

Bitcoin: A Secure, Distributed System

Part 1

David Koleczek

Goals

What is Bitcoin and cryptocurrency?

- See the difference between government issued fiat money (USD)

Why does it have any value? Ex. 1 BTC = \$30,000+

What are its “investment” characteristics?

- Risks
- Future outlook

How do we reason about similar technologies like other cryptocurrencies?

My goal is **not** to answer these questions directly, but to give the technical background so you can reason about them yourself.

How?

Cryptocurrency or more generally, *blockchain*, is a combination of many CS topics

- Applied Cryptography & Security:
 - Kerckhoffs's principle
 - Cryptographic hashes (SHA-256, RIPEMD-160)
 - Public Key Cryptography (RSA, Elliptic-curve cryptography, Digital Signatures)
- Networking: (Peer-to-peer, message passing)
- Systems & Hardware: Scalability, computational power, ASICs
- Software Engineering
 - testing, high coding standards, bugs and exploits are catastrophic
- Algorithms and Data Structures
 - Merkle trees, bloom filters

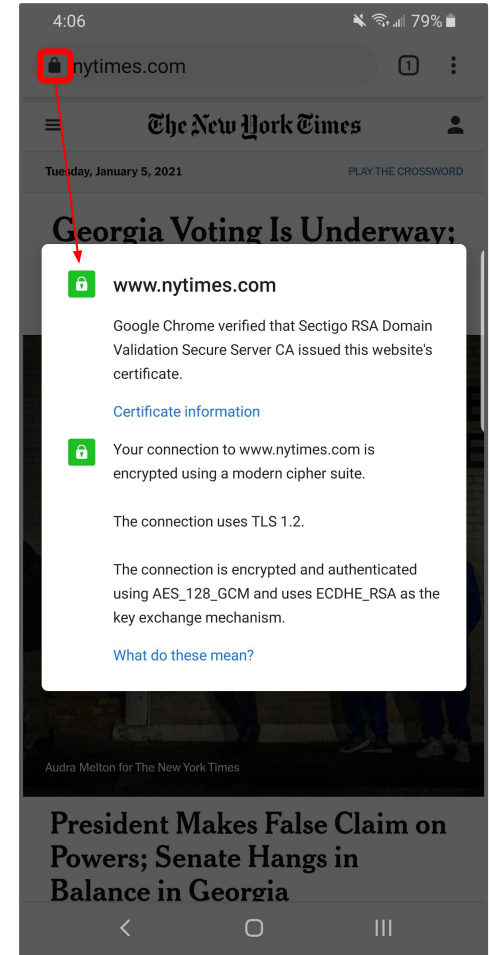
Kerckhoff's Principle

A cryptosystem should be secure even if the attacker knows all the details of the system, with the exception of the secret key.

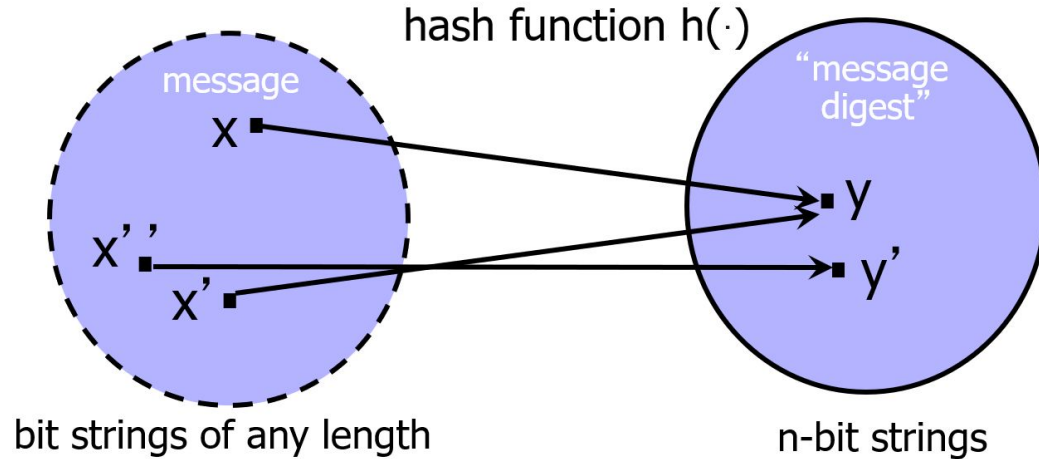
Implications: You can know **all** details about a system.
 i.e. source code, algorithms used for security, etc
 Yet the system is secure.

In contrast, *security by obscurity*

- "System security should not depend on the secrecy of the implementation or its components." - NIST



Cryptographic Hash Functions



Definition: $h(x) = y$

y is the resulting hash and has a small fixed length (ex. 160 bits)

x is called the *preimage* of y and can be of any size

Cryptographic Hash Functions

Several properties of such functions (MD5, SHA-256, RIPEMD-160) make them cryptographically strong

1. Preimage resistance (one-wayness)
 - a. For any y , it is computationally infeasible to find an x such that $h(x) = y$
2. Second preimage resistance (weak collision resistance)
 - a. Given x_1 and $h(x_1)$, it is infeasible to find an $x_2 \neq x_1$ such that $h(x_2) = h(x_1)$
3. Collision resistance (birthday attack)
 - a. It is infeasible to find **any** $x_1 \neq x_2$ such that $h(x_1) = h(x_2)$
 - b. Brute force attack is $O(2^{n/2})$ where n is the number of bits
 - i. See [birthday problem](#)
4. Corollary: Even changing just **one** bit of the preimage will flip each bit of the output with $\frac{1}{2}$ probability. “[Avalanche effect](#)”

Application 1: Storing Passwords

Instead of storing a user's password in plaintext, store `hash(password)`

- When user enters a password, compute its hash and compare with the entry in the password file
- Why is hashing better than encryption?
- System does not store actual passwords and can't go from hash to password!

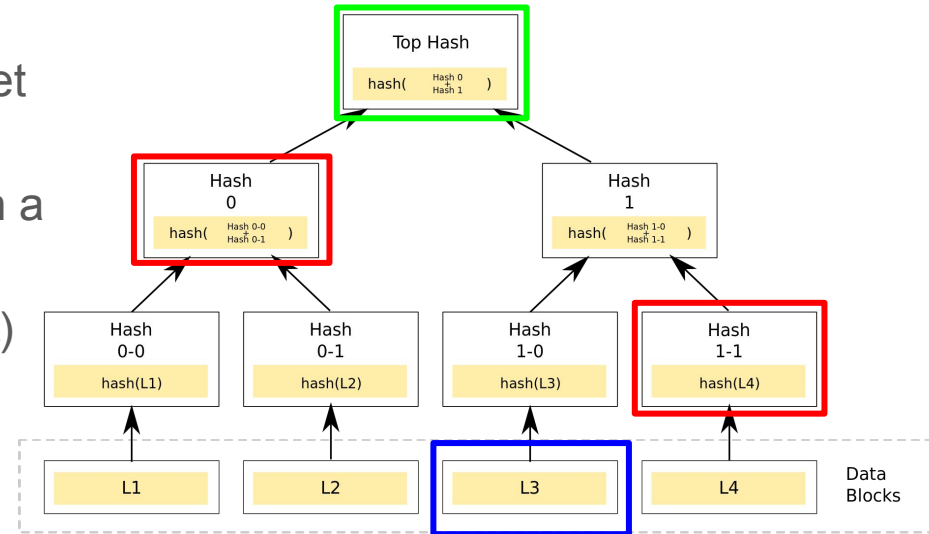
Does hashing protect weak, easily guessable passwords? No. Statistical attacks, lookup-table, or [rainbow table](#) attacks.

- Solution: Hash **Salt**
 - When a user creates a pw, generate a random value (a salt) store it.
 - Concatenate the salt and password, store `hash(random salt + password)`

Application 2: Merkle Trees

Used for checking if an item is an ordered set or to check overall set equivalence.

- Create a tree such that each element in a set L is a leaf node
- Each leaf L_i has a parent hash: $h_i = h(L_i)$
- Each subsequent parent is:
 $h(\text{concatenation of its children})$



https://en.wikipedia.org/wiki/Merkle_tree

To prove an element is in the set:

- Provide $\log_2(|\text{inner nodes}|)$ values from the inner tree
- Check if the roots match

Application 2: Merkle Trees

Replica synchronization in distributed key-value stores
(Dynamo, Apache Cassandra, Riak)

- Data is replicated on N different “replicas” for **when** servers fail
- Merkle trees used to detect inconsistencies between replicas and minimize the amount of transferred data

Each replica maintains a Merkle tree where each key is a leaf node

- If hash values of the roots of different replicas are identical - no synchronization required
- If not, exchange hashes of children until the leaves. The differences let you pinpoint what data is out of sync

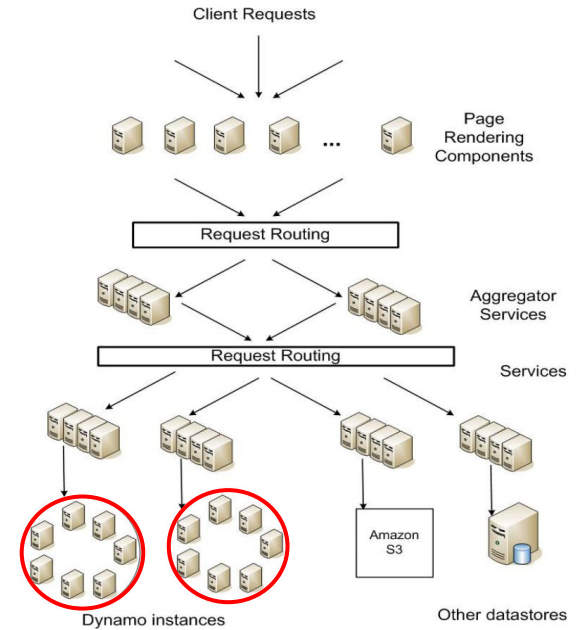


Figure 1: Service-oriented architecture of Amazon's platform

DeCandia, Giuseppe, et al. "Dynamo: amazon's highly available key-value store." ACM SIGOPS operating systems review 41.6 (2007)

Public-key (Asymmetric) Cryptography

A tale of two keys - data is encrypted with one key, but can only be decrypted to plaintext by the other key

Given: Everyone knows Alice's public key. Only Alice knows her private key

Goals:

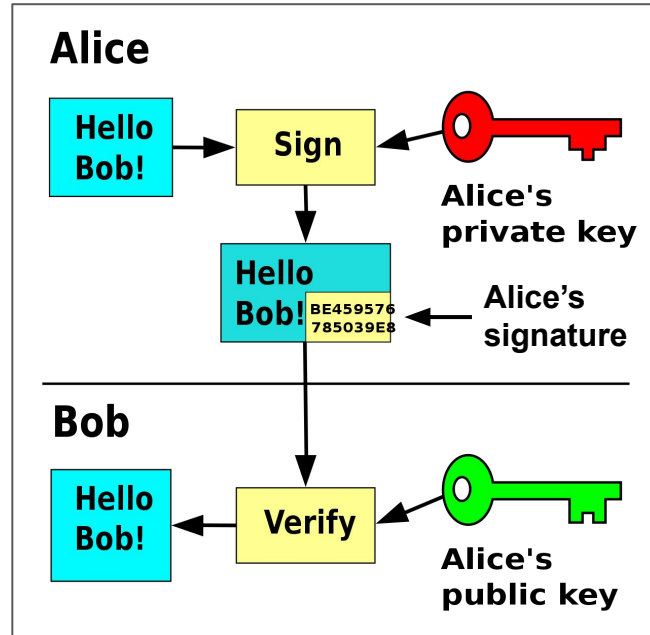
1. Bob wants to send a message that only Alice can read
 - a. RSA (secure data transmission), TLS (the internet protocol for secure communication)
2. Alice wants to send a message that only Alice could have written
 - a. Digital Signatures like RSA, DSA, ECDSA, and Schnorr signature

Public-key (Asymmetric) Cryptography

Key generation

- Computationally easy to generate (public key PK, private key SK)

Digital Signatures



Public-key Crypto for Digital Signatures

Group - Set of elements G and a group operator \circ that satisfy the following axioms

1. **Closure:** $a \circ b = c$, where c is a member of G
2. **Associativity:** $a \circ (b \circ c) = (a \circ b) \circ c$
3. **Identity Element:** there exists an element e such that for all a in G ,
 $a \circ e = a$ and $e \circ a = a$
4. **Inverse:** For all a in G , there exists the inverse a^{-1} such that
 $a \circ a^{-1} = e$ and $a^{-1} \circ a = e$
5. *Abelian* groups are also **commutative:** For all a, b in G :
 $a \circ b = b \circ a$
6. *Cyclic* groups have a *generator*
With a generator and the operator, we can generate all elements in the group

Public-key Crypto for Digital Signatures

Schnorr group: Operator is multiplication modulo p

- p : pick a large prime number
- $p = qr + 1$, where q is prime
 - $p-1$ must be composite so it can be factored into qr where q is prime
- h : Pick h such that $h^r \not\equiv 1 \pmod{p}$, where $1 < h < p$
- g : $h^r \pmod{p}$, which is our generator of order q

Simple Example of a Schnorr Group

- Let $p = 13$, $q = 3$, $r = 4$
- Choose $h = 2$
- $g = 2^4 \pmod{13} = 3$
- Use the generator: $3^1 \pmod{13}$, $3^2 \pmod{13}$, $3^3 \pmod{13}$, ...
- $G = \{3, 9, 1\}$

Digital Signatures: Schnorr Signature

Schnorr Signature

- Uses a key pair consisting of a public and private key
- Private key used to generate a digital signature for a message
- The signature can be verified by using the signer's corresponding public key
- Common use case: ensure a message came from who you think sent it

Operation

1. Key Generation (choosing parameters and creating the PK/SK pair)
2. Key Distribution (the signer publishes their public key)
3. Signing
4. Verifying

Schnorr Signature

Parameters

- Choose a group G of order q , with generator g (typically a Schnorr group)
- Choose a cryptographic hash function (SHA-256)
- M : our message

Key Generation

- Private key (SK): Any integer $0 < SK < q$
- Public (verification) key (PK): $PK = g^{SK}$
 - where exponentiation means repeatedly apply the group operator

Schnorr Signature

Signing msg M :

$k = \text{random nonce } 0 < k < q$

$$r = g^k$$

$$e = H(r || M)$$

$$s = k - \text{sk} \cdot e$$

Send (M, s, e)

Verifying msg M given (M, s, e)
 (PK, H, g)

$$r_v = g^s \cdot PK^e$$

$$e_v = H(r_v || M)$$

M is verified iff $e_v = e$

Proof of correctness

$$r_v = g^s \cdot PK^e = \underbrace{g^{k - \text{sk} \cdot e}}_{\text{plug in } s} \underbrace{g^{\text{sk} \cdot e}}_{\text{by } PK = g^{\text{sk}}} = \underbrace{g^k}_{\text{exp rules}} = r$$

$$\text{so then } e_v = H(\underline{r_v} || M) = H(\underline{r} || M) = e$$

Schnorr Signature

Why is this secure?

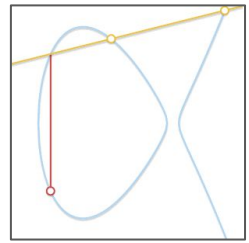
Discrete Logarithm Problem

- A discrete logarithm $\log_b a$ is an integer k such that $b^k = a$

In Schnorr we have

- $PK = g^{SK}$
- Solving for SK would require computing $SK = \log_g PK$
- We don't know how to compute this efficiently over the carefully chosen groups using large enough prime numbers

Elliptic Curve Digital Signature Algo (ECDSA)



Similar to Schnorr signatures, but based on the more complicated DSA (created to get around Schnorr's patent) and elliptic curves for its group

- Elliptic Curve Cryptography: a gentle introduction [Part 1](#) [Part 2](#)
- Security is based on the difficulty of the elliptic curve discrete logarithm
 - No known efficient algorithms...
 - Shor's Algorithm for Quantum computers
 - A powerful enough quantum computer will let us solve the discrete logarithm problem in polynomial time
 - Quantum Resource Estimates for Computing Elliptic Curve Discrete Logarithms ([Microsoft Research 2018](#))
 - This could happen in our lifetimes! Something to keep in mind when considering risks of crypto
 - Great related [article from Imperial College London](#) - "Committing to quantum resistance: a slow defence for Bitcoin against a fast quantum computing attack"

Consensus

We often trade tangible items with the rest of the world using cash or gold.

- Possession is equivalent to ownership

Some tangible items require a bit more process

- houses, cars, shares of company

We have many artifacts and mechanisms to regulate the trading process

- registries, titles, certificates, arbiters, exchanges, judges, authorities

All of this in place is to answer *consensus!*

- Who owns this?
- How much money is this account?
- Is the transaction authorized?

Consensus

It's critical that money is owned by one person at a time.
How do we manage this in real life?

Move_Money(value, source, destination):

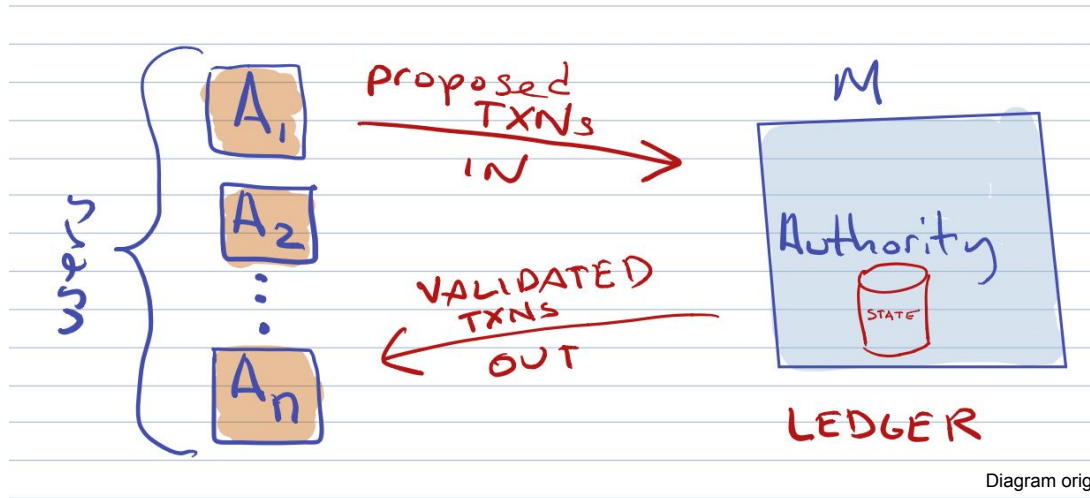
1. Check source exists
2. Check authorization
3. Check destination exists
4. Check sufficient funds
5. Remove funds from source
6. Insert funds into destination

Problems?

- What if funds are removed by a concurrent call to Move_Money before 5, but after 4
- What if 5 completes, but 6 fails?

Consensus

It's easiest to correctly offer consensus services using a single authorized central authority



Consensus

Downsides of a single authority?

- Single point of failure
- Single point of attack
- Single point of trust

We will see how Bitcoin addresses these downsides.

Bitcoin is a **secure**, **distributed** system that manages **consensus** about the state of accounts and the authorized transactions among them.

Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto (2008)

Abstract. A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. *Digital signatures* provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent *double-spending*. We propose a solution to the double-spending problem using a **peer-to-peer network**. The network timestamps transactions by *hashing* them into an ongoing *chain* of *hash-based proof-of-work*, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.

Important Takeaways

Cryptographic Hash Functions

- $h(\text{arbitrary bits})$ gives you, “with probability 1”, a unique output
- Consider the use case of Merkle trees that let you prove a item is in a set and prove set equivalence

Public-key Cryptography

- Digital Signatures: Your message, your signature, and your PK is enough for a third party to validate that you sent that message.

Consensus

- Nakamoto’s blockchain is a secure distributed system that manages consensus amongst accounts and the transactions between them

Bitcoin: A Secure, Distributed System

Part 2

Review

Cryptographic Hash Functions

- h (arbitrary amount of bits) gives you “with probability 1” a unique, fixed-length output

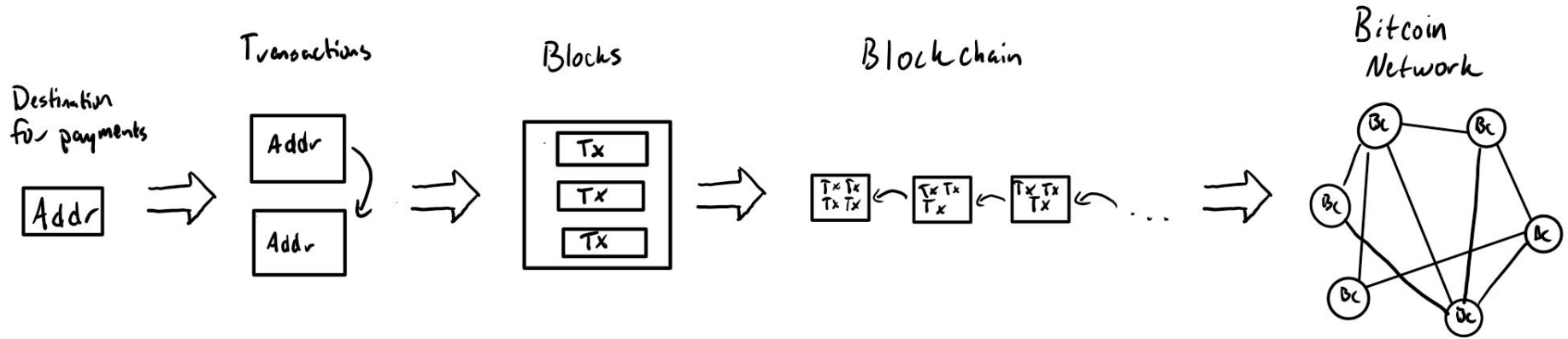
Public-key Crypto

- Digital Signatures: Your message, your signature, and your PK is enough for a third party to validate that you sent that message.

Consensus

- Nakamoto’s blockchain is a secure distributed system that manages consensus amongst accounts and the transactions between them

Big Picture of how Bitcoin Works



Big Picture of how Bitcoin Works (in words)

1. User creates an address (pub key), which is a destination for payments
2. To spend coin, users create transactions (txs) that are broadcast to the network
 - a. Tx references a previous tx that authorizes you to spend that coin
 - b. Scripting system validates the coin can be redeemed with your pub key and signature
3. Miners validate transactions and place them into blocks which form the blockchain
4. To get everyone to agree that any miner's block is THE block in the chain, they are the first to compute the answer to a computationally expensive mathematical problem
 - a. Their answer is propagated through the Bitcoin network, creating a single distributed ledger of authorized txs

Bitcoin Addresses

The hash of a public ECDSA key

- Represents a possible destination for Bitcoin payment

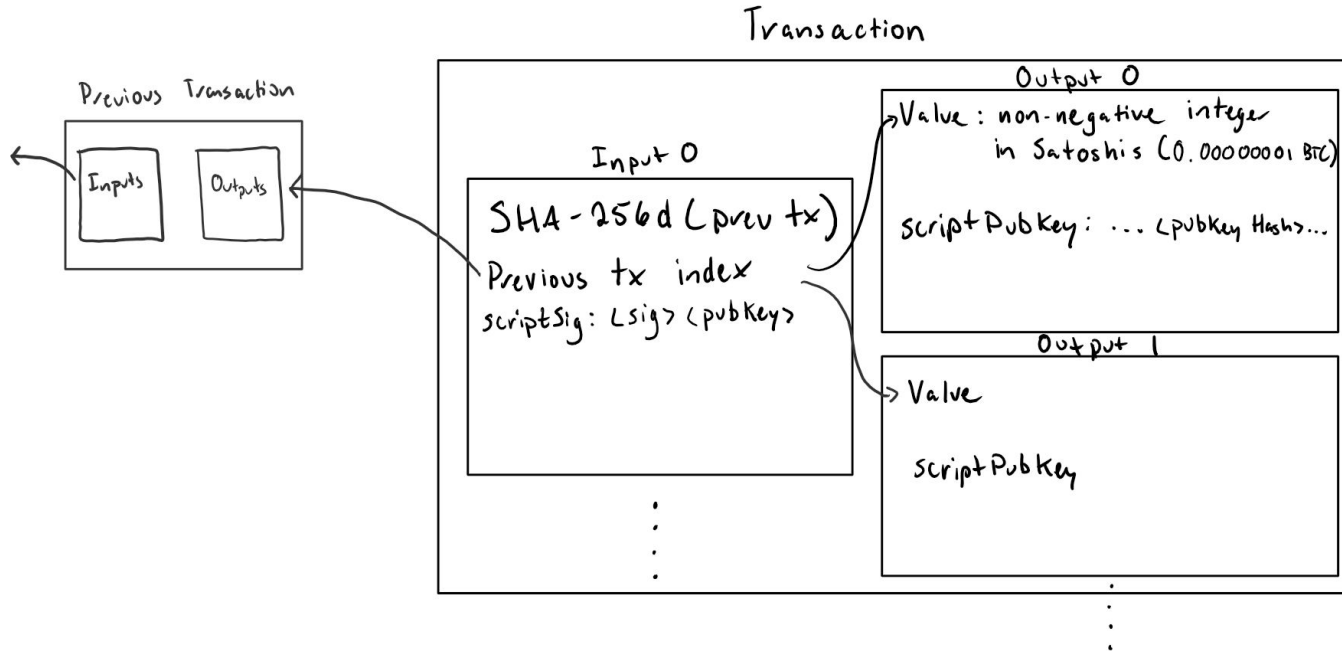
How a Bitcoin address is created:

1. Private *ECDSA* key: `18e14a7b6a307f426a94f8114701e7c8e774e7f9a47e2c2035db29a206321725`
2. Corresponding public key: `0250863ad64a87ae8a2fe83c1af1a8403cb53f53e486d8511dad8a04887e5b2352`
3. *SHA-256* of step 2: `0b7c28c9b7290c98d7438e70b3d3f7c848fbd7d1dc194ff83f4f7cc9b1378e98`
4. *RIPEMD-160* of step 3: `f54a5851e9372b87810a8e60cdd2e7cfd80b6e31`
5. Add version byte (0x00 for main network): `00f54a5851e9372b87810a8e60cdd2e7cfd80b6e31`
6. *Base58Check* encoding (several steps omitted): `1PMycacnJaSqwWJqjXBERnLsZ7RkXUAs`

Why hash here? Ask Satoshi, probably done to get shorter addresses.

Transactions

Transfer of *Bitcoin value* that is broadcast to the *Bitcoin network* and then collected into a *block*.



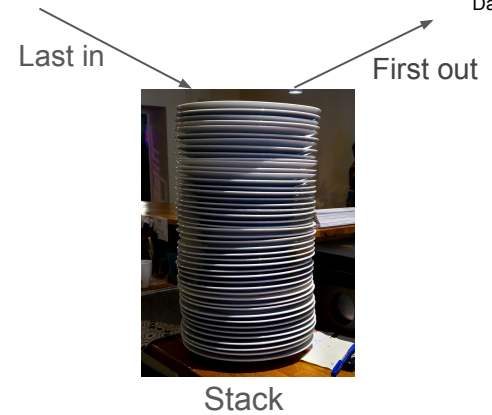
Scripting System

Bitcoin uses a scripting system for transactions

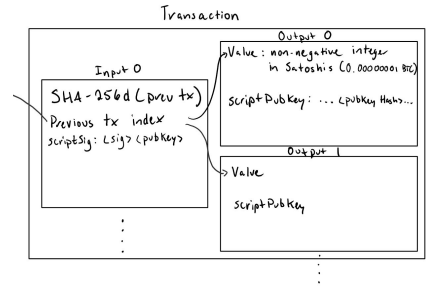
- Uses its own simple programming language
- Stack-based
- Not Turing-complete (cannot compute any computable function) and no loops

The standard form of a transaction is pay-to-pubkey-hash (P2PKH)

- `scriptSig` and `scriptPubKey`



Pay-to-pubkey-hash (P2PKH)



scriptPubKey says something like:

- “This coin can be redeemed by only a signature from the owner of address x ”

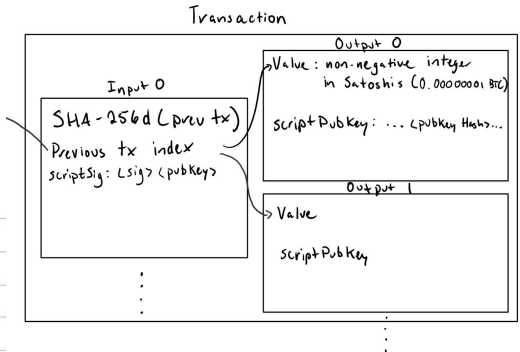
But since x is a hash, it can't be used to validate the signature, so in reality it says:

- “This coin can be redeemed by a public key that hashes to x , along with a signature from the owner of that key”

A signature of what?

- A signed copy of the transaction itself
- Transactions are redemptions of a previous transaction by signing with the spender's private key

Pay-to-pubkey-hash



this txn was written by someone who gave control to Alice.

This txn can be redeemed by only a txn signed by public key K_{A+} such that $H(K_{A+}) = \langle \text{Addr} \rangle$



output scriptpubkey contains the hash of the public key allowed to spend next.

- $\langle \text{Addr} \rangle \leftarrow H(K_{A+})$
- some other things

txn₂ META INPUT OUTPUT

This tx can be redeemed by only a tx signed by public key K_{B+} such that $H(K_{B+}) = \langle \text{Addr} \rangle$

input says:

- $h(\text{txn}_1)$ ← previous txn's ID
- slot θ ← slot of previous txn K_{A+} (PK)
- here's my public key
- here's a signature of this transaction txn_2 $\langle \text{sig} \rangle$ to show I know my private key.

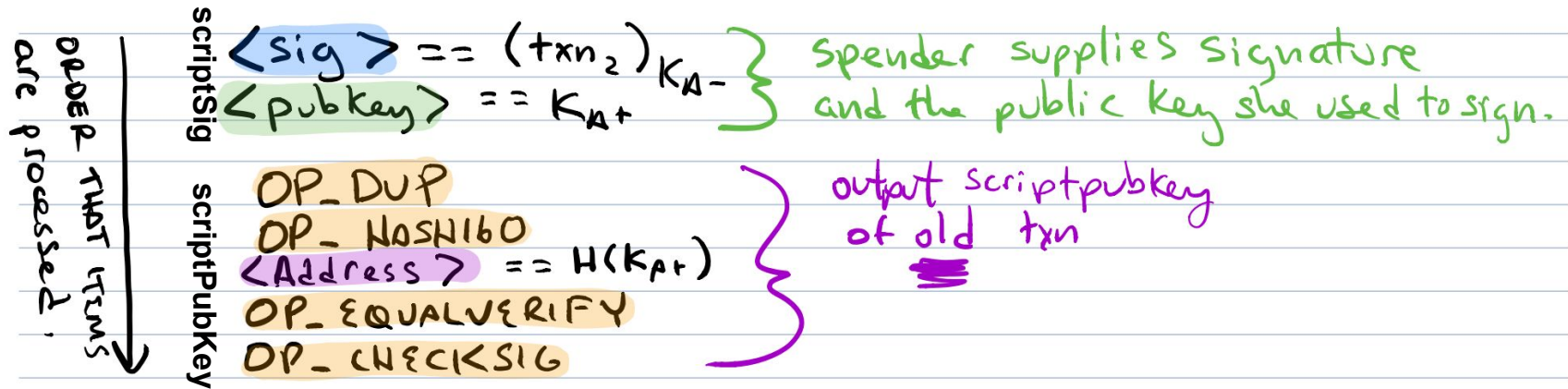
written by Alice

← she's giving control to Bob by listing $h(K_{B+})$ in the output

Pay-to-pubkey-hash

To validate a new transaction, we attach it to a previous txn.

- We concatenate the previous txn's output with the new txn's input

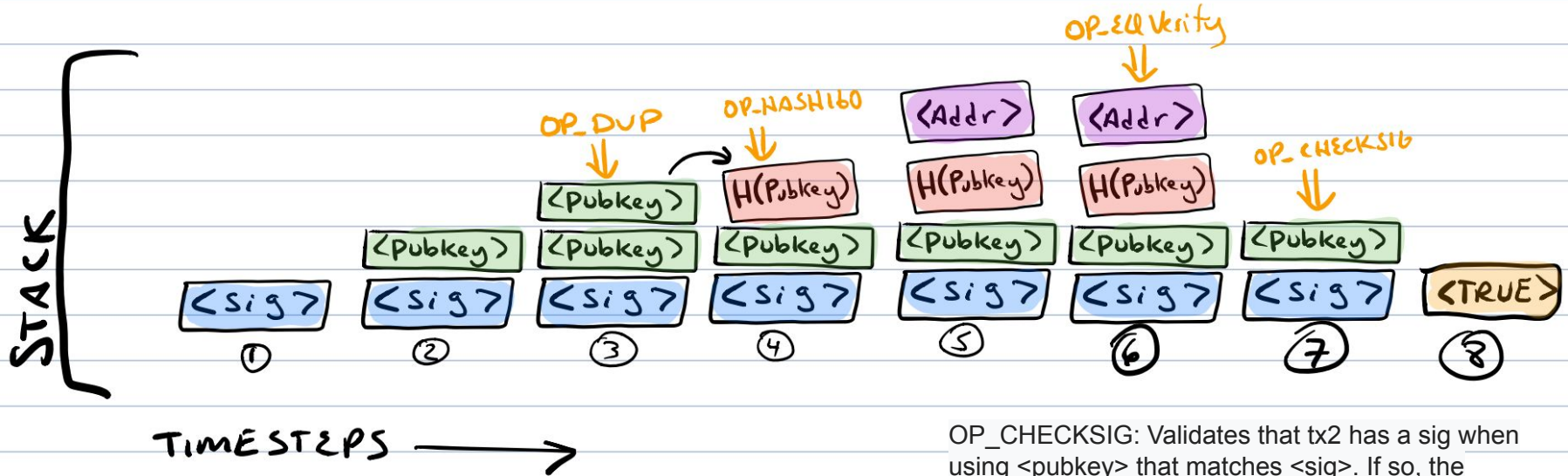


Pay-to-pubkey-hash

Executing the scripts to validate a transaction


$\langle \text{sig} \rangle == (tx_2)_{K_A}$
 $\langle \text{pubkey} \rangle == K_{A+}$

OP_DUP
OP_HASH160
 $\langle \text{Address} \rangle == H(K_{A+})$
OP_EQUALVERIFY
OP_CHECKSIG



OP_CHECKSIG: Validates that tx2 has a sig when using $\langle \text{pubkey} \rangle$ that matches $\langle \text{sig} \rangle$. If so, the $\langle \text{pubkey} \rangle$ specified in tx1 was used to sign tx2.

Example Transaction [\(Visualized on btc.com\)](#)

| Home / Block - 270953 / Transaction 1c12443203a48f42cdf7b1acee5b4b1c1fedc144cb909a3bf5edbffafb0cd204 | | | |
|--|-----------------------------|--|--|
| Summary | | | |
| Height | 270953 | Input | 194,993.50000004 BTC |
| Confirmations | 393892 | Output | 194,993.50000004 BTC |
| Timestamp | 2013-11-22 12:05:19 | Sigops | 8 |
| Size (rawtx) | 7,027 Bytes | Fees | 0.00000000 BTC |
| Virtual Size | 7,027 Bytes | Fees Rate (BTC / kVB) | 0.00000000 BTC |
| Weight | 28,108 | Other Explorers |  BLOCKCHAIR |
| Input (47) | 194,993.50000004 BTC | Output (2) | 194,993.50000004 BTC |
| ◀ 1AKpfwYmBUYfRAKfVgMwLeJxJXFHdu1DCT | 90.00000000 | 12sEnWECeRSmTeD... (Bitstamp Aud...) | 194,993.00000000 ▶ |
| ◀ 16os9VWytjHNL9HTuVLASDZmaVF9pMQW1P | 67.00000000 | 1NEoiC44bcEptPHmfZ4emG88hJBzj6EDse | 0.50000004 ▶ |
| ◀ 16os9VWytjHNL9HTuVLASDZmaVF9pMQW1P | 15.00000000 | | |
| Input Scripts | | | |
| 473044022071d0e6b7799f0dac3c5521b36af1b8567e7acd03fb2b09d257e1ef13298f563302202494cf9ca1c08ad563c624f72699899157240dbbffaeb4904069d628893818012102904c965c66618965aaf836396a7735d5563ad6c41f4af878a7e779108316c413 | | | |
| 48304502206a66daf45400df480049636d4aefc223c0609280508f6f6ed8ac76f442bf10e0022100c95157f91226d96d0117d066a87c7e75df59bdf947b7c5d569319ede2862a63693303238990146235c84351b8146846744574631220255c63188511854f4533 | | | |
| Output Scripts | | | |
| P2PKH OP_DUP OP_HASH160 147817677f02b8902230608f186d239d61785563 OP_EQUALVERIFY OP_CHECKSIG | | | |
| P2PKH OP_DUP OP_HASH160 e8f70a68a1afe290ab40edea3631d629a4be566b OP_EQUALVERIFY OP_CHECKSIG | | | |

Blocks

We have a means of making sure spending coin is possible only by the owner of the coin, but how do we prevent them from spending it more than once (doublespending) without a central authority?

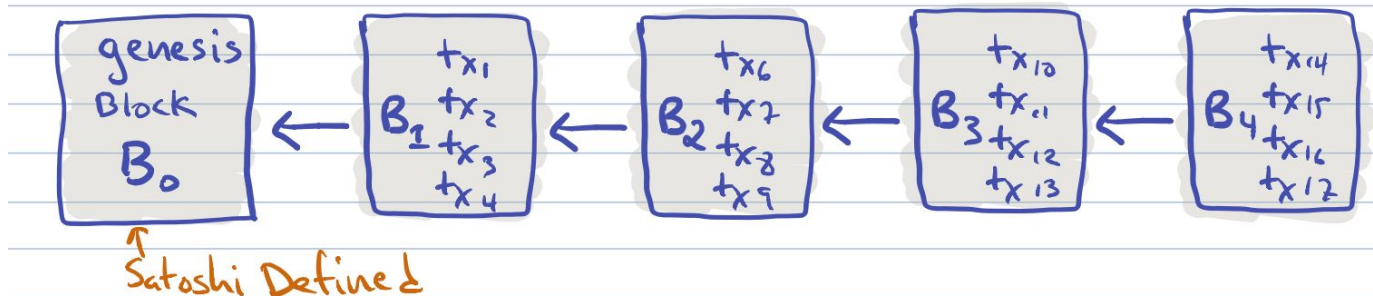
First let's look at *blocks*.

- Transaction data is permanently recorded in files called *blocks*.

Miners and Proof-of-Work

Blocks are organized into a linear sequence over time (the block chain).

- New transactions are constantly being processed by *miners* into new blocks which are added to the end of the chain.
- They are called miners because of the process used to elect which miner gets to add a new block.



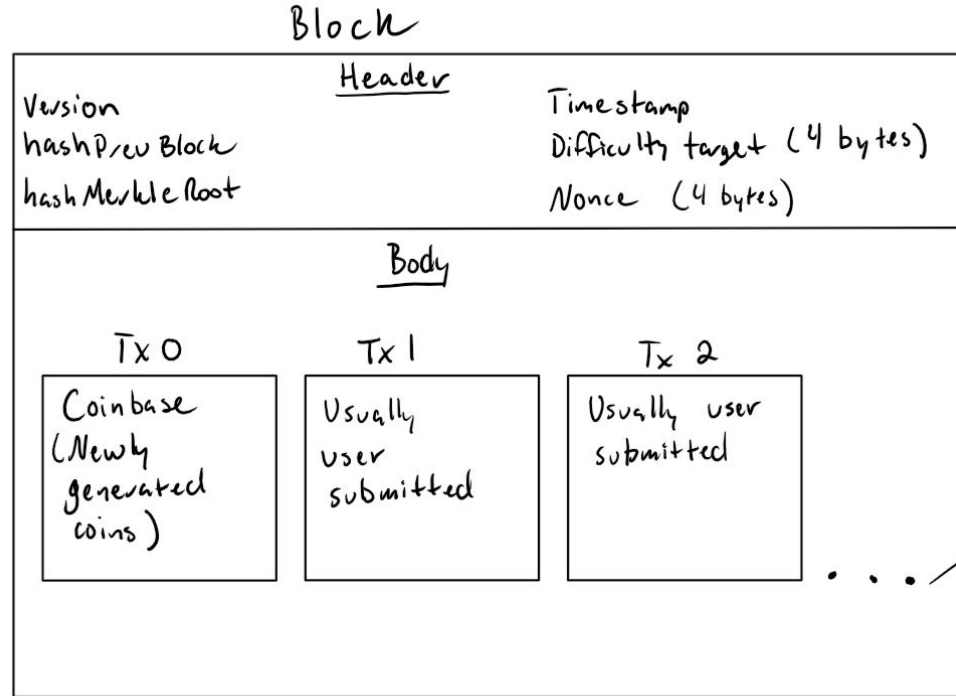
Blocks

hashMerkleRoot is root hash of the Merkle tree built from each tx as a leaf

Each new block is allowed a coinbase tx, which consists of **newly generated coins**.

Currently 6.25 BTC are generated per block.

“The halvening” happens about every four years.



Miners and Proof-of-Work (POW)

Hashcash POW algorithm; “hash until the hash is less than some target”

1. Each miner constructs a candidate block with the header and a set of transactions
2. Let $D = \text{difficulty}$ and let $t = 2^{256} / D$
 - a. The block represents valid **proof-of-work** (POW) if the mining criteria is met: $H(\text{header}) \leq t$
 - b. If the mining criteria is not met, the miner can change the nonce, time, or merkle root of txs
3. If the mining criteria is met, the block is announced to all miners and users
 - a. The transactions that created the hashMerkleRoot are inside the block
 - b. The other miners confirm that:
 - i. $H(\text{header}) \leq t$
 - ii. Each transaction is authorized
 - iii. Each transaction is valid given the prior block
 - iv. All header values are valid
 - c. Then, the miners will add to the blockchain after its prior

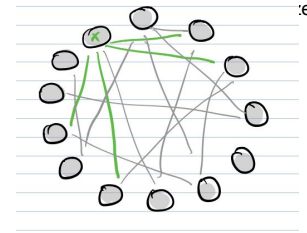
Miners and Proof-of-Work

By adding the new block to their copy of the chain, the miners remove txs from the *mempool*, the pool of all unconfirmed txs

They then start mining a new candidate block, with a new set of txs from the pool and a new prior block in the header

Note: POW is probabilistic.

- Example: No guarantee that more than one miner performed more than a single hash of one header.
- But for a sequence of blocks, the number of hashes needed to confirm a block is close to the expectation
- Difficulty is adjusted such that it takes on average 10 minutes to mine a block



Bitcoin Network

An Internet-based peer-to-peer (P2P) network connects miners and users

- BitTorrent clients, Gnutella, Microsoft's "Delivery Optimization" for Windows updates

Users join as a miner, full node, or lite client

- Full nodes keep a full, up-to-date copy of the blockchain (ex. a merchant)
- Lite clients want to verify specific payments (using Simple Payment Verification)

At the start, new peers contact a bootstrap node

- Hardcoded nodes in the software
- The bootstrap tells peers of other peers already joined
- Peers ensure they have 8 outgoing connections and accept up to 117 incoming connections

Bitcoin Network

To send a transaction, send an *inv* message containing it to all your peers.

- If peers consider the transaction valid after receiving it, they will also broadcast the transaction to all of their peers with an *inv*, and so on
- Same idea for miners sending out new blocks

What if two miners simultaneously announce a new block, eg. a fork?

- Each miner will mine on the block they received first
- Eventually, one subset will produce more blocks, breaking the tie



High-Level Summary

1. User creates an address (pub key), which is a destination for payments
2. To spend coin, create a tx that is broadcast to the Bitcoin network
 - a. Tx references a previous transaction that authorizes you to spend that coin
 - b. Scripting system validates the coin can be redeemed with your pub key and signature
3. Miners validate transactions and place them into blocks
 - a. To get everyone to agree that any miner's block is THE block in the chain, they are the first to compute the answer to the hashcash proof-of-work algorithm
4. Valid blocks are broadcast to the Bitcoin network. Other miners and full nodes add them to their local copy of the Bitcoin blockchain.
 - a. This is the idea of a distributed database, where thousands of identical copies exist throughout the world

So much more...

Scalability

- Visa handles about 65,000 transactions/second. Bitcoin? Max of about 4
- Other currencies and ideas are trying to address this limitation

Security Risks and Attacks

- 51% attack, selfish mining, eclipse attacks
- Exploits in software

Simplified Payment Verification

- Only download the parts of the chain you need to verify txs

How useful are cryptocurrencies and blockchain?

Blockchain, the amazing solution for almost nothing

- Story about how a student in a small Dutch town became famous for building a blockchain based app for tracking children's poverty aid - that didn't actually use blockchain
 - “Maybe this is blockchain's greatest merit: it's an awareness campaign, albeit an expensive one. ‘Back-office management’ isn't an item on the agenda in board meetings, but ‘blockchain’ and ‘innovation’ are”
- Opinion: Similar to hype behind Data Science and Machine Learning, except the only successful use case has been cryptocurrency

Environmental Impact

One mining machine (ASIC) can use 1300+ Watts

- Ran 24/7, exceeds how much monthly power an average household uses in 24 days

University of Cambridge: Bitcoin accounts for **0.51%** of all energy consumed in the world^[source]

- BBC: [“Bitcoin consumes more electricity than Argentina”](#)

One Bitcoin transaction equates to ~887 kWh or driving a Tesla Model 3 for over 3,800 miles

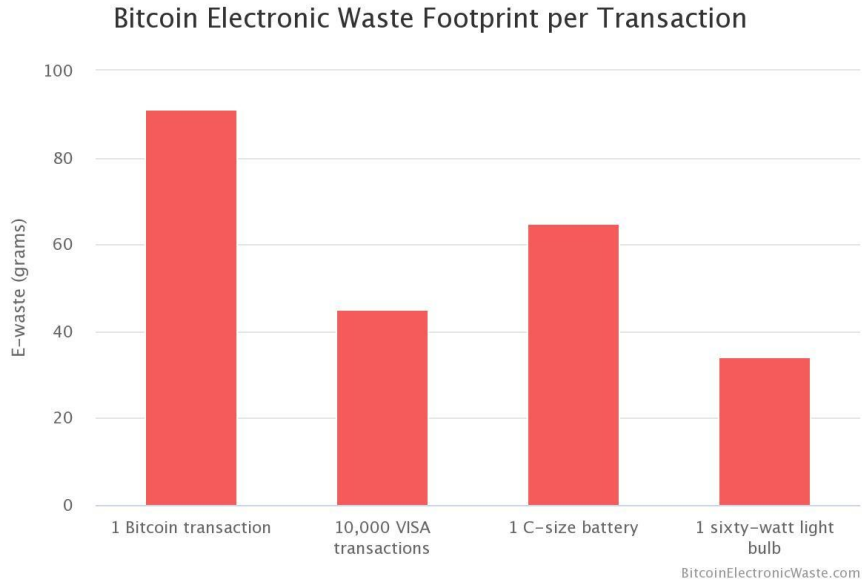
- Assumes 100TWh Bitcoin yearly energy usage; 112,735,854 yearly transactions; and a Model 3 goes 322 miles on a 75kWh battery pack

Ironically, Tesla bought \$1.5B in Bitcoin...

- Reuters: [Elon Musk wants clean power. But Tesla's carrying bitcoin's dirty baggage](#)

Environmental Impact

Electronic waste



Source: [Digiconomist](#)

[Bloomberg: Global chip shortage](#)

- Ford plant shutdowns
- Price of laptops
- Can't buy a new GPU or PS5 at MSRP
- NVIDIA [announces](#) dedicated mining GPUs, begin throttling other SKUs



Image Credit: wccftch - [NVIDIA GeForce RTX 3000 Gaming Laptops Powered Massive Cryptocurrency Mining Farm...](#)



Dogecoin

Based off Litecoin, which is a (git) fork of Bitcoin omg developers

- Uses script instead of SHA-256 as its hashcash POW hash function
- Blocks are mined every 1 minute
 - **Much blocks!**
- 10,000 coins are minted per block (since ~2015)
 - This is fixed, i.e. no halvings
 - **So many puppies!**
- Over 128 billion currently in circulation [\[source\]](#)
 - **To the moon!** 🚀🚀🚀
- Such Schnorr!



Ethereum (ETH)

Differences from Bitcoin

- Blocks are mined about every 15 seconds vs 10 minutes
- Coin rewards for *ommers* (orphaned blocks)
- Moving towards *Proof-of-Stake*, i.e. users with a stake (32 ETH) are chosen at random and asked to validate blocks. If you attest to malicious blocks, you lose your stake.
- Includes a Turing complete language which allows for complex *smart contracts* rather than just transactions ([See the rise and fall of The DAO](#))
- Transaction fees differ by computational complexity, bandwidth use, and storage needs (in a system known as gas)
- **Many other differences**

Exchanges

Crypto exchanges are **market makers**

- Very similar to NASDAQ
- Maintain inventory of an asset (like BTC)
- Both sells and buys that asset from its clients
- Profits off of the **bid-ask** spread
 - Bid: highest price a buyer is willing to pay
 - Ask: lowest price a seller is willing to accept
- Allow for withdrawals into the actual asset
 - Important/Confusing point: You don't own the actual BTC on the blockchain until you withdraw it from the exchange
 - Everyone wanting to withdraw their BTC at once probably would be like a run on the bank

Exchanges

Let's look at some live EUR to BTC exchanges!

- [Coinbase Pro \(formerly GDAX\)](#)
- [Binance](#)

Data is more accessible?

- Market Caps, Volume
 - <https://coinmarketcap.com/>
- Historical data by minute for many exchanges
 - <http://www.cryptodatadownload.com/data/>
- Binance gives you all trades between two timestamps
 - <https://github.com/binance/binance-spot-api-docs/blob/master/rest-api.md#compressedaggregate-trades-list>

Avenues for Further Exploration and Research

Scalability, Security, and Risks

- Challenges that arise in distributed computing systems
- (Proposed) solutions to the scalability problem
- Think like an adversary - how can one attack these systems?

Software and Systems

- Learn about implementations (most are on Github), build your own blockchain, [Ethereum Decentralized Apps](#)

Almost anything you want to do with equity or forex applies to crypto.

- Portfolio management/allocation, price prediction, trading systems, etc)
- Benefits from more accessible data, open 24/7, less regulation

Attributions

Many portions of this presentation were inspired by and used materials from professors and lecturers at the University of Massachusetts Amherst.

Brian Neil Levine

- UMass CS461/661 - Secure Distributed Systems

Andrew Stone

- UMass CS461/661 - Secure Distributed Systems
- Creator of Bitcoin Unlimited

Amir Houmansadr

- UMass CS460 - Computer and Network Security