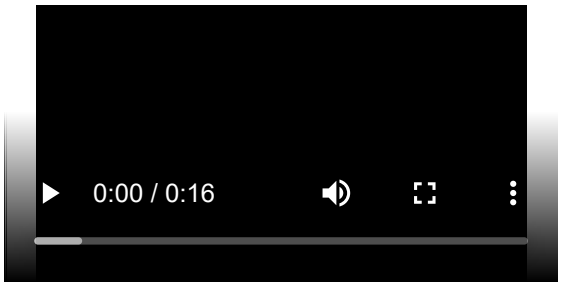


Identity (Stage 1)

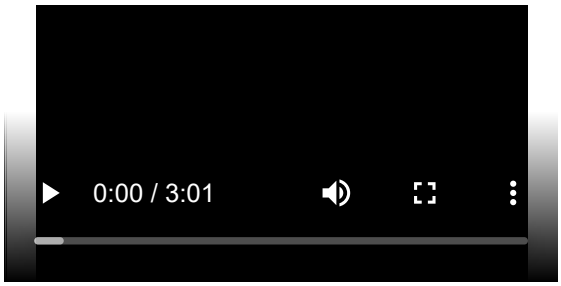


Subscribe

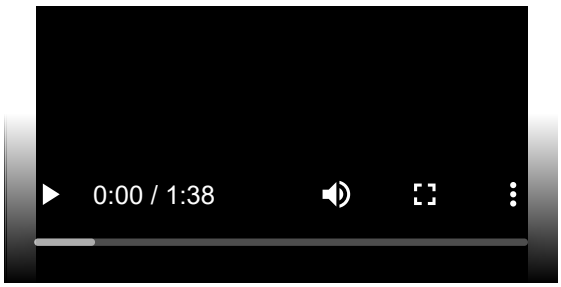
Intro: Transactions (Stage 2)



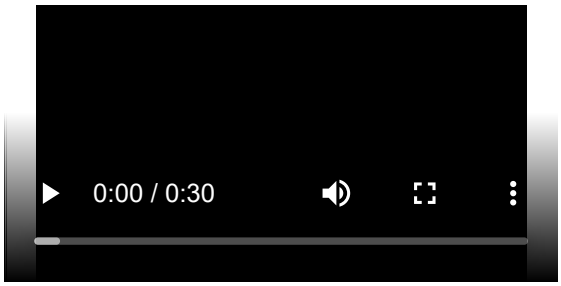
Transactions (Stage 2)



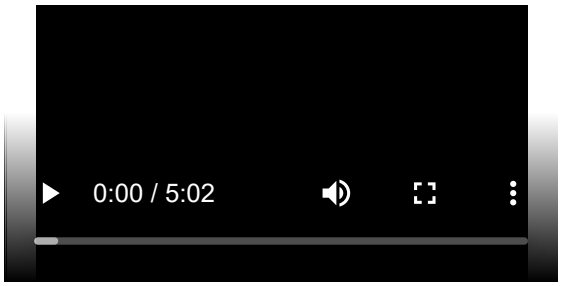
Demo: UTXOs



Intro: Record Keeping (Stage 3)

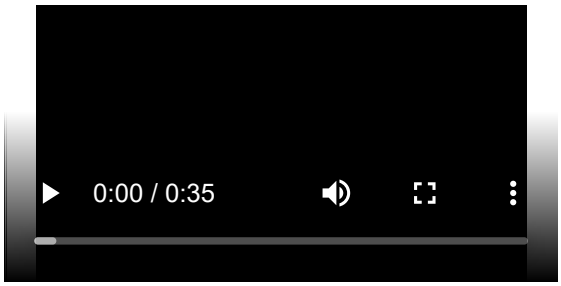


Record Keeping (Stage 3)

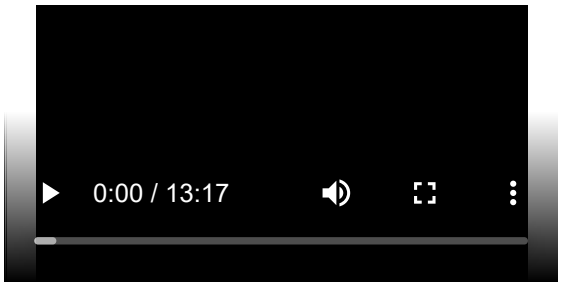


Subscribe

Intro: Consensus (Stage 4)



Consensus (Stage 4)



Supplement: 51% Attacks

Author: Rea Savla

51% Attack

The Bitcoin network requires Proof-of-Work to add new blocks to the blockchain. Users in the Bitcoin network vote on new blocks, and come to consensus on whether or not new blocks should be included in the blockchain. Proof-of-Work ties voting power to computational power rather than digital identity, and was designed to prevent the Sybil attack, where a malicious actor creates many identities to skew the vote. However, due to an uneven distribution of computational power, Satoshi Nakamoto's "1 CPU 1 Vote" vision is not reflected perfectly in reality.

Bitcoin's correct operation hinges on one key assumption: that there is an honest majority of computational power. An honest majority would be able to mine faster than a malicious minority, and thus have a higher probability of creating the next block. Once the network comes to consensus on these new blocks, generally it is in a miner's best interest to follow protocol and mine on the longest observed blockchain. The longest chain is seen as the "true" valid transaction history because it has had the most work put into it. Therefore, the majority defines the transaction history.

However, if a malicious entity controls more than 50% of the mining power (say 51%), it has the majority and is now able to mine an alternative chain (with a different transaction history) and make it the longest chain. Bitcoin users would then accept that chain as the "true" transaction history. This happens for the same reasons that an honest majority might be able to maintain the longest chain. Giving power to the majority is a requirement of the decentralization Bitcoin aims to achieve. If we allow an honest minority to control the transaction history, then we've created a centralized entity consisting of the collection of these honest actors and thus defeated the purpose of decentralization.

With 51% of the mining power, malicious actors can double spend, and use the same bitcoins for two different transactions. A malicious actor may send the same bitcoin to a third party and then to itself, choosing to include and validate the latter transaction and avoiding payment altogether.

One entity holding majority mining power also makes the network susceptible to the Goldfinger Attack, where a 51% is used to destroy the value of a cryptocurrency.

We'll explore all these attacks in further detail in future course modules.

Forking

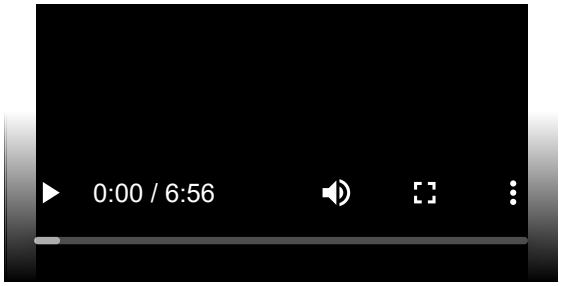
Sometimes, different miners may create different blocks, either intentionally (e.g. double spending) or unintentionally, to add at the same point on the blockchain. This creates multiple chains: multiple different versions of the transaction history. We say that the blocks are competing at the same block height, and that there has been a fork. Following protocol, miners eventually resolve the fork and agree upon one of the chains to be the valid blockchain, and continue to build blocks upon it.

While some forks occur naturally, and some are the result of double spending attempts, there also exist purposeful cases of forking, used to make changes to the Bitcoin protocol.

Two categories of these protocol changes are soft and hard forks. Soft forks implement protocol updates that strictly reduces the set of valid transactions, while hard forks, conversely, allow for previously invalid transactions to become valid.

We'll explain more in the course modules for Week 4.

Video: Bitcoin Review



Text: Lecture 1 Summry
Author: Rea Savla

I. What is Bitcoin?

Cryptocurrency is a completely digital, formless currency, tied together using computer science, cryptography and economics. Bitcoin is the first and most widely used cryptocurrency.

Bitcoin is inspired by the Cypherpunk Movement in the 1980s, which advocated for protection of privacy from external entities using cryptography. Satoshi Nakamoto first outlined and created Bitcoin in 2008 and 2009.

Bitcoin aims to be pseudonymous, trustless, decentralized, and immutable. In addition, anyone with a computer and internet connection can join the Bitcoin network. Each computer is a node in the Bitcoin network, and each node may verify and audit the transaction history of their own funds. In Bitcoin, the minting and distribution of bitcoins is determined through mining; since anyone can mine and win bitcoins, this process also aims to be decentralized.

Some of the challenges Bitcoin addresses are:

- The difficulty to ensure every Bitcoin node holds a consistent version of the transaction history
- The difficulty to identify malicious actors

These conditions may normally allow a node to conduct a Double Spend Attack, in which the one spends the same funds more than once by tricking parts of the network to believe

different versions of the transaction history. However, Satoshi overcame this problem using blockchain and Proof-of-Work.

Bitcoin is robust because it serves the same functions as a bank:

- Account management; the Bitcoin protocol gives users a way to create and manage their own identities (account)
- Legitimacy; It ensures we are legitimate owners and accessors of our accounts
- Record-keeping; It honestly records account balances at each transactions.

Unlike a bank, Bitcoin is decentralized and ensures a high degree of privacy and trust.

Trust is built on the blockchain due to a high level of transparency: blockchain is a publicly verifiable ledger, not owned by any entity, and it prevents any single point of failure.

To maintain this trust, we need identity in Bitcoin for authentication and assigning blame. Bitcoin uses public keys to send funds and private keys to prove ownership of the public key and redeem the sent funds. Each individual is responsible for creating and managing their own private and public keys. Public keys are generated from Private keys and are used to send/receive funds. Private keys are randomly generated and used to prove ownership of the public key. The chances of guessing the same private key are very low.

In addition to proof of ownership, in order to be considered valid, transactions must also have enough available funds to spend from and guarantee that no other transaction uses the same funds. Bitcoin uses the Unspent Transaction Output (UTXO), in which users spend directly from transactions made to them.

Instead of storing transactions individually, Bitcoin batches them into “blocks,” built off of their previous blocks, thus forming the tamper-evident, blockchain data structure.

Users on the blockchain must come to a consensus on which updates and blocks to add to the blockchain. Doing so also prevents Double Spend Attacks. Bitcoin uses a form of peer validation to build a shared transaction history; everyone on the network casts votes on the validity of a transaction. To prevent a Sybil Attack, where users create multiple identities for malicious purposes, Bitcoin employs Proof of Work, where voting power is based on computational power, to make voting expensive.

Previous Next

Readings
The shy college student who helped build Bitcoin into a global phenomenon Opens in new window

The Essence of How Bitcoin Works (Non-Technical) Opens in new window

(Optional) The Untold Story of Silk Road, Part 1 Opens in new window

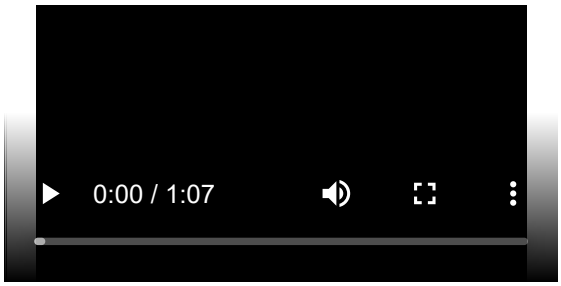
(Optional) Bitcoin: A Peer-to-Peer Electronic Cash System Opens in new window

(Optional) Bitcoin Developer Guide Opens in new window (up to but not including "P2PKH Script Validation")

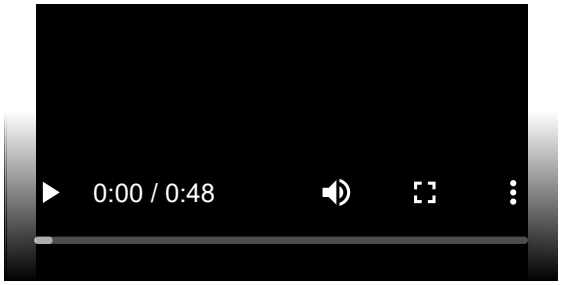
(Optional) How Bitcoin Works in 5 Minutes (Technical)

Blockchain History: From the Cypherpunk Movement to JP Morgan Chase

Welcome to Week 2

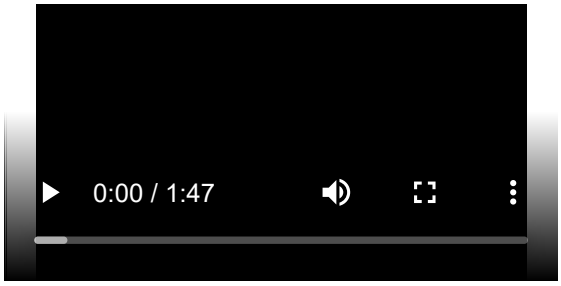


Intro: Pre-Bitcoin

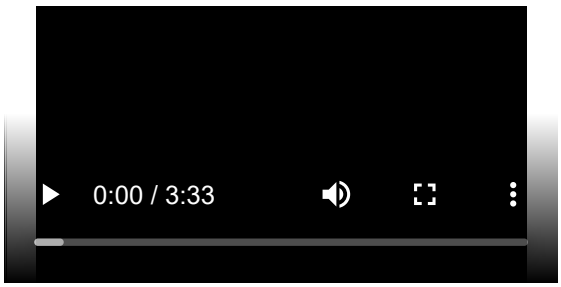


Subscribe

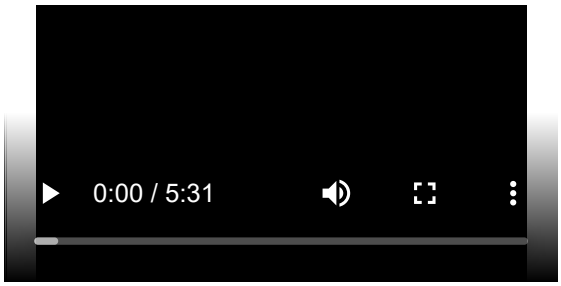
Libertarian Dreams



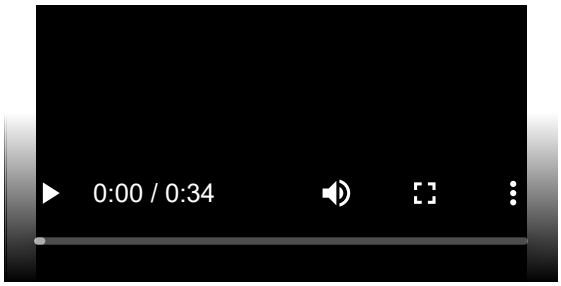
Bitcoin Precursors



Bitcoin Invention

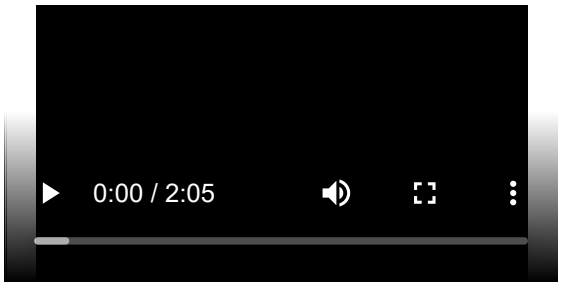


Intro: Early Bitcoin

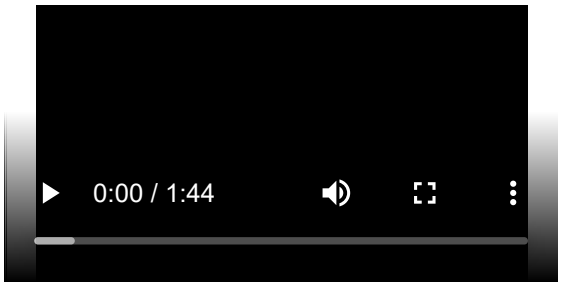


Subscribe

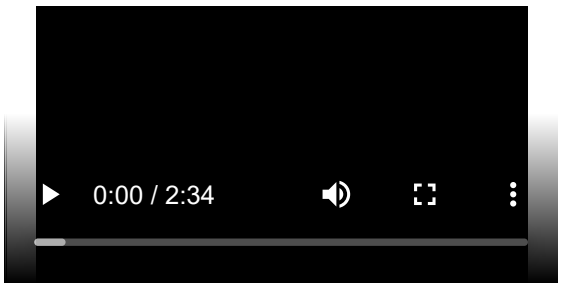
Hacks and Scandals



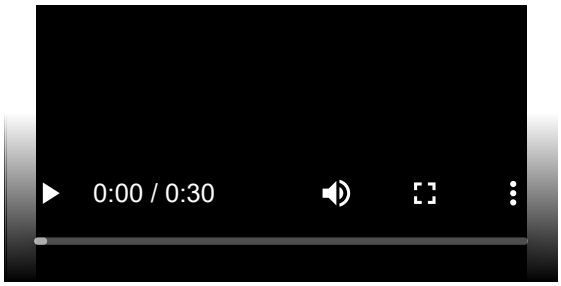
Bitcoin Bubble and Altcoins



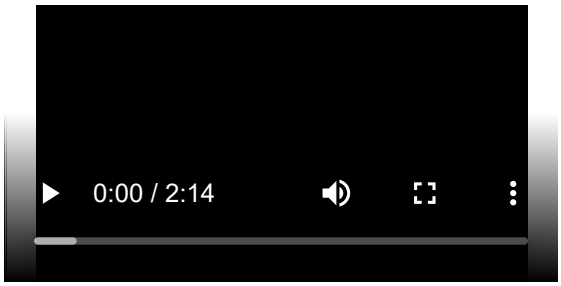
Bitcoin Headlines



Intro: Scalability

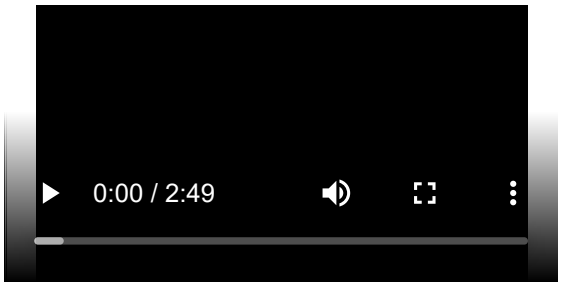


Bitcoin Struggles to Scale

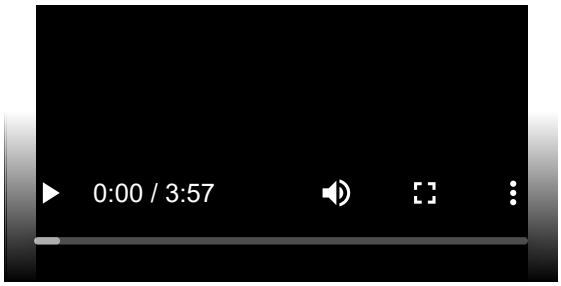


Subscribe

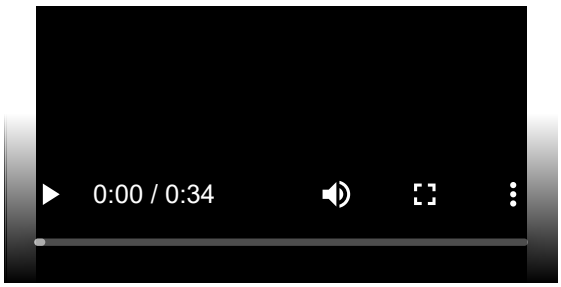
Ethereum Timeline



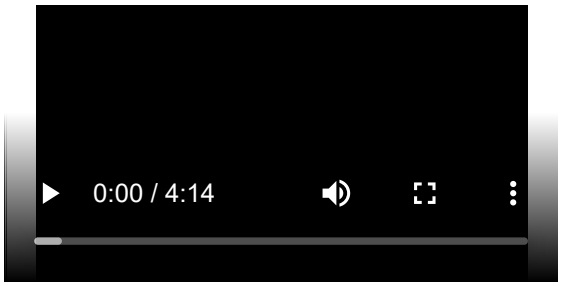
Ethereum Bubble



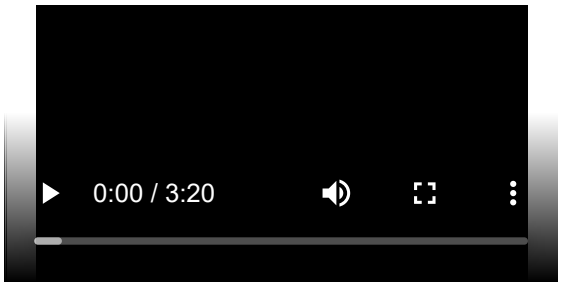
Intro: Enterprise Blockchain



Banks and Blockchain

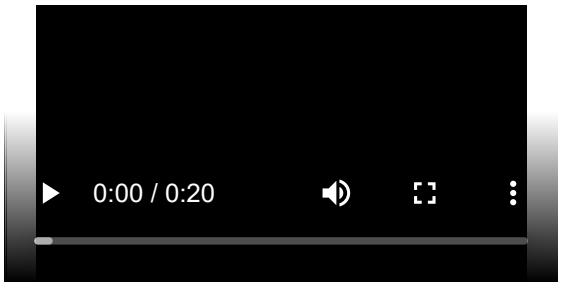


Blockchain Community and Politics

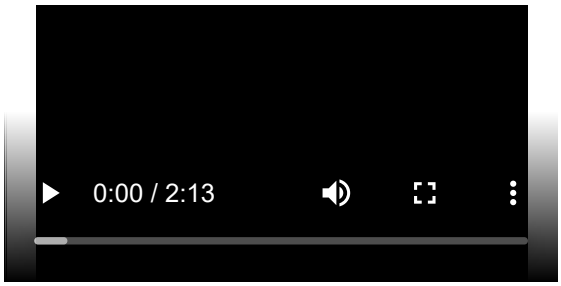


Subscribe

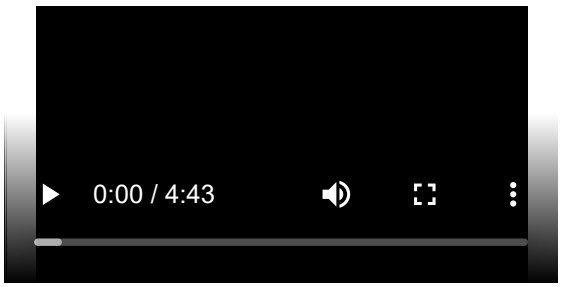
Intro: State of the Industry



ICOs



Industry Projects



Text: Lecture 2 Summary
Author: Rea Savla

I. Pre-Bitcoin: Libertarian Dreams

In the face of increasingly powerful banks and national agencies, the Cypherpunks and Crypto-anarchists of the late 1980s advocated the use of cryptography to preserve privacy, which they defined as the power to selectively reveal oneself. They sought to develop an anonymous digital transaction system.

In October 2008, Satoshi Nakamoto released the Bitcoin whitepaper, which outlined, for the first time, an anonymous, trustless, decentralized cryptocurrency.Bitcoin was built upon a history of failed cryptocurrencies, including Digicash, Hashcash, and B-money. Bitcoin relies on Proof-of-Work, a peer validation protocol introduced by Hashcash, that expends computational power to solve cryptographic puzzles and to cast votes. As in Digicash, each node in Bitcoin maintains their own identity through public and private keys, authenticating transactions using blind signatures. As in B-money, every Bitcoin full node maintains a copy of the blockchain. Bitcoin is a deflationary currency, with 21 million total bitcoins that will be slowly introduced to the bitcoin supply via block rewards.

The first transaction using Bitcoin to purchase a tangible asset occured in 2010, when Laszlo Hanyecz exchanged 10,000 bitcoin for $25 worth of pizza. This exchange gave bitcoins real value, a first step towards legitimacy.

Image of Laszlo Hanyecz's pizzas that he ordered online. His post: "I just want to report that I successfully traded 10,000 bitcoins for pizza.

II. Early Bitcoin: Scandals, Hacks, and Illegal Activity

As Bitcoin began increasing in popularity, it also began facing increasing cases of thefts, hacks, and illegal activity. The first was a large hack on Magic: The Gathering Online Exchange (Mt. Gox), one of the first Bitcoin exchange website created in 2010.

Bitcoin was also used for the purchase of illegal substances on the dark web, especially through the website “Silk Road,” nicknamed the “eBay for Drugs.” As Bitcoin became more accessible and useful to the public, it steadily grew in value, reaching a bubble in 2013.

Altcoins, such as Litecoin, ZCash, Stellar, Ripple, Ethereum, Dogecoin, DASH, and Monero, began popping up soon after Bitcoin’s success, each serving a different functionality.

In 2014, merchants, such as Overstock.com and PayPal (with Coinbase), started accepting Bitcoin. Bitcoin startups and wallet companies, including Coinbase, Bitpay, and Blockchain.info, started appearing around this time as well. Around this time, people started to differentiate between the term blockchain from Bitcoin.

III. Scalability Debates and Ethereum

Bitcoin is far from perfect. One of the biggest technological challenges Bitcoin faces is that of scalability. Such issues raise concern of decentralized governance. Currently, Bitcoin users may propose Bitcoin Improvement Protocols on online forums and gather ad hoc community votes on proposed matters.

Another influential blockchain platform is Ethereum. While Bitcoin is a storage of value, Ethereum is a platform designed to execute arbitrary code called “smart contracts”. Users pay for code execution using Ethereum’s internal token, called ether. With Ethereum’s introduction, Distributed Autonomous Organizations (DAOs), programs on the Ethereum blockchain that create a distributed government, gained popularity. “The DAO” was a decentralized Venture Capital DAO that suffered a $120 M hack in 2016. Disagreement about whether to roll back the history of transactions to before the hack led to two versions of Ethereum. Ethereum(the main chain) rewound the transaction history to before the hack, whereas Ethereum Classic continued the original chain despite the hack.

While concerns of the SEC’s reaction to The DAO hack initially lowered Ethereum’s price, the growing popularity of Initial Coin Offerings and cryptocurrency exchange-traded-funds increased Ethereum’s price overall. Economic and political changes in early 2017 along wit an expansion of the crypto user base to include millennials further contributed to growth i Ethereum value and transaction volume.

IV. Enterprise Blockchain

Meanwhile, banks started seeking ways to apply blockchain technology leading to an increased interest in “private blockchains.” Enterprise blockchain technologies today include R3’s Corda, Chain, JP Morgan’s Quorum and Juno, Digital Asset Holdings, and IBM’s hyperledger. Blockchain has come a long way — from online ideation amongst Cypherpunks to adoption by JP Morgan Chase.

The blockchain space has expanded to include not only major financial institutions, but also the general public. Initial Coin Offerings, equity-less fundraising schemes for new crypto startups that allow anyone to participate, and the advent of Cryptokitties, an online marketplace for virtual cats, indicate the spread of blockchain across communities and also to popular culture.

Readings
A Cypherpunk’s Manifesto Opens in new window by Eric Hughes

Coindesk: A Bot Named Willy: Did Mt. Gox’s Automated Trading Pump Bitcoin’s Price? Opens in new window

The DAO, The Hack, The Soft Fork and The Hard Fork Opens in new window

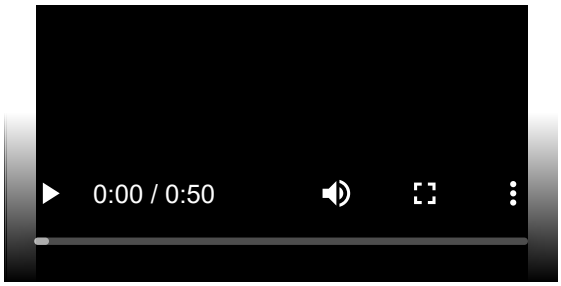
(Optional, to Get Ahead) Princeton Textbook 1.1 Cryptographic Hash Functions (pages 23-31)

(Optional) All You Need to Know About ICOs Opens in new window

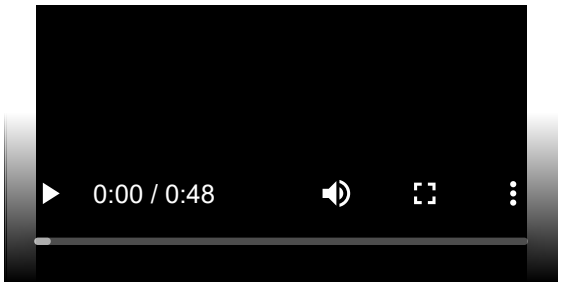
(Optional) Digital Gold Opens in new window by Nathaniel Popper

Bitcoin Mechanics & Optimizations: A Technical Overview

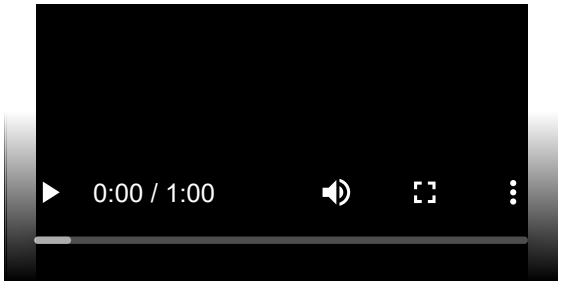
Welcome to Week 3



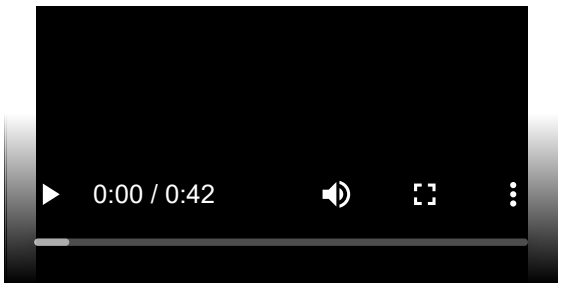
Intro: Motivations and Definitions



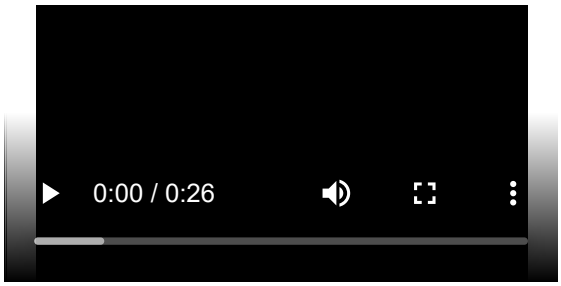
Motivations and Definitions: Integrity of Information



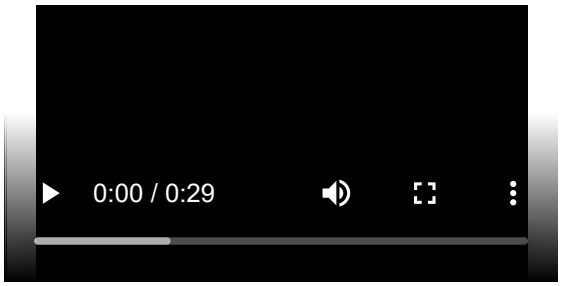
Motivations and Definitions: Cryptographic Hash Functions



Key Properties of Cryptographic Hash Functions

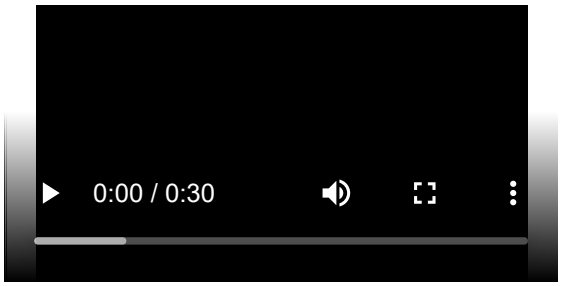


Key Properties of Cryptographic Hash Functions: Preimage Resistance

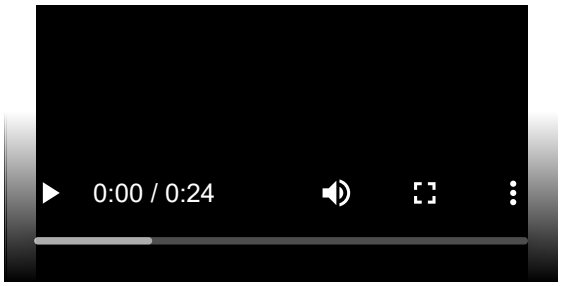


Subscribe

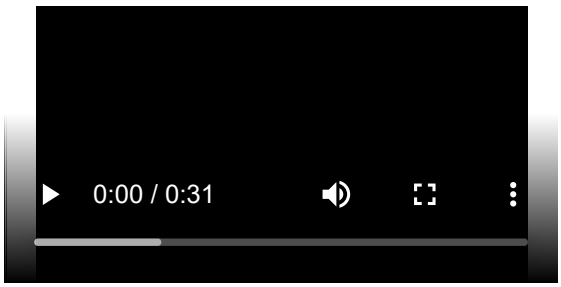
Key Properties of Cryptographic Hash Functions: Second Preimage Resistance



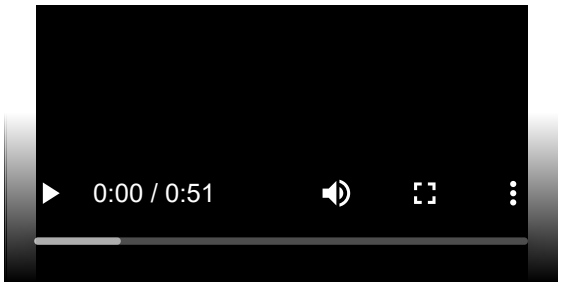
Key Properties of Cryptographic Hash Functions: Collision Resistance



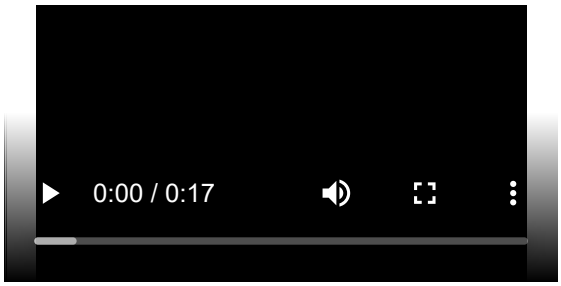
Key Properties of Cryptographic Hash Functions: Avalanche Effect



SHA256d

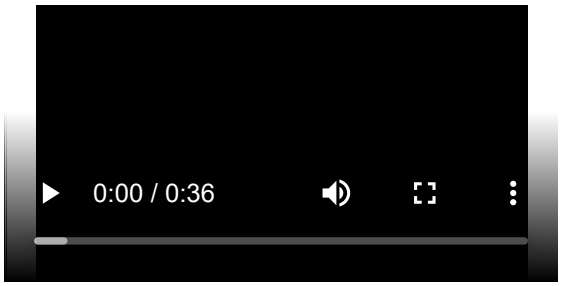


Intro: A Tamper-Evident Database

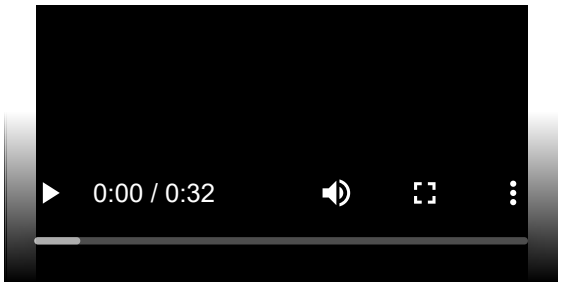


Subscribe

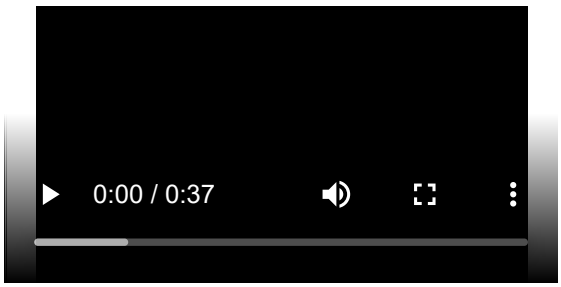
Tamper-Evidence



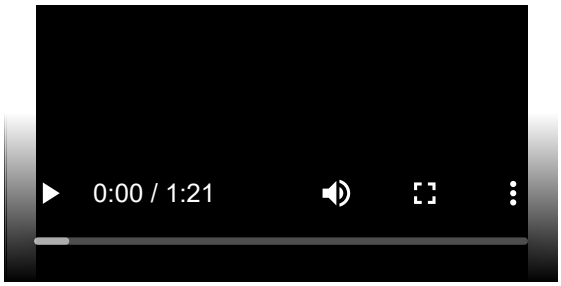
Dissecting a Block



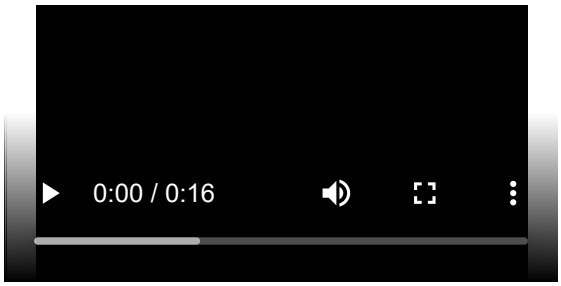
Dissecting a Block: The Block Header



Merkle Trees

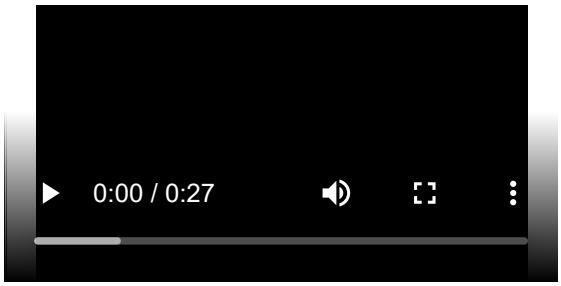


Merkle Trees: Tamper Detection

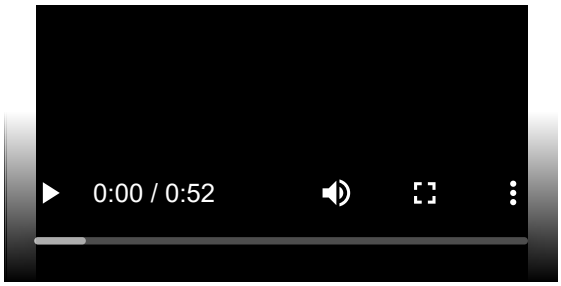


Subscribe

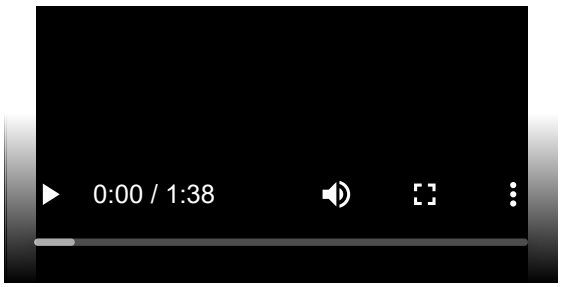
Merkle Trees: Proof of Inclusion



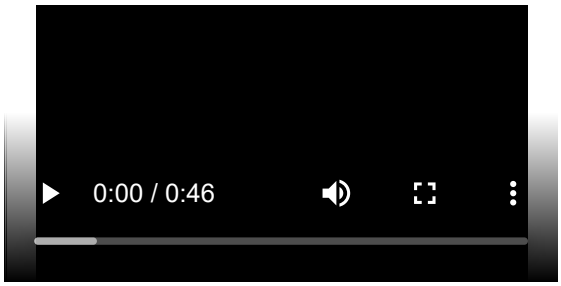
Previous Block Hash



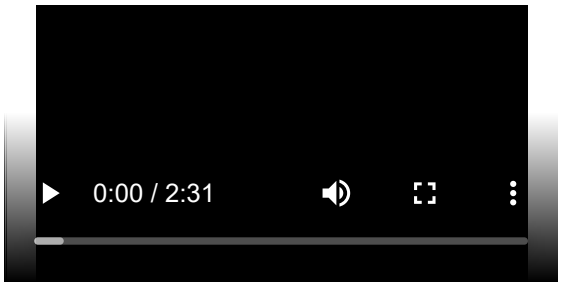
Proof-of-Work: Partial Preimage Hash Puzzle



Proof-of-Work: Mining

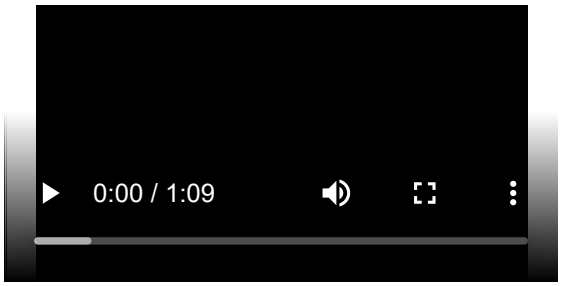


Proof-of-Work: Block Difficulty

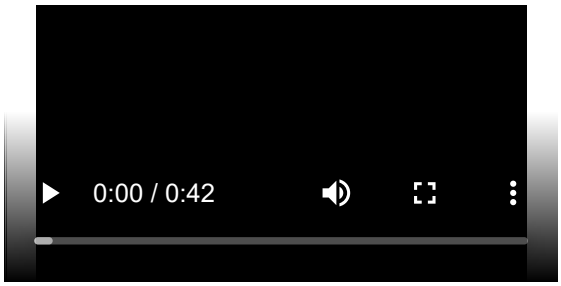


Subscribe

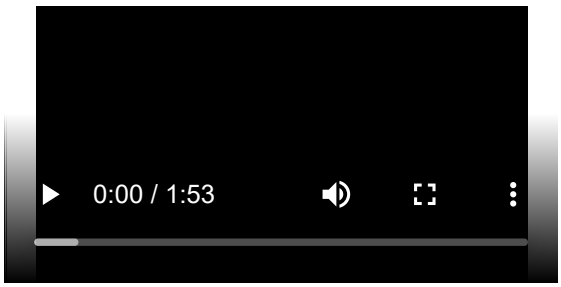
Proof-of-Work: Coinbase Transaction



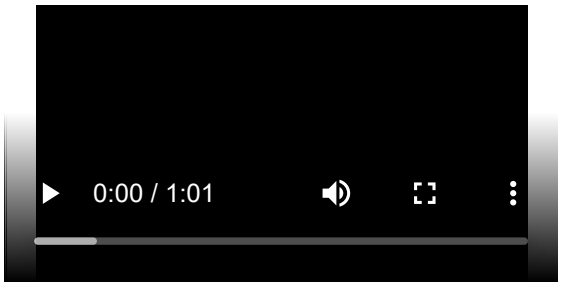
Intro: Signatures and Authentication



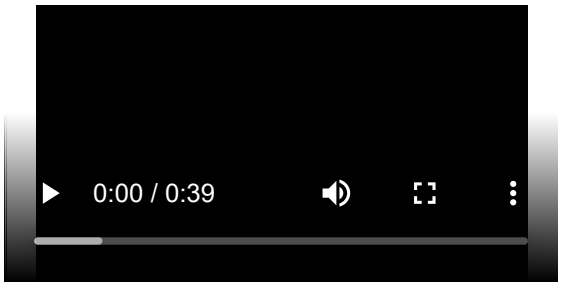
Digital Signature Schemes



DSS Key Definitions and Takeaways

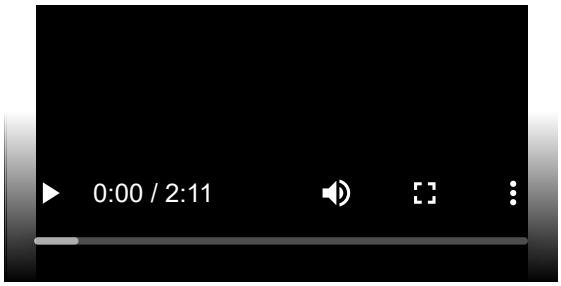


From Private Keys to Addresses

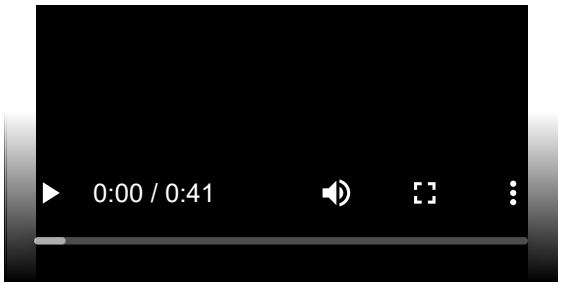


Subscribe

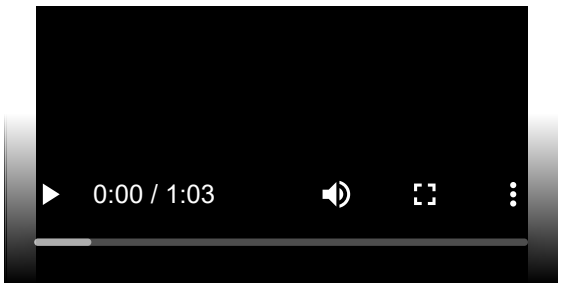
Elliptic Curve Cryptography



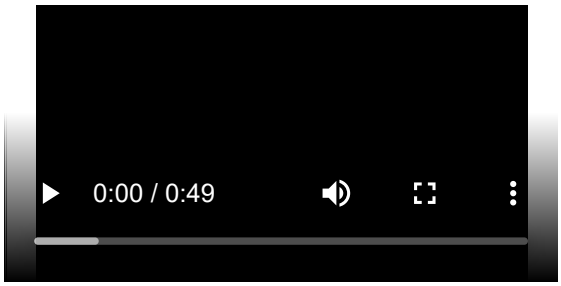
Elliptic Curve Cryptography: Demo



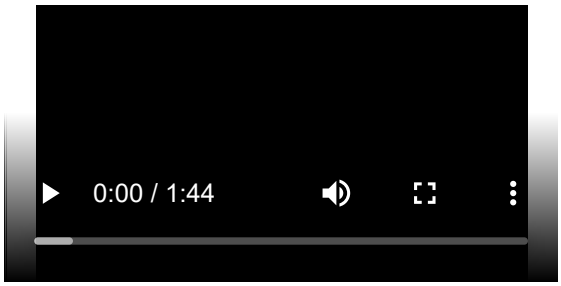
Elliptic Curve Cryptography: Security



Public Key to Public Key Hash

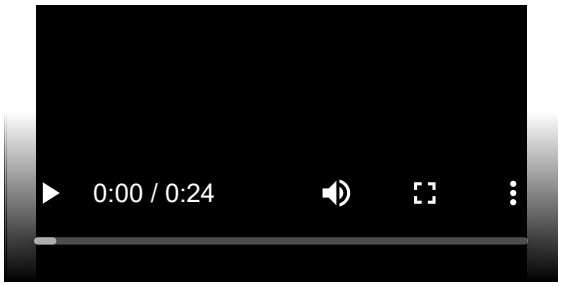


Public Key Hash to Address

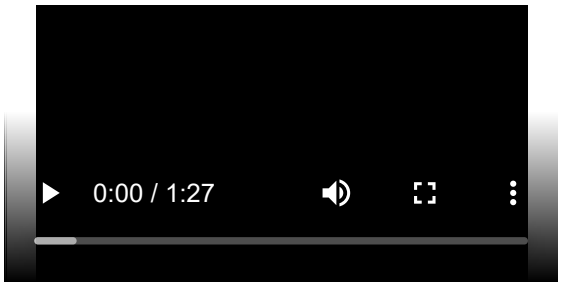


Subscribe

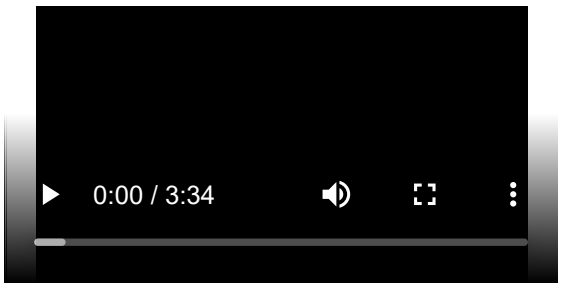
Intro: Bitcoin Script



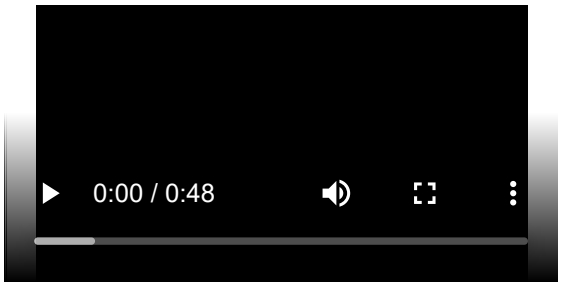
Remember the UTXO Model?



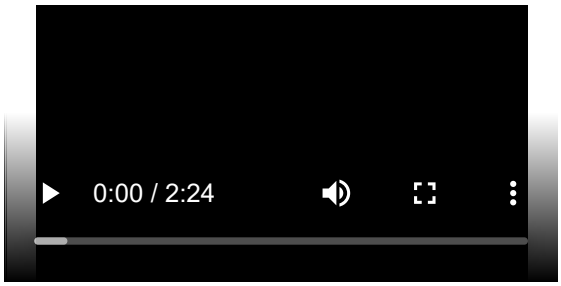
Contents of a Transaction



Bitcoin Script Reminders

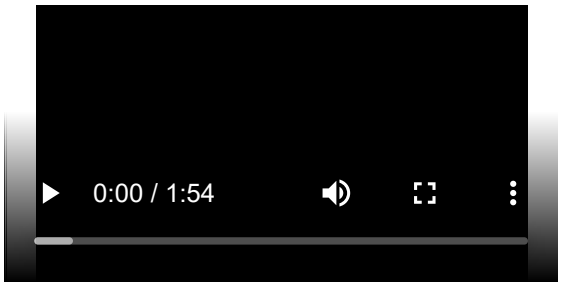


Intro to P2PKH

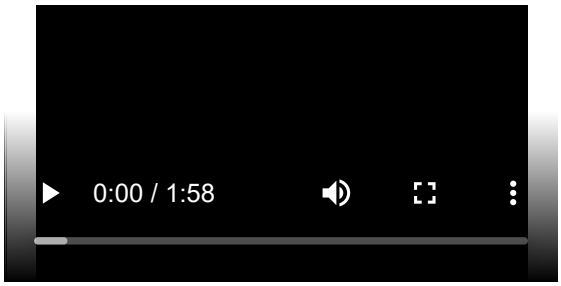


Subscribe

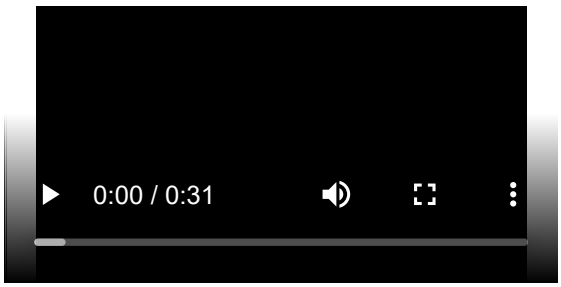
Demo: P2PKH Example



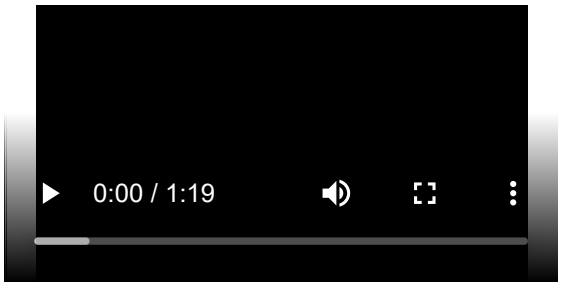
Proof of Burn



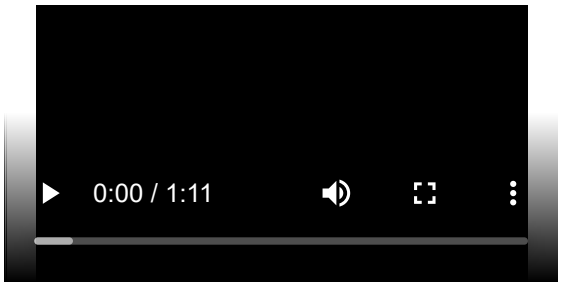
Intro: Advanced Bitcoin Script and P2SH



P2PKH vs P2SH

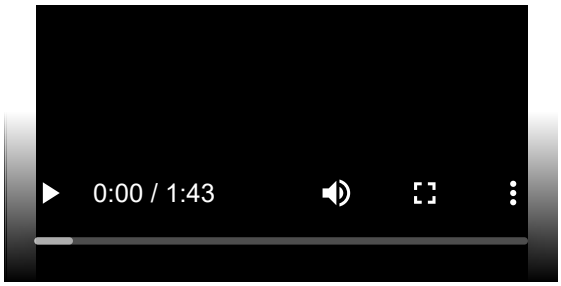


How P2SH Works

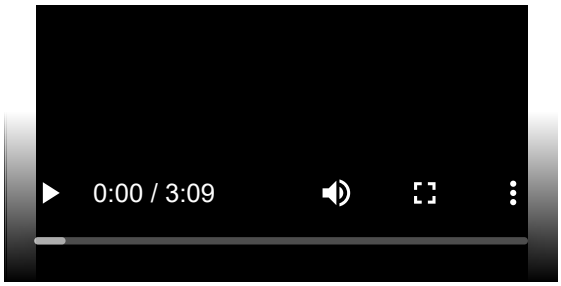


Subscribe

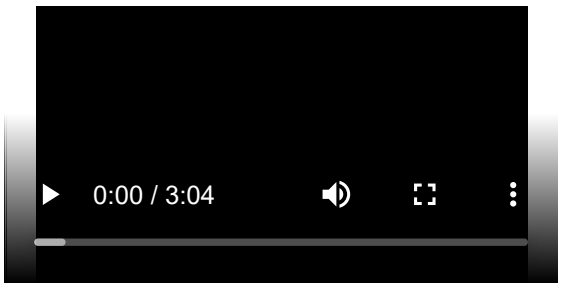
Why use P2SH?



Multisignature



Timelocks



Text: Lecture 3 Summary

Author: Rea Savla

I. Cryptographic Hash Functions

In this lecture, we dove into the low-level specifics of Bitcoin that make it work. Bitcoin was innovative because it allowed a decentralized network to reach consensus. It achieved this via tamper-evidence, which means although one can modify the information that passes along the Bitcoin network, it would be obvious that some modification has been made. This tamper evident system allows us to be sure any update on Bitcoin is the same for everyone.

We achieve a tamper evident system using cryptographic hash functions to produce standardized random "fingerprints" of our data. If the data changes, so will the fingerprints. Cryptographic hash functions do the following:

Cryptographic hash functions are pseudorandom: although the output for any given input seems random, the output will remain consistent for that input.

Important Properties of Cryptographic Hash Functions:

1. Pre-image Resistance: Given H(x), it is computationally difficult to determine x
2. Second-image Resistance: Given x, it is computationally difficult to find some value x' such that H(x) == H(x')
3. Collision Resistance: It is computationally difficult to find x and y such that H(x) == H(y)

These properties produce the Avalanche effect, where even any small change in the input leads to a significant pseudorandom change in the output.

The particular hash function Bitcoin uses is SHA256, which takes in an input of size less than 2^64 bits and produces a 256 bit fix sized output.

II. A Tamper Evident Database

This cryptographic hash function is used to make an entire tamper evident database in Bitcoin. The Block Header of a block on Bitcoin, is a hash of many contents within the block, most notably its Merkle Root, Previous Block Hash, and Nonce fields. The Merkle Root represents a summary of transactions, the Previous Block Hash represents the chaining, and the Nonce represents the Proof-of-Work.

The Merkle Root is the head of the Merkle Tree, a binary tree of hashes of all the previous transactions. The Previous Block Hash contains the hash of the previous block. Both of these hashes change if any of the previous transactions or blocks is modified.

The Nonce is the manifestation of the proof-of-work in Bitcoin; it is a numerical value that must be found to solve the partial preimage hash puzzle. Miners hash the entire block header (the input) and tweak the nonce and coinbase until they find an output that solves the hash puzzle.

Hash puzzles must be:

1. Computationally Difficult: The solution to the hash puzzle cannot be easily found
2. Parameterizable: The difficulty of the hash puzzle should be adjustable
3. Easily Verifiable: Computers should have to do little work to ensure the answer is correct

The difficulty of the hash puzzle in Bitcoin is:

difficulty = difficulty * two weeks / time to mine previous 2016 blocks

Once miners solve the puzzle, they receive bitcoin via a coinbase transaction. Whenever miners produce a block, they first create a coinbase transaction, which is the first transaction of the Merkle Tree.

Using cryptographic hash functions, we ensure previous blocks remain tamper evident; we now turn our attention to how digital signatures help us ensure current transactions are tamper evident as well. Public and Private keys in Bitcoin are generated using Elliptic Curve Digital Signature Algorithm (ECDSA). ECDSA has three key properties:

1. Given the encrypted message and the sender's public key, the recipient should be able to identify the message origin. Since the message has been signed by the sender's private key, the ability to encode it using the public key demonstrates the original sender has authorized this message.
2. The digital signature scheme must also ensure non-repudiation: once the sender signs the message, they should not be able to undo it.

3. Finally, the scheme must maintain integrity; since messages are signed with the private key, they cannot be modified after signing.

Identity in Bitcoin is derived from private keys, which are generated randomly. Public keys are the result of Elliptic curve point multiplication of the private key against a known generator point on the curve. Given the public key, it is computationally infeasible to arrive at the private key.

We can apply these concepts of private and public keys to understand how transactions in Bitcoin work. Spending bitcoin is the act of **redeeming** previous transaction outputs with a proof that you are the legitimate redeemer, and then **specifying who can redeem** the output of the transaction you are now creating, by encoding that per's information in your transaction.

A transaction has three main sections:

1. Metadata: Contains housekeeping data, a unique ID of this transaction, locktime, and size
2. Inputs: Contains a list of previously created UTXOs and proof of eligibility to redeem this money
3. Outputs: Contains a list of new UTXOs that will be sent to new addresses. These values are locked by a script only the intended redeemer can unlock.

Bitcoin uses the stack-based, Turing-incomplete language named Script to create transactions. Locking and Unlocking Scripts are contained in transaction input and previou transaction output and are used to redeem the output of a previous transaction and specify requirements for redeeming transactions, respectively. Senders specify a Locking Script, and recipients specify an Unlocking Script. In Pay-to-Pub-Key-Hash (P2PKH), the recipient says "send your coins to the hash of this Public Key." In Pay-to-Script-Hash (P2SH), the recipient says "Send your coins to the hash of this Script; I will provide the script and the data to make the script evaluate to true when I redeem the coins." The latter is popular among customer-vendor transactions, where the vendor (recipient) is responsible for writing the script.

Readings
Where is Double hashing performed in Bitcoin? Opens in new window

Princeton Textbook 5.1-5.4 (pg. 131 – 157)

Bitcoin Wallets Explained: How to Choose the Best Wallet for You Opens in new window

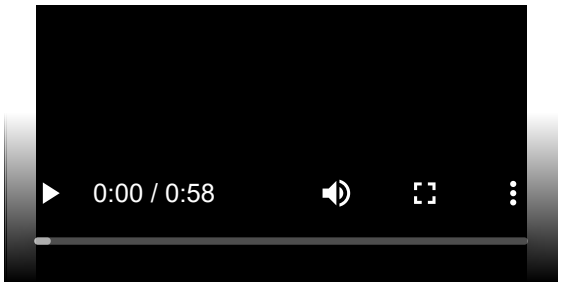
(Optional) Bitcoin Developer Guide Opens in new window (There's a lot; don't try to read it all in one day)

(Optional) Tech explained: Hash puzzles and proofs of work Opens in new window

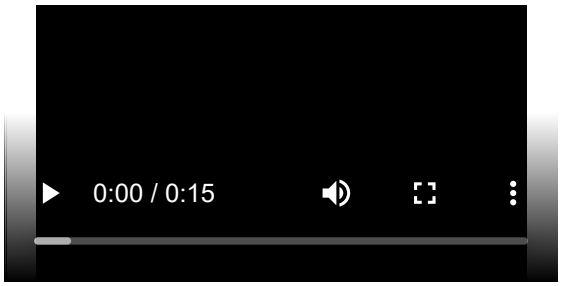
(Optional) Secure Hash Standard (SHS) Opens in new window (Insane math: a blessing or curse depending on your preference)

Bitcoin in Real Life: Wallet Mechanics, Mining, & More

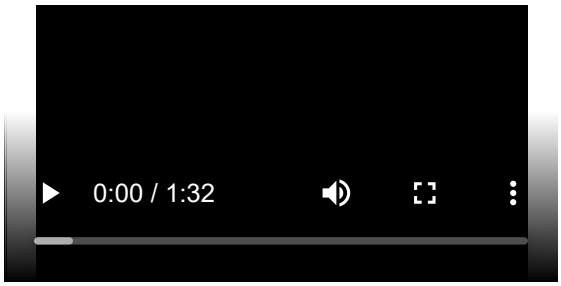
Welcome to Week 4



Intro: Types of Users

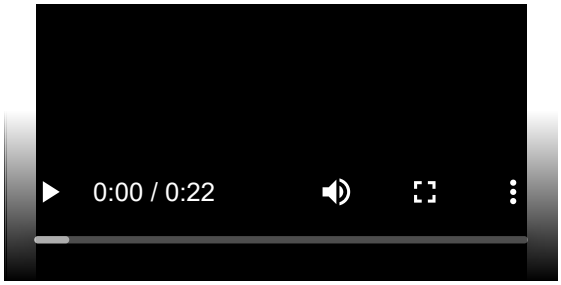


Key Components

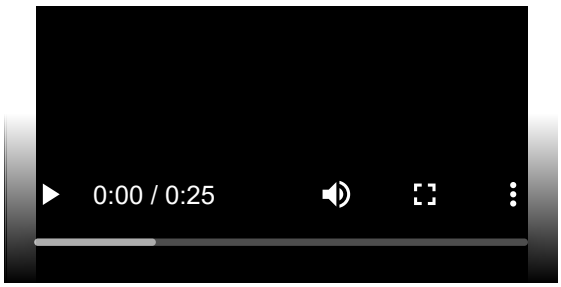


Subscribe

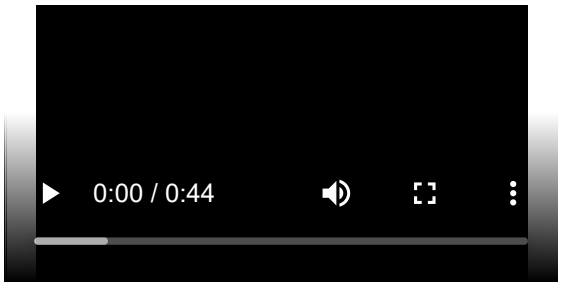
Intro: Wallets



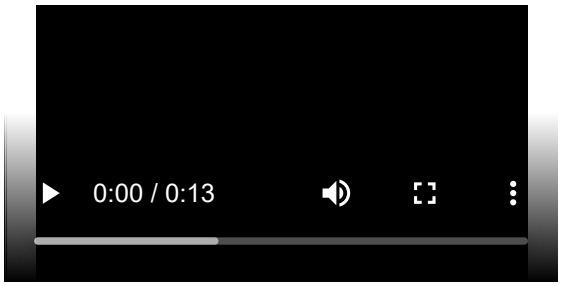
Purpose of a Wallet



Types of Wallets

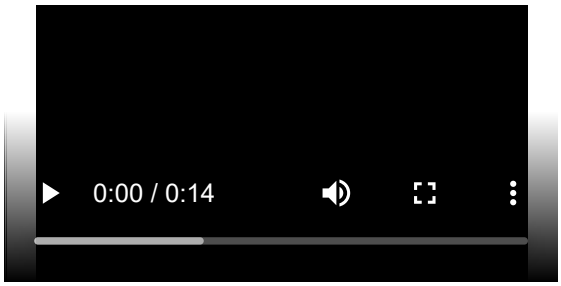


Types of Wallets: Hot and Cold

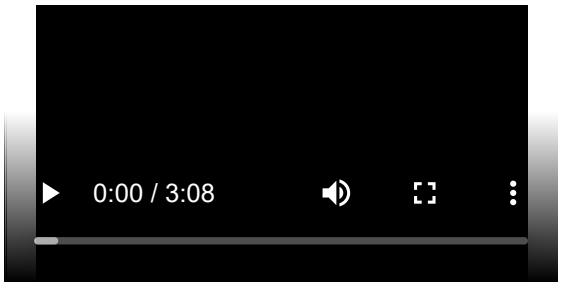


Subscribe

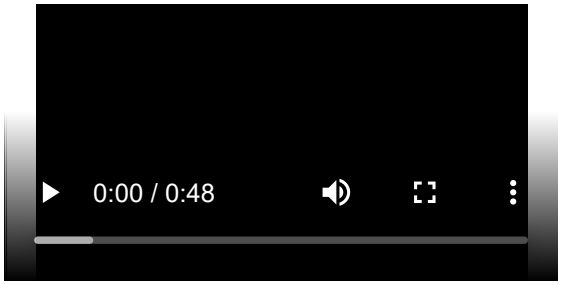
Types of Wallets: Hot Wallets



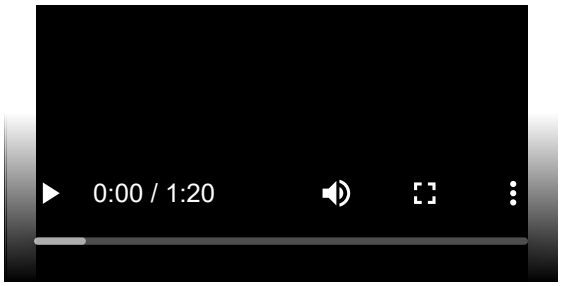
Types of Wallets: Cold Wallets



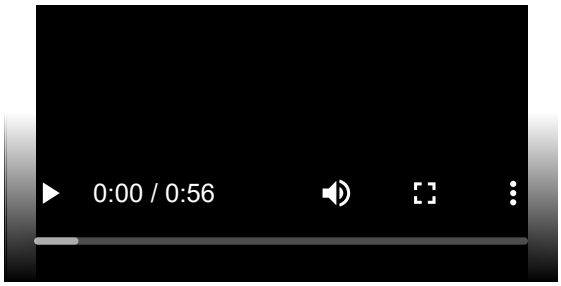
Key Stretching



Choosing a Wallet

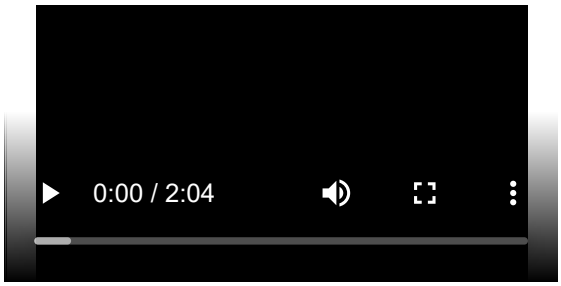


Bitcoin ATMs

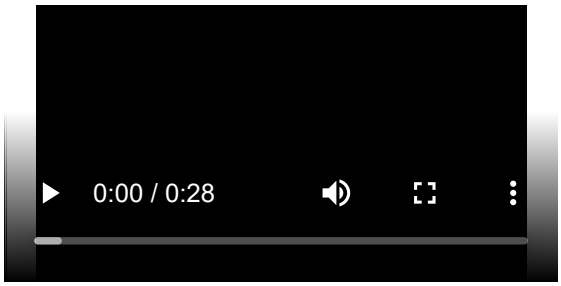


Subscribe

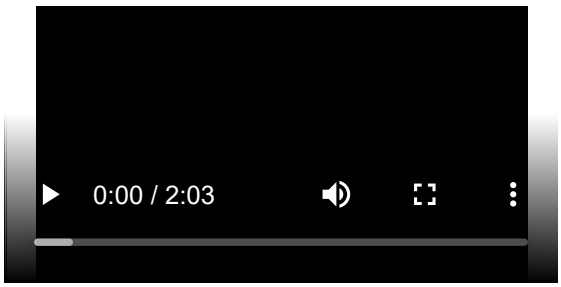
Exchanges



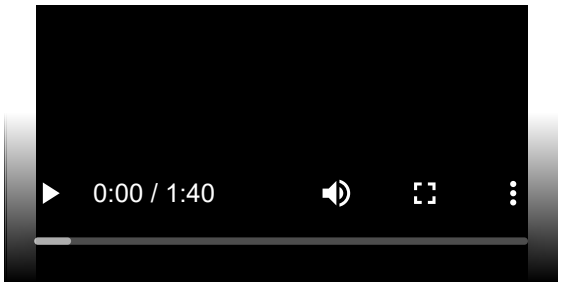
Intro: Wallet Mechanics



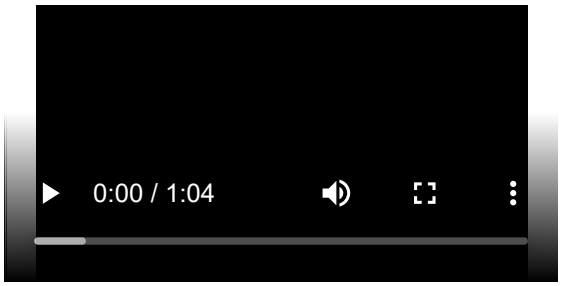
Simple Payment Verification



Multisignature

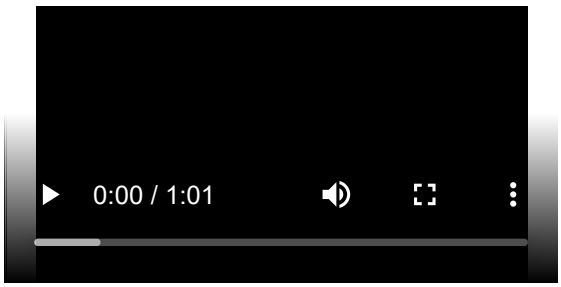


Key Generation Best Practices

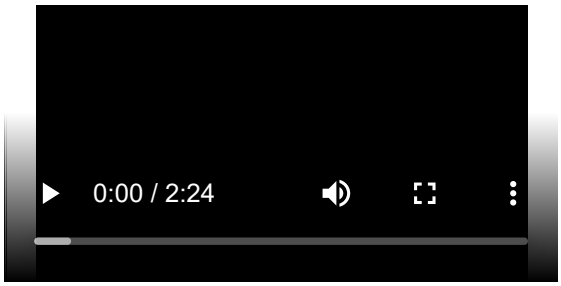


Subscribe

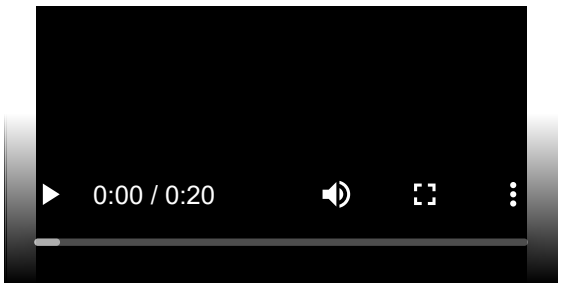
JBOK Wallets



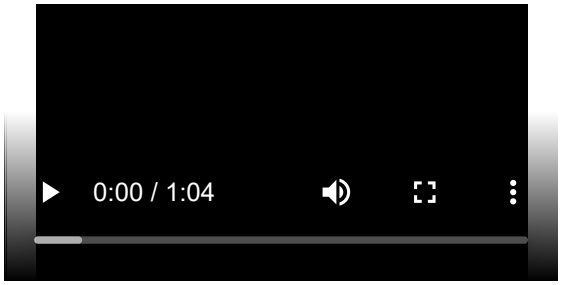
HD Wallets



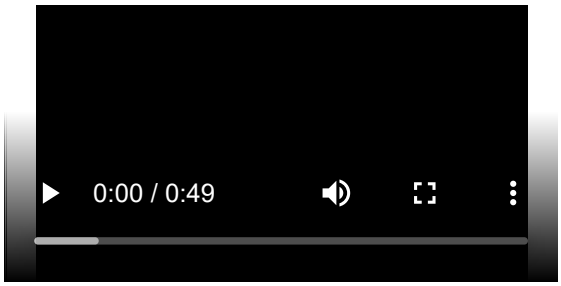
Intro: Mining



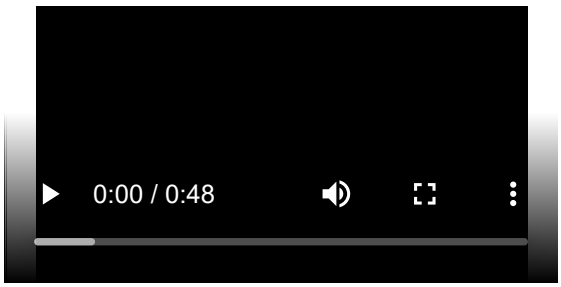
Recipe for Mining: Overview



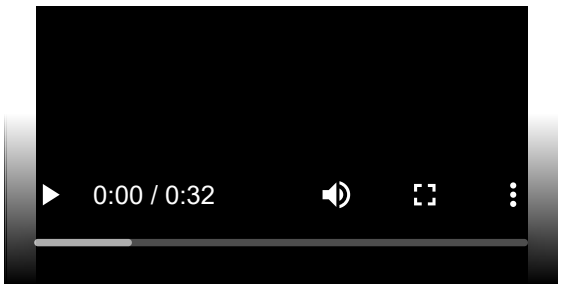
Recipe for Mining: Step 0



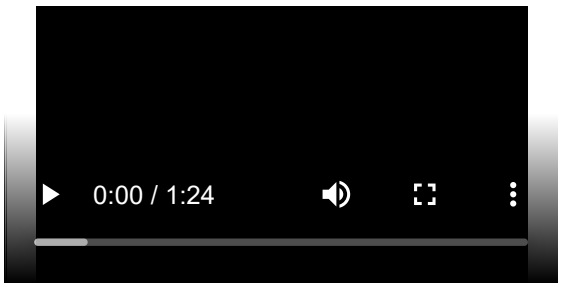
Recipe for Mining: Step 1



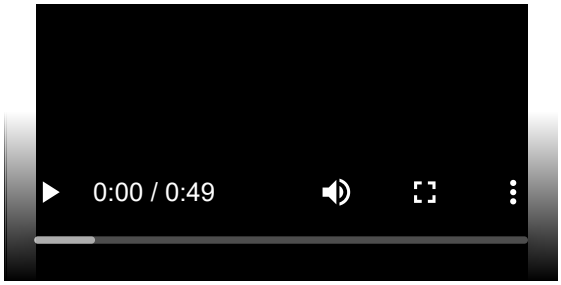
Recipe for Mining: Step 2



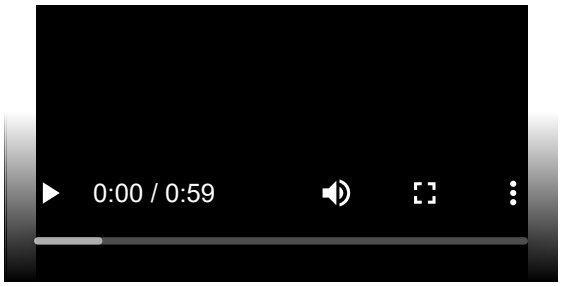
Recipe for Mining: Step 3



Recipe for Mining: Step 4

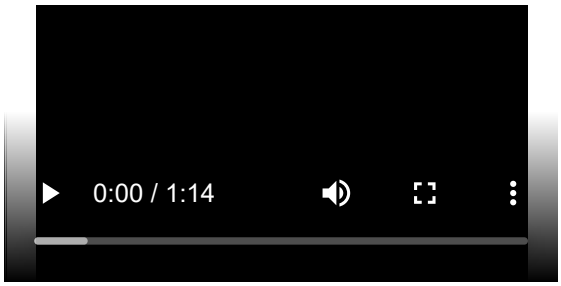


Recipe for Mining: Step 5

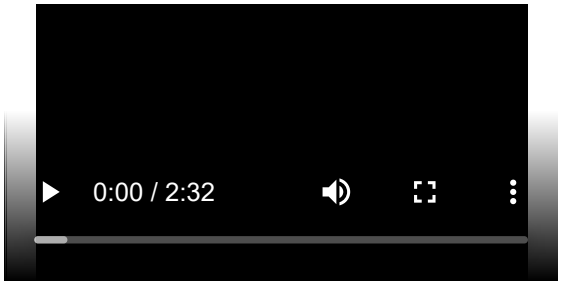


Text: Recipe for Mining

Mining Incentives: Intro

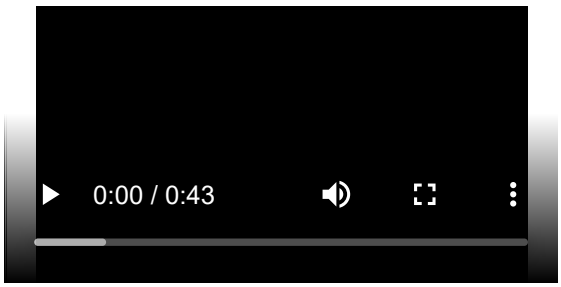


Mining Incentives: Block Reward

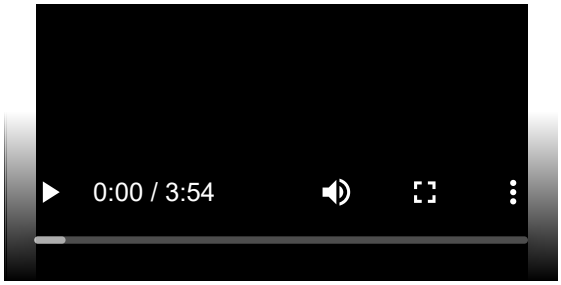


Mining Incentives: Transaction Fees

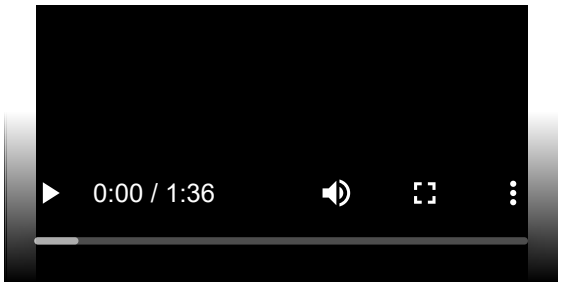
Mining Costs: Fixed Costs



Types of Hardware

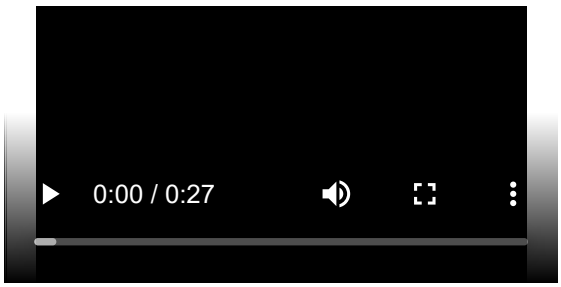


Mining Costs: Operating Costs

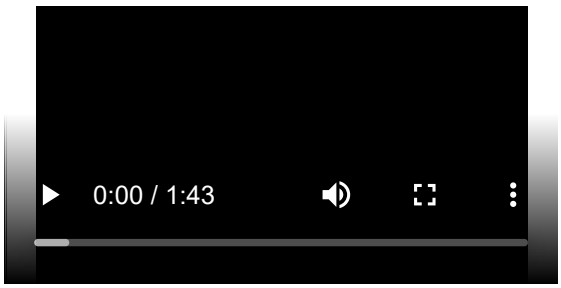


Intro: Real World Mining

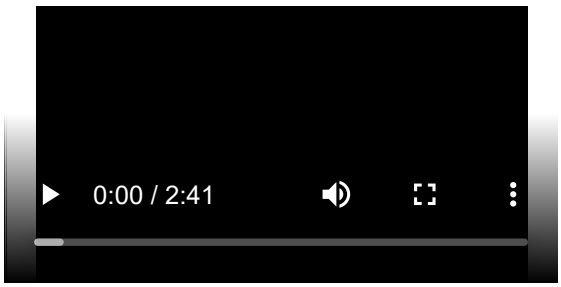
Subscribe



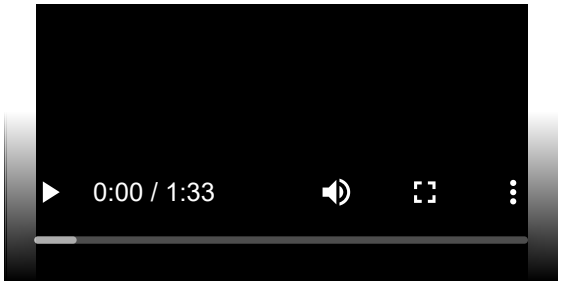
ASICs



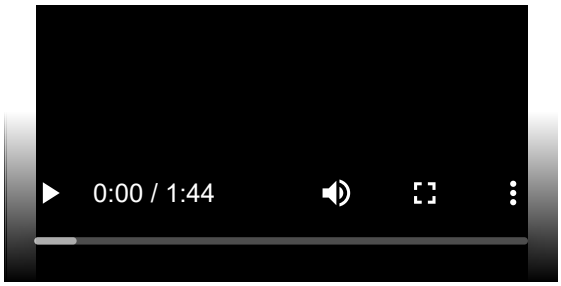
Mining Pools



Mining Pool Schemes

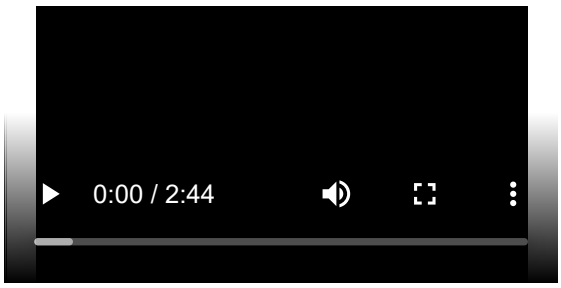


Mining Pool Pros and Cons

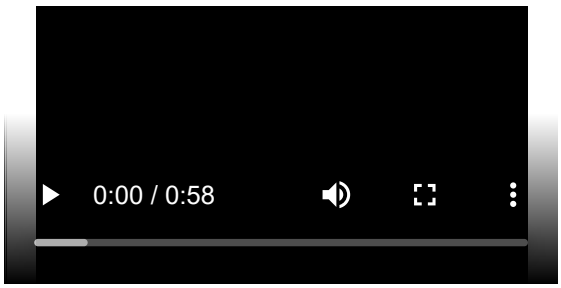


Mining Pool Stats

Subscribe



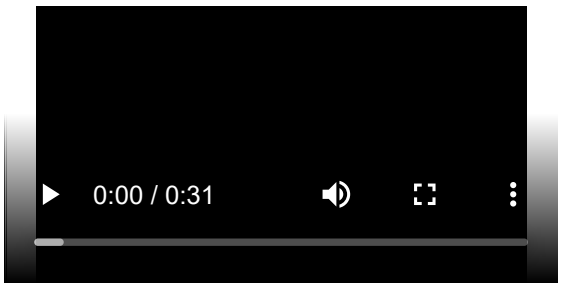
Types of Miners



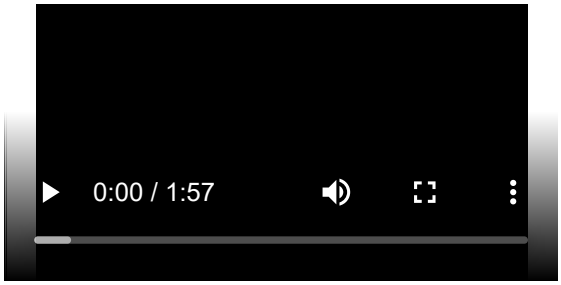
Supplement: Lightweight Mining

Bitcoin in Real Life: Wallet Mechanics, Mining, & More

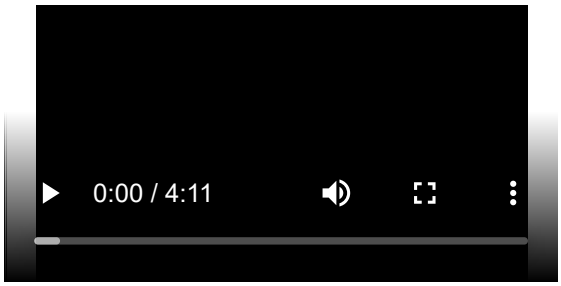
Intro: Bitcoin Governance



Ensuring Decentralization

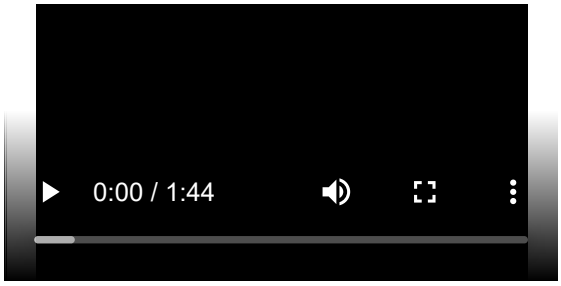


Ensuring Decentralization: ASIC Resistance

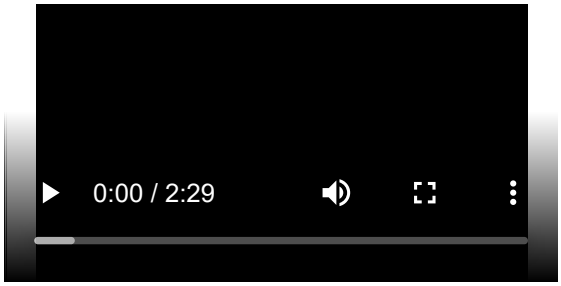


Ensuring Decentralization: ASIC Debate

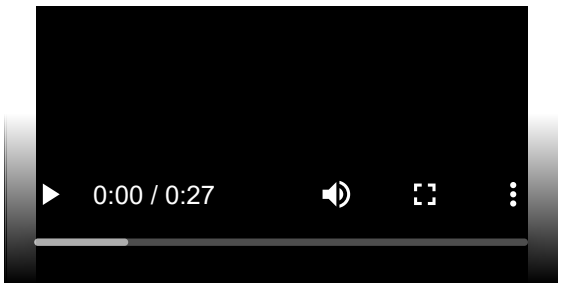
Subscribe



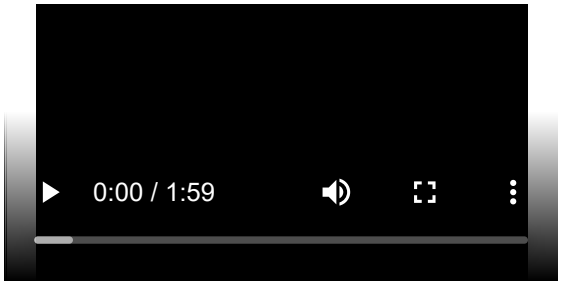
Eliminating Waste



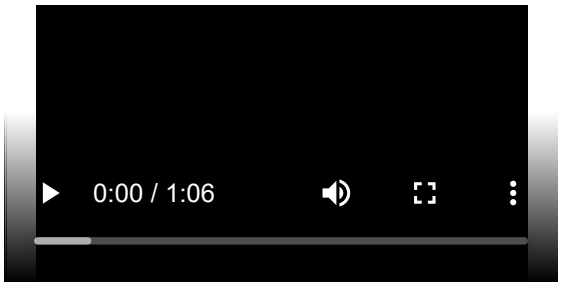
Consensus Updates: Bitcoin Core



Consensus Updates: Forks

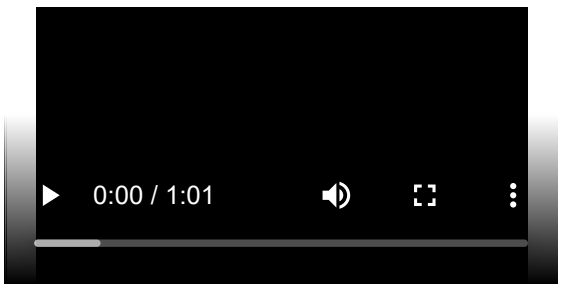


Consensus Updates: BIPs

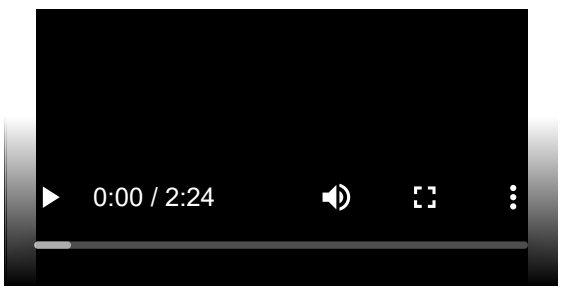


Subscribe

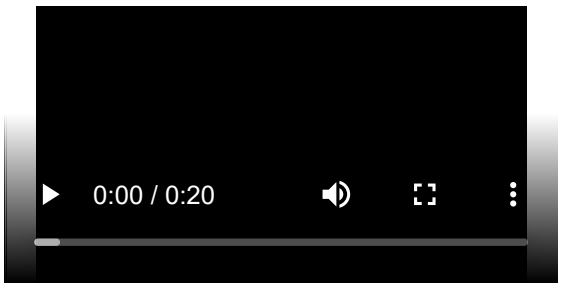
JBOK Wallets



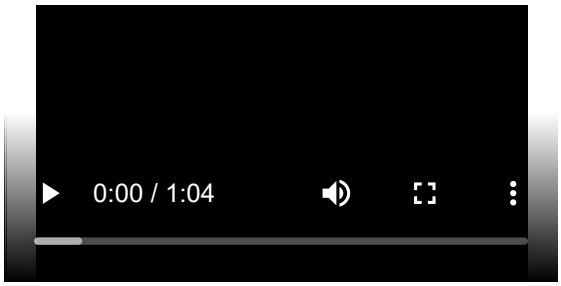
HD Wallets



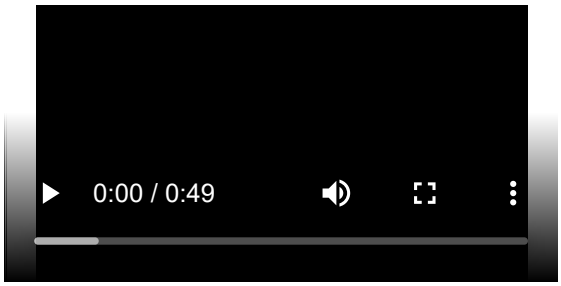
Intro: Mining



Recipe for Mining: Overview

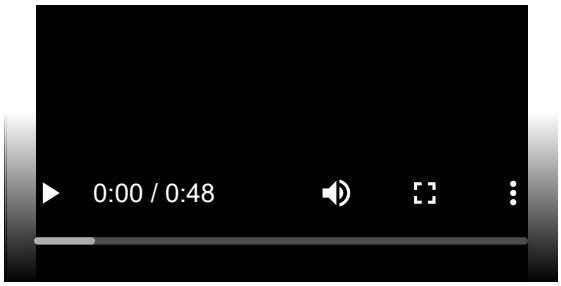


Recipe for Mining: Step 0

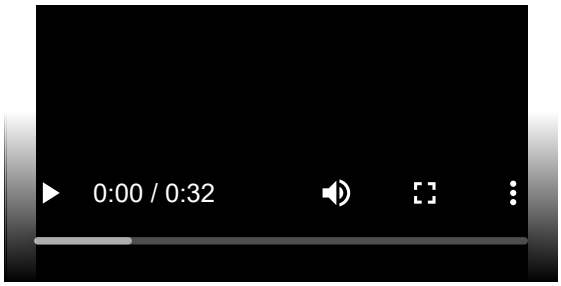


Subscribe

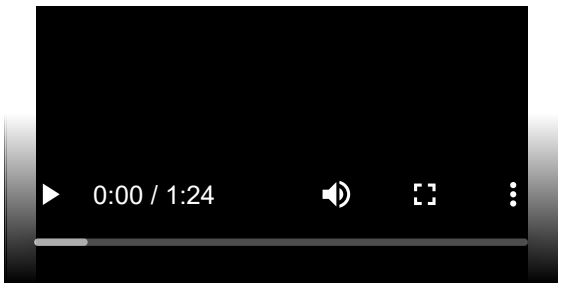
Recipe for Mining: Step 1



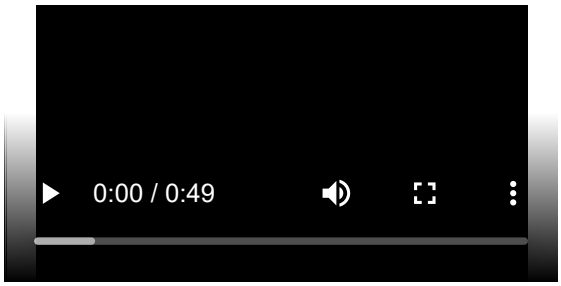
Recipe for Mining: Step 2



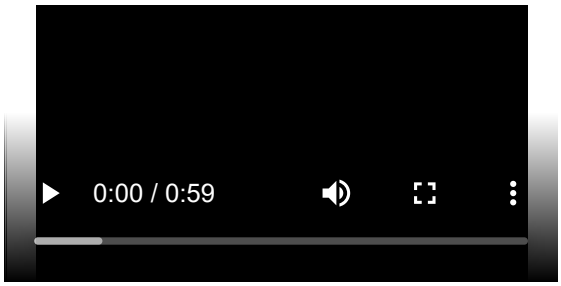
Recipe for Mining: Step 3



Recipe for Mining: Step 4



Recipe for Mining: Step 5



Text: Recipe for Mining

Author: Rea Savla

Step 0:

- Download the entire Bitcoin blockchain. This only has to be done once.
  - This allows us to know the history so we can validate future transactions.
  - Note: This step is optional if you mine in a mining pool or are doing lightweight mining.

Step 1:

- Verify transactions.

- You store newly received, unprocessed transactions in a “mempool,” where all pending transactions live before making their way into a block.
- You then choose the transactions with the highest fee per byte or size ratio to verify.
- You verify the validity of each transaction by running the unlocking script.
- If that script runs successfully, then the transaction is included within our block.

Step 2:

- Create the block with the given transactions and necessary metadata, such as time, version, and target.
  - Construct the block data from our list of valid transactions.
  - Construct the Merkle Root by hashing the hashes of each pair of transactions
  - Construct the Previous Block Hash by hashing the previous block’s header

Step 3:

- Find the proof-of-work that solves the partial preimage hash puzzle.
  - A valid nonce makes the hash of the block header less than some algorithmically generated value known as the “target.”
  - By finding the nonce, we have translated the energy burned in computation into voting power, as designed by the Proof-of-Work consensus protocol.
  - Note: There are two different nonces, the header nonce and the coinbase nonce. In the event that no permutation of the header nonce solves the hash puzzle, alter the coinbase nonce. This changes the Merkle root, yielding an entirely different hash puzzle.

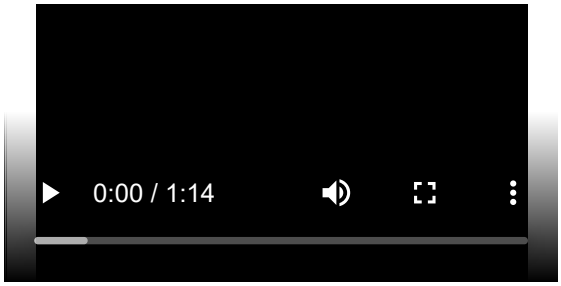
Step 4:

- Broadcast your block if you have not yet seen any competitor blocks.
  - After finding the valid proof-of-work, broadcast the block as soon as possible.
  - Other miners will validate the block for themselves before accepting it into their chain and propagating it further through the network.

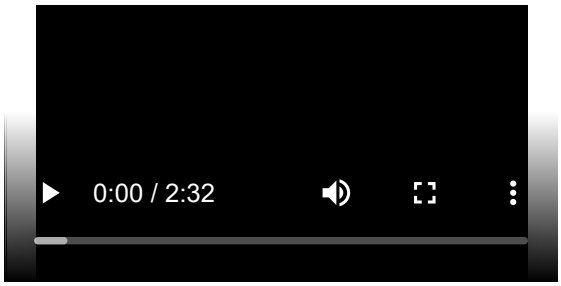
Step 5:

- If your block makes it into the longest chain, you profit!
  - You receive both the block reward and the transaction fees,
  - All the transactions within that block are added to the transaction history.
  - Note: When two valid blocks are submitted to the network at roughly the same time, resulting in a fork, honest miners choose to mine on whichever block they see first. You will not receive block reward if the other fork grows longer.

Mining Incentives: Intro

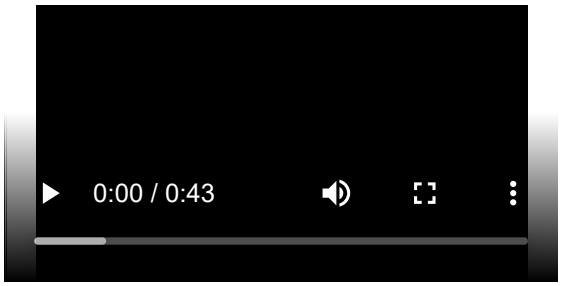


Mining Incentives: Block Reward



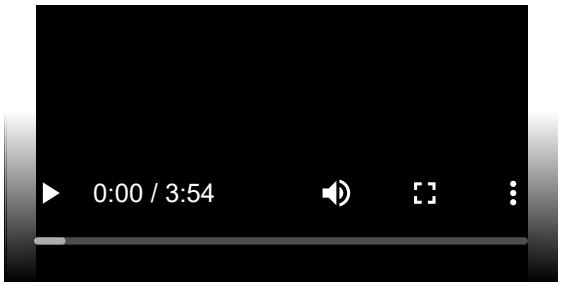
Mining Incentives: Transaction Fees

Mining Costs: Fixed Costs

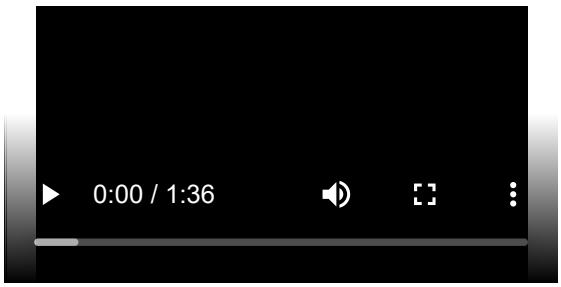


Subscribe

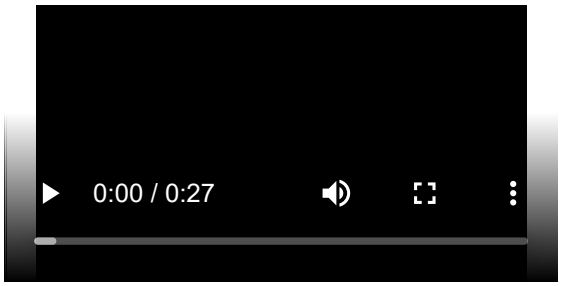
Types of Hardware



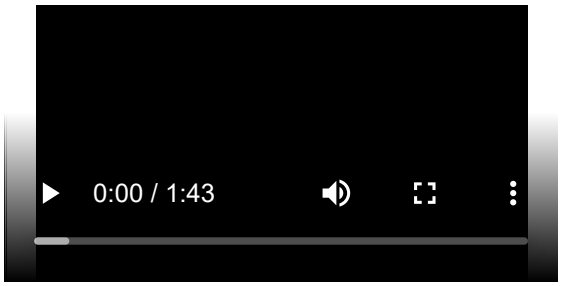
Mining Costs: Operating Costs



Intro: Real World Mining

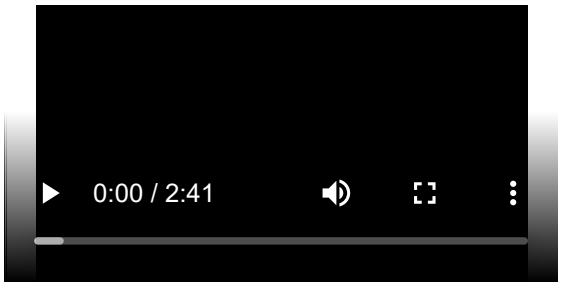


ASICs

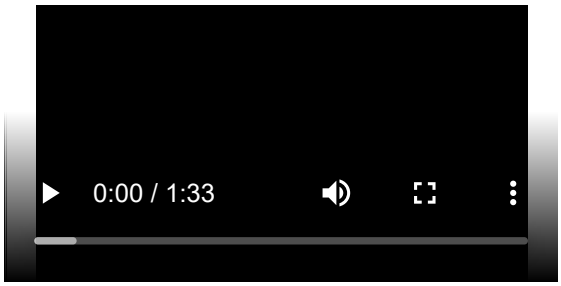


Subscribe

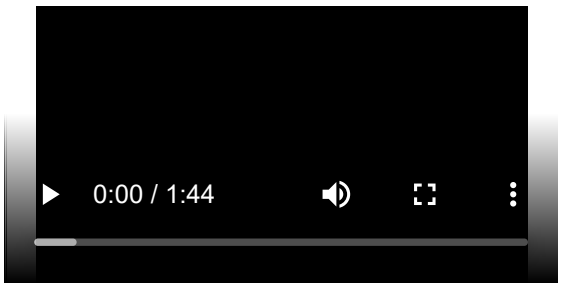
Mining Pools



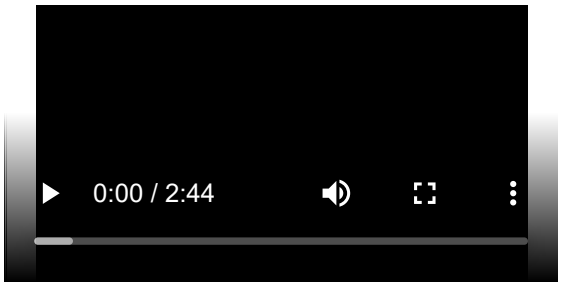
Mining Pool Schemes



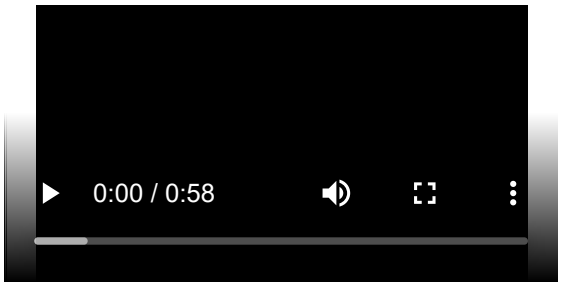
Mining Pool Pros and Cons



Mining Pool Stats



Types of Miners



Subscribe

Supplement: Lightweight Mining

Author: David Luo

Overview

Simple Payment Verification (SPV) is a method for verifying if particular transactions are included in a block by only having to download the block headers of each block, instead of the entire block, which includes all of the transactions. These SPV/lightweight clients are especially useful for clients who have limited memory space and simply want bitcoin wallets without downloading the entire blockchain.

There is also lightweight mining software, which differs from lightweight wallet software. Instead of using SPV, users can mine using other methods without downloading any data from the blockchain. One way users with these lightweight clients are able to mine is by joining a **mining pool**. These miners work outside the network and only need to mine the transactions the pool operator gives them, trusting that the operators have the entire blockchain and the transactions given to them are valid. It is up to the pool operator to listen for new blocks and validate the transactions. To do this, the operator sends the miner a template of the block to be working on. If a block is found, it is bound to this template, which

includes the coinbase transaction that gives the reward to the pool operator. This way, a user cannot steal the reward for themself if they find a block.

getwork, getblocktemplate, and Stratum

getwork is a remote procedure call method that a client can call to get a block header from a mining pool operator. However, a caller of this method does not receive any information about the block or the transactions within it. A malicious server could easily send this user a block header hash that contains invalid transactions, and the user would mine on this invalid block.

getwork has been replaced by **getblocktemplate**, which allows miners to create their own blocks instead of having to trust somebody else (e.g. a pool operator) to provide them a valid block to mine on. Only a block template is provided with specific configuration parameters, with block data that the client can modify if needed. This way, the control is given back to the miner, because they can now choose the transactions to be included in the block they are mining on. However, miners are incentivized to include pool specific data, such as a coinbase transaction to the pool operator in order to make valid shares and get rewards from the mining pool.

Although getblocktemplate does provide good functionality, it requires a lot of bandwidth due to the transferring of the entire block. Because of this scalability issue, some pools now use **Stratum**, which is another protocol for mining. Similar to getblocktemplate, the server sends the client a block template. However, in Stratum only the block header and first transaction are included, instead of the entire block.

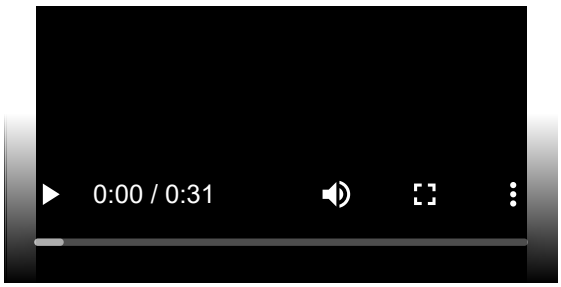
Validationless Mining

In order to mine, a client needs to have the block header hash of the previous block, which comes from the block header of the previous block, which in turn is generated from previous block. One piece of information our new block needs to have is the block header hash, as we must reference the previous block in the chain. They can simply receive this value from an external source and mine on that block. This comes with one caveat, however, as the miner cannot validate the block header hash and must place trust in whatever source such as a pool operator they are getting the block header hash from.

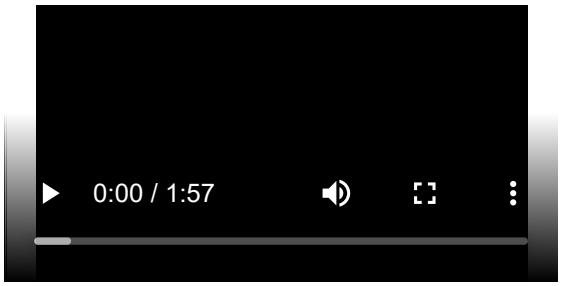
Spy Mining

In mining pools, once a block is found, the block header hash is sent immediately to the miners so they can all start mining on the new block as soon as possible. Competing miners that are outside of this mining pool can take advantage of this by **connecting to the mining pool**, listen for the **block header hash**, and **mine on it for themselves**.

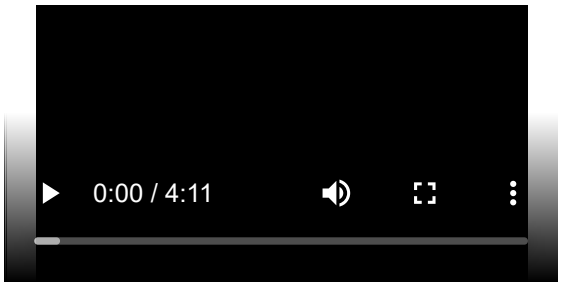
Intro: Bitcoin Governance



Ensuring Decentralization

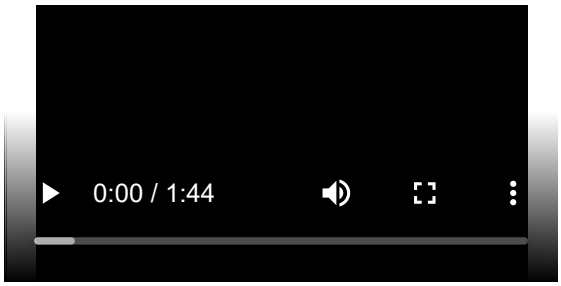


Ensuring Decentralization: ASIC Resistance

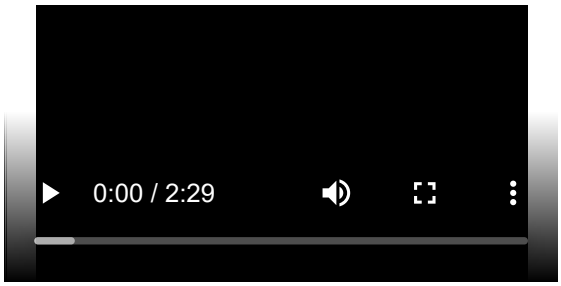


Subscribe

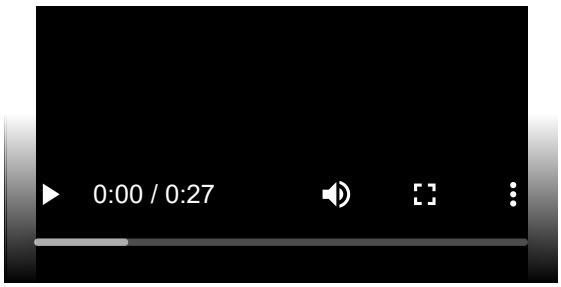
Ensuring Decentralization: ASIC Debate



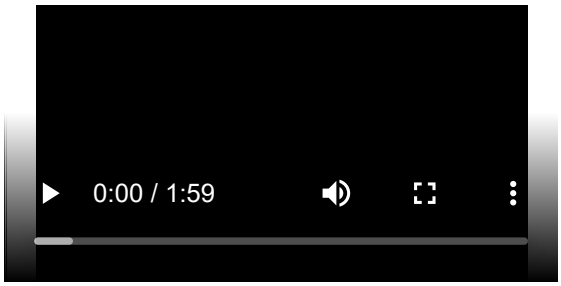
Eliminating Waste



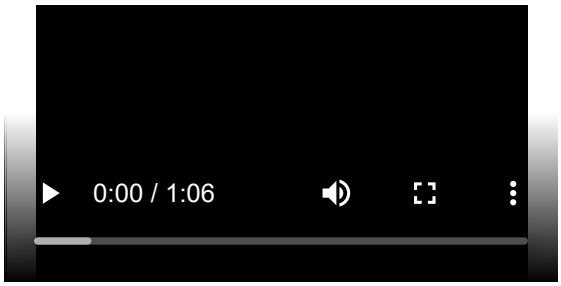
Consensus Updates: Bitcoin Core



Consensus Updates: Forks



Consensus Updates: BIPs



Subscribe

Text: Lecture 4 Summary

Author: Rea Savla

I. Types of Users

Bitcoin makes the distinction of 4 key functionalities (Mastering Bitcoin Opens in new window, p.152) that every node in the network is a combination of:

1. Wallets: management of keys and addresses
2. Mining: the voting process which expends computational power
3. Full Blockchain: a copy of the full Bitcoin blockchain
4. Routing: software that allows you to talk to other Bitcoin nodes

We can make powerful generalizations using these 4 key characteristics, however there exist other distinctions between users too.

> Image Source: Mastering Bitcoin Opens in new window

II. Wallets

The primary function of Bitcoin wallets is to keep track of your identity. In Bitcoin, a private key gives you a claim to your virtual identity; it unlocks your virtual identity and gives you access to your Bitcoin address. Wallets primarily store your private keys, but most wallet softwares also generate public/private keys with each new transaction and track your transactions. Unlike the physical wallets we use for money, Bitcoin wallets do not store physical bitcoins. They're simply impossible to store. Hence, control over private keys is essentially control over identity and bitcoins.

There are two types of Bitcoin wallets: hot wallets (Mycelium, AirBitz, Webapp, Mobile App, etc) are connected at some point to the internet, while cold wallets (i.e. paper and hardware) are not.

You can find them at Bitcoin ATMs, where you can trade cash for bitcoins. You can also get bitcoins through an exchange, where you can trade between different types of fiat currencies and cryptocurrencies. Centralized exchanges are easy to use and access, but are single points of failure which may be vulnerable to hacks. Decentralized exchanges (i.e. Bitshares and Bitsquare) conduct trades directly between users by creating proxy tokens and are mostly trustless but may suffer from liquidity problems.

III. Wallet Mechanics

For wallets to successfully keep track of your identity and bitcoin, they must implement mechanisms to verify transactions and keep track of many keys.

When using a wallet software, you do not have to download the full Bitcoin blockchain thanks to Simple Payment Verification (SPV). SPV is a method for verifying if particular transactions

(specifically your own transactions) are included in a block without having to download entire blocks. Instead, you can simply ask your neighbors for relevant info about the blocks, though you must trust they will give you legitimate data.

To maintain anonymity, it is best practice never to reuse pseudonyms. Bitcoin addresses are cheap to generate, and using a new pseudonym for each transaction makes tracking your activity and discovering your identity much more difficult.

The traditional way to keep track of a large number of keys is through a "Just a Bunch of Keys" (JBOK) wallet: a new key pair is created and backed up for every transaction. However, JBOKs are hard to scale. You can see how exchanges have to deal with thousands of transactions a day for thousands of users would need something more efficient than JBOK wallets.

Hierarchically Deterministic (HD) Wallets overcome this problem by deriving all new keys from an original random seed value. In HD Wallets, anyone controlling the parent key can also generate and control child keys.

IV. Mining

Mining in Bitcoin requires miners to verify incoming transactions, create a block, find a valid nonce, broadcast the block, and earn a profit in bitcoins.

Miners are primarily motivated to maintain the network, or execute the Proof-of-Work consensus mechanism and bring new bitcoin into circulation, by earning profit. Miners onl earn a profit from mining when the revenues from the block reward and the transaction fe are greater than the fixed and variable costs of mining.

A block reward is a reward that goes to miners whose blocks are included in the longest chain. Bitcoin incentivizes honest mining by rewarding miners in bitcoin. The block reward halves every 210,000 blocks, approximately 2 years, and will fall to 0 once 21 million bitcoins have been mined.

Regardless of the number of bitcoins in circulation, miners will always be rewarded via transaction fees.

The main fixed cost in mining is the hardware that allows users to mine. CPUs, GPUs, FPGAs, and ASICs are all types of hardware that have been used for mining, and each is more computationally powerful than the last. ASICs, which stands for Application-Specific Integrated Circuits, are currently capable of computing 14 trillion hashes per second. They are fully specialized devices that can only do one activity: mine for Bitcoin.

The main variable costs associated with mining are energy and infrastructure costs (i.e. warehouse and personnel). Energy costs are further divided into three types: Embodied Energy, to produce hardware; Electricity, to power hardware; and Cooling, to maintain hardware.

V. Real World Mining

To get a better sense of how mining looks in real life, here is what a real-world ASIC mining farm in China looks like:

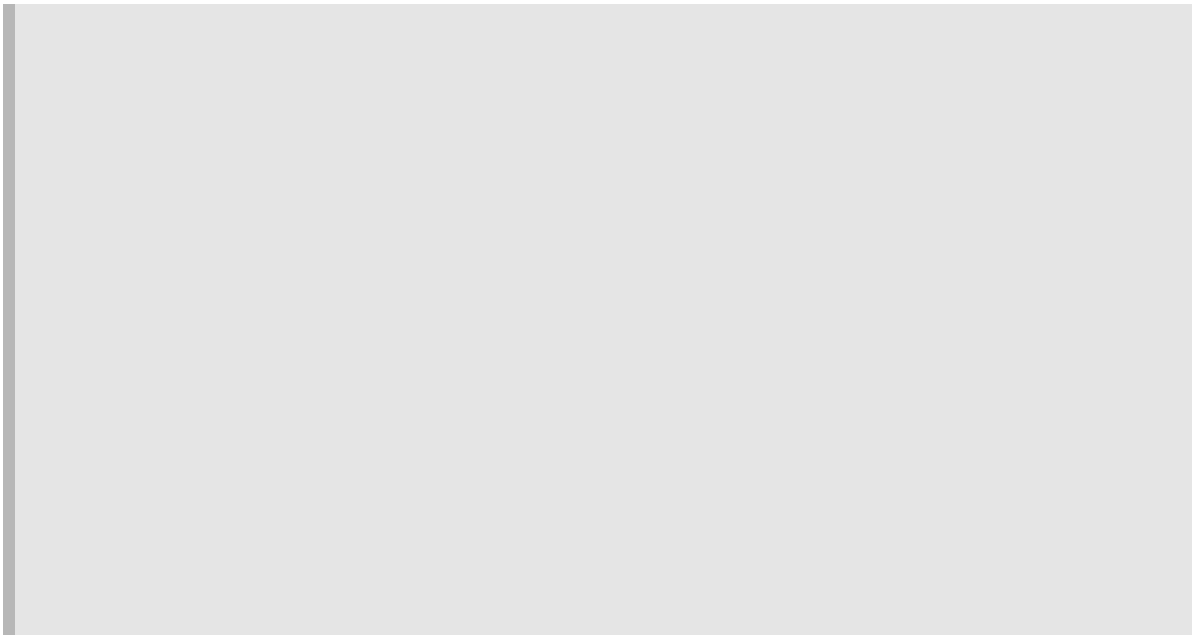


Image Source: The Register Opens in new window

Even with the most impressive technology, it still takes a great amount of time and power to find a block. Mining pools allow miners to join their hash power together so that they will collectively find blocks and collect rewards more often, reducing variance. As a member of a mining pool, your reward is calculated by the number of "shares" you contribute. A share is defined as a near valid block, used to estimate the mining power you contributed. More shares equals more attempts at finding a valid nonce equals a larger slice of the pie.

The two main payment mechanisms in mining pools are:

1. Pay-per-Share: where miners are paid a fixed amount of money for each share, regardless of how much reward the pool makes
2. Proportional Schemes, where miners receive a reward proportional to the number of shares submitted before the block was found

Pay-per-Share pools are advantageous to the miner because of guaranteed payouts, but risky for the pool because miners have no incentive to submit valid blocks to the pool. Proportional Schemes are advantageous for the pool because the pool operator only pays out after a block is found, but more risky for the miner due to greater variance of reward.

VI. Changing Bitcoin

One of the current challenges of Bitcoin is that mining is tending towards centralization with the development of ASICs, mining pools, and mining farms. ASICs cost a lot of money, which means only those with access to high volumes of capital can afford them, thereby concentrating the mining power into the hands of a few. To address this issue, we must

develop a tweak to the protocol that reduces centralization but still adheres by the puzzle requirements (quick to verify, adjustable difficulty, and computationally difficult).

We can get around the ASIC domination by creating a memory-hard or memory-bound puzzle, periodically switching mining puzzles, or replacing Proof of Work by Proof of Useful Work.

While ASIC resistance may decentralize mining and make it a more democratic process, achieving ASIC resistance may also cause disruptions to the mining community that would undermine the network.

Real changes are implemented in Bitcoin via hard or soft forks. Hard forks result from a new protocol that does things not allowed in the old protocol. In a soft fork, the rules of the protocol are only restrained. A soft fork is backwards compatible but not forwards compatible.

Hard Fork:

Soft Fork:

In Bitcoin, changes to the protocol come in the form of BIPs, or Bitcoin Improvement Proposals. These can be changes in the network protocol, block or transaction validation, or anything affecting interoperability. There are three main kinds of BIPS: standard, informational, process.

Readings

Readings

- Mining Bitcoin with pencil and paper: 0.67 hashes per day Opens in new window
- ViaBTC Rises: How A Mysterious Miner Could Decide Bitcoin's Future Opens in new window
- Antbleed: Bitcoin's Newest New Controversy Explained Opens in new window
- (Optional) Time stamped chatlog: Why Jihan and Jiang want to block segwit at all cost Opens in new window
- (Optional) Bitcoin Hash Rate vs Difficulty Opens in new window (play around with different variables!)
- (Optional) BlockShell Opens in new window (set up and play around with the CLI)

Game Theory & Network Attacks: How to Destroy Bitcoin

Text: Lecture 5 Summary

Author: Rea Savla

I. Pool Strategies

I.I Pool Hopping

Miners can maximize profits via pool hopping. This occurs when miners switch between Pay-per-Share and Proportional scheme mining pools to whichever payment protocol produces a higher rewards per additional share. In this scenario, honest and loyal pool miners will be cheated out of their profits by miners who pool hop and cause inconsistency in the pool's hash rate.

I.II Pool Cannibalization

In this attack, pool miners distribute a small percent of mining power equally among other pools, without ever submitting valid blocks. In doing so, we increase personal profits to the detriment of our mining pools. Pool Cannibalization is very difficult to detect, unless

statistically significant. Since it is more profitable to be dishonest than honest, miners are incentivized to spend mining power cannibalizing each other rather than increasing the useful hashrate for finding the next valid block.

If we model the choice between attacking and not attacking the Bitcoin network using game theory, we see that the dominant strategy is for each miner to attack the network, leading to the ultimate detriment of Bitcoin.

In this situation, the Nash Equilibrium, or the position from which no actor can change their position to improve their outcome, is also known as the Tragedy of the Commons,where each individual actor finds it individually beneficial to exploit the public good, leading to the ultimate deterioration of the public good.

II. The Double Spend Attack

The double spend attack occurs when an individual successfully spends the same value more than once. You can double spend via race attack:

1. The Race Attack occurs when you trick someone into thinking the transaction is complete before the transaction even enters a block.
2. While you can show your victim a valid transaction, you can also send into the network a conflicting transaction that sends bitcoins from that same UTXO to an account you control, and incentivize miners to include this conflicting transaction instead of the first transaction by offering a higher transaction fee. In other words, you “race” with the conflicting transaction to get included in the longest chain.

The Race Attack occurs when you trick someone into thinking the transaction is complete before the transaction even enters a block.

While you can show your victim a valid transaction, you can also send into the network a conflicting transaction that sends bitcoins from that same UTXO to an account you control, and incentivize miners to include this conflicting transaction instead of the first transaction by offering a higher transaction fee. In other words, you “race” with the conflicting transaction to get included in the longest chain.

To protect against this type of attack, the victim should wait for kconfirmations, kblocks built off of some particular block. In practice, kequals 6. If you still want to double spend, you must create a longer private chain with the same UTXO. If the victim waits kconfirmations, you must mine k+1blocks upon your private chain and then publish your private chain to the network, since no one else can see it when it is private, to complete the Double Spend Attack.

Using the following graph, we can see that if you have over 50 percent of the hashing power, the probability that you will eventually get a longer chain than the honest chain is 100 percent, and you will thus successfully conduct a race attack.

III. The 51 Percent Attack

If you, the attacker, have any more than 50 percent of network hashrate, then you will always be able to double spend because you will always be able to create the longest chain, and since you would be in full control of which blocks are included within that chain. Since individuals on the network would just leave the network when they see a 51 percent attack, and subsequently devalue bitcoin, conducting such an attack is too risky.

IV. The Goldfinger Attack

The Goldfinger Attack allows attackers to still profit off of the destruction of Bitcoin. Attackers can "short," or place a bet on the devaluation of bitcoin.

V. Censorship

In addition to attacking Bitcoin by leveraging hashpower, you can also attack the network via censorship. With censorship, we can choose to ignore the transactions from an individual or group of individuals, isolating them from the network and effectively rendering their bitcoins useless.

A mining pool with more than 51 percent of the network hashrate can decide not to work on a chain containing transactions spending from a particular address. Other miners will see that if they include a transaction with that address, their block will not be included in the chain, since the mining pool with the majority hashrate can fork and create a longer proof-of-work chain. Therefore, they will be incentivized to exclude the transactions with that address from their chain. This is called punitive forking.

Even without 51 percent of the network hashrate, you can censor another address' transactions using feather forking. In feather forking, the attacker announces they will attempt to fork if they see a block with a certain address' transactions, but will give up after kconfirmations. Even though the attacker has a low chance of orphaning, or excluding, that

block, miners will ignore blocks with the given address. This happens because miners are incentivized primarily by the profit they expect to earn from mining a particular block. Even if there is a small chance of the attacker forking a block with the given address, the math works out such that the expected profit of ignoring that transaction will exceed that of including it. Thus the victim has to pay a much higher transaction fee to incentivize miners to include their transaction.

VI. Selfish Mining

Upon being the first to find a valid block, you can withhold that block from the network and continue to work on it privately. If you find two blocks on your secret chain before the network finds the next one, then you suddenly have the longest chain. This means you get at least two block rewards and you have fooled the network into working on an honest chain that you will render useless. You can keep doing this and submit your longer, secret chain to the network right before you think the network is going to catch up to you. This is selfish mining.

VII. Defense

We discussed several theorized defenses to selfish mining including

- Block Validation Using Time Signatures Opens in new window, proposed by Schultz (2015) and Solat and Potop-Butucaru (2016)
- Fork-punishment rule Opens in new window, proposed by Lear Bahack (2013)
- Uniform Tie-Breaking Rule Opens in new window, proposed Eyal and Sirer (2014)
- Publish or Perish Opens in new window, proposed by Zhang and Preneel (2017).

However, according to Vitalik Buterin, “in practice, most Bitcoin miners act altruistically to support the network, both out of ideological considerations and because they do not want to destabilize the source of their own revenue. Such higher-level economic concerns are beyond the scope of Eyal and Sirer’s paper, but they seriously reduce the chance that this economic attack will work in practice” (Bitcoin Magazine Opens in new window, 2013).

VIII. The Bitcoin Network

In addition to individual attacks, the architecture of the Bitcoin network itself contains vulnerabilities. The Bitcoin protocol is P2P, in which messages get sent via a gossip protocol, where each node passes a message to its connected nodes. Though we would like an even topology, this is not the case. Nodes with higher hashrate, perhaps from running mining pools, may have miners connected to them directly on a lightweight protocol, or a secret subgraph. Due to this hidden graph topology, a user could potentially hide that it has more than 50% of the network hash power.

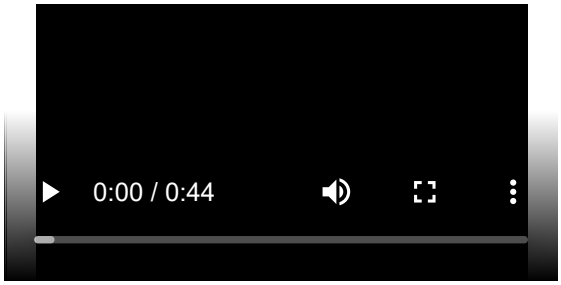
In addition to network topology, network latency also poses as an issue. Nodes that are better connected to the network see things faster and can send out messages and blocks to the network more quickly. This leads to disproportionate profits in miners.

Readings

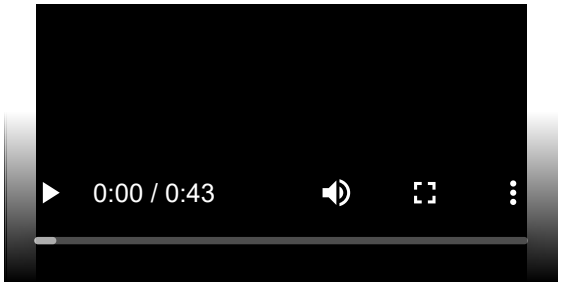
Readings

- Selfish Mining: A 25% Attack Against the Bitcoin Network Opens in new window
- "Ethereum Whitepaper up to "Miscellanea And Concerns"

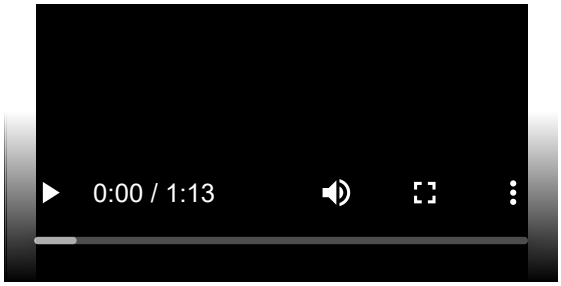
Welcome to Week 5



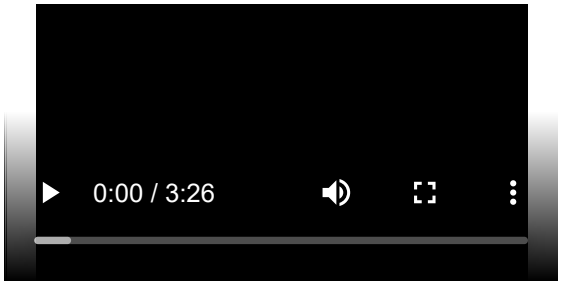
Intro: Pool Strategies



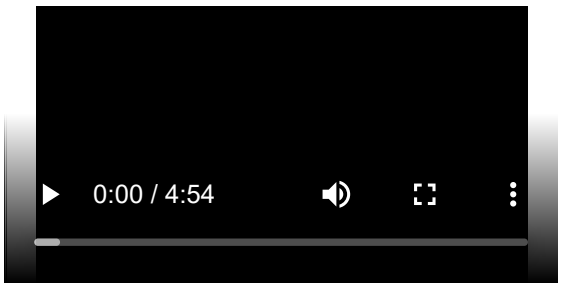
Pool Reward Scheme Review



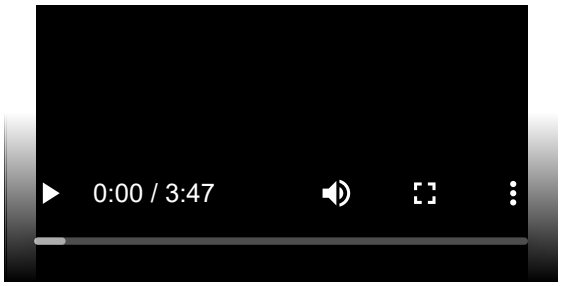
Pool Hopping



Pool Cannibalization



Nash Equilibrium and the Tragedy of the Commons



Supplement: Game Theory

Author: Rea Savla

We can model the behavior of different actors in the Bitcoin network for a given situation using a game theoretical analysis. Our first assumption is that actors will act in a rational way, where rationality is defined as taking the actions that maximize their utility. In our scenario utility is defined by the monetary gains resulting from an action. While not everyone in Bitcoin is purely motivated by monetary gain, this is still a powerful generalization.

A ***Pure Strategy Nash Equilibrium*** is the set of actions that maximize each actor's utility given the responses of all the other actors. Note, the utility each player receives depends on the player's own decisions and the decisions of all the other actors, in the same way that the rewards a miner gets in the mining pool depend on the miner's own decision to attack or cooperate and the rest of the pool's decision to attack or cooperate. Players in the Bitcoin network or in a mining pool will converge to acting according to the Pure Strategy Nash Equilibrium for a given scenario, assuming they all behave rationally.

Let's take a look at a simple scenario. Suppose there are only 2 mining pools in the Bitcoin network, Pool A and Pool B. Their utilities are shown below. The numerical value of utility is arbitrary in economics; it does not have any associated units, and it is calculated using a utility function of other parameters, defined by the economist. We are only interested in the comparative value of an action's utility, whether it is higher or lower than another; we are not interested in its absolute value. In this example, the respective utilities are derived from the monetary gains each player would receive from the given scenarios of attacking or cooperating with the other.

The specific numerical values for the utilities in this example were chosen arbitrarily to reflect a scenario where players are incentivized to act dishonestly, but fare worse when both players are dishonest than when both players are honest. Let's take the perspective of Player A. Given Player B acts honestly, player A gains the most utility from acting dishonestly, since a utility of 3 is greater than the utility of 2 that A would receive by acting honestly.

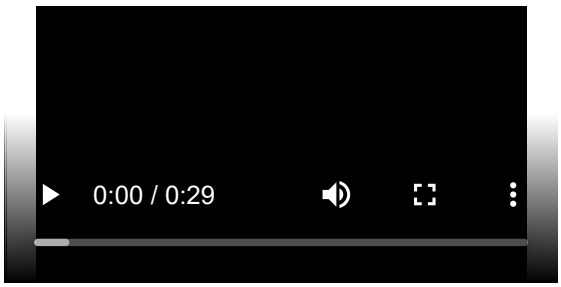
Given Player B acts dishonestly, Player A would prefer also acting dishonestly since a utility of 1 >utility of 0.

Player B makes the same conclusions, and makes preferences as follows:

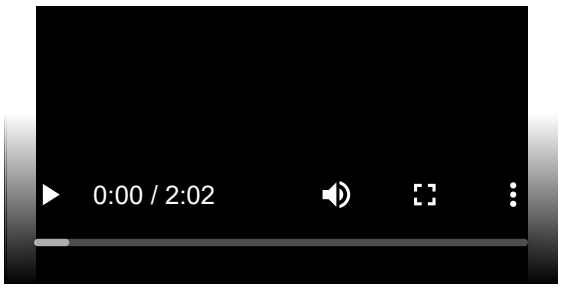
We can see that the scenario in which both players choose to act dishonestly is the Pure Strategy Nash Equilibrium, since at this position, both players are maximizing their payoffs given the other player's actions. Thus, despite receiving higher returns from both acting honestly, Players A and B will both act dishonestly.

If you are interested in learning more about introductory game theory, we recommend the following reading: A Brief Introduction to Basics of Game Theory Opens in new window by Matthew O. Jackson, Stanford University.

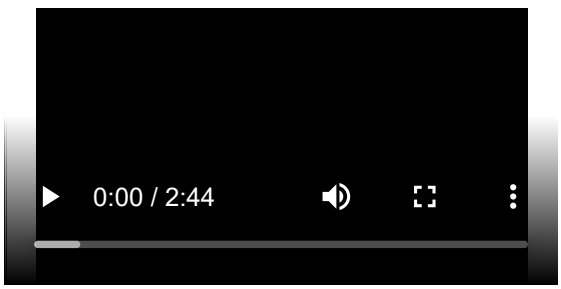
Intro: Double Spending



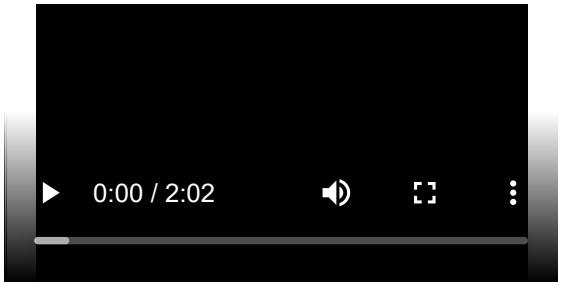
Race Attack



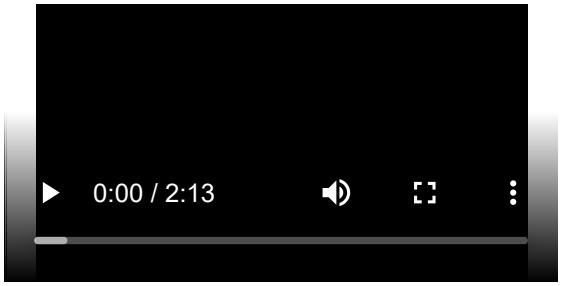
Defense: Confirmations



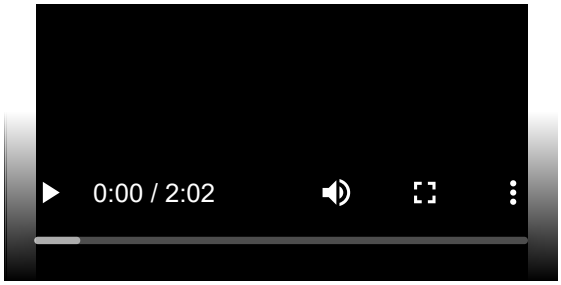
Demo: Confirmations



Probability Analysis

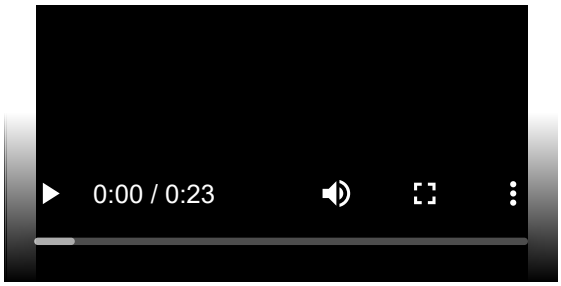


Goldfinger Attack

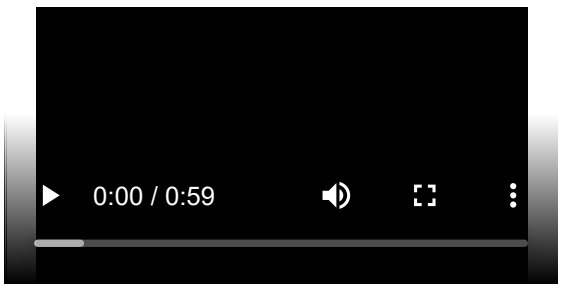


Subscribe

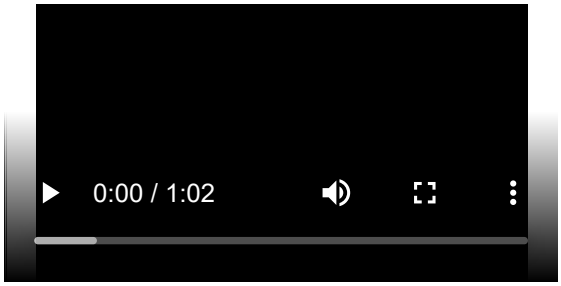
Intro: Censorship Attacks



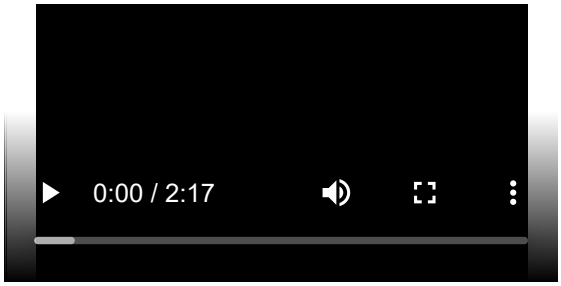
Censorship Attacks: Naive Censorship



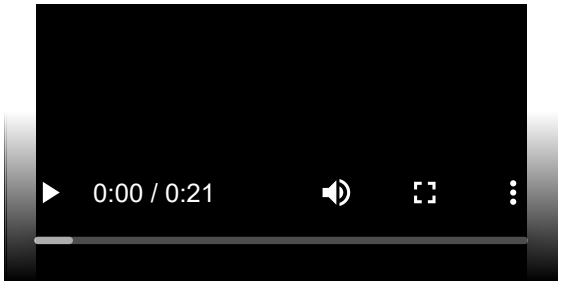
Censorship Attacks: Punitive Forking



Censorship Attacks: Feather Forking

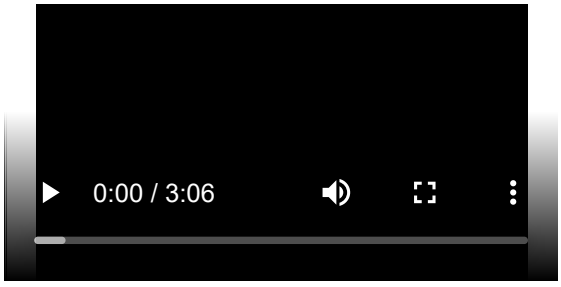


Intro: Selfish Mining

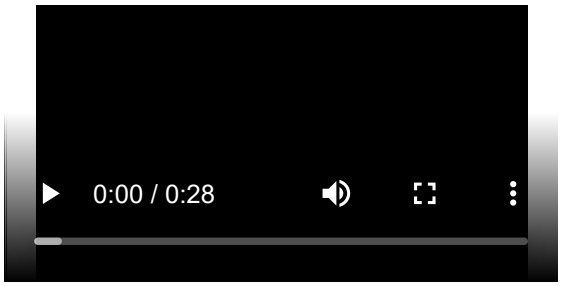


Subscribe

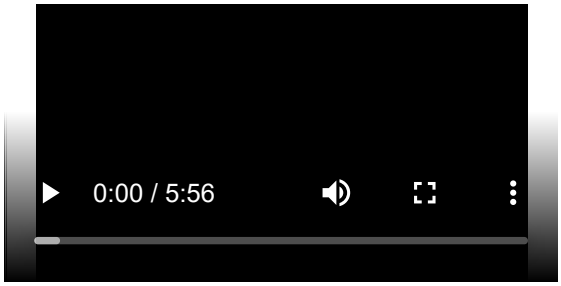
Selfish Mining Attacks



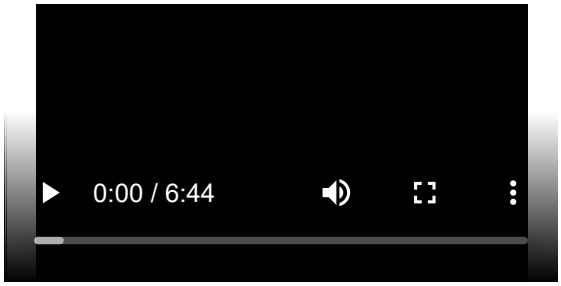
Intro: Defenses



Publish or Perish: Analysis

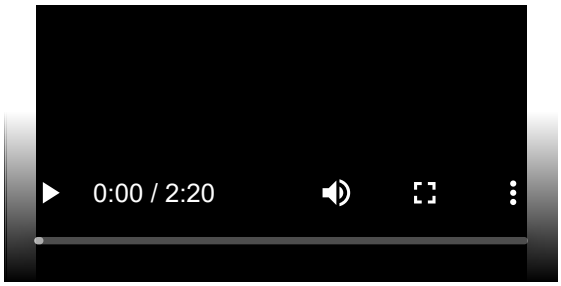


The Bitcoin Network

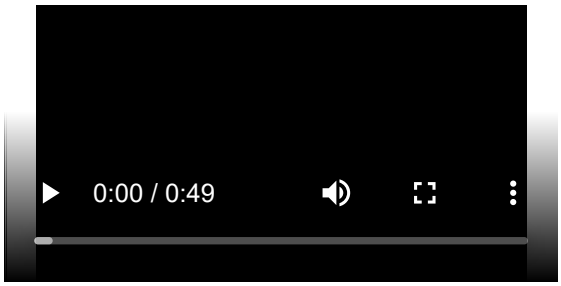


Ethereum & Smart Contracts: Enabling a Decentralized Future

Welcome to Week 6

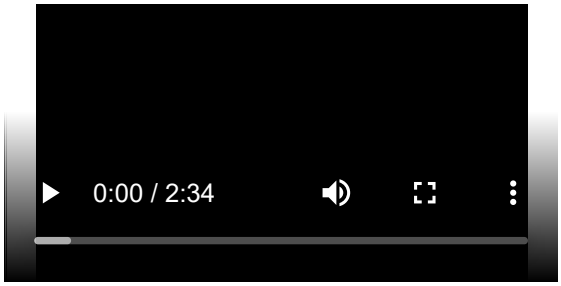


Intro: Smart Contracts

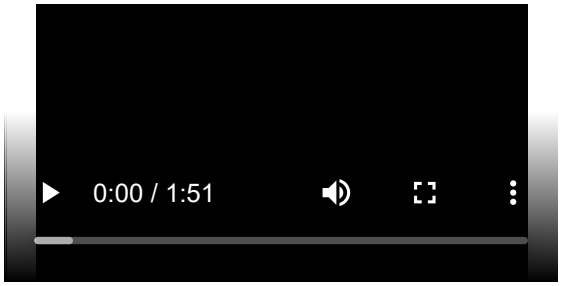


Bitcoin Review

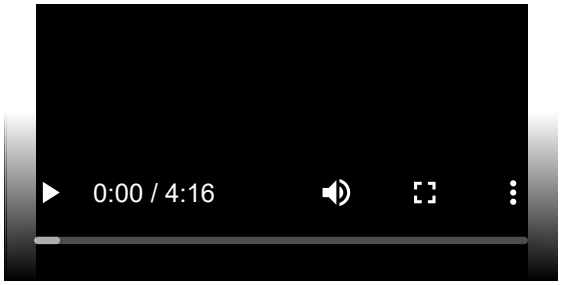
Block Validation



Fork Publishment

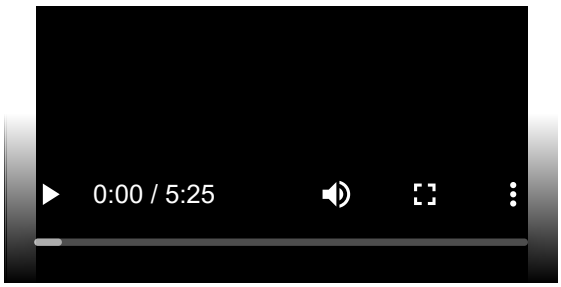


Uniform Tie Breaking

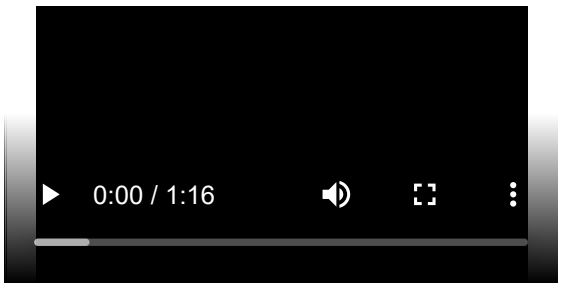


Subscribe

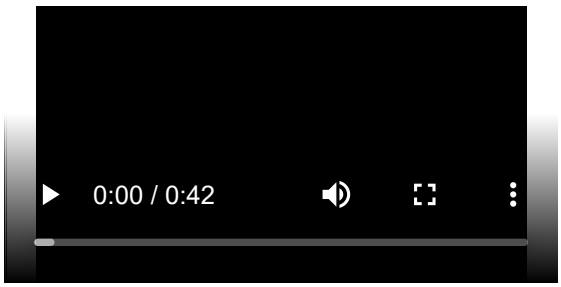
Publish or Perish: Overview



Smart Contracts Overview

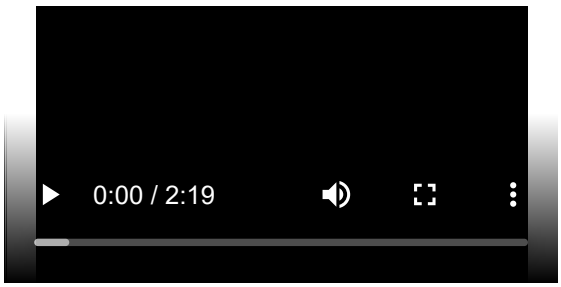


Intro: Ethereum

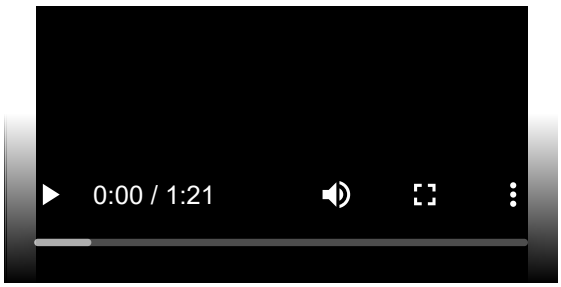


Subscribe

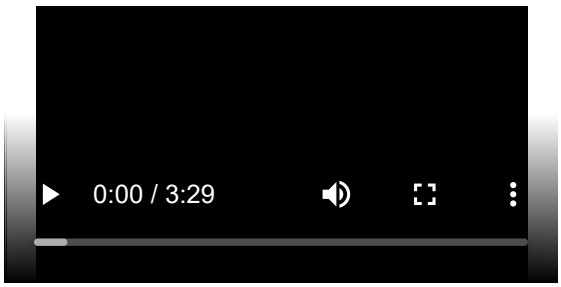
High Level Overview



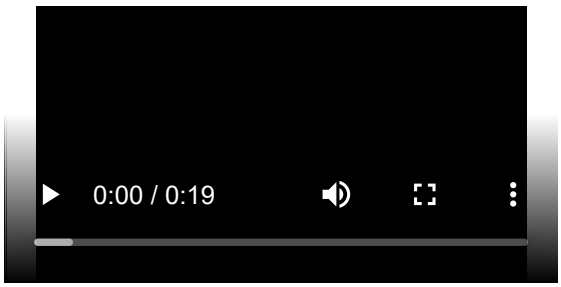
Ethereum vs. Bitcoin



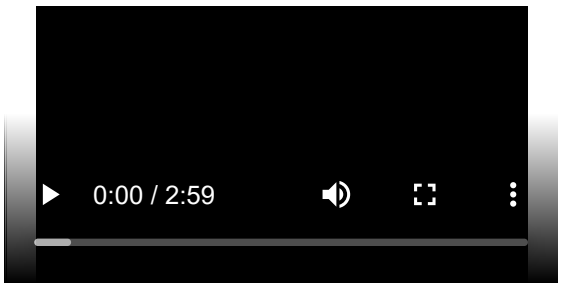
Ethereum Features



Intro: Ethereum Virtual Machine

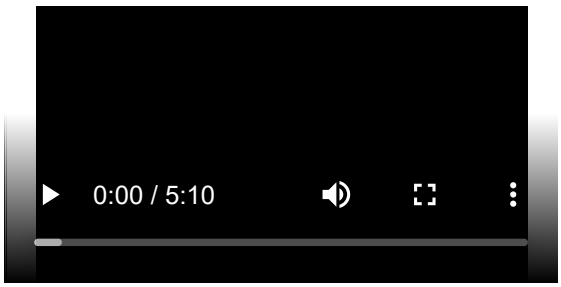


Understanding the Ethereum Virtual Machine

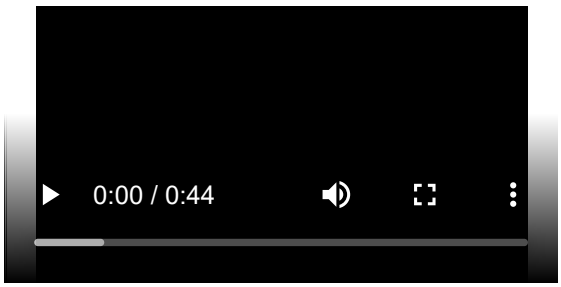


Subscribe

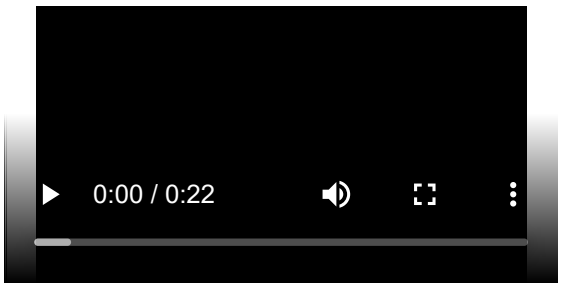
Ethereum Virtual Machine: Overview



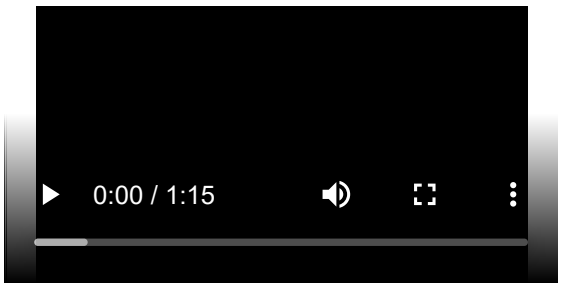
Ethereum Conclusions



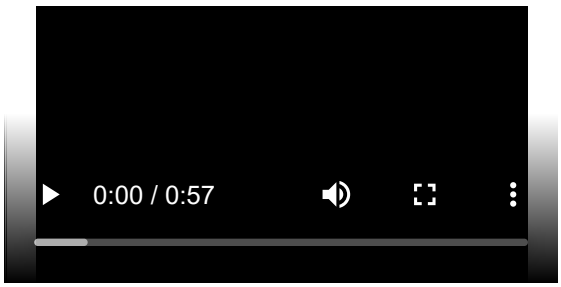
Intro: Ethereum Use Cases



Basic Use Cases: Smart Assets

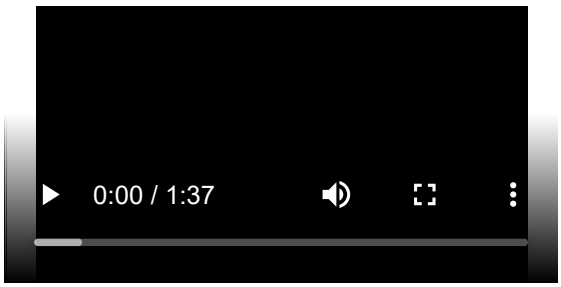


Basic Use Cases: Multisig

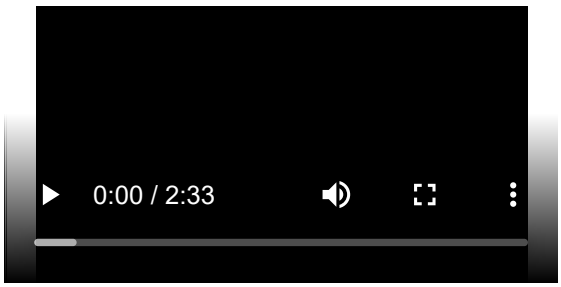


Subscribe

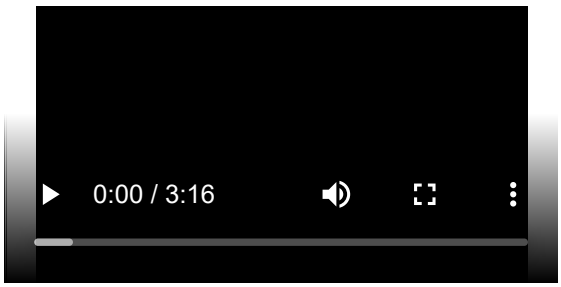
Basic Use Cases: Proof of Existence



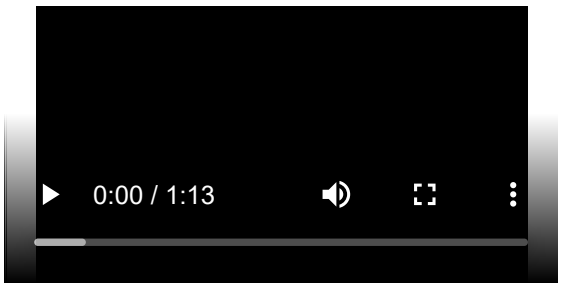
Advanced Use Cases: Land Titles



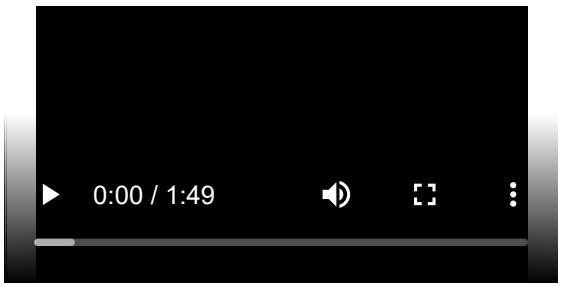
Advanced Use Cases: Prediction Markets



Advanced Use Cases: Supply Chain

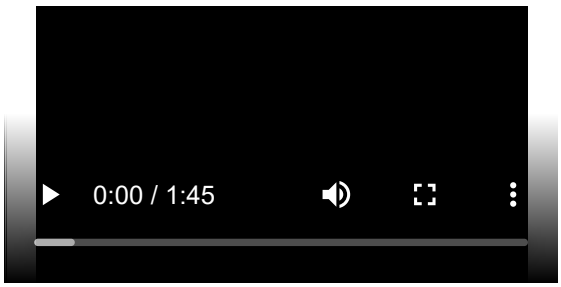


Advanced Use Cases: Smart Energy Grids

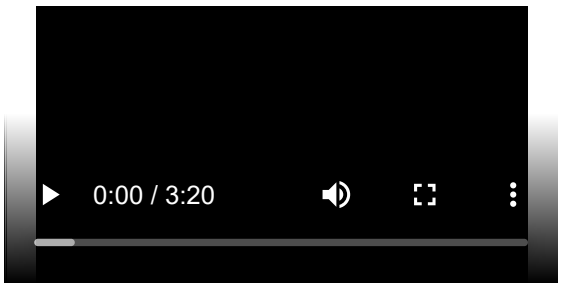


Subscribe

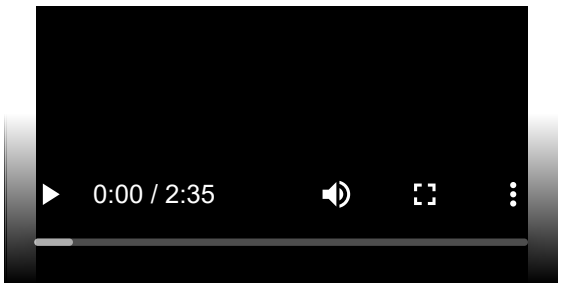
Advanced Use Cases: After Hours Trading



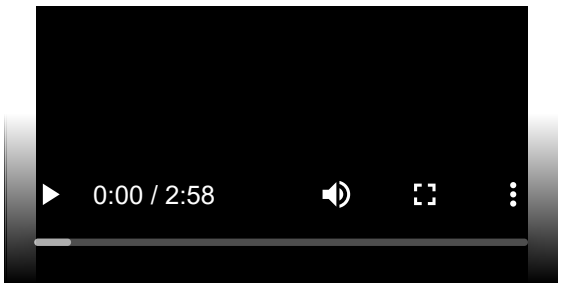
Blockchain vs. Internet



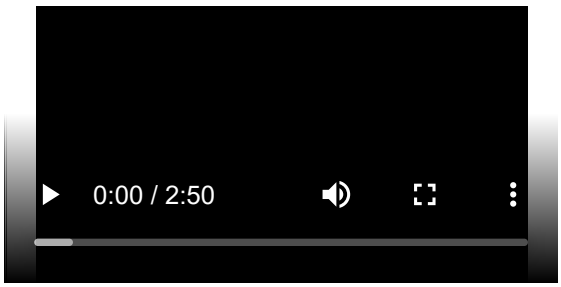
Blockchain Generalizations: Part 1



Blockchain Generalizations: Part 2



Blockchain Generalizations: Part 3



Supplement: Use Cases

Authors: Rea Savla and David Luo

I. Basic Use Cases

The first use case is a smart asset, or token, built on a currently existing blockchain. All that needs to be done is to ensure that a user who wants to spend money has the funds, has been authenticated, and is not double spending.

Another meaningful use case for smart contracts is multisignature wallets. On Ethereum, we can leverage the platform's built-in authentication protocol to create a functionality that allows us to have an m of n signature scheme (refer to Lecture 3).

We can also create Proof-of-Existence use cases using smart contracts, which hash intellectual property onto a blockchain to prove the existence of some piece of information at some point in time. With Proof-of-Existence, we leverage both the public auditability and immutability of the blockchain. We can use Proof-of-Existence to record document ownership or implement a decentralized DNS system.

II. Advanced Use Cases

Many developing nations struggle with broken land titles systems; flawed paperwork, forged signatures, and unclear documents make tracing ownership of land difficult. The pitfalls which keep a central organization from solving this problem are mistrust between citizens and these central parties.

Proof-of-Ownership helps address these issues by tracking ownership of documents, similar to Proof-of-Existence. We can create an association between users and document hashes to prove ownership. This provides transparency and immutability, and it limits centralization. A major caveat to this use case is “Garbage in Garbage Out” (GIGO), which states that if the inputs are incorrect, the outputs will also be incorrect. We refer to the entities that send information from the real world into the blockchain as “oracles.”

Prediction markets are another advanced use case of smart contracts. A prediction market allows users to bet on the future and receive winnings if they bet correctly.We can use prediction markets to “buy information.” Use cases of prediction markets include insurance, bug bounties, ICO signaling, and futarchy.

Smart contracts also have significant potential to secure supply chains. Everledger was created to tokenize and track a diamond’s path from mining to current product.

In addition to providing a digital service, blockchain can also be used to provide a real world service, such as providing infrastructure in the absence of government action to do so. A common infrastructure problem is that of energy grids. Individual households can create smart contracts that promise to pay a certain amount of money to build an electric grid if neighboring households commit to doing the same. A contractor can then redeem the mor in exchange of working on the grid. Through smart contracts, we can coordinate between untrusting parties without a central entity.

Finally, blockchain can also allow us to conduct after hours trading by solving the “Liquidity Problem,” which causes the SEC to restrict after hour trading to placing limit orders. With blockchain, brokers can tokenize stocks by creating legal contracts underwriting them. Stock tokens can be traded worldwide even after the markets close and redeemed for their respective stocks during business hours.

III. Blockchain Generalizations: Essential Properties

Although using blockchain provides numerous properties if implemented in a system, there are often other solutions that provide the desired properties without using a blockchain.

Blockchains provide guarantees of privacy, decentralization, and more. However, these properties are sometimes unaffordable for practical use depending on the user base, goals, and scale of the application.

III.I Efficiency

The word “efficiency” is often used when used to describe blockchains, but blockchains are not always efficient for their proposed use cases they are proposed for. For example, if one

wanted to purchase a coffee using bitcoin, it is far less efficient to wait 10 minutes for a confirmation than it is to simply swipe a credit card or hand the cashier some money. However, sending money across the world using Bitcoin would take a mere ten minutes, compared to the days for coordination between banks internationally. As shown, there are numerous properties that a blockchain provides if implemented, but sometimes there is no need to implement a blockchain in order to achieve these properties for specific uses cases. Efficiency depends on whether the immense cost and complexity of decentralized consensus is overcome by the benefits of blockchain.

In addition, properties like data immutability , integrity, auditability, authenticity, and public key identity infrastructures are all very powerful properties provides by blockchains that a blockchain can possess, but can all be achievable by using previously existing technologies, like cryptography and implementing a system with a hash chains. Redundant, mission-critical, fault-tolerant systems already exist in the form of secure servers or cloud services like such as Amazon Web Services. In addition, these centralized solutions are literally millions of times less expensive to run than current decentralized solutions. Although by definition a blockchain possesses these properties, a centralized solution can just as well implement these, usually cheaper as well.

III.II Solving coordination failures

One major property of a blockchain system is its ability to create arbitrary incentive structures so as to solve coordination failures. By offering rewards for users to act a certain way, a system can therefore incentivize them to follow those certain expectations. Because of this, blockchain can be considered a “technological solution to a social problem.” This is especially useful in use cases where individuals do not trust each, but could benefit from working together. Blockchain allows for them to still coordinate or collaborate without the need for a trusted third party. Funding public infrastructure/crowdfunding is one such example for coordination, such as with the smart energy grid example, which uses smart contracts in order to create a common law without a third party.

III.III Horizontal Integration

Blockchains create a standardized platform for access and interaction, combining the power of all users to enhance all of their capabilities. One such example of this is the combining of data silos. Since all information that is stored on a blockchain is shared, it is accessible for all users on the chain. Any individual contribution to the chain increases the protocol’s value for all. This property, along with that of enforcing having a common formatting standards, is known as network effects, which is the increased value of potential of a product for each additional user. Users support the entire community while also benefiting themselves.

III.IV Pure decentralization

One of the largest selling points of using a blockchain is its decentralization. Pure decentralization refers to decentralization for decentralization’s sake, such as Bitcoin avoiding the functioning bank system out of distrust. So if it required to avoid a centralized authority, a blockchain would be a good solution to this issue. This extends to scenarios of corruption or power saturation in central authorities, such as governments and financial

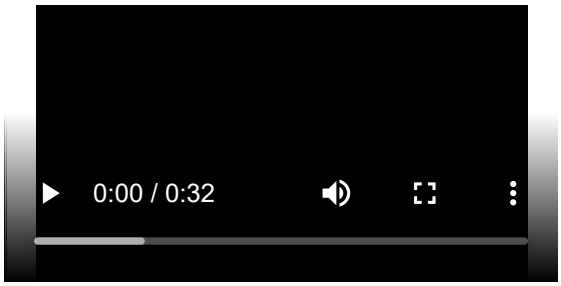
institutions. In a situation where there is a lack of trust in the government due to corruption, a blockchain could be implemented. This decentralization paves the path to also create censorship resistance and disintermediation of power. Because blockchains are globally accessible, they are unstoppable as long as there is a community that supports them.

III.V Advantages of centralized solutions

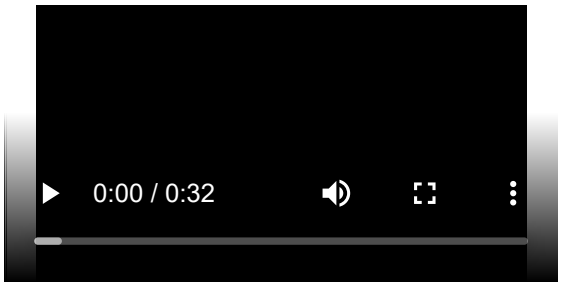
A central solution is fully controlled by a single organization, keeping everything under the same umbrella. One of the benefits of this is simpler and faster software updates, meaning it is easy to make changes and have the update for all users. Contrast this with a blockchain, where everyone instead must have to voluntarily agree to upgrade. It is much easier to make security patches when you don't need to reach consensus on upgrades across all users. Centralized solutions are also more efficient, as instead of having to do a computation for every user in a network, it only needs to be done once, and there is no need for verification. Access control is also much easier, as the central authority can simply keep track of access in a centrally controlled database, something that cannot be done in a decentralized system. Most significantly, centralized systems can make decisions with their human judgment, especially helpful when tackling complex decisions. In a decentralized system, smart contracts may not be able to cover or formalize all of the edge cases, and may not be suitable for use in more complex issues. For example, if Airbnb were replaced by used smart contracts, how would a smart contract be able to determine what to do if household possessions were damaged? Which individuals could be trusted to report this information without stripping away the values of decentralization? How would a smart contract handle human error?

In the end, it is important to always ask why a blockchain is better than a centralized soluti Neither is better than the other universally, so it is important to compare the pros and cons of both to determine which one is better for each specific use case.

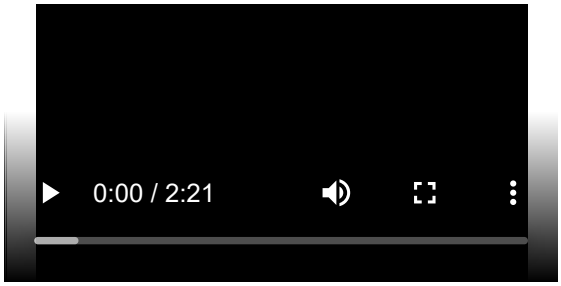
Blockchain Use Cases Takeaways



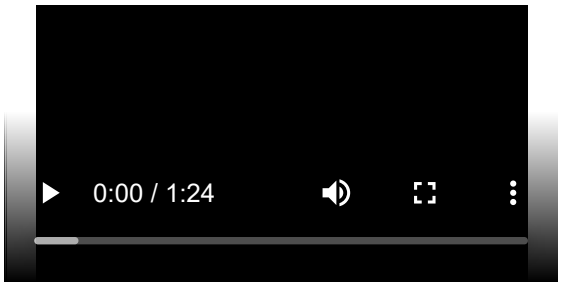
Intro: Ethereum Ecosystem



Community Leaders



Ethereum Research



Text: Lecture 6 Summary

Author: Rea Savla

I. Smart Contracts

Subscribe

Smart contracts on Ethereum enable us to execute arbitrary computations on the blockch
A smart contract is a piece of code that facilitates, verifies, or enforces the negotiation or execution of a digital contract. For us to reach consensus, a trusted entity must run this code.

II. Ethereum

Ethereum is a decentralized platform designed to run smart contracts. It is a distributed computer spread across many nodes around the world that executes the code people feed in it, without any possibility of downtime, fraud, censorship, or third party intervention.

Here's a review of the similarity and differences between Ethereum and Bitcoin:

There are two kinds of Ethereum accounts:

- Externally Owned Accounts contain an address that allows people to send them ether and a balance of ether
- Contract Accounts contain an address, persistent storage, and code which is executed by transactions or function calls

Ethereum smart contracts generally serve four main purposes:

- Store and maintain data
- Manage contract or relationship between untrusting users
- Provide functions to other contracts
- Complex authentication

III. Ethereum Virtual Machines

The Ethereum Virtual Machine enables the Ethereum blockchain to be programmable. Ethereum Smart Contracts are written in high level programming languages like Solidity, which have to be compiled down to Ethereum Virtual Machine (EVM) code. Every node runs EVM as part of its block verification procedure. Once the contract code is compiled to EVM code, every node in the Ethereum network executes the code. Nodes then come to consensus on the new system state. Note: Ethereum is redundantly parallel; its main goal is not to optimize computational efficiency, but to enable distributed and trustless computation.

As in Bitcoin, the nodes come to consensus using Proof-of-Work, removing the need for a trusted third party. Miners in Ethereum competitively create blocks by executing EVM code and searching for a solution to a mining puzzle.

The developers of Ethereum implemented ‘gas’ to protect against the Denial of Service (DoS) Attack, where an attacker includes an infinite loop in the smart contract to disable nodes from executing any other contracts. Gas is what fuels a given contract. You can think of purchasing gas as the price you have to pay to use this distributed, trustless computational

power. This disincentivizes DoS attacks because attackers would have to pay an absurdly large amount of ether to execute this attack.

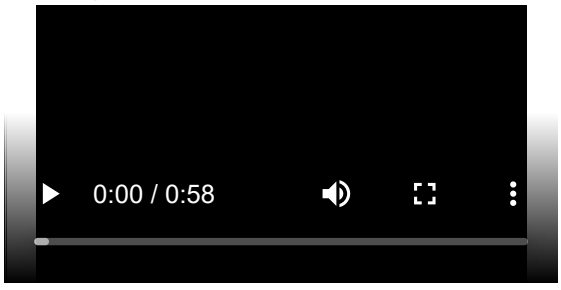
Ethereum is implemented as a distributed state machine, and transactions on the network change the global state of the system. In the transaction execution, or "state transition function," nodes in Ethereum come to consensus on the network state, and their code execution on the EVM changes the global state from the previous state to the new state. The function looks like this:

Readings

- Ethereum Whitepaper: A Next-Generation Smart Contract and Decentralized Application Platform Opens in new window
- (Optional) Explore decentralized applications Opens in new window Check out all the different Ethereum Apps that have been created!
- (Optional) Ethereum: A Secure Decentralized Generalised Transaction Ledger Opens in new window
- Blockchain, Cryptocurrencies & The New Decentralized Economy Opens in new window

The End

Thank you!



Next Steps

Thanks for taking our course. We hope you enjoyed taking it as much as we enjoyed teaching and facilitating it.

This course honed your Bitcoin and cryptocurrency thinking skills, but barely touched upon blockchain technology as a whole. Having understood cryptocurrency, it's now time to learn blockchain in general. The next course in the Blockchain Fundamentals program will cover topic such as:

- Alternative consensus, removing the need for useless and expensive Proof-of-Work computation
- Cryptoeconomics and Proof-of-Stake
- Blockchain Scalability
- Enterprise Blockchain and Real-World Applications
- and more...

Congratulations again for getting through this first course. We hope to see you in class again soon!

Read More

Leave a Reply

Your email address will not be published. Required fields are marked *

COMMENT

NAME *

EMAIL *

Save my name, email, and website in this browser for the next time I comment.

Post Comment

Search ...

Subscribe

LEARNTHINGS.ONLINE TELEGRAM GROUP

IPFS for Beginners – Interact With IPFS By Javascript

In this article, we'll learn how to interact with IPFS by javaScript programming language. It's one way to make your own application to interact with IPFS. The post IPFS for Beginners – Interact With IPFS By Javascript appeared first on LearnThings.Online.

Subscribe

Facebook Rename Its Libra Wallet Project Calibra to Novi

2020 May 26, Facebook rename its Libra wallet project Calibra to Novi. It makes its name more separate from Libra. Novi plans to launch its App in 2020. The post Facebook Rename Its Libra Wallet Project Calibra to Novi appeared first on LearnThings.Online.

Libra Appoints It's General Counsel, a Former HSBC, and Goldman Sachs

On May 19th, 2020, the Libra association appoint Robert Werner, an Ex-HSBC & Ex-Goldman Sachs the founder and CEO of GRH Consulting, as its general counsel. The post Libra Appoints It's General Counsel, a Former HSBC, and Goldman Sachs appeared first on LearnThings.Online.

Subscribe