Introduction to digital currencies

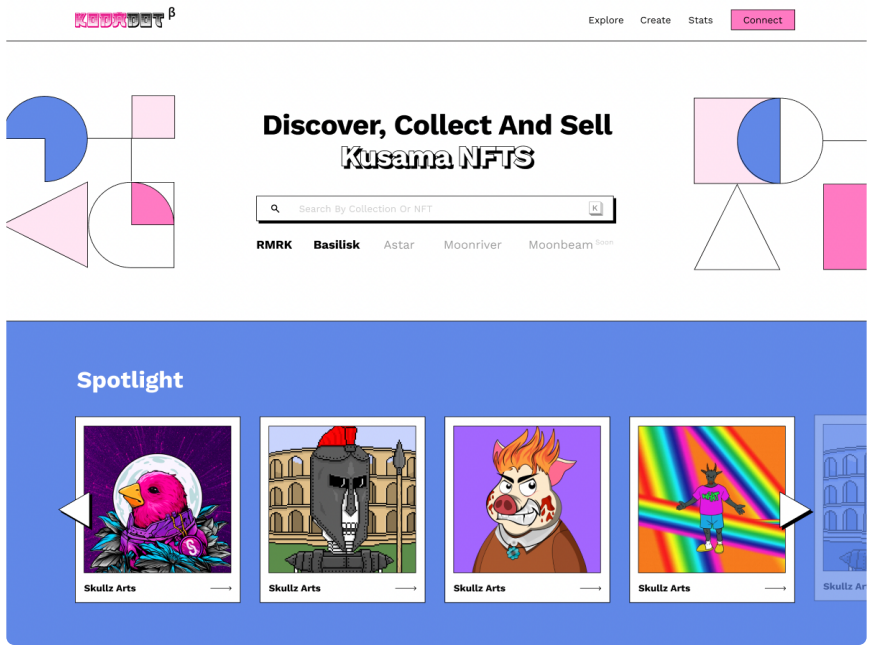
and other fancy stuff

Hey, I am Viki 👋

- co-founder of KodaDot
- external PhD student at STU FIIT
- technical wizard
- bleeding edge implementations in KodaDot
- Github / Twitter: @vikiival
- Discord: vikiival|KodaDot#0001
- Web: vik.ink

What is KodaDot?

- Open-source NFT marketplace in the Dotsama ecosystem
- Multichain powered - Kusama, Basilisk, Moonbeam, Moonriver
- Technology agnostic - RMRK strings, runtime pallets, EVM, ink!
- Bounty based - bringing dev talent to the ecosystem
- **330 + forks, 220+ stars, 80+ contributors**



Attempts for digital money

Why digital money?

- **Digital money** is a **digital representation** of value that can be **transferred** or **stored** electronically.
- **Digital money** is a **medium of exchange** that is **digital** and has some **unique properties** that **distinguish it** from **traditional money**.

(credits to David Stancel @ CoinStory.tech)

Timeline

- **Digicash** - 1990
- **E-gold** - 1996
- **Hashcash** - 1997
- **B-money** - 1998

Digicash

- 1990 with excelent team
- first serious attempt to create digital money
- allowing anonymous transactions
- secure - using digital signatures RSA

How it works

1. Obtain - withdraw cash from bank account
2. Swap - exchange cash for digital cash - stored on PC
3. Spend - send payment request to the merchant
4. Verify & Accept - merchant verifies the payment request and accepts it
5. Blind signatures - wrap the payment request in a digital envelope.

E-gold

- digital currency backed by physical gold
- SSL encrypted
- immediate settlement
- API - in 1996!
- anonymous, irreversible transactions
- 1/10th of cost of credit card transaction
- good ground for Ponzi schemes

Hashcash

- proposed in 1997 by Adam Back
- released as a paper in 2002
- used as a spam prevention mechanism in email
- used cryptograpic puzzle to prove work
- used as a proof of work in Bitcoin

B-money

- 1998 by Wei Dai
- first digital money with proof of work
- used as a base for Bitcoin
- idea of turning electricity into money

BitGold

- created 1998 by Nick Szabo
- worked on Digicash and was sceptical about it
- released whitepaper in 2005
- goal: "Get rid of the trusted third party"
- the foundation of Bitcoin

How BitGold works

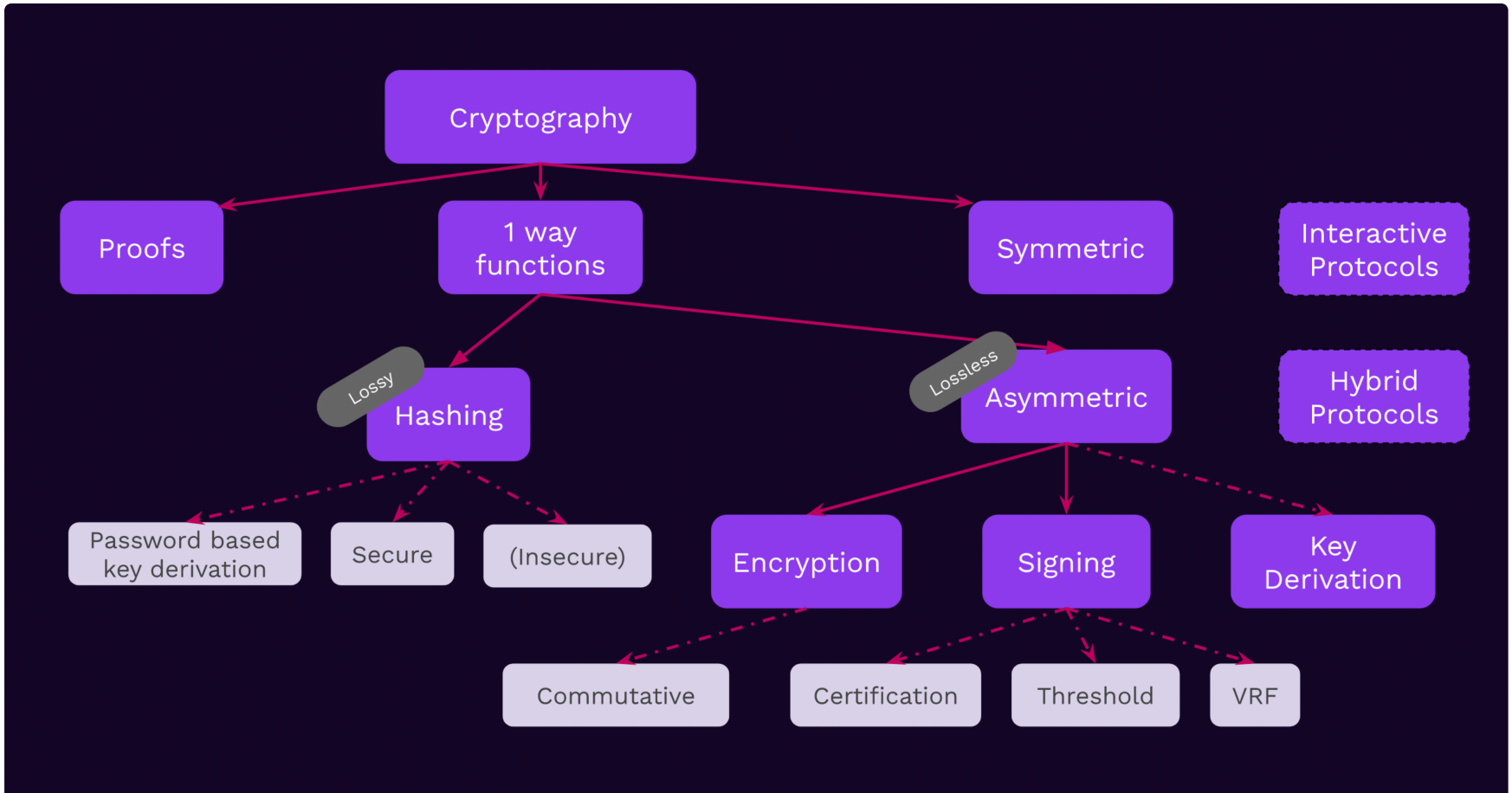
1. A public string of bits, the "challenge string," is created (see step 5).
2. Alice on her computer generates the proof of work string from the challenge bits using a benchmark function.
3. The proof of work is securely timestamped. This should work in a distributed fashion, with several different timestamp services so that no particular timestamp service need be substantially relied on.
4. Alice adds the challenge string and the timestamped proof of work string to a distributed property title registry for bit gold. Here, too, no single server is substantially relied on to properly operate the registry.
5. The last-created string of bit gold provides the challenge bits for the next-created string.
6. To verify that Alice is the owner of a particular string of bit gold, Bob checks the unforgeable chain of title in the bit gold title registry.
7. To assay the value of a string of bit gold, Bob checks and verifies the challenge bits, the proof of work string, and the timestamp.

What is tech stack behind blockchain?

Few things to remember:

- 🔑 **Asymmetric encryption**
- 📩 **Hash functions**
- 👨‍🔬 **Elliptic Curve Cryptography**
- 🎄 **Merkle Trees**
- 🤝 **Consensus**
- 📦 **Blocks**

Fast lesson on cryptography



Why is cryptography useful?

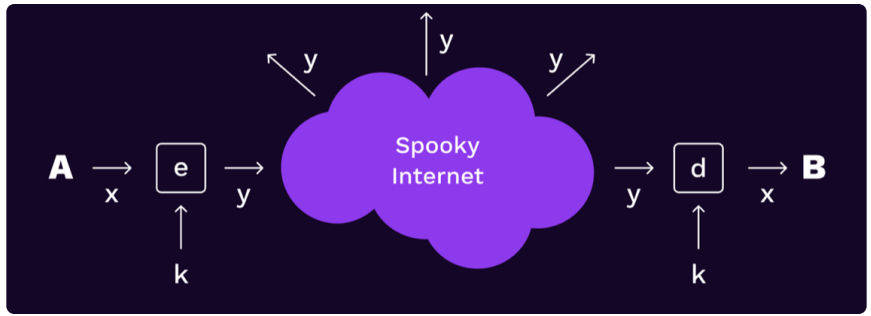
- **Secure communication** - ensure that only the right person can read the message
- **Data accessibility** - gain access to data if we have the right key
- **Message authenticity** - prove that the message was sent by the right person
- **Data integrity** - physical signatures are easy to forge, digital signatures are not

Why is cryptography useful?

- **Secure communication** - ensure that only the right person can read the message
- **Data accessibility** - gain access to data if we have the right key
- **Message authenticity** - prove that the message was sent by the right person
- **Data integrity** - physical signatures are easy to forge, digital signatures are not

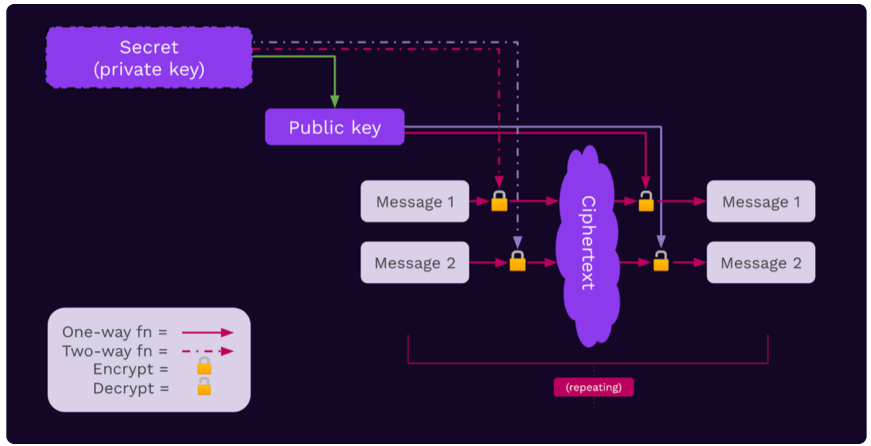
Symmetric encryption

- uses the same key for encryption and decryption
- the key is shared between sender and receiver
- super fast
- requires trust between sender and receiver



Asymmetric encryption

- uses two different keys for encryption and decryption
- pretty slow
- preserves algebraic properties
- less trust, more truth
- public key to encrypt information
- private key decrypts the information



What is a hash function?

- **one-way function** - it's easy to compute the hash, but hard to find the input
- **deterministic** - the same input always produces the same output
- **collision-resistant** - it's hard to find two different inputs that produce the same output
- **accept any input size** -
- **output should have bounded size** -
- **be fast to compute**

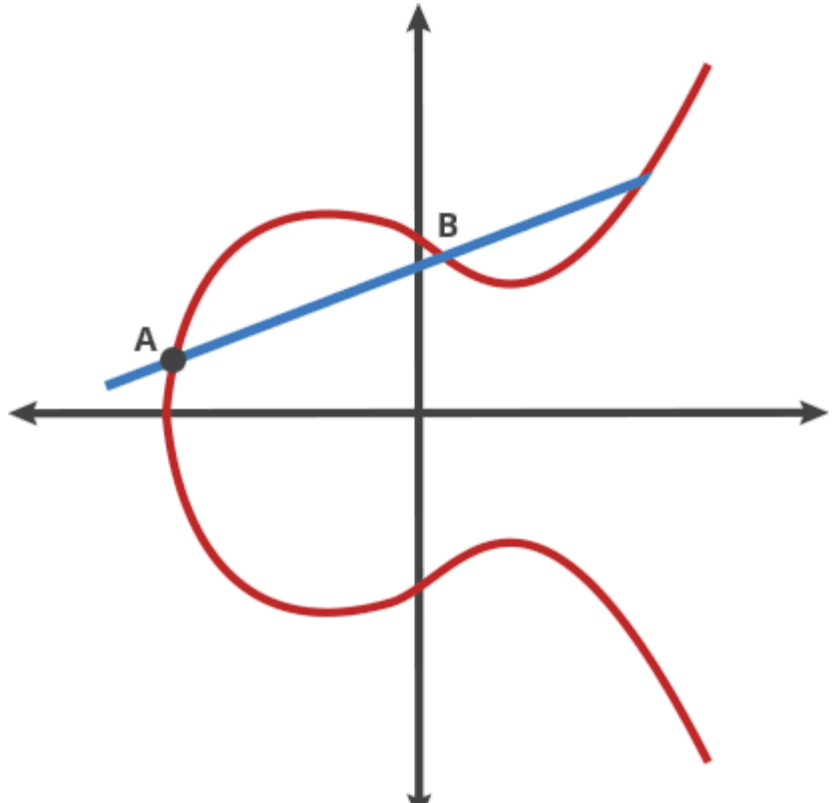
Usage

- **representation of data object** - history, state, etc
- **keys in db** - to encrypt a message
- **digital signatures** - to prove that the message was sent by the right person
- **key derivation** - to generate a key from a password
- **pseudo-random numbers** - to generate a random number

Eliptic curve cryptography

Eliptic curve cryptography is a type of public key cryptography. It is based on the theory of elliptic curves over finite fields. It is used in many applications, including cryptocurrencies.

- most powerful but least understood types of cryptography in wide use today
- TL;DR is: ECC is the next generation of public key cryptography and, based on currently understood mathematics, provides a significantly more secure foundation than first generation public key cryptography systems like RSA
- A (Relatively Easy To Understand) Primer on Elliptic Curve Cryptography



Can we touch blockchain? 🙏

- anders94/blockchain-demo
- blockchaindemo.io

Sooo what is blockchain?

- A tamper-proof, shared digital append-only ledger that records transactions grouped into blocks in a decentralized peer-to- peer network.
- The permanent recording of transactions in the Blockchain stores permanently the history of asset exchanges that take place between the peers.
- Updating the ledger (usually) requires solving Byzantine Agreements (hash) with economically incentivized participation, secured by cryptography

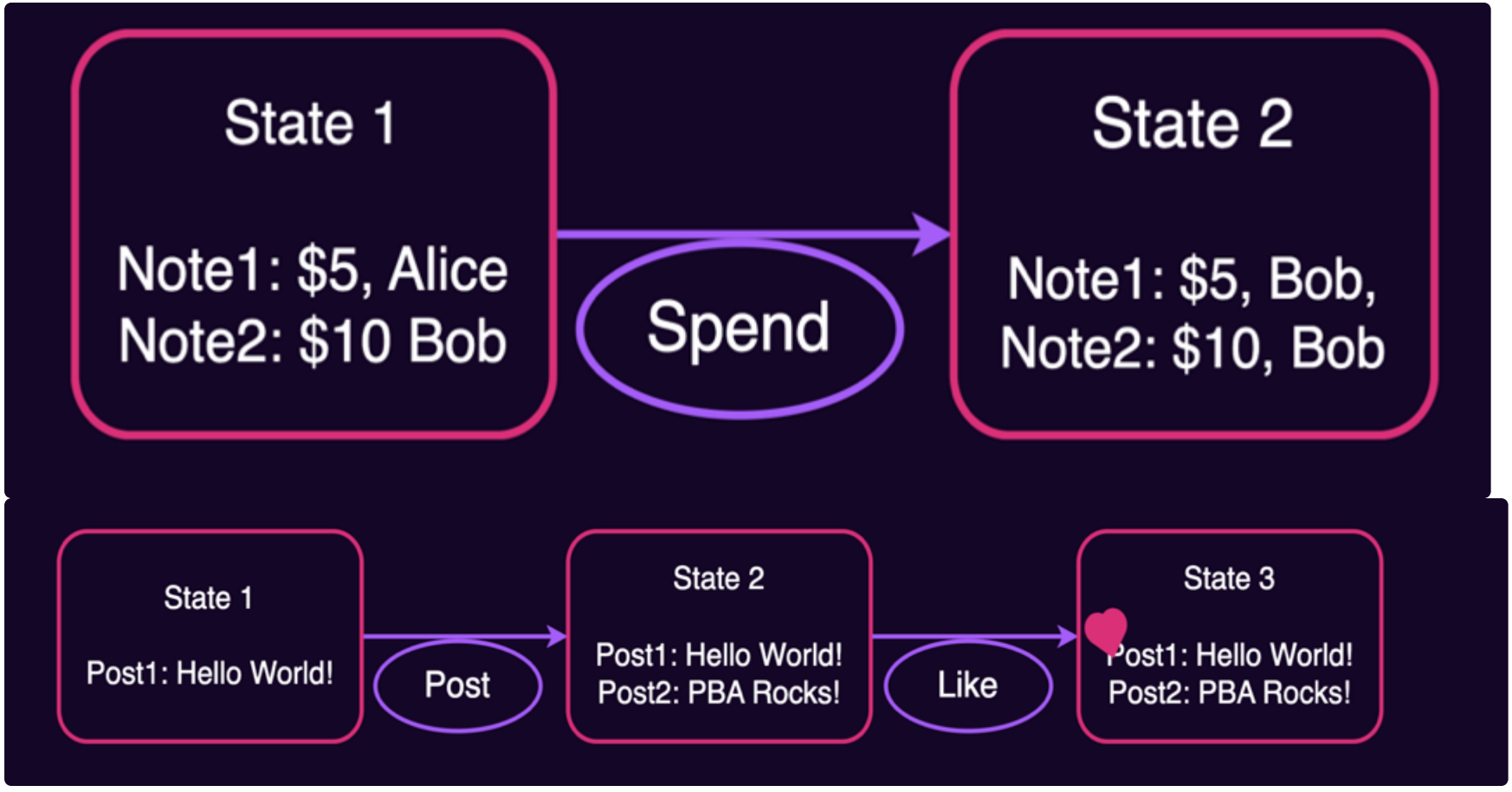
Which properties should blokchain have?

- **Permissionless access** - Anyone should be able to access and interact with the system.
- **Privacy** - Users should have credible expectations about what information they give up about themselves.
- **Authenticity** - Users should have credible expectations about the messages they see, regardless of the platform the messages are on.
- **Finality** - Users should be able to form credible expectations about when a state transition is final.
- **Unstoppability** -No individual actor, company, state, or coalition should be able to degrade any of these properties.
- **A shared history** - formalize the history of the system using state machine

OK, plz Viki what is blockchain?

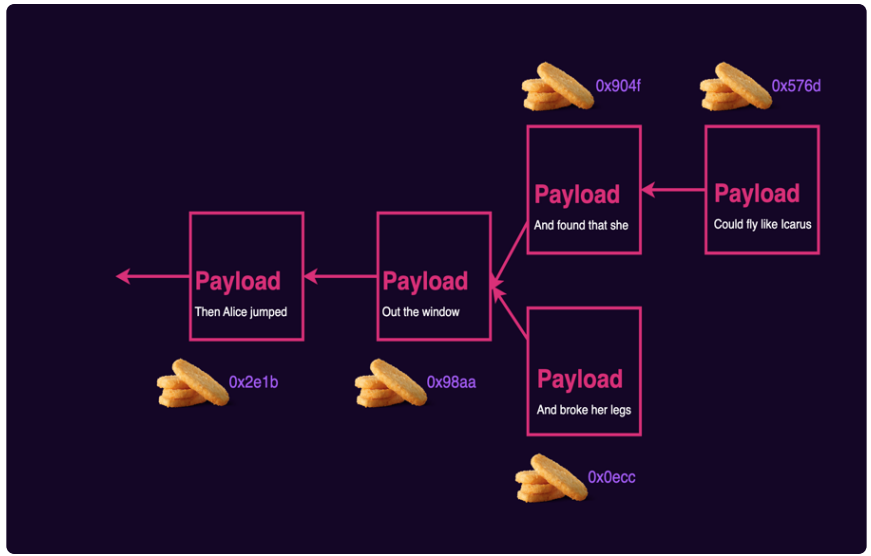
- **State Machine**
  - What does the state hold?
  - What are the rules to change it?
- **Data Structure**
  - How can we cryptographically represent a history so it is tamper-proof?
- **Consensus**
  - Which history is the real one?
  - What part of history is final?

Story of state machine



What is the data structure of blockchain?

- **linked list of blocks**
- each block contains: a hash of the previous block, a timestamp, and transaction data.
- By making each block point to the previous one, you are forming a chain.
- extremely difficult to modify a past block, because doing so would require changing all subsequent blocks.



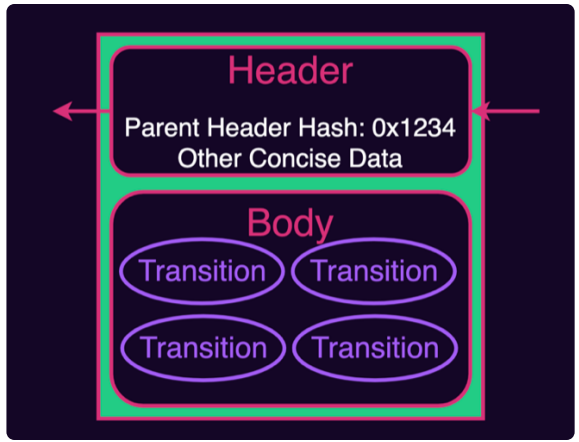
Can you show me a block? 🥺

Bitcoin

- Version
- Previous Hash
- Tx Merkle Root
- Time
- N Bits
- Nonce

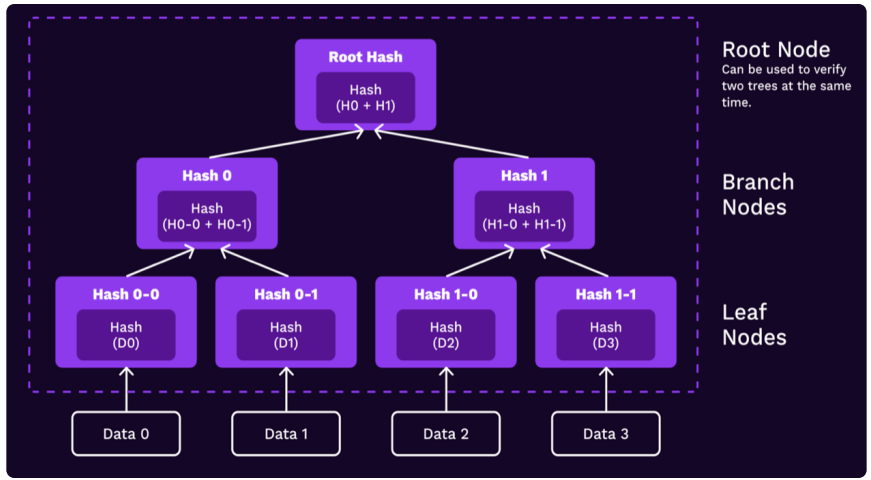
Ethereum

- Time
- Block Number
- Base Fee
- Difficultv
- Mix Hash
- Parent Hash
- State Root
- Nonce

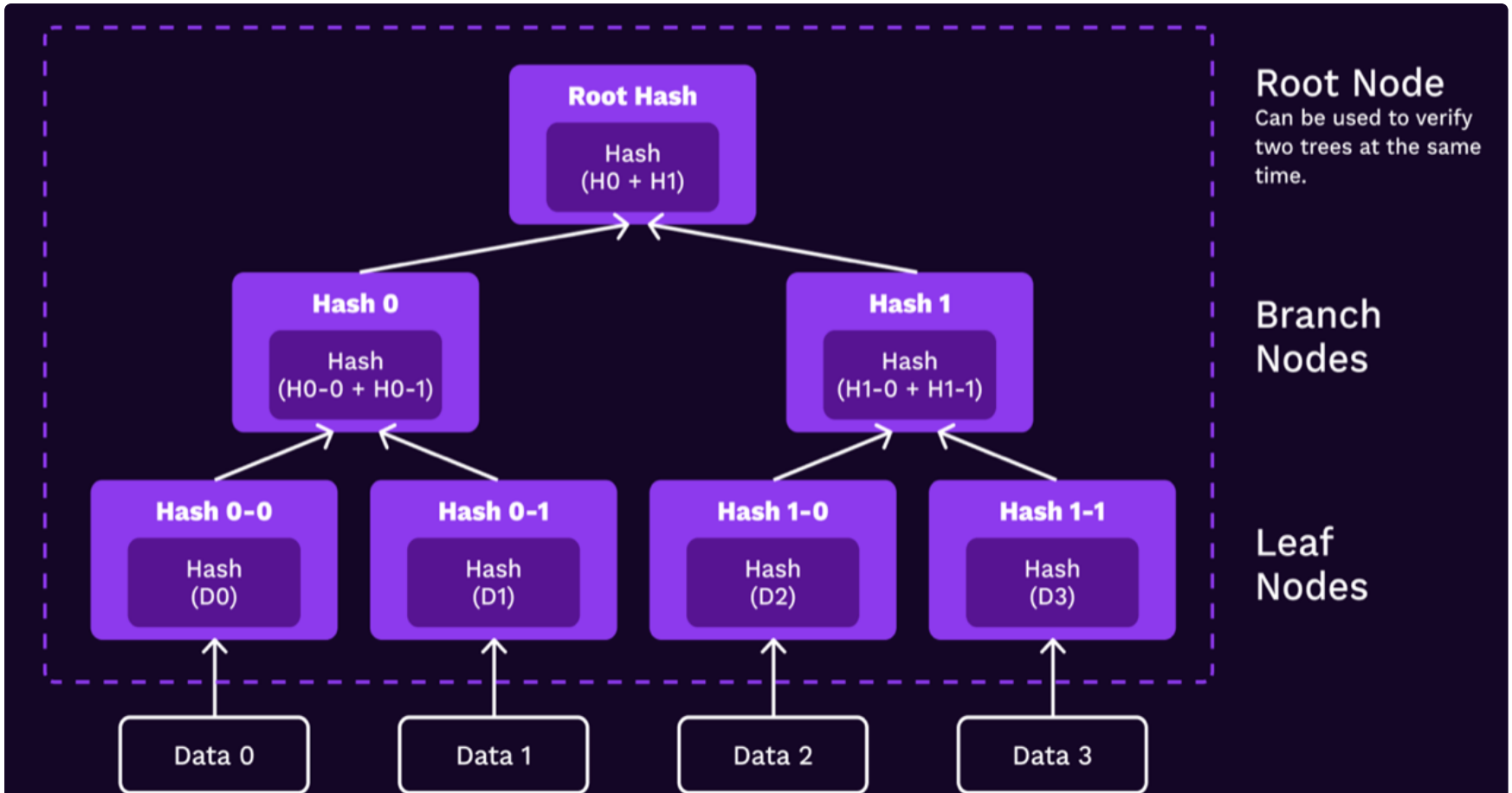


What is Merkle Tree?

- A data structure that allows you to verify the integrity of a large set of data
- It is a binary tree where each leaf node is a hash of a data block
- Each non-leaf node is a hash of the two child nodes
- The root node is a hash of the entire tree
- The tree is built bottom-up
- The tree is traversed top-down
- The tree is immutable
- The tree is append-only



Merkle Tree but bigger

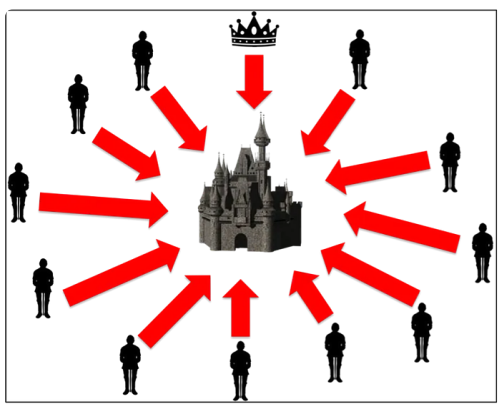


The Byzantine Generals Problem

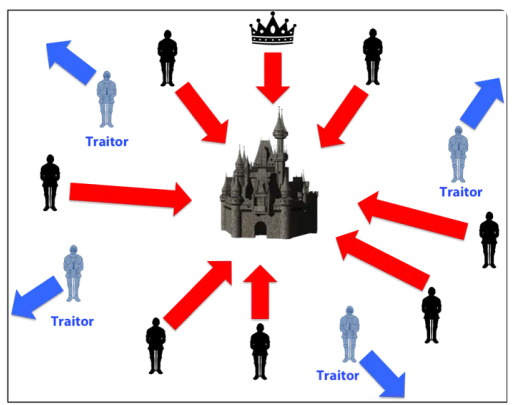
- involves a group of generals who must coordinate their attack or retreat on an enemy city through a series of messengers
- **problem:** some of the generals may be traitors who are trying to sabotage the mission, and they may send false messages to confuse the loyal generals.
- often used as an analogy for the challenges of achieving consensus in a decentralized system, such as a blockchain
- **solution:** involve redundancy and error-checking mechanisms to ensure that false messages are detected and corrected
- related to the concept of the "Byzantine fault tolerance," which refers to a system's ability to continue functioning even when some of its components fail or behave maliciously
- blockchain network involves multiple nodes that must reach consensus on the state of the ledger, even in the presence of malicious actors who may attempt to disrupt the system

The Byzantine Generals Problem in blockchain

- blockchain network involves multiple nodes that must reach consensus on the state of the ledger, even in the presence of malicious actors who may attempt to disrupt the system
- the consensus mechanism is used to ensure that all nodes in the network agree on the state of the ledger
- Just as the generals must agree on a single course of action, the nodes in a blockchain network must agree on a single version of the ledger. If there are disagreements or inconsistencies, the network may be vulnerable to attacks or fraud



Coordinated Attack Leading to Victory



Uncoordinated Attack Leading to Defeat

Can we agree or not?

> Blockchain consensus a decentralized consensus system to reach agreement over a shared history of a state machine.

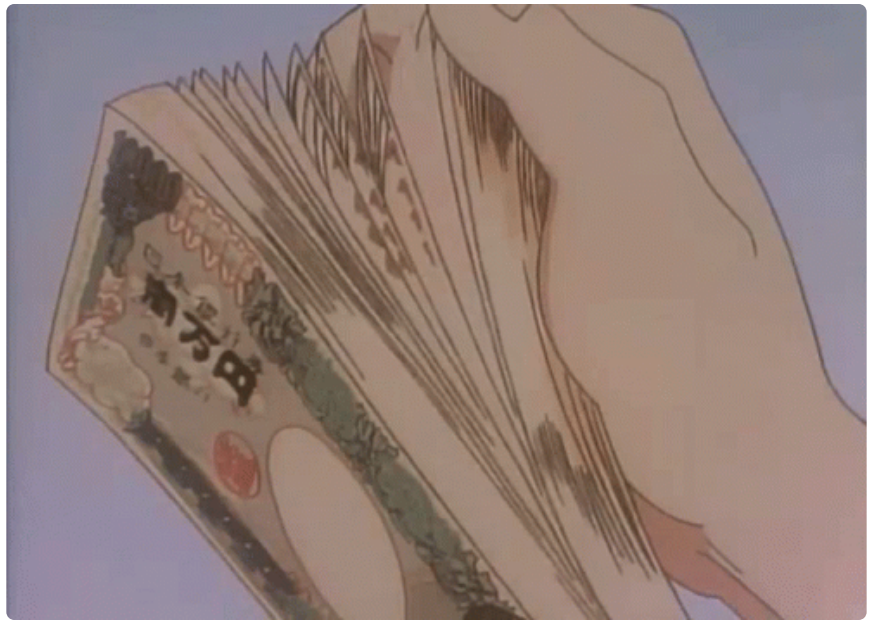
- used in blockchain to reach an agreement among network participants about the current state of the ledger, which contains all the transactions that have been made on the blockchain
- **proof of work (PoW), proof of stake (PoS), and delegated proof of stake (DPoS).**
- Consensus algorithms aim to ensure that the blockchain is secure, decentralized, and resistant to attacks and fraudulent activity.
- Some of the key factors to consider when evaluating a consensus algorithm include scalability, energy efficiency, security, and decentralization.

Proof-of-Work

- Mining is the process of adding new blocks to the blockchain.
- Miners compete with each other to find the correct nonce for a block and achieve the required hash.
- The miner who finds the correct hash first is rewarded with a coinbase transaction.
- Finding the correct hash requires a lot of energy, but cheating is not possible due to the one-way nature of the hash function.
- Other nodes in the network can easily verify the correctness of the hash by reading the block.
- If the hash is correct, the block is added to the blockchain and the miner is rewarded with a coinbase transaction.

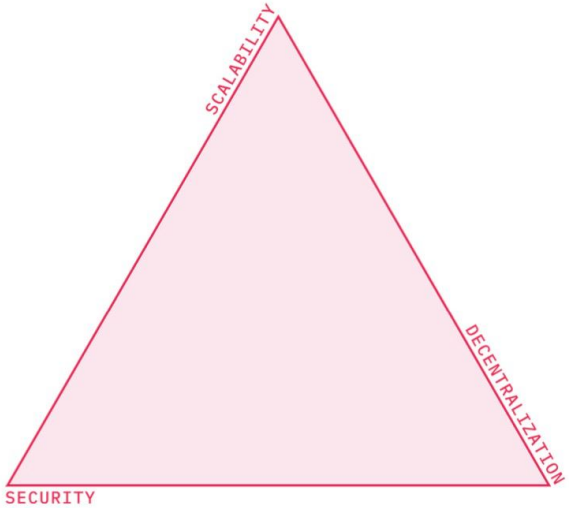
Proof-of-Stake

- validators validate blocks and the data
- they do not compete to find the correct hash for a block
- reward in the form of transaction fees from the transactions included in the block they validate.
- need to stake a certain amount of coins to become validators and secure the network.
- if they cheat, they lose their stake.
- PoS requires less computing power and is less energy-intensive compared to PoW.
- Ethereum requires 32 ETH to become a validator.



Blockchain Trilema

- The blockchain trilema is a concept that describes the trade-offs between **decentralization, security, and scalability.**



Credits & Acknowledgements

- David Stancel @ CoinStory.tech
- Polkadot Blockchain Academy
- Internet for anime gifs
- KodaDot for inspiration
- Roman Bitarovsky for consensus algorithms
- **SubWork hackathon**

Reading & Resources:

- http://www.datnos.com/bc-demos/public-private-keys/keys.html
- A. Back - Hashcash
- N. Szabo - Bitgold
- W. Dai - B-money
- S. Nakamoto - Bitcoin Whitepaper
- H. Finney - RPOW
- Lamport, Shostak, Peace - The Byzantine General Problem
- Diffie, Hellman - New Directions in Cryptogprahy
- D. Chaum - Blind Signatures for Untraceable Payment
- S. Haber, S. Stornetta - How to Timestamp a Digital Document
- D. Chaum - Dinning Cryptographers Problem

How did we actually get to blockchain?

> Blockchain ecosystem is usually refered as Web 3.0

- **Web 0.0** - 1960s - 1970s
  - ARPANET
  - email
- **Web 1.0** - 1990s
  - web browser
  - web search engine
  - web hosting
  - web server
  - web page
- **Web 2.0** - 2000s
  - social network