Course Outline

- ❑ Introduction
- ❑ Money and token economy
- ❑ Cryptography
- ❑ Blockchain data structure / Filecoin
- ❑ Mining and PoW
- ❑ Consensus algorithms
- ❑ Bitcoin as a platform
- ❑ Ethereum and smart contracts
- ❑ Distributed applications & enterprise DLT
- ❑ Security
- ❑ Scalability
- ❑ Privacy

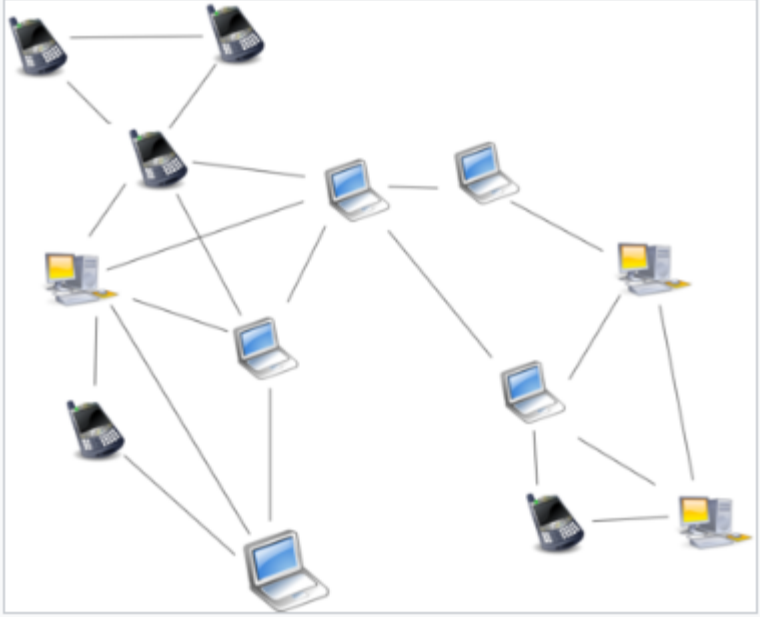
Peer-to-Peer Network

P2P v.s. C/S Model

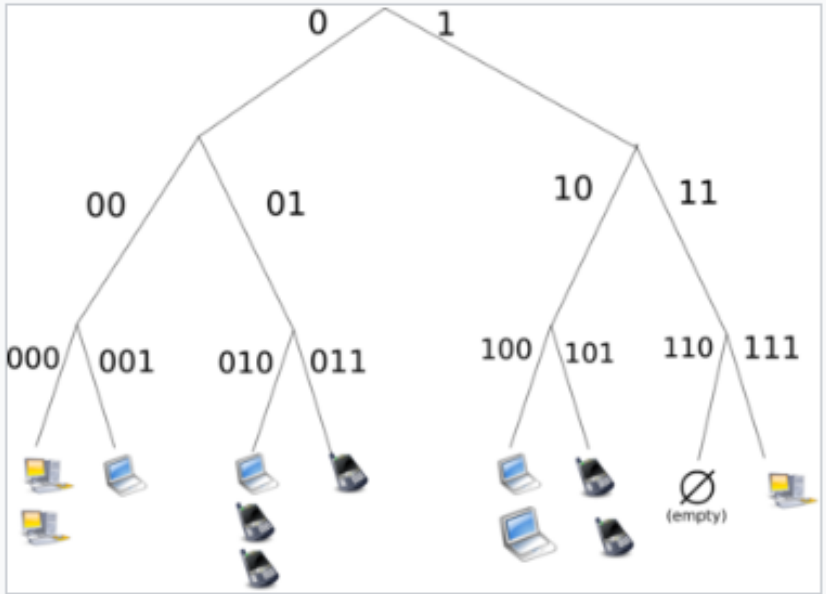
A network based on the **client-server model**, where individual *clients* request services and resources from centralized servers

A **peer-to-peer (P2P) network** in which interconnected nodes ("peers") share resources amongst each other without the use of a centralized administrative system

Unstructured v.s. Structured P2P Network



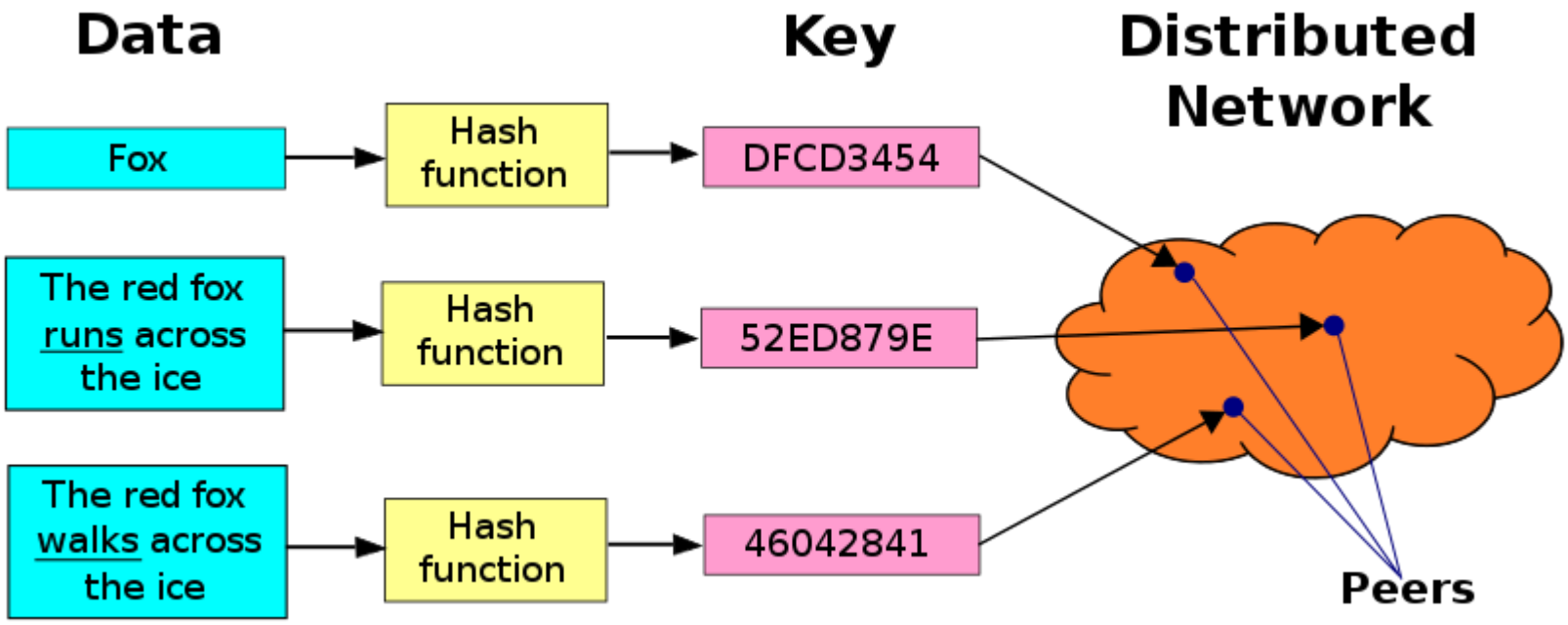
Overlay network diagram for an **unstructured P2P network**, illustrating the ad hoc nature of the connections between nodes



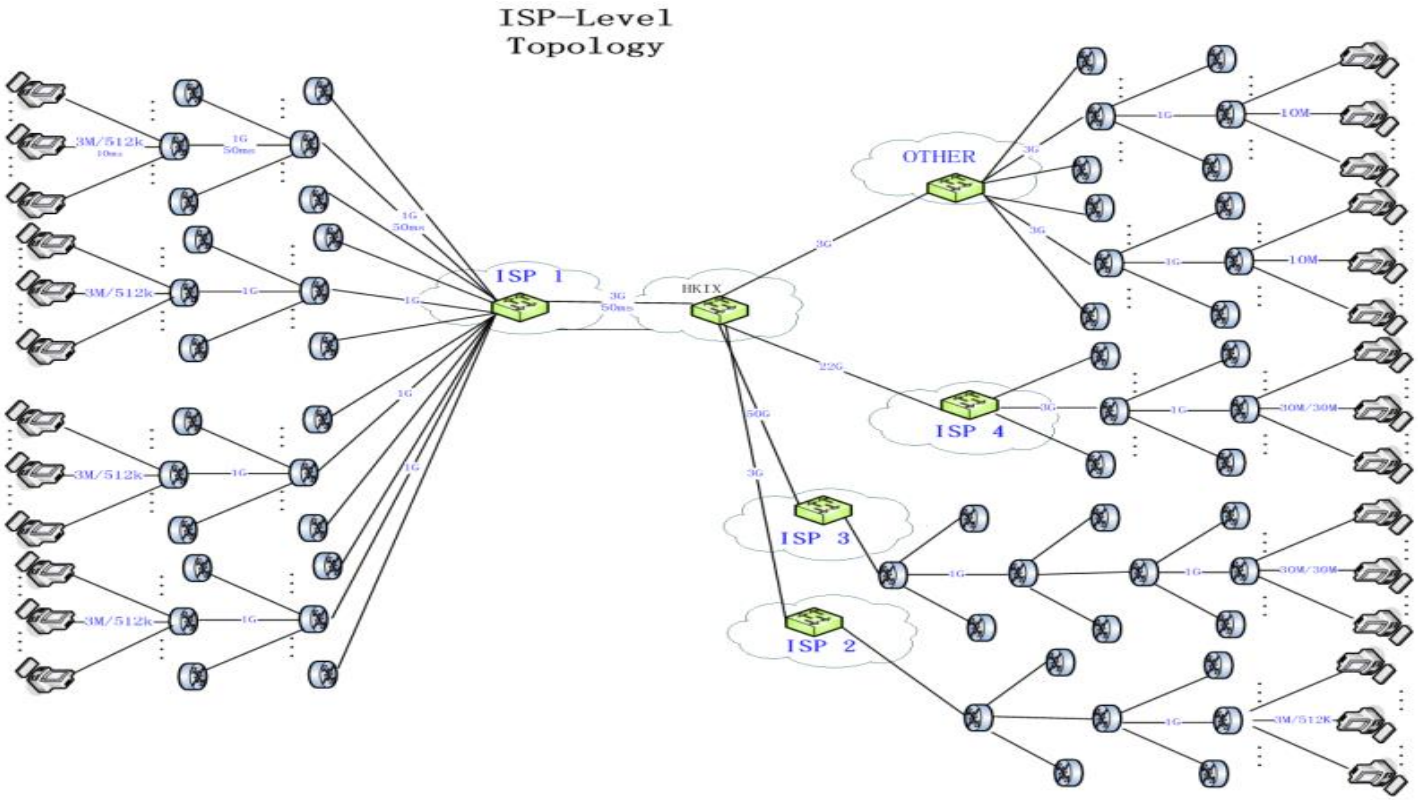
Overlay network diagram for a **structured P2P network**, using a distributed hash table (DHT) to identify and locate nodes/resources

Distributed Hash Table

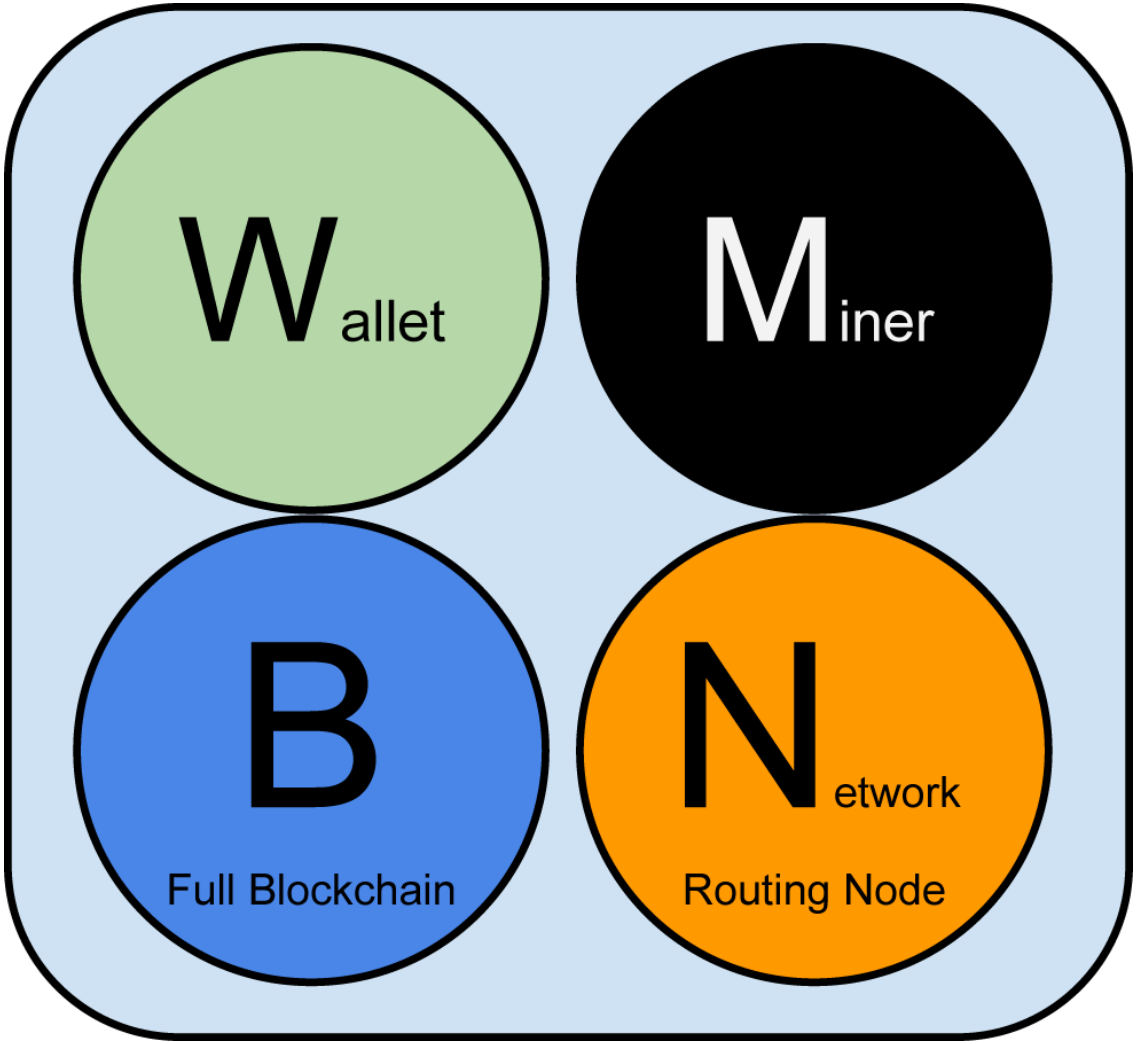
- Hashing is used to assign file to a particular peer.
- This enables peers to search for resources on the network using a hash table: that is, (*key*, *value*) pairs are stored in the DHT
- P2P resource discovery - cost of advertising/discovering resources and static and dynamic load imbalance



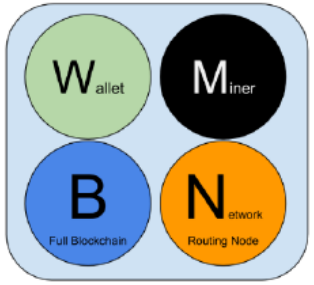
P2P is Overlay Network v.s. Physical Network



Bitcoin Network Nodes

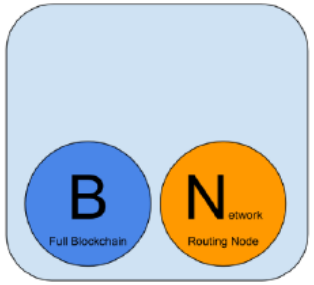


Bitcoin Network Nodes



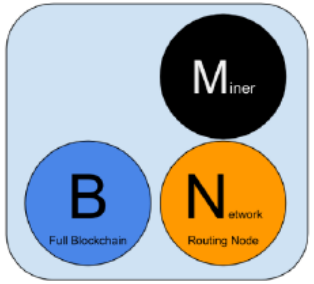
Reference Client (Bitcoin Core)

Contains a Wallet, Miner, full Blockchain database, and Network routing node on the bitcoin P2P network.



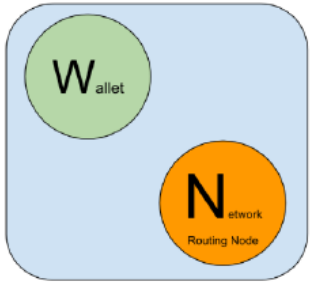
Full Block Chain Node

Contains a full Blockchain database, and Network routing node on the bitcoin P2P network.



Solo Miner

Contains a mining function with a full copy of the blockchain and a bitcoin P2P network routing node.



Lightweight (SPV) wallet

Contains a Wallet and a Network node on the bitcoin P2P protocol, without a blockchain.

- Main bitcoin network, running the bitcoin P2P protocol, consists of around 10,000 listening nodes running bitcoin reference client (Bitcoin Core)
- A small percentage of the nodes are mining nodes, and network edge routers, allowing various other services (exchanges, wallets, block explorers, merchant payment processing) to be built on top

Network Discovery

- Upon initialization, a node queries DNS servers (DNS seeds – maintained by Bitcoin community) for a list of IP addresses of full nodes
- The node performs up to 8 outgoing connection attempts (called entry nodes)
- Once one or more connections established, new node sends an *addr* message containing its own IP address to its neighbors. The neighbors will forward to neighbors, ensuring the new node is well known and connected
- A node must connect to a few different peers to establish diverse paths. Paths are not reliable—nodes come and go—and so the node must continue to discover new nodes as it loses old connections, and assist other nodes when they bootstrap
- Nodes will periodically send message to maintain the connection. If a node has not communicated for 90 minutes, it is assumed disconnected. A new node is sought

Full Nodes and SPV (Lightweight Nodes)

- When nodes connect, need sync up blockchain data and exchange transactions
- Full nodes maintain a full blockchain with all transactions. Full node can independently and authoritatively verify any transaction without reliance on any other node. Full node relies on network to receive updates about new blocks of transactions, which it then verifies and incorporates into its local copy of blockchain
- Simplified payment verification (SPV) or lightweight node doesn't store full blockchain. SPV nodes download only the block headers, not transactions, hence 1,000 times smaller than full blockchain
- SPV nodes cannot construct a full picture of all the UTXOs that are available for spending because they do not know about all the transactions on the network
- A full node verifies a transaction by checking the entire chain of blocks below it, whereas an SPV node checks how deep the block is buried by a handful of blocks above it (i.e., depth in the blockchain instead of the height)
- An SPV node can prove a transaction exists but cannot verify that a transaction doesn't exist because it doesn't have a record of all transactions - double-spending attack – SPV strikes a balance between resource and security

Dedicated Relay Networks

- **Miners increase their advantage by (or lose profit mining outdated blocks)**
  - Receive new blocks as fast as possible
  - Broadcasting the mined blocks as fast as possible to the majority of other miners
- **Private peering network or alternative relay network – optimized to circulate blocks across miners (global latency 100-300ms)**
- **Relay network performs partial object validation and does not follow the request management system of Bitcoin to ensure speed.**

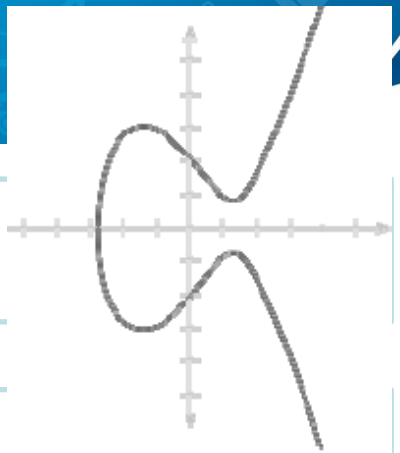
http://bitcoinfibre.org/public-network.html

Internal Reputation Management System

- **What if peer nodes behave badly? – prevent ill-formed blocks/transactions**
- **Check received blocks/transactions before forwarding**
  - First validate syntax and size (oversized objects discarded)
  - Then for transactions, verify signature, and input/output coins
  - For blocks, PoW in the block headers is verified wrt current network difficulty
- **A receiving node locally assigns a penalty to peers who broadcast ill-formed objects**
  - Once a node reached 100 penalty points, disconnected for 24 hours
  - Invalid alert messages get 10 penalty points
  - Invalid transaction signature immediately assign 100 points
  - Penalties also apply to ill-formed control messages such as *inv* (inventory) or *addr* commands
  - Locally assigned penalties are not transmitted to other peers

Blockchain Structure - Identity

Identity in Bitcoin



Send money between pseudonyms

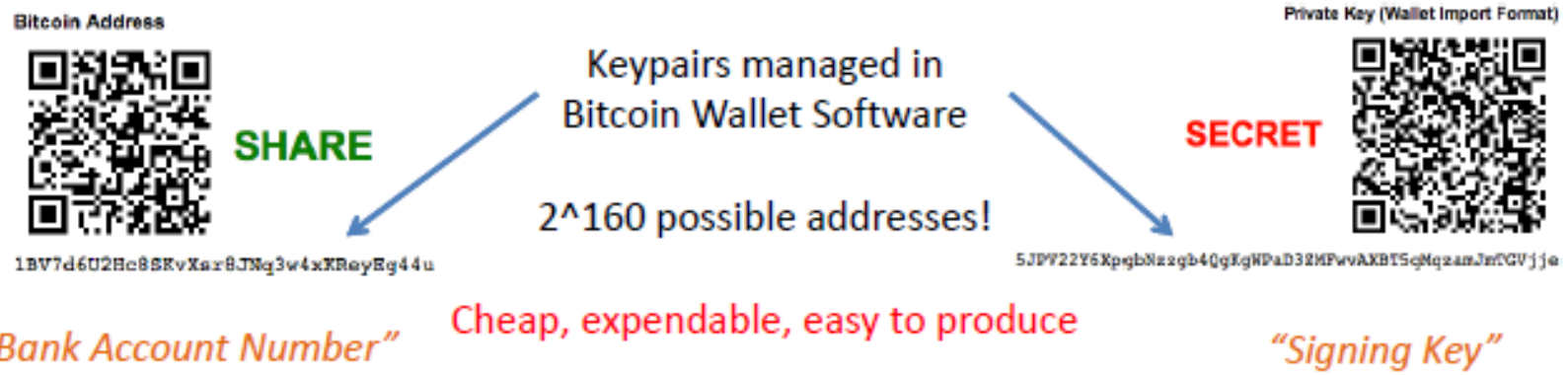
- pseudonym == address == hash of public key

Public Key/Private Key Pair

- sign a message using private key so signature is unforgeably tied to the public key
- Anyone can use public key to verify the message was created with the private key

Client wallet generates public/private key pairs using Elliptic Curve Cryptography (ECDSA)

- Public key is encoded into a 27-34 character address string
- Private key is used to spend coins by digitally signing transaction messages
- Example Address: 1FtQU9X78hdshngJiCBw9tbE2MYpx87eLT



Openning The Account

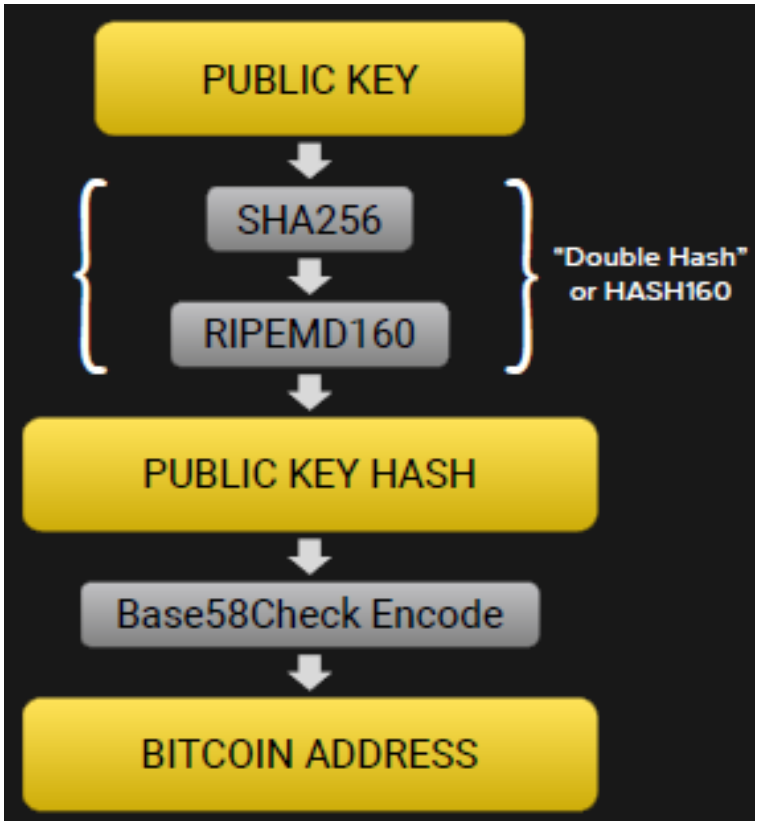
Users can generate arbitrarily many key pairs.

The probability of your generating a private key that is able to unlock another person's funds is nearly 0.

2^160 possible addresses
(1,461,501,637,330,902,918,203,684,832,716,283,019,655,932,542,976)
Grains of sand on earth: 2^63

This is to help put into perspective the concept of joining the Bitcoin network and opening accounts. You're never going to collide with anyone else.

From Public Key to Bitcoin Address



PUB KEY HASH = RIPEMD160 (SHA256 (K))

- where K = public key
- SHA-256 (Secure Hashing Algorithm)
- RIPEMD (RACE Integrity Primitives Evaluation Message Digest), produces 160-bit number

Bitcoin Addresses: Base58Check Encoded

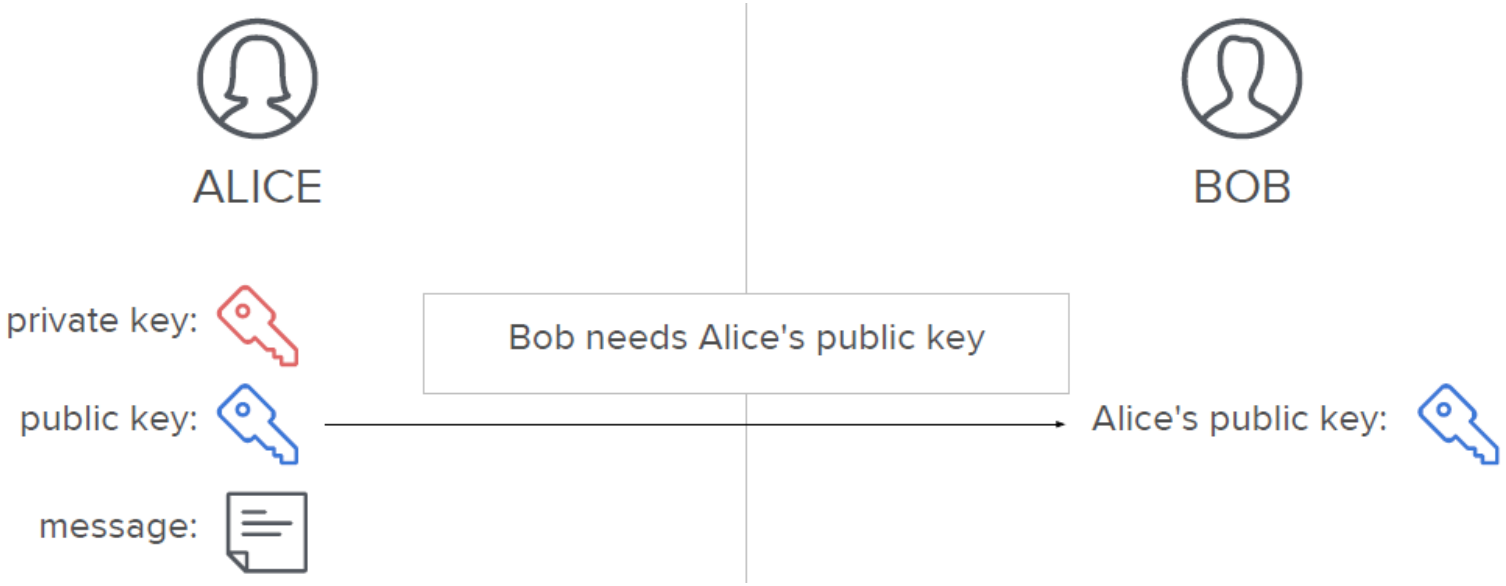
- Base-58 alphabet: 1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz, among these 62 chars omits (0, O, I, l)

Digital Signature Schemes Used with Identity

Sender puts digital signature (similar to a handwritten signature)
- Recipients can verify that the message was from you
- Others cannot forge a signature

Recipients given the (message, signature) pair can verify:

- ❑ Message Integrity - message cannot have been modified since sending
- ❑ Message Origin - original sender (owner of private key) has authorized this message/transaction
- ❑ Non-repudiation - the original sender cannot backtrack and claim they did not send the message



Digital Signature Schemes

Digital signature scheme consists of two algorithms:

A signing algorithm, Sign, which uses a secret key, sk.
s = Sign(m, sk), s is the signature for message m.

A verification algorithm, Verify, which uses a public key, pk.
valid = Verify(Sign(m, sk), pk),
invalid = Verify(s', pk) for all s' != s.

ECDSA : Elliptic Curve Digital Signature Algorithm

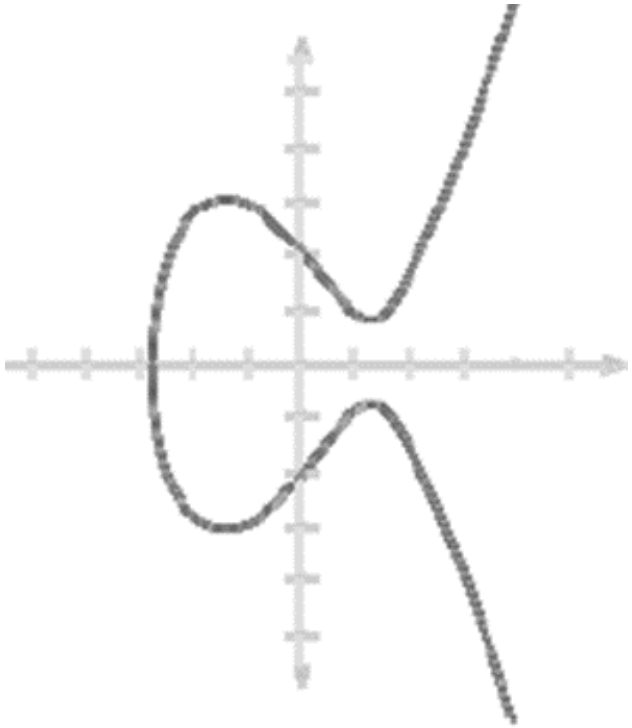
ECDSA is defined by:

E: an elliptic curve

g: a generator point of the elliptic curve with large prime order, p

p: a large, prime integer where $g^p = O$

H: a cryptographic hash function



Secret Key:

sk, chosen randomly from [0, …, p-1]

Public Key:

$pk = g^{sk}$

ECDSA - Sign

Sign(m, sk):

$h = H(m)$ // Hash the message

$z = h[0 : \log_2 p]$ // Take the $\log_2 p$ left-most bits of h

k = randomly chosen from $[1, \ldots, p-1]$ // k is kept secret

$r = x\text{-coord}(g^k) \pmod p$ // x-coord(P = (x, y)) = x

$s = (z + sk \cdot r) \cdot k^{-1} \pmod p$

return (r, s) //signature for the message

ECDSA - Verify

Verify(r, s, m, pk)

ECDSA Quantities

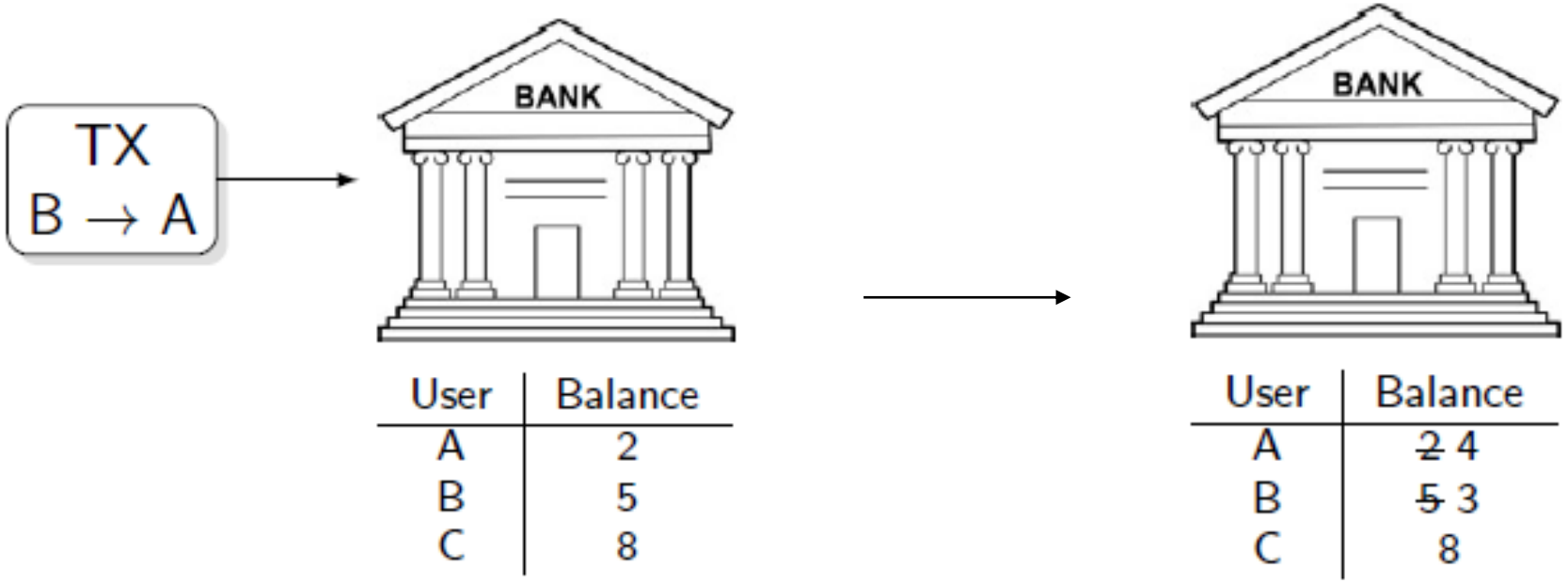
- Private key: 256 bits
- Public key, uncompressed: 512 bits
- Public key, compressed: 257 bits
- Message to be signed: 256 bits
- Signature: 512 bits

Messages are always hashed before being signed, so effectively any size message can be signed.

Blockchain Structure - Transaction

The Bank of Bitcoin

Customer B transfers 2 BTC to A



Before

After

Bitcoin Transaction

- **Bitcoin exists as software**
  - Transactions conducted through wallet app
  - Wallet creation generates a Bitcoin address

- **To receive money, you share your address**
  - Sender specifies address and amount

- **The transaction is broadcast to the network, where "miners" verify it and add it to the transaction history**

Wallet interface

Transaction - UTXO Model

Source: Wikipedia

Analogous to Rai Stones of the Yap Islands

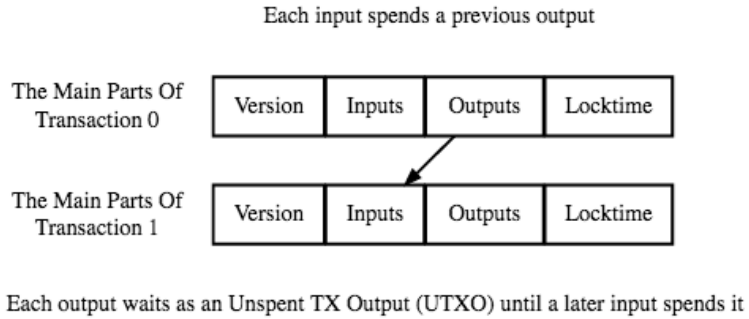
- Rai Stones never moved
- Instead: Agreed on change of ownership

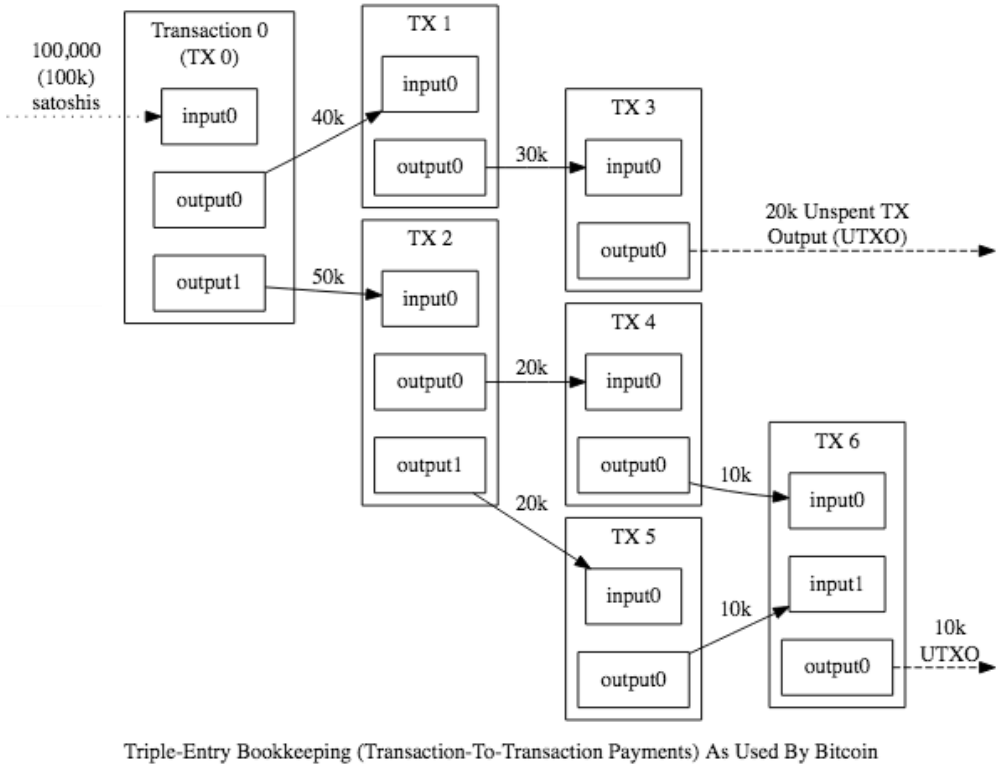
UTXOs stands for "Unspent Transaction Outputs"

- "I'm spending THIS bitcoin," not "I'm spending A bitcoin.“
- Only unspent outputs can be used as inputs to a transaction
- When a transaction happens, inputs are deleted and outputs are created as new UTXOs that may be consumed in future
- What if there is left-over – gives back to sender as UTXO

Transaction – Basic Concepts

- Maps inputs addresses to output addresses
  - Outputs can only be spent once
- Transactions contain signature of owner of funds
- Spending Bitcoin is redeeming previous transaction outputs
- Typical tx: one input, two outputs (with left-over)
- Fees are implicit

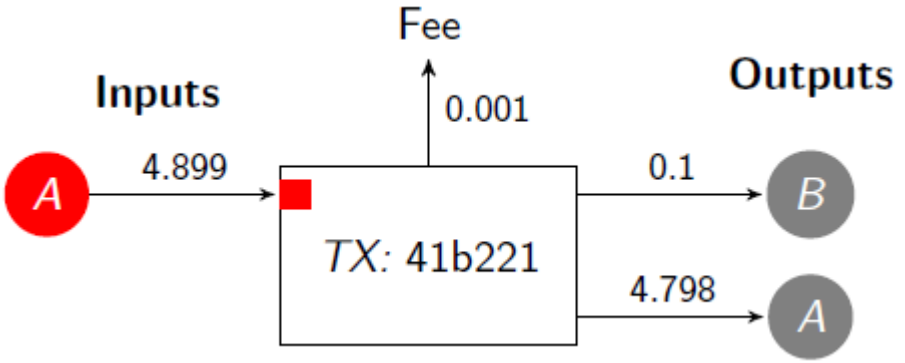


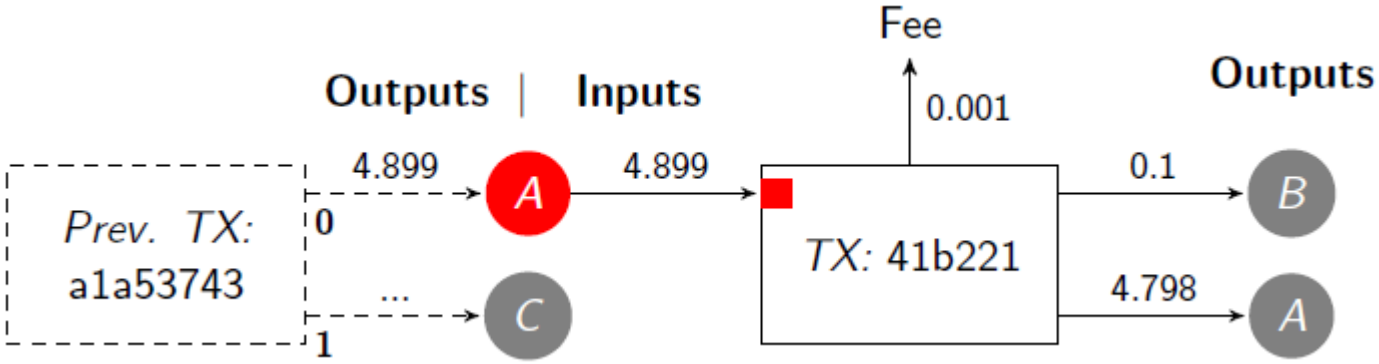


Triple-Entry Bookkeeping (Transaction-To-Transaction Payments) As Used By Bitcoin

Source: Bitcoin Developer Guide

Transferring Bitcoins





Contents of a Bitcoin Transaction

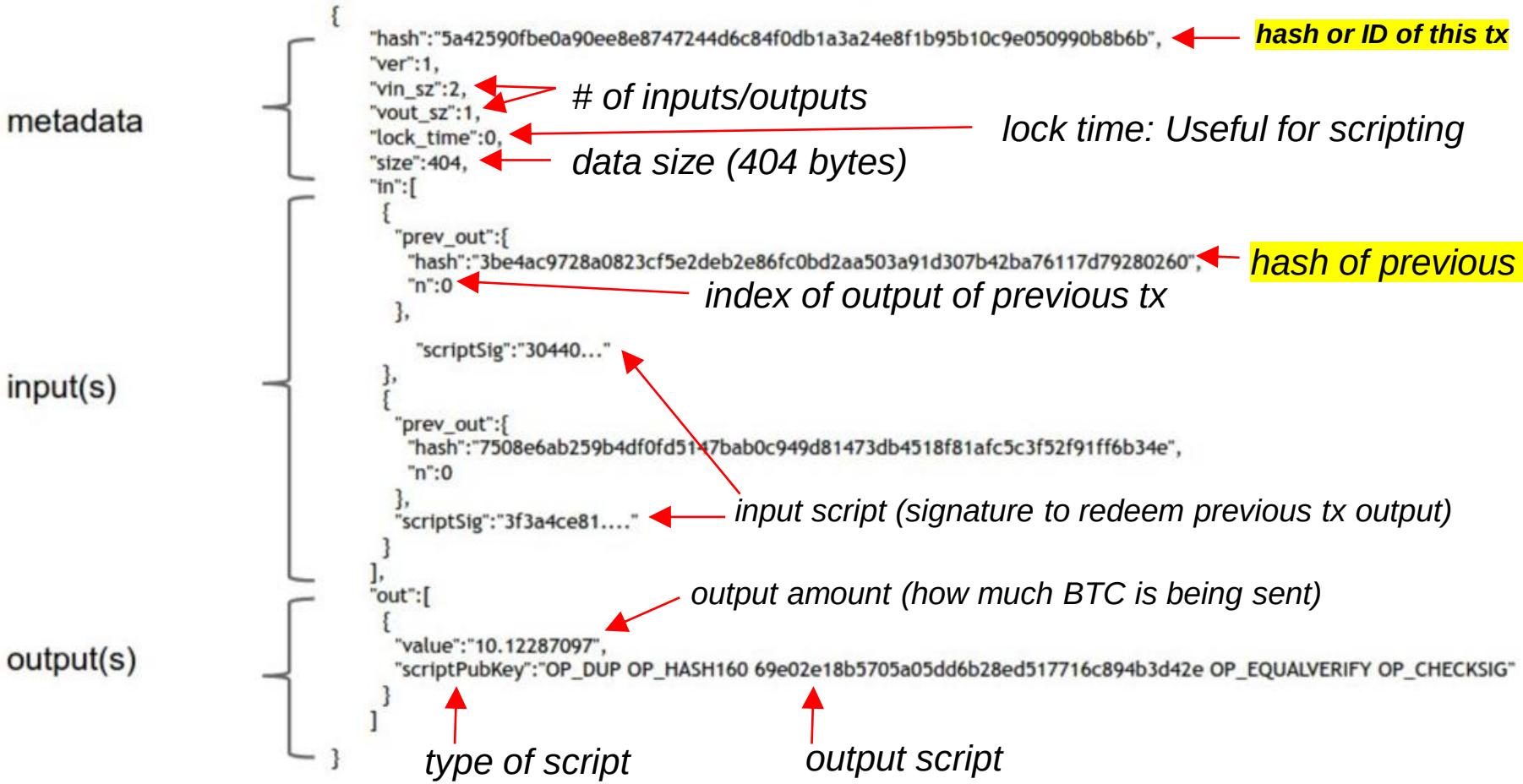


Figure 3.3 **An actual Bitcoin transaction.**

Bitcoin Scripting

Output "addresses" are really scripts.

"scriptPubKey": "OP_DUP OP_HASH160 69e02e18b5705a05dd6b28ed517716c894b3d42e OP_EQUALVERIFY OP_CHECKSIG"

➔ This particular Output Script: "This amount can be redeemed by the public key that hashes to address X, plus a signature from the owner of that public key"

Inputs and outputs through scripting allows for more functionality and future extensibility

Script or "Bitcoin Scripting Language": Language built specifically for Bitcoin

- Stack based
- Native support for cryptography
- Simple, not Turing complete (no loops)

Bitcoin Scripting

"scriptPubKey": "OP_DUP OP_HASH160 69e02e18b5705a05dd6b28ed517716c894b3d42e OP_EQUALVERIFY OP_CHECKSIG"

- **locking script:** found in previous tx output, specifies requirements for redeeming tx
- **unlocking script:** found in tx input, redeems the output of a previous tx
- Output script specifies a public key; input script specifies a signature for that public key
- bitcoin validating node will execute the locking and unlocking scripts in sequence to verify a redeeming transaction is valid

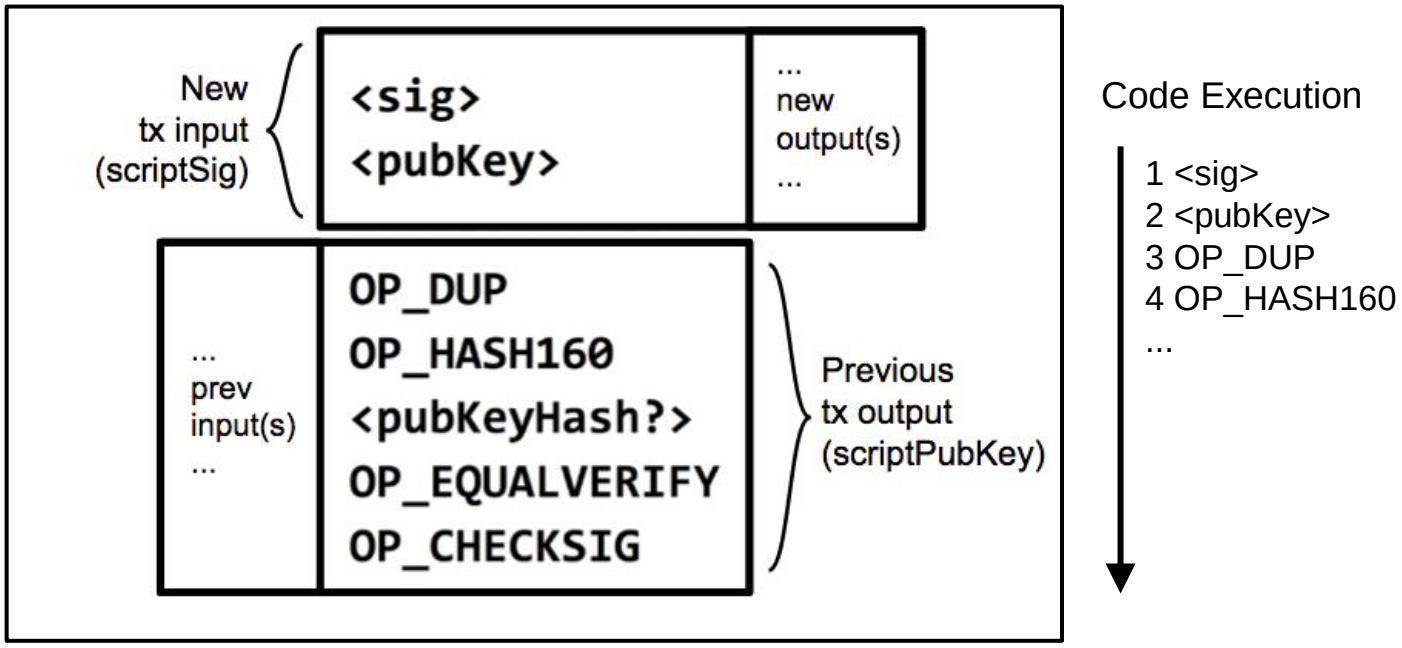
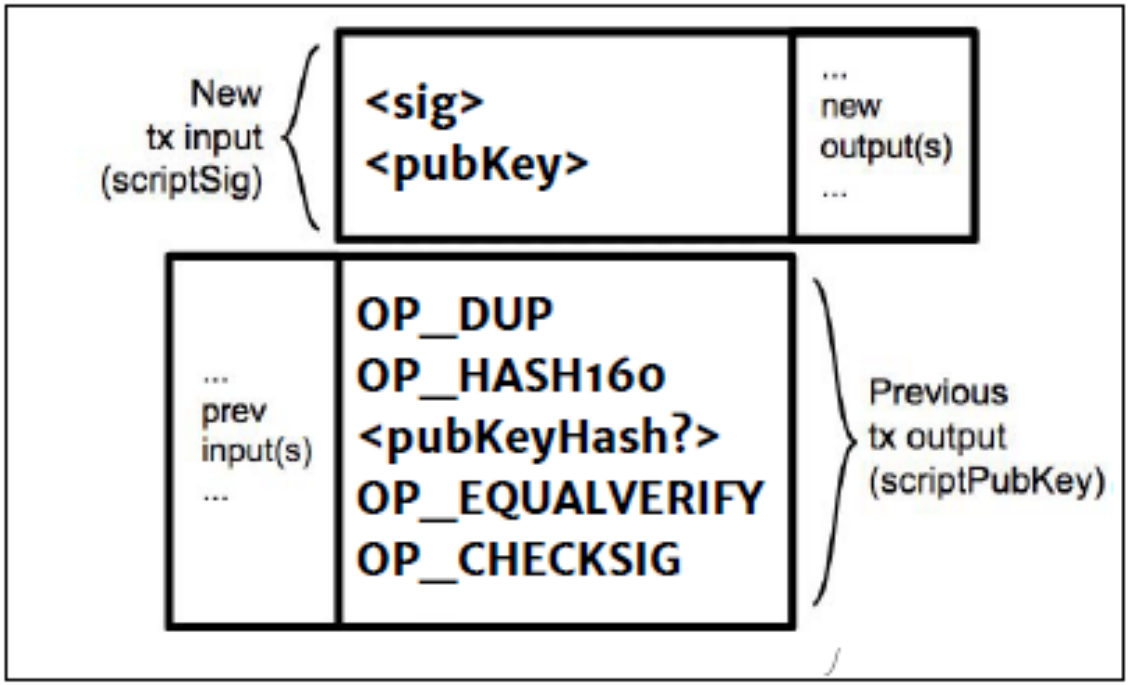


Figure: Two transactions along with their input and output scripts

Bitcoin Scripting

"scriptPubKey": "OP_DUP OP_HASH160 69e02e18b5705a05dd6b28ed517716c894b3d42e OP_EQUALVERIFY OP_CHECKSIG"

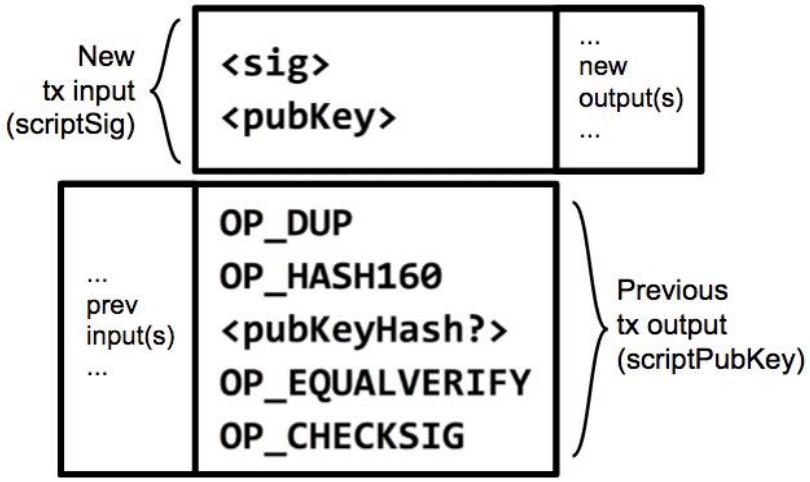


Code
Execution

"scriptPubKey": "
OP_DUP
OP_HASH160
69e02e18b5705a05dd6b28ed5
17716c894b3d42e
OP_EQUALVERIFY
OP_CHECKSIG"

- Output script scriptPubKey specifies a public key
- Input script scriptSig specifies a signature for that public key
- Putting a scriptSig and a scriptPubKey together results in a working script that Bitcoin clients can run to verify that a redeeming transaction is valid

Bitcoin Scripting

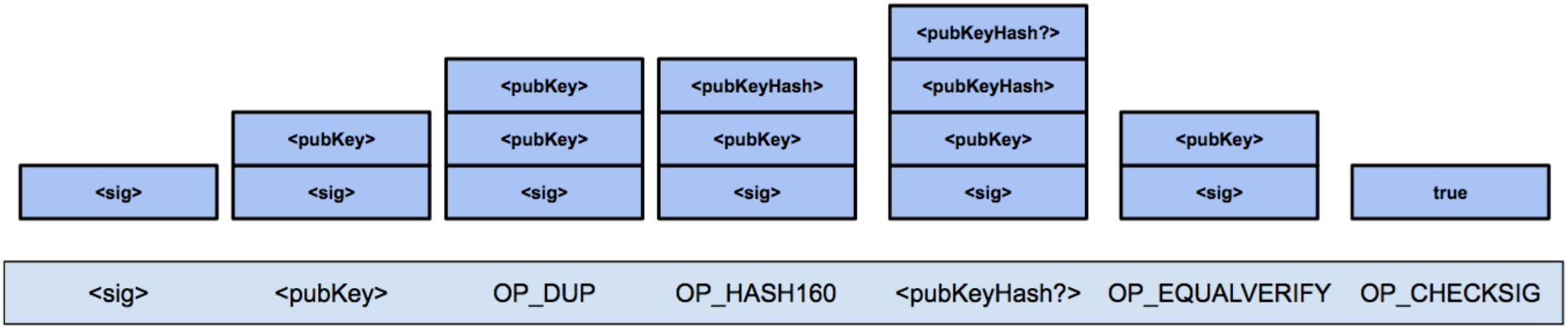


Output says: "This amount can be redeemed by

1) the **<pubKey>** that hashes to address **<pubKeyHash?>**

2) plus a **<sig>** from the owner of that **<pubKey>**

...that will make this script evaluate to **true**."



Bitcoin Scripting – Proof of Burn

Output script:

OP_RETURN

<arbitrary data>

How to write arbitrary data into the Bitcoin blockchain?

Proof of Burn

- OP_RETURN throws an error if reached
- Output script can't be spent - you prove that you destroyed some currency
- Anything after OP_RETURN is not processed, so arbitrary data can be entered

Use cases

- Prove existence of something at a particular point in time, Ex. A word you coined, hash of a document/music/creative works
- Bootstrap AltCoin by requiring that you destroy some Bitcoin to get AltCoin (alternative coin)

Transaction View information about a bitcoin transaction

447cb6623db32b5f28c94ac1055180207 5f053208fe995204a145197e2904bb9

3Nxwenay9Z8Lc9JBiywExpnEFiLp6Afp8v (44,000 BTC - Output)	3LrLWTSdd69oZVVQ6dtWaAAaBLn7N3rRjz - (Spent)	333.33328889 BTC
	3QkXtcSWJA9w77eCujnMBKDWFe7F7zwxTg - (Spent)	333.33328889 BTC
	3Qd7hXZoZ1iyXZznrbdUwUQBxHMmujdqhJ - (Spent)	333.33328889 BTC
	3ECJwvx9VgfotcUuEJMVNvmWnTGVMk179L - (Spent)	333.33328889 BTC
	3BuQmbmdce3e31GEovq5SgowLdfMgJzLDE - (Spent)	333.33328889 BTC
	3NwKLjJjzXSnBFQWokXRgBG3JeuF3bsnfE - (Spent)	333.33328889 BTC
	3GEaT8ZRXELcjMSFvGro6eZcC5S1LSLZuN - (Spent)	333.33328889 BTC
	35DVAzDtZDKAU94kFT9sxoscnuLCTxgwYc - (Spent)	333.33328889 BTC
	3Nxwenay9Z8Lc9JBiywExpnEFiLp6Afp8v - (Unspent)	38,000 BTC
	35mwqShnStDro6uEB4bmsgbyBo8en6Byfm - (Spent)	333.33328889 BTC
	39pvSqfNcUosc8RGVWxyzKM3ny96a3uSkW - (Spent)	333.33328889 BTC
	39QNJSgQg5JnBXAtbF8ezkDn72VqWdPZPJ - (Spent)	333.33328889 BTC
	3L9qAGBQLbXkFAB2GpijnJXPScSVjuiJio - (Spent)	333.33328889 BTC
	37WSkANPVUQ8uuktf8hv671CejRtBtQ4tJ - (Spent)	333.33328887 BTC
	3EEwPZZ6pYRJSotCz9RBoVYPRnoWyGWEka - (Spent)	333.33328889 BTC
	3C4ABC7iPcAAKBh6SJXfvUSDBew3abCtw3 - (Spent)	333.33328889 BTC
	3HpQozfTzoXAsHf87m2mwJXUQ14LVtLgK4 - (Spent)	333.33328889 BTC
	337RfngTLRTpU7RT9sKWQWDdmfcdmWnugi - (Spent)	333.33328889 BTC
	3P2eoKr3vAeZhJcTzon3VFkv5r7DqSXW9G - (Spent)	333.33328889 BTC
		43,999.9992 BTC

Summary	
Size	1055 (bytes)
Received Time	2016-08-30 11:45:03
Included In Blocks	427512 (2016-08-30 11:51:09 + 6 minutes)
Confirmations	854 Confirmations
Relayed by IP	5.39.93.85 (whois)
Visualize	View Tree Chart

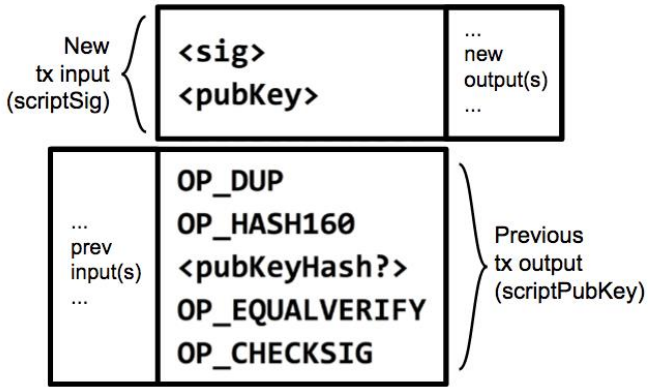
Inputs and Outputs	
Total Input	44,000 BTC
Total Output	43,999.9992 BTC
Fees	0.0008 BTC
Estimated BTC Transacted	333.33328887 BTC
Scripts	Hide scripts & coinbase

Pay-to-PubkeyHash vs. Pay-to-Script-Hash

Who specifies the script?

- **In Bitcoin, senders specify the locking script, recipients provide an unlocking script.**
- **P2PKH: Vendor (recipient of transaction) says "Send your coins to the hash of this PubKey."**
  - Simplest case
  - By far the most common case
- **Pay-to-Script-Hash (P2SH): Vendor 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.”**
  - A vendor cannot say, “To pay me, write a complicated output script that will allow me to spend using multiple signatures.”

It is provided by vendor. How would the sender know?



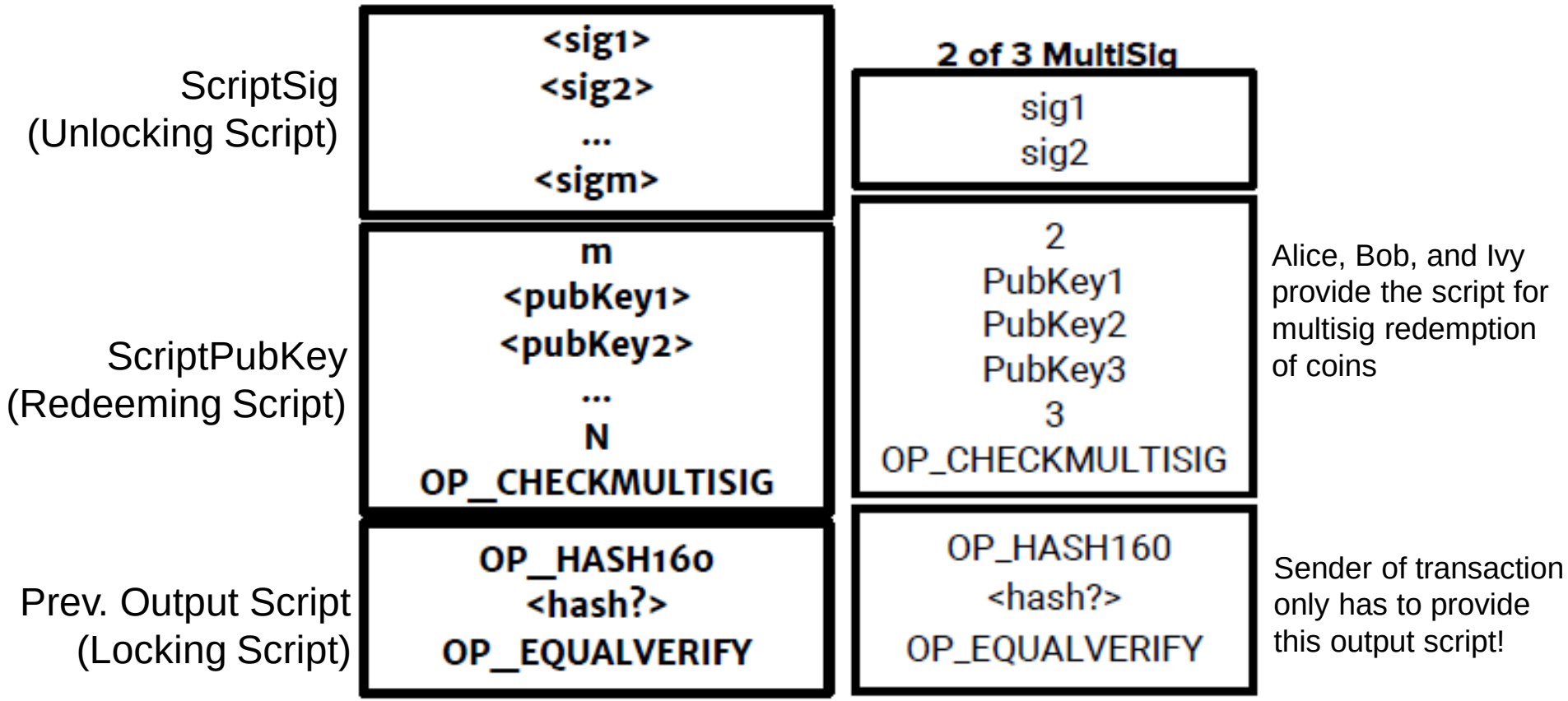
Simple P2PKH script - recipient only has to provide signature and public key

Why Pay-to-Script-Hash?

- **Offloads complicated script writing to recipients**
- **Makes more sense from a payer-payee standpoint**
  - Merchant is responsible for writing correct and secure script, and give the hash of script to the customer
  - Customer doesn't care what the script actually is, or how company holds their funds in order to easily send the money
- **P2SH is the most important improvement to Bitcoin since inception**
- **Example: MultiSig**
  - M of N specified signatures can redeem and spend the output of this transaction

MULTISIG Example

Alice, Bob, and Ivy are in charge of a joint account and make all spending decisions together: 2 out 3 persons can sign to authorize.



Blockchain Structure – Ledger
Ledger Data Structure

Triple-Entry Accounting

- Double entry accounting: a transaction has a debit and a credit
- With Bitcoin, there is a 3rd entry. Every transaction goes into a repository of common knowledge.
- This repository or public ledger is highly secure and maintained by everyone on the network
- The public ledger is the final word – so there can be no disagreement about the debits and credits and there can be no "double spending"
- The public ledger is called the "blockchain" or the "World Wide Ledger"

Account-based vs. Transaction-based ledger

Account-based

- must track every transaction affecting Alice
- Requires additional maintenance, error-prone

Downside: keep track of account balances

Question: Does Alice have the 15 coins?

Create 25 coins and credit to Alice$_{\text{ASSERTED BY MINERS}}$
Transfer 17 coins from Alice to Bob$_{\text{SIGNED(Alice)}}$
Transfer 8 coins from Bob to Carol$_{\text{SIGNED(Bob)}}$
Transfer 5 coins from Carol to Alice$_{\text{SIGNED(Carol)}}$
Transfer 15 coins from Alice to David$_{\text{SIGNED(Alice)}}$

Figure 3.1 an account-based ledger

Bitcoin keeps tracks of transactions (triple-entry accounting).

Features:

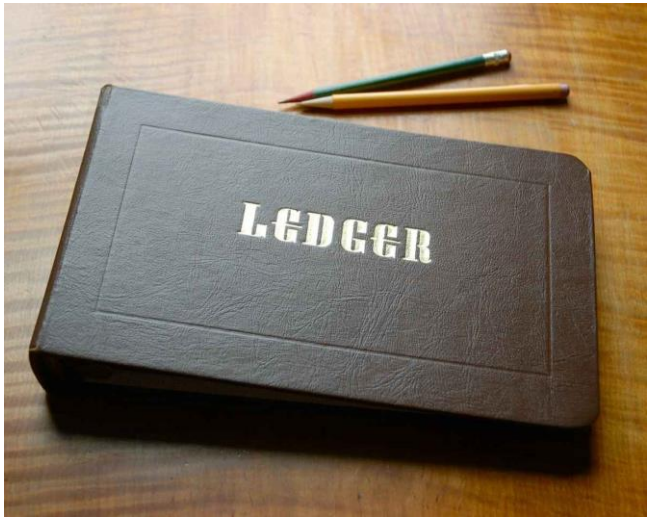
- Change addresses - Required since tx outputs only spent once
- Efficient verification - only read recent history
- Joint payments - Alice + Bob form 1 tx
- Bitcoin is optimized to be extremely simple and robust, leaving as little room for error as possible

1	Inputs: Ø Outputs: 25.0→Alice	
2	Inputs: 1[0] Outputs: 17.0→Bob, 8.0→Alice	SIGNED(Alice)
3	Inputs: 2[0] Outputs: 8.0→Carol, 9.0→Bob	SIGNED(Bob)
4	Inputs: 2[1] Outputs: 6.0→David, 2.0→Alice	SIGNED(Alice)

Figure 3.2 a transaction-based ledger, which is very close to Bitcoin

Distributed Ledger Technology

- **DLT is a protocol for building a replicated and shared record ledger system. Such a system may be used to record a wide range of items, such as asset ownership, asset transfer transactions, and contract agreements.**
  - Bitcoin, one of the best-know applications of DLT
- **The ledger function is similar to that of a conventional paper-based or electronic-based ledger system**
- **It provides a new way of constructing a secure record system that offers stakeholders more transparency, and encourages member participation in its operations.**



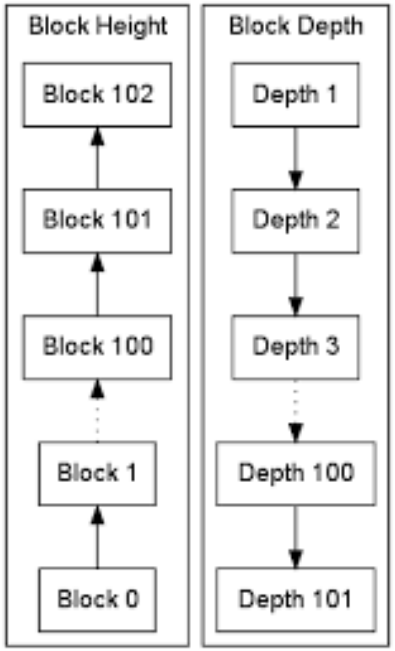
Basic Concepts - Blocks + Blockchain

Blocks

- Contains an ordered bunch of transactions
  - Timestamps the transactions, are immutable
- Each block references a previous block
- Each block has height and depth (confirmations)
  - Currently 564k blocks

Blockchain

- The entire series of blocks 'chained' together



Block Height Compared To Block Depth

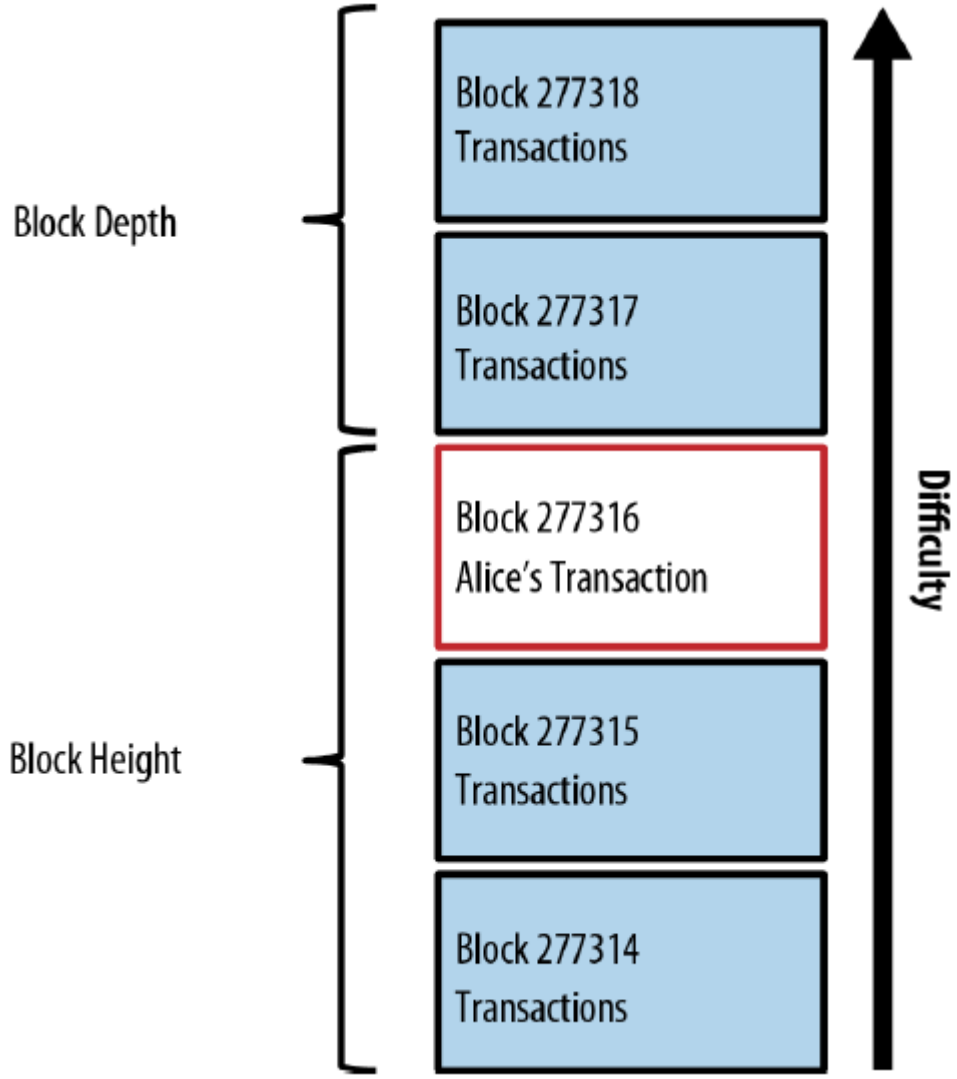
LATEST BLOCKS

Height	Age	Transactions	Miner	Size (bytes)
564886	2 minutes	2069	BTC.com	1,174,260
564885	3 minutes	1684	AntPool	1,123,996
564884	4 minutes	2669	F2Pool	1,271,307
564883	16 minutes	1685	ViaBTC	1,111,724
564882	19 minutes	2835	DPOOL	1,205,018

Source: Bitcoin Developer Guide

Basic Concepts - Blocks + Blockchain

- Alice's transaction included in block #277316
- Block height is: 277316
- Block depth is: 2



Basic Concepts - Blocks + Blockchain

Transaction View information about a bitcoin transaction

0627052b6f28912f2703066a912ea577f2ce4da4caa5a5fbd8a57286c345c2f2

1Cdid9KFAaatwczBwBttQcwXYCpvK8h7FK (0.1 BTC - Output)

1GdK9UzpHBzqzX2A9JFP3Di4weBwqgmoQA - (Unspent) 0.015 BTC

1Cdid9KFAaatwczBwBttQcwXYCpvK8h7FK - (Unspent) 0.0845 BTC

0.0995 BTC

Summary	
Size	258 (bytes)
Weight	1032
Received Time	2013-12-27 23:03:05
Included In Blocks	277316 (2013-12-27 23:11:54 + 9 minutes)
Confirmations	209489 Confirmations
Visualize	View Tree Chart

Inputs and Outputs	
Total Input	0.1 BTC
Total Output	0.0995 BTC
Fees	0.0005 BTC
Fee per byte	193.798 sat/B
Fee per weight unit	48.45 sat/WU
Estimated BTC Transacted	0.015 BTC
Scripts	Hide scripts & coinbase

Alice's transaction to Bob's Cafe

Basic Concepts - Blocks + Blockchain

<< Previous **Blocks mined on: 24/09/2017** Next >>

Height	Time	Relayed By	Hash	Size (kB)
486805 (Main Chain)	2017-09-24 17:16:53	AntPool	00000000000000000f0b7f68eb408ba5351ca05bdd839fb4446e816240e0d9e	9.19
486804 (Main Chain)	2017-09-24 17:07:33	BTC.com	0000000000000000000227136ac7e966957d838e4aa0e5ecb58b41236c77147c5	1,006.25
486803 (Main Chain)	2017-09-24 17:00:44	AntPool	00000000000000000008b0df39abc9688a4883cb2bcdaa57b4922d04e24737ec4	1,004.93
486802 (Main Chain)	2017-09-24 16:32:07	F2Pool	00000000000000000049b0123a7180b8c899fca74c4dd71a09949376569d2080	705.71
486801 (Main Chain)	2017-09-24 16:20:58	AntPool	00000000000000000024476b790ed0a11e91d967b8b04c9f7059a9836701764c	624.38
486800 (Main Chain)	2017-09-24 16:12:18	ViaBTC	00000000000000000003b6eefa619ddcdce375b7cc83f4dcaec2761f65913a8d1	1,032.99
486799 (Main Chain)	2017-09-24 15:59:55	Unknown	00000000000000000a06d8dc0b1ff8c624005d3e6c8681e3ae86e12cc871f60	498.85
486798 (Main Chain)	2017-09-24 15:50:06	1Hash	00000000000000000007ea205ad905dd063b44e7d666e4350e13b9fd508b89cec	998.13
486797 (Main Chain)	2017-09-24 15:47:02	AntPool	00000000000000000a32f2694ca47df9e5b5db7fdae65b6f0beefc6790b5cc1	1,001.74
486796 (Main Chain)	2017-09-24 15:31:17	BTC.TOP	00000000000000000db3c23e8c84b4fbb16036fc9c5da59074e44e71d4d7f80	999.22

Basic Concepts - Blocks + Blockchain

Block #486804

Summary	
Number Of Transactions	1674
Output Total	9,966.30652121 BTC
Estimated Transaction Volume	528.80390433 BTC
Transaction Fees	1.21061888 BTC
Height	486804 (Main Chain)
Timestamp	2017-09-24 17:07:33
Received Time	2017-09-24 17:07:33
Relayed By	BTC.com
Difficulty	1,103,400,932,964.29

Hashes	
Hash	00000000000000000227136ac7e966957d838e4aa0e5ecb58b41236c77147c5
Previous Block	00000000000000008b0df39abc9688a4883cb2bcdaa57b4922d04e24737ec4
Next Block(s)	00000000000000000f0b7f68eb408ba5351ca05bdd839fb4446e816240e0d9e
Merkle Root	d74b74c40805ddcb9139e492957747616acd23b93e6467a8e9ebadc23a42f965

Transactions

61da61fd58b25f0217dd0c575386e4320457ac8868eaeafd085a5df008485386 **(Size: 244 bytes) 2017-09-24 17:07:33**

No Inputs (Newly Generated Coins) → 3EhLZarJUNSfV6TWMZY1Nh5mi3FMsdHa5U - (Unspent) 13.71061888 BTC
Unable to decode output address - (Unspent) 0 BTC
13.71061888 BTC

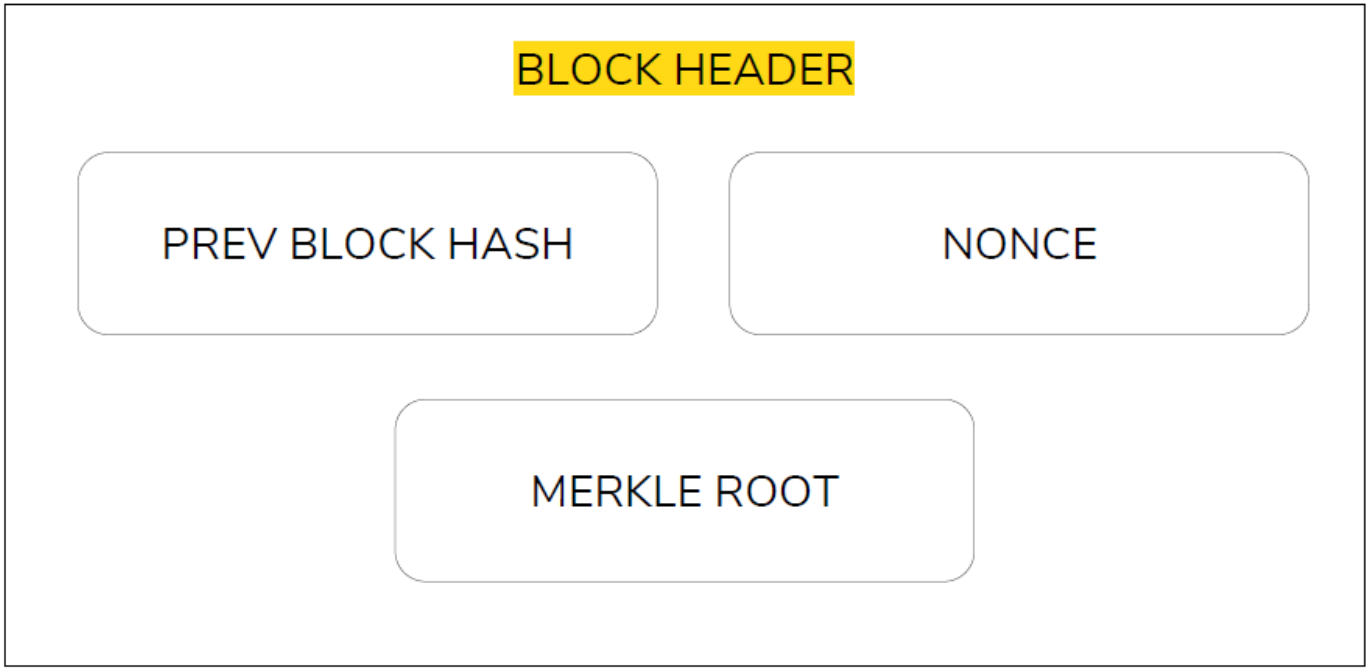
e5b8b760e9f94be6fd6061c82ceb7db1c4be2dc3dfee9b0c7670f5cb01c52107 **(Fee: 0.00331775 BTC - 368.64 sat/WU - 1,474 sat/B - Size: 225 bytes) 2017-09-24 17:01:33**

1FpooPkQvLBnTnPsa8oQtzJ9Jt7b67s8Ug (0.2975335 BTC - Output) → 1u4VsS1jnJChf91QGtG9W8H2ziSdQMT8e - (Unspent) 0.13271 BTC
1FpooPkQvLBnTnPsa8oQtzJ9Jt7b67s8Ug - (Unspent) 0.16150575 BTC
0.29421575 BTC

9244131167a8f0ed742d196a58b51860a388e53ea494905b19689e3d4bf2880d **(Fee: 0.0014 BTC - 154.87 sat/WU - 619.47 sat/B - Size: 226 bytes) 2017-09-24 17:04:14**

17a1HFTG7cG5VSrHn3AytdoQ2MHxybXN8i (0.05066915 BTC - Output) → 1cY2JsCr6rvM5kKR4ENnDwfrX6qHt7Ayh - (Unspent) 0.00026915 BTC
1LCGvTsxA6G8JBqe3hcd9Dhf1m8Huwz7Y1 - (Unspent) 0.049 BTC
0.04926915 BTC

What is in a Block?



blockID = **H(blockHeader)** = H(prevBlockHash || merkleRoot || nonce)

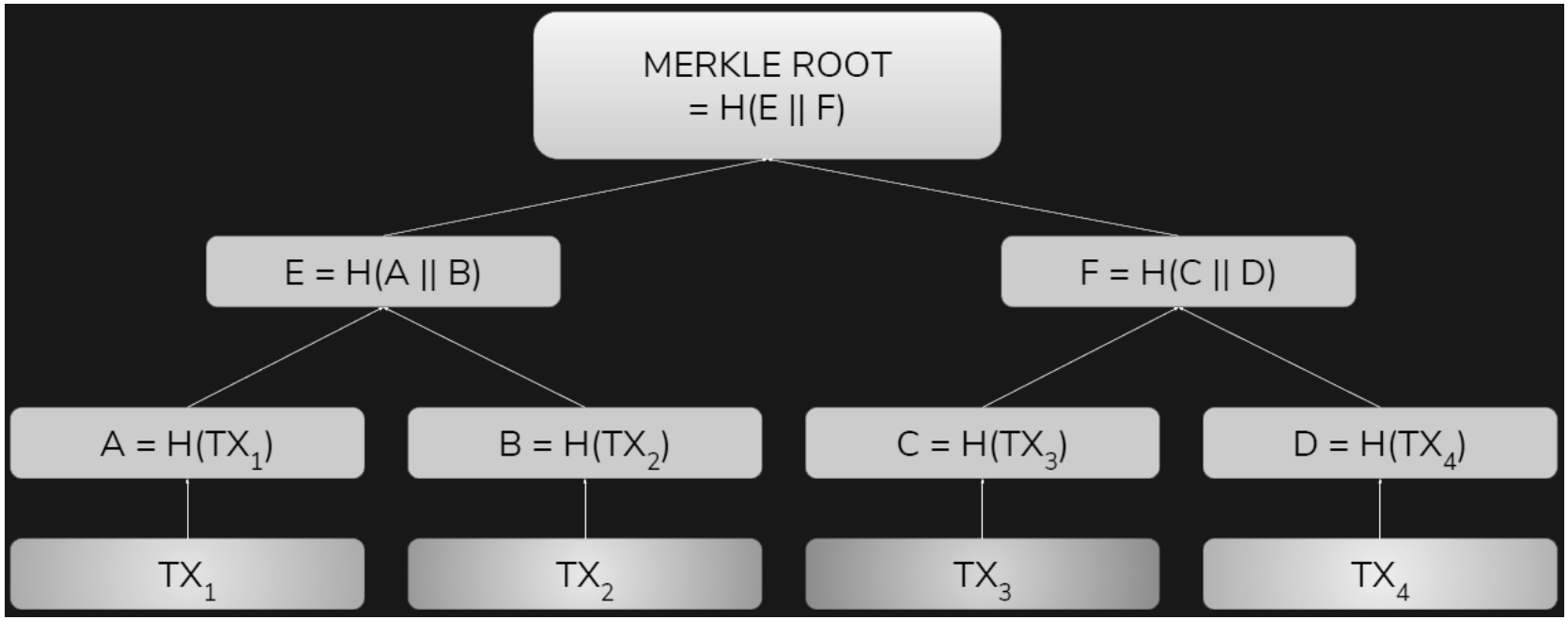
What is in a Block? – More Details

- **To identify a block in the blockchain via its block header hash**
- **The block header hash is calculated by running the block header through the SHA256 algorithm twice**
- **A block header hash is not sent through the network but instead is calculated by each node as part of the verification process of each block**

The block header data (80-byte)

Size	Field	Description
4 bytes	Version	The Bitcoin Version Number
32 bytes	Previous Block Hash	The previous block header hash
32 bytes	Merkle Root	A hash of the root of the merkle tree of this block's transactions
4 bytes	Timestamp	The timestamp of the block in UNIX.
4 bytes	Difficulty Target	The difficulty target for the block.
4 bytes	Nonce	The counter used by miners to generate a correct hash.

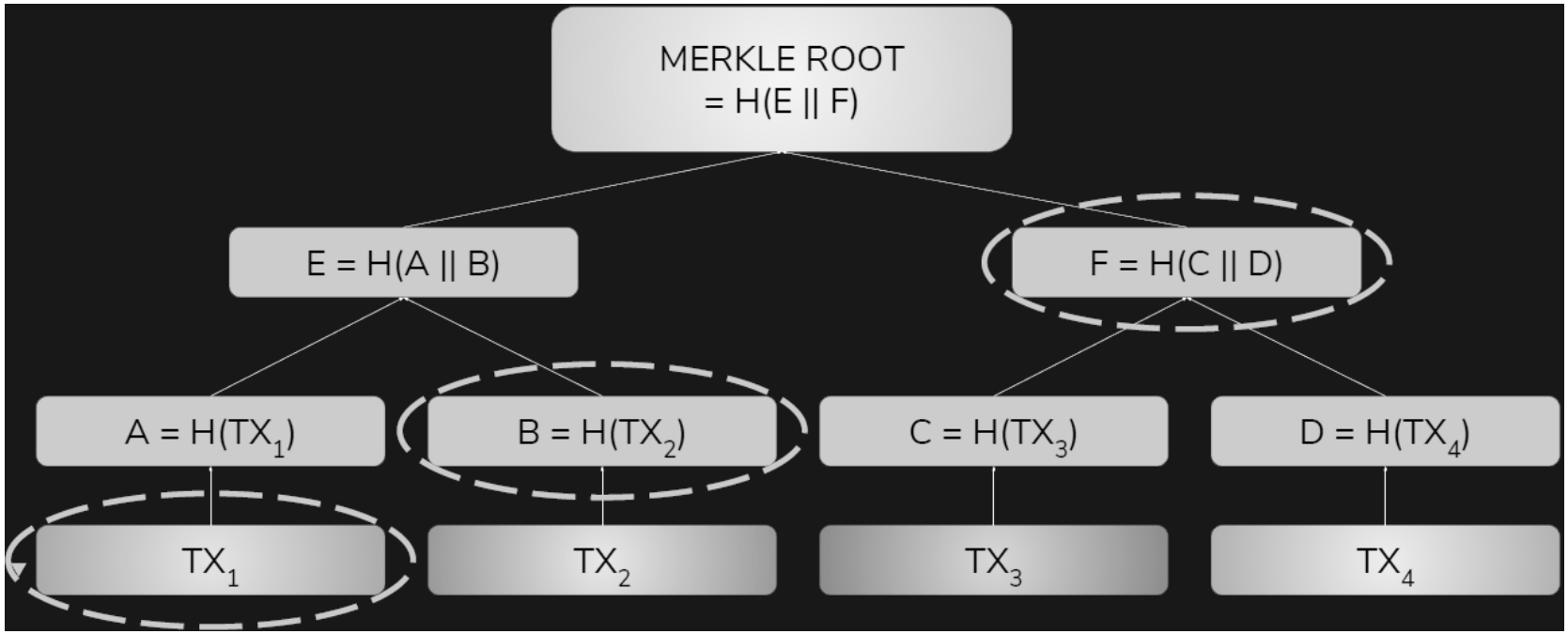
MERKLE Tree Stores Transaction Data



A binary tree of hash pointers:

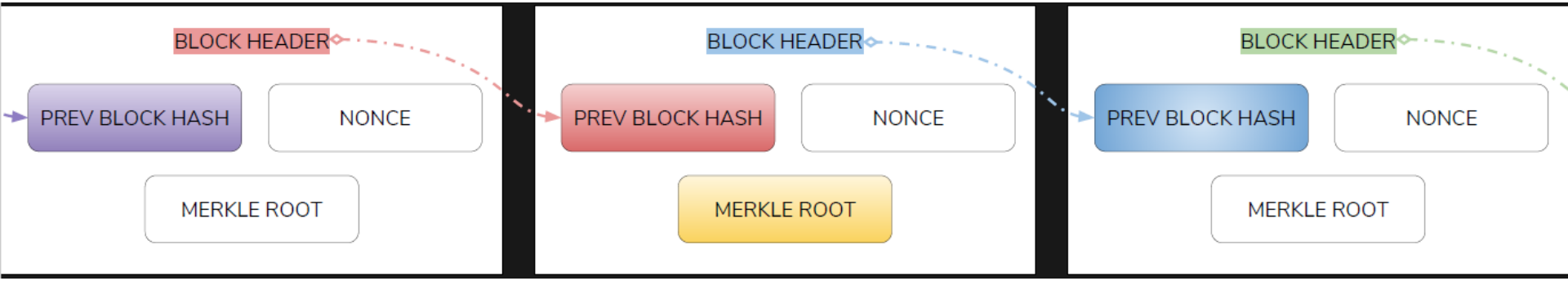
- **Hash a number of data and their hashes together, ending up with a single hash called Merkle Root**
- **Prove your data existed in the tree corresponding to the Merkle Root without saving all of the data**
- **Ex: For data in 3rd node, to prove inclusion in the tree, you simply save 3 intermediate hashes, and show you can produce the Merkle Root with this data**

MERKLE Branch & Proof of Inclusion



- For example: for data in TX1, to prove inclusion in the tree, you simply save 2 intermediate hashes (B and F), and you can produce Merkle Root with TX1, B and F

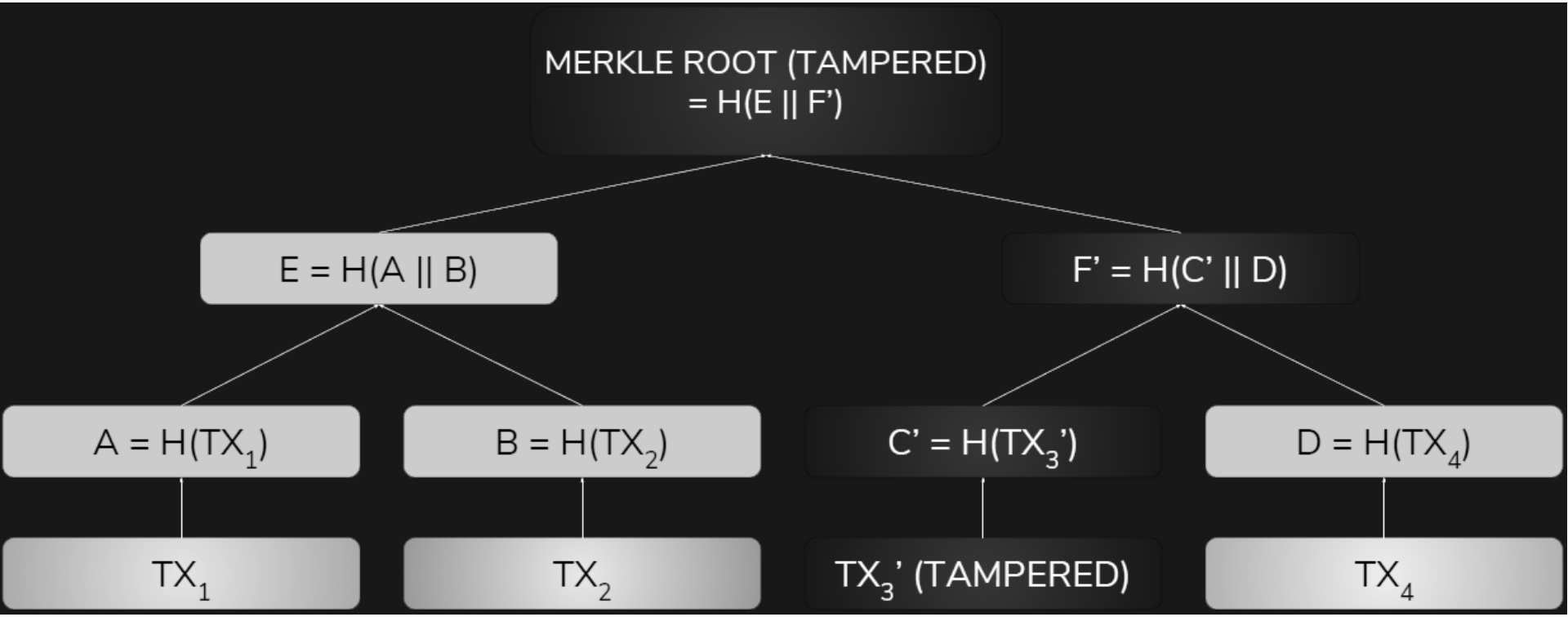
How Blocks are Chained Together?



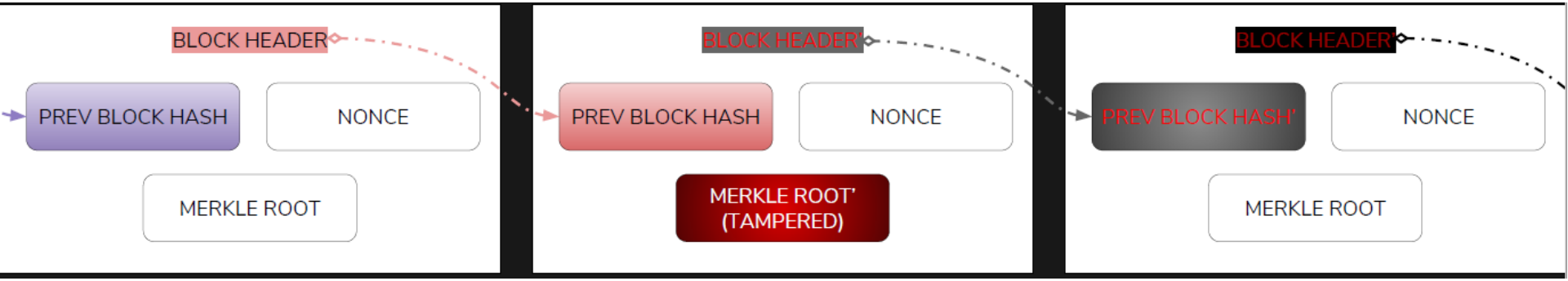
prevBlockHash = H(blockHeader) = H(prevBlockHash || merkleRoot || nonce)

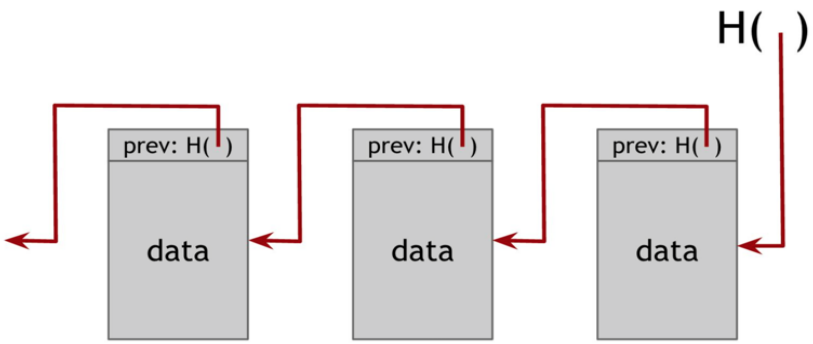
Hash is SHA256(SHA256(x))

Blockchains are tamper evident

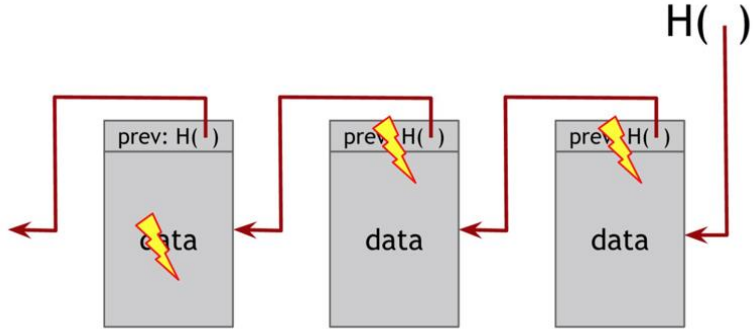


Blockchains are tamper evident





Clean Blockchain



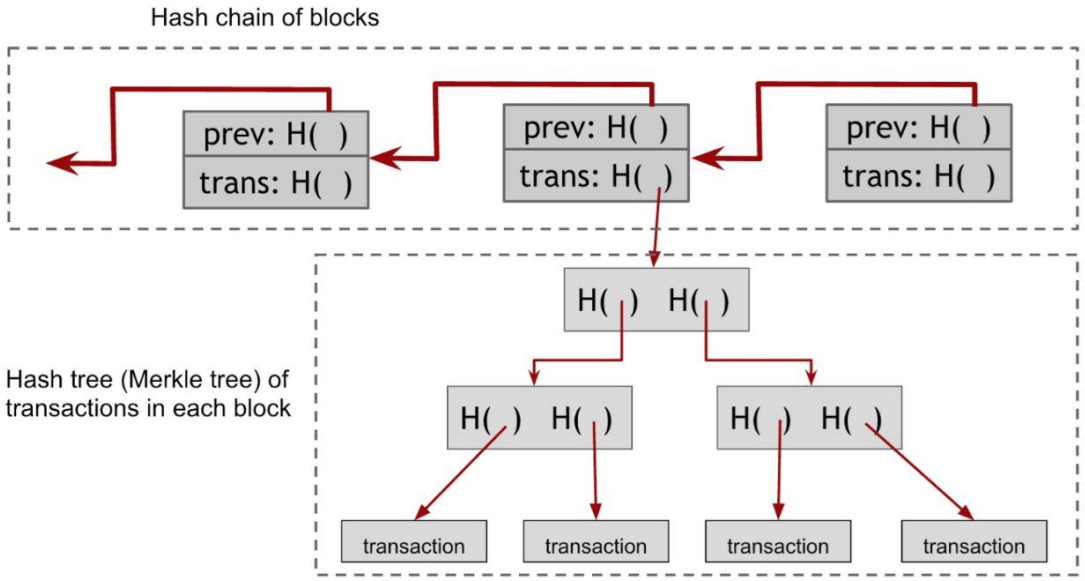
Tampered Blockchain

Summarize – Blockchain Structure

Transactions are leaves in the Merkle tree, includes a coinbase transaction

Two hash structures

1. **Hash chain of blocks - these blocks are linked together and based off of each other.**
2. **A Merkle tree of txs, internal to each block**



Princeton Textbook Figure 3.7/3.8

Blockchain Structure – Ledger
Ledger Update

Recap - The Innovation of Satoshi Nakamoto

- **First ever decentralized, trustless system for transactions**

- **Nakamoto solved the Double Spending problem**
  - Prevent someone from spending the same asset twice
  - Solution? The blockchain + PoW - those who helped secure the network were rewarded with Bitcoin, and those who tried to compromise the Bitcoin network had to make a massive financial investment to do so, with relatively little benefit

Byzantine Generals Problem

Bitcoin and decentralized currency is actually an instantiation of the Byzantine generals problem

Group of generals surrounding a city must vote and agree on a plan of action

Constraints:

- Generals are physically separated; must use messengers
  - Messengers may fail
- Generals may be loyal or intentionally traitorous
- Assume majority of generals are loyal
- "Byzantine Fault Tolerance" achieved if loyal generals have unanimous agreement on strategy

In Bitcoin, this is an agreement on the history of transactions

Sketch of Bitcoin Mining - Proof of Work

- **Solution to the Byzantine Generals Problem: Proof-of-work**
  - Miners compete to solve a computationally difficult problem
- **Proof of work criteria: Easy to verify - Hard to compute**
- **SHA-256 Hash function satisfies these**
  - SHA-256 (Secure Hash Algorithm) developed by the NSA
  - Used for SSL (Secure Sockets Layer)/TLS for secure traffic on the Internet
  - Input can be any arbitrary size data (maximum 2^64-1 bits)
  - Output is 64 numbers/characters (called hexadecimal, a-f + 0-9)
  - Pretty much random (very useful property)
- **Example**
  - SHA256("Donald Trump") == "e4f2e1f0e2ae4d3ce7018cf3b4f3577c99714bdc9f5a4ac28e3e7cb2c505db6c"
  - SHA256("Donald trump") == "6ad2fa6a5caaee9143578931456322c4433a92ae2af8f0d5c9b4f9bb080f49d6"

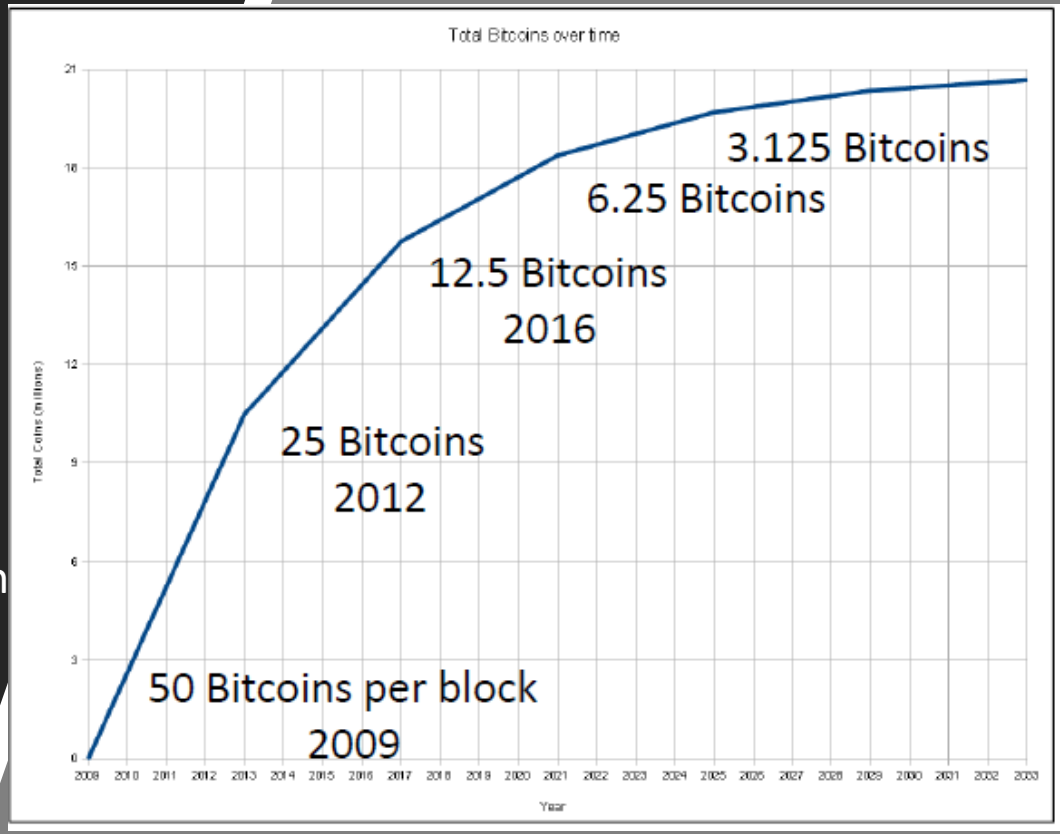
Sketch of Bitcoin Mining - Finding blocks

Finding the PoW => 'found' a block; can add block to blockchain

- Miner who found block adds "coinbase transaction“
- contains mining reward (currently 6.25 BTC, halved in May 2020)
- Miner broadcasts block
- Other nodes verify, then add to their own copy of the blockchain

Timeline + stats

- This happens roughly every 10 minutes
- Difficulty of the problem adjusted every 2 weeks
- Block reward halving every 4 years
- Bitcoin is in limited supply - 21 million bitcoins by 2140 - Deflationary
- 17.2 million bitcoins currently in circulation



The Mining Problem – Preimage Hash Puzzle

Bitcoin's partial preimage hash puzzle:

A problem with a requirement to find a nonce that satisfies the following inequality:

H(prevBlockHash || merkleRoot || nonce) < target

Used to implement Proof-of-Work in Bitcoin (and every other PoW cryptocurrency)

Hash puzzles need to be:

1. **Computationally difficult.**
2. **Parameterizable.**
3. **Easily verifiable**

The Mining Problem – Preimage Hash Puzzle

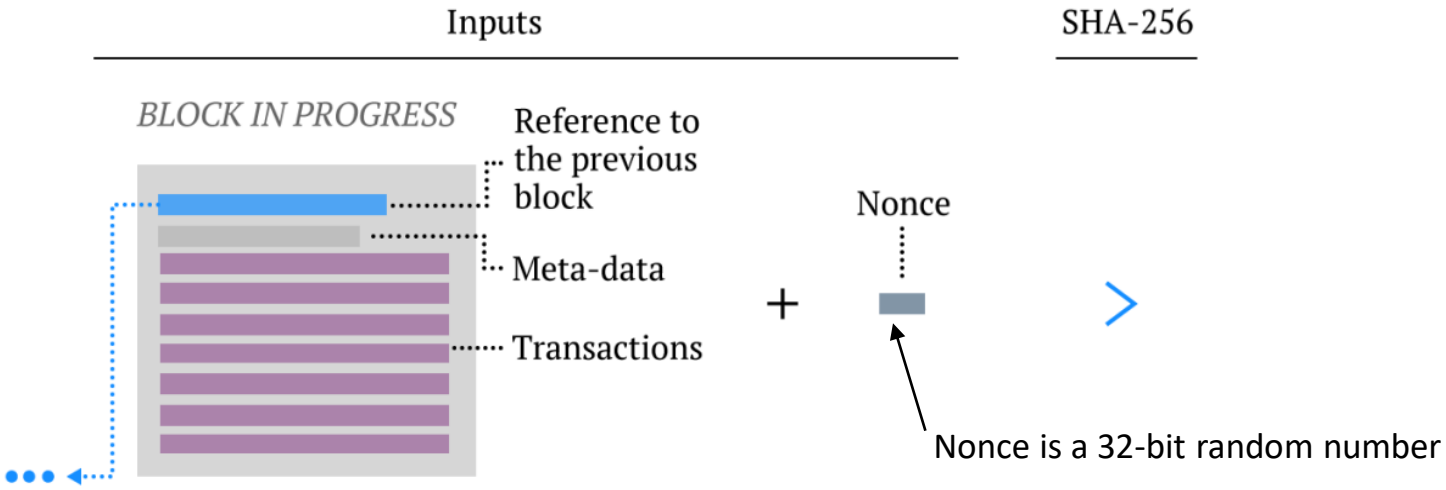
Solution (Proof-of-work): an output with a requisite number of leading 0s

- The number of the 0 bits is the difficulty
- Difficulty adjusts every 2016 blocks (~2 weeks) to regulate block creation

Difficulty: as expected number of computations to find a block

- Implemented as requirement of leading number of 0s
- Adjusts with global hashrate
- difficulty *= two_weeks /time_to_mine_prev_2016_blocks

Hash in The Mining



- Their goal is to find a hash that has at least a certain number of leading zeroes, e.g.

00000eb9c313a3c87d4b1fadb69a9d1395cdbc802b10707fa7e620ad722c0f63

- More leading zeroes means fewer solutions – and more time to solve the problem – it determines the "difficulty" (currently 17 zeros)

Probability of a single hash being the solution: 0.00000000000000000000477
Probability of winning lottery twice in a row: 0.0000000000000000000000019

CPU Mining Pseudocode

```
TARGET = (65535 << 208) / DIFFICULTY;
coinbase_nonce = 0;
while (1) {
      header = makeBlockHeader(transactions, coinbase_nonce);
      for (header_nonce = 0; header_nonce < (1 << 32); header_nonce++){
            if (SHA256(SHA256(makeBlock(header, header_nonce))) <
      TARGET)
                  break; //block found!
      }
      coinbase_nonce++;
}
```

Mining – Changing a Nonce in Block Header

Previously, hash of:

- Merkle Root
- PrevBlockHas
- Nonce (varied value)

below some target value.

Actually two nonces:

1. In the block header
2. In the coinbase tx

Hash of

- PrevBlockHash
- Coinbase nonce (varied value)
  - Affects the Merkle Root
- Block header nonce (varied value)

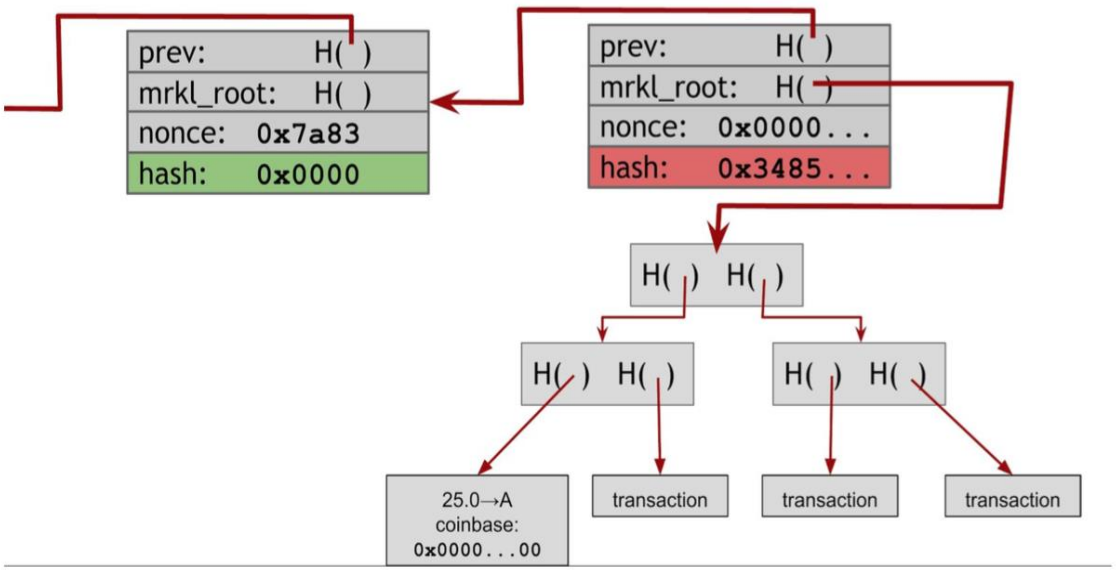


Figure 5.1: Finding a valid block. In this example, the miner tries a nonce of all 0s. It does not produce a valid hash output, so the miner would then proceed to try a different nonce.

Princeton Textbook

Mining – Changing a Nonce in Coinbase

What if there is no solution?

- **Block header nonce is 32 bits**
  - Antminer S9 hashes 14 TH/s
  - How long does it take to try all combinations?
  - 2^32 / 14,000,000,000,000 = 0.00031 seconds
  - Exhausted 3260 times per second
- **Therefore, must change Merkle root**
  - Increment coinbase nonce, then run through block header nonce again
  - Incrementing coinbase nonce less efficient because it must propagate up the tree

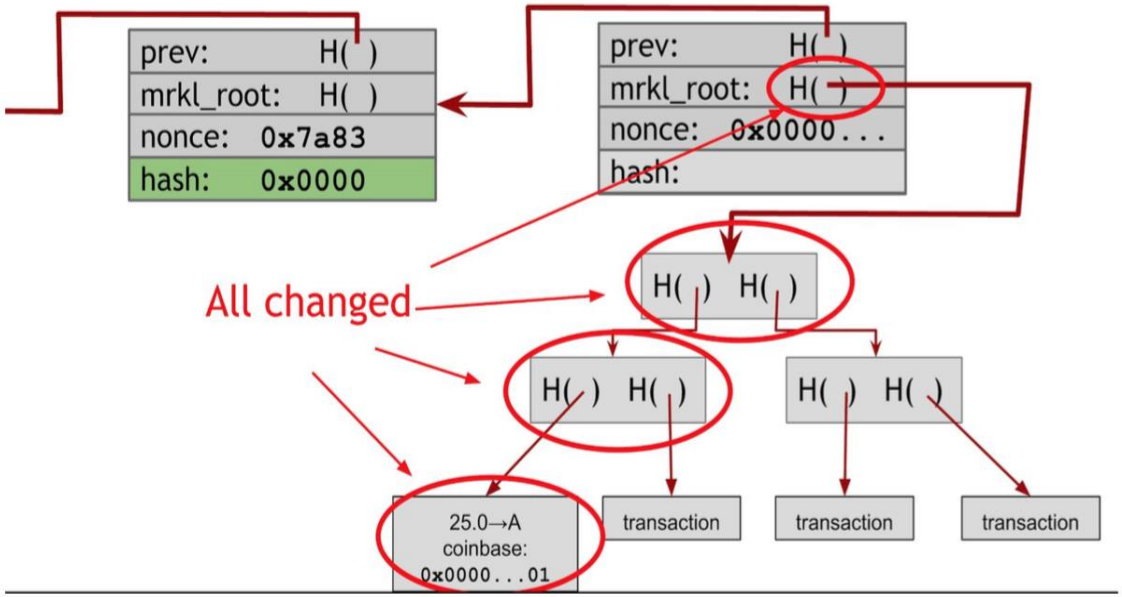
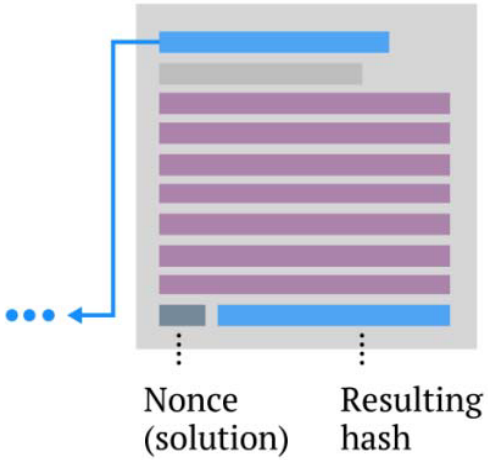


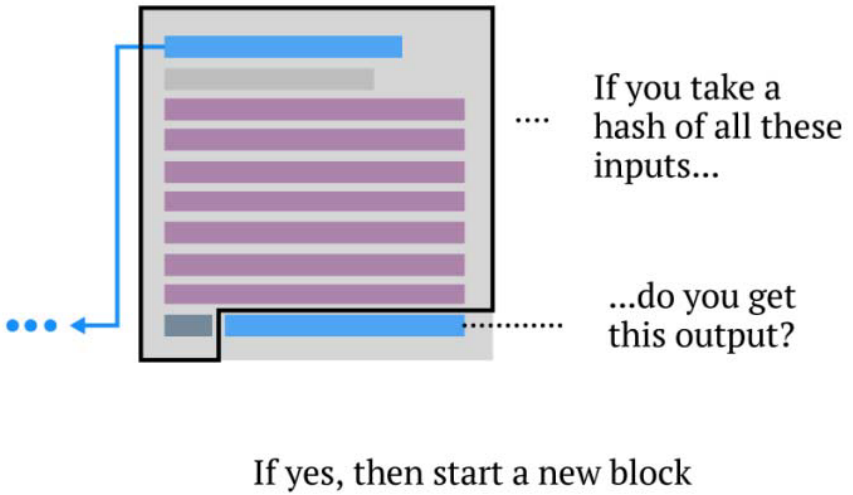
Figure 5.2: Changing a nonce in the coinbase transaction propagates all the way up the Merkle tree.

Princeton Textbook

Mining – Win the Block

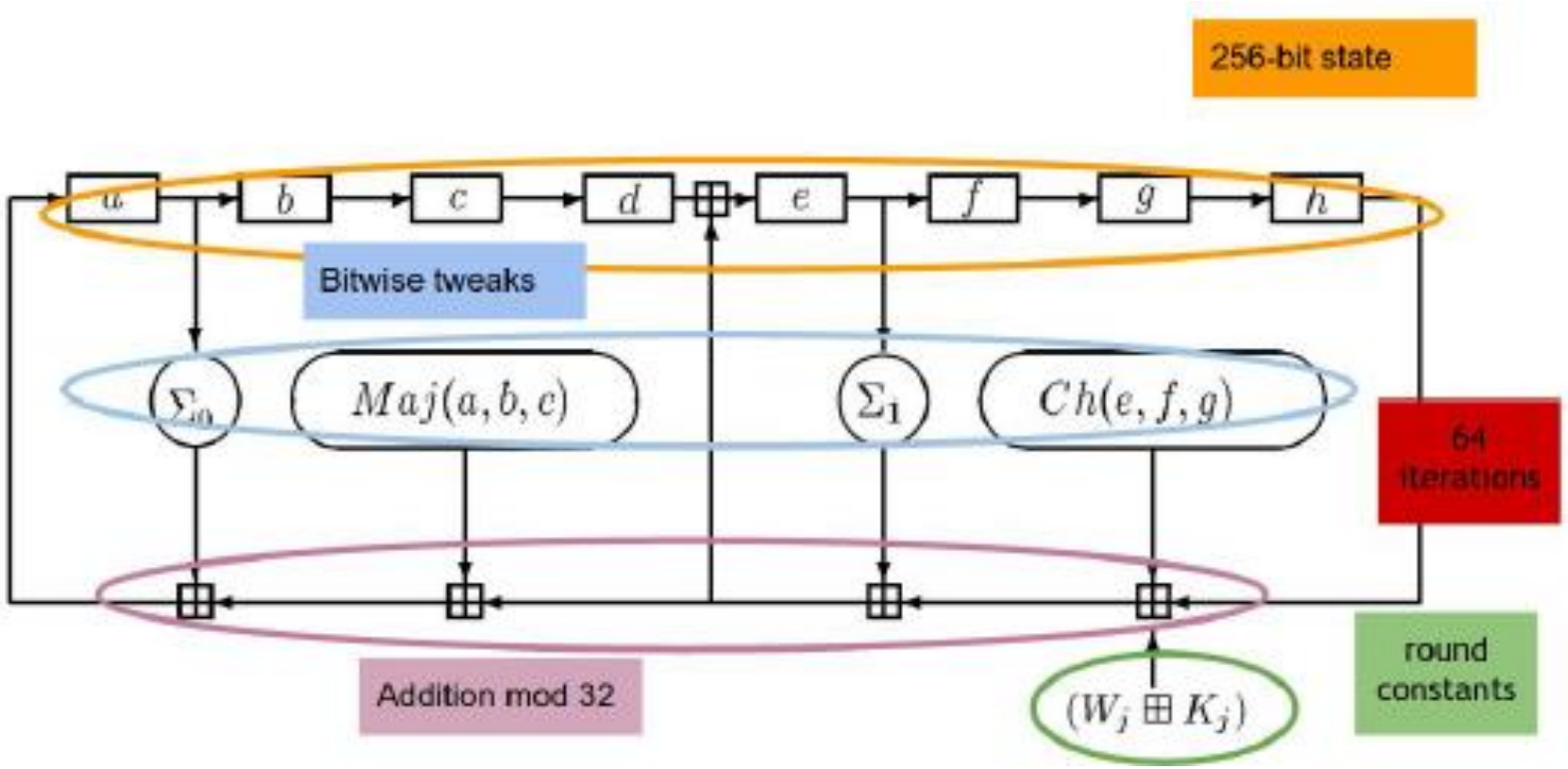
- When miner finds the nonce that works, they "win" the block
- They provide the nonce with block and everyone verifies
- Work is hard to solve but easy to verify





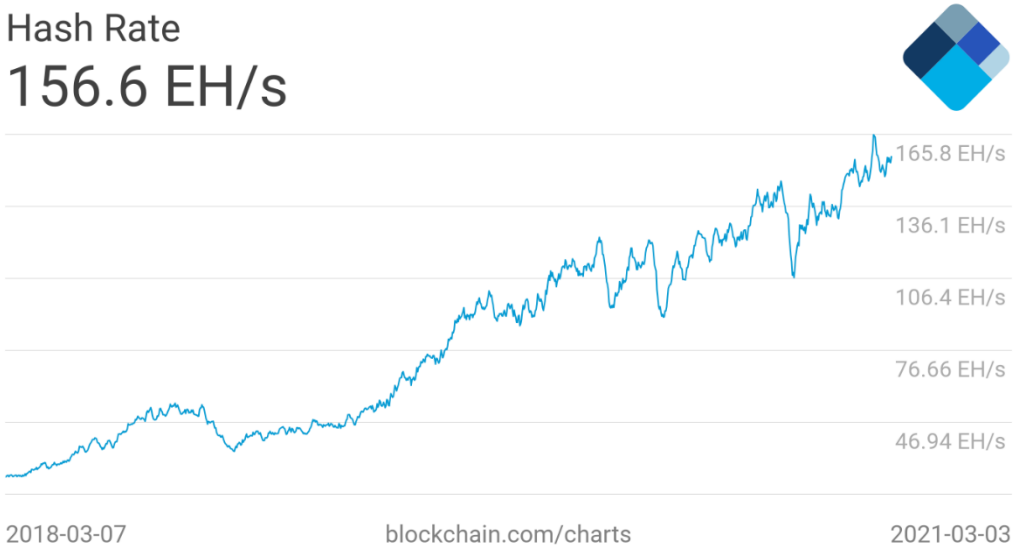
What is inside a Hash? Structure of SHA-256

This is one round of the compression function.
A complete computation of SHA-256 does this for 64 iterations.
During each round, different constants are applied so no iteration is the same.



Hash Rate of Bitcoin Network

Number of tera hashes per second (trillions of hashes per second) (blockchain.info)



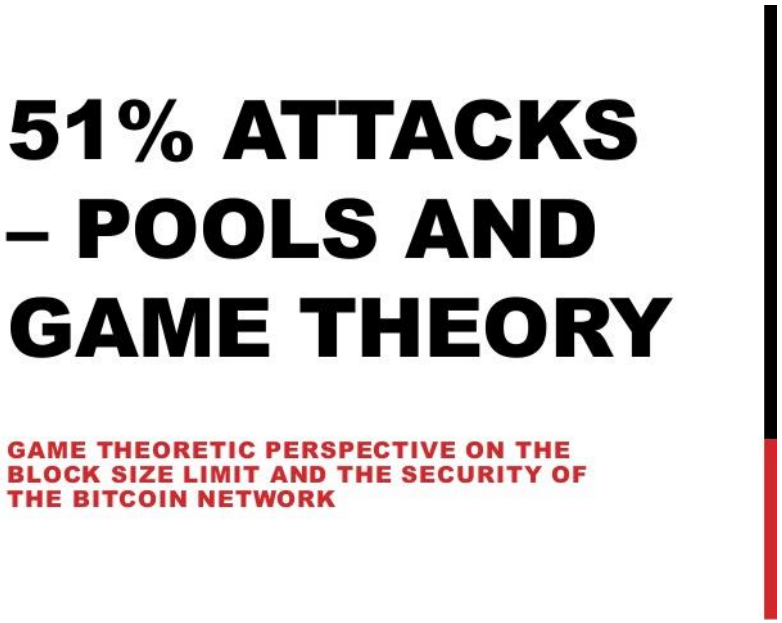
Sketch of Bitcoin Mining - 51% Attacks

Major assumption of Bitcoin:

No more than 51% percent of the network is dishonest

An honest majority will always form the longest proof-of-work chain

51% Attack: Attempt to overwhelm the mining power of the network



Source: Martin Koppelmann presenting at SF Bitcoin Devs

Sketch of Bitcoin Mining - Summary

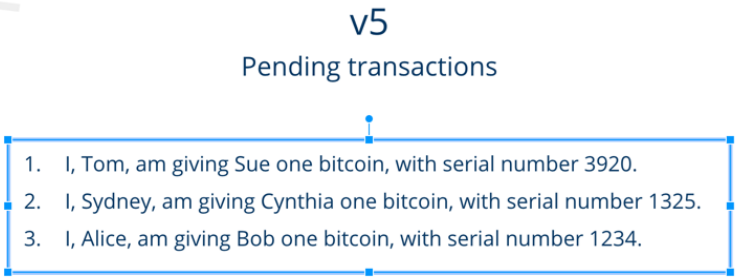
Functions as:

- A minting mechanism that ensures coins are distributed in a fair way
- An incentive for people to help secure the network
- Key component that enables you reach consensus in a decentralized currency
- Mining is similar to a lottery, except instead of buying lottery tickets, you contribute computational power

Bringing it all together - Back to a transaction

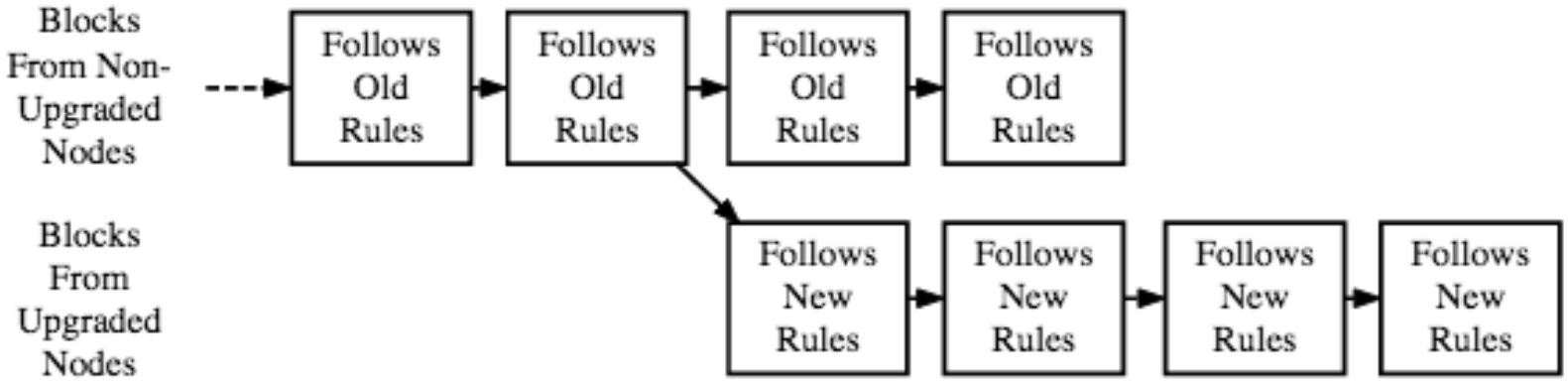
Slide by Viget

- I want to send money to Sunny
  - Sign transaction
  - Broadcast to network

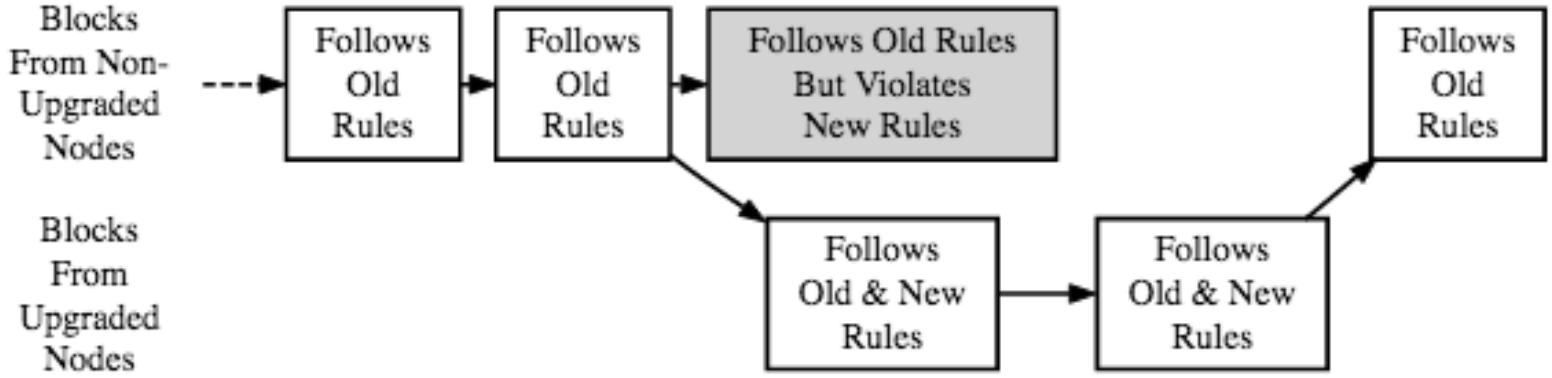


- Miners receives transaction, adds to “zero-conf pool”
  - Verify transaction: i.e. signature matches, enough money,
- Miner finds PoW, broadcasts block
  - Block propagates; others verify
- Miners work on the next problem

Forking + Consensus Updates

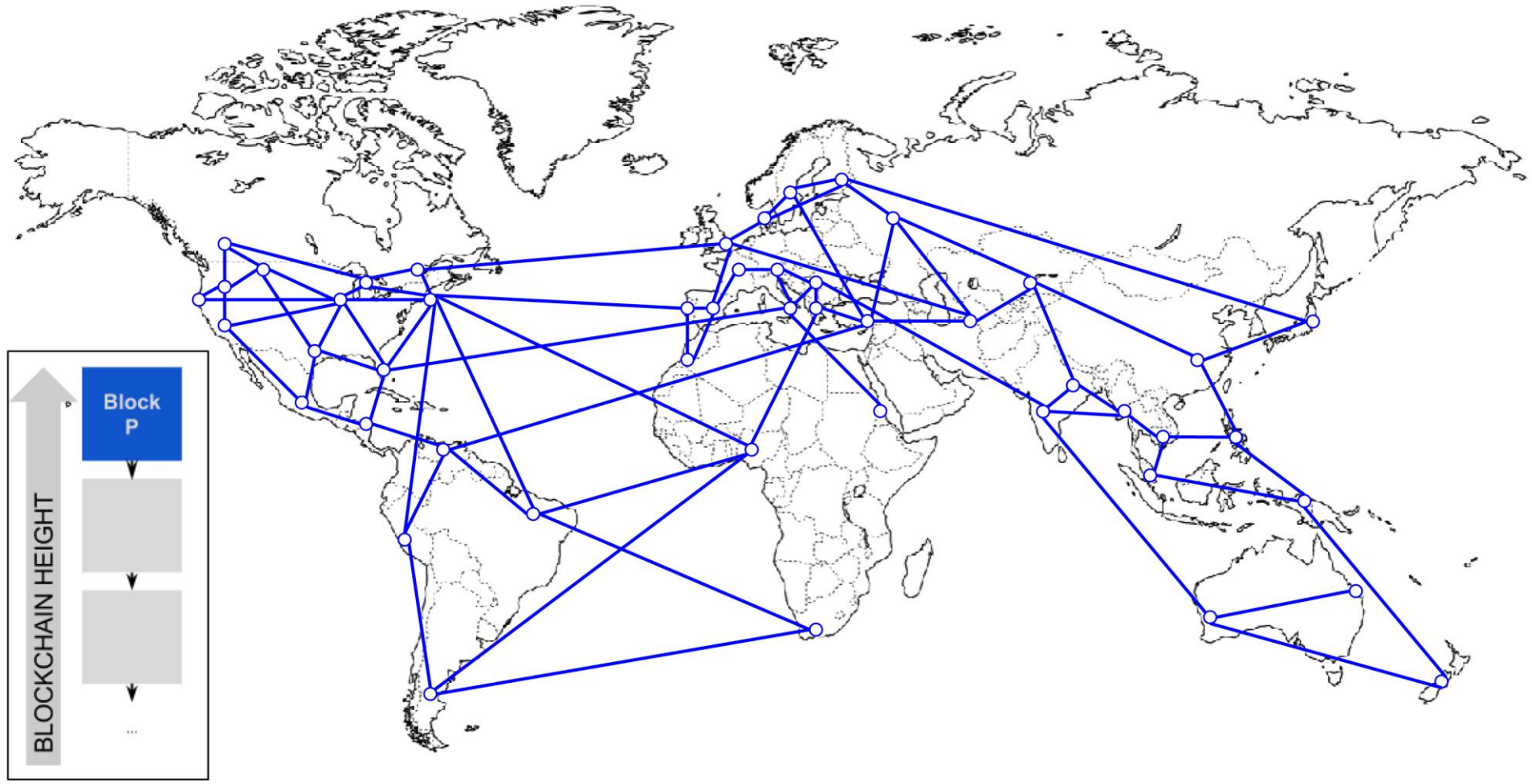


A Hard Fork: Non-Upgraded Nodes Reject The New Rules, Diverging The Chain

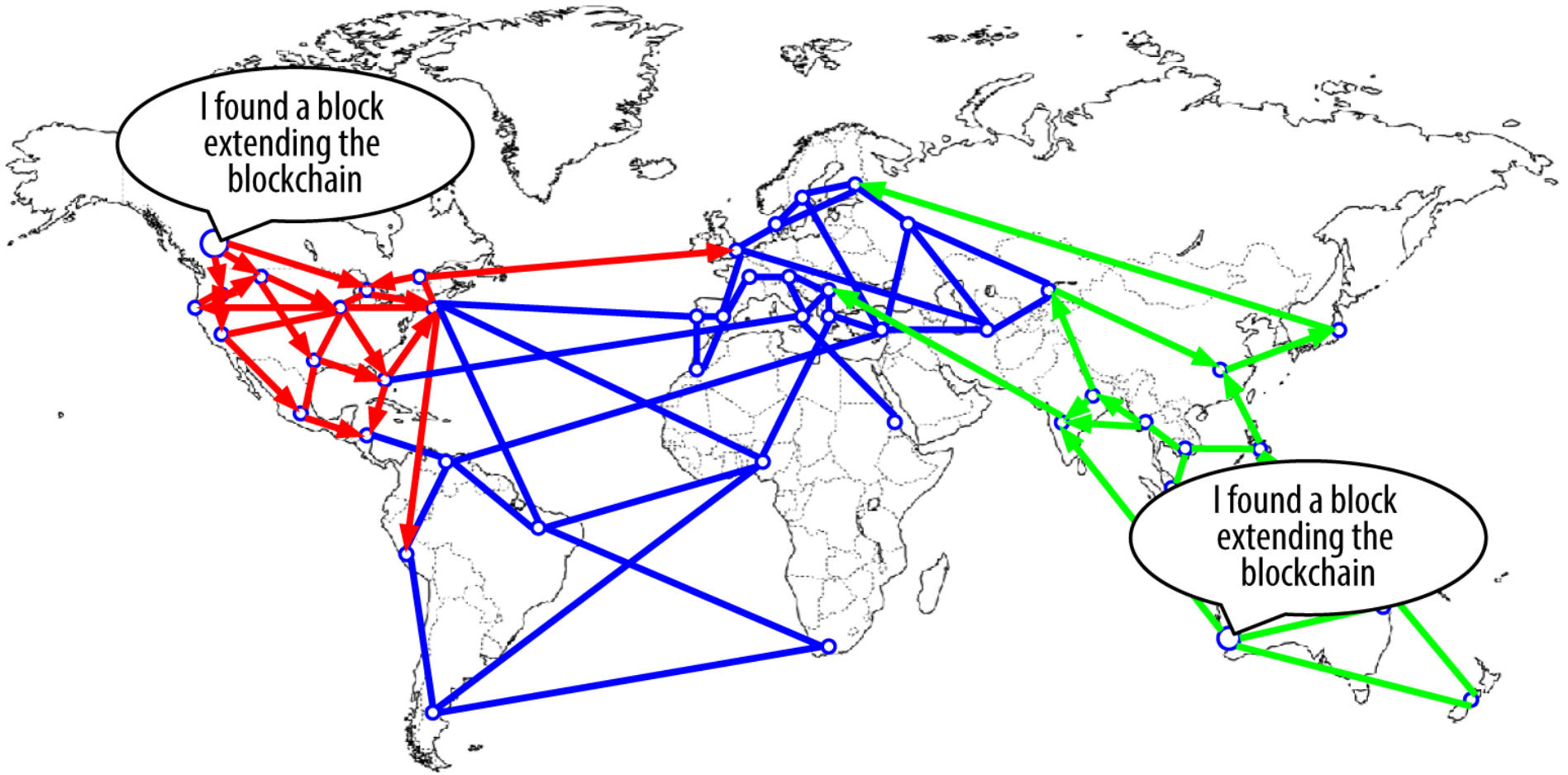


A Soft Fork: Blocks Violating New Rules Are Made Stale By The Upgraded Mining Majority

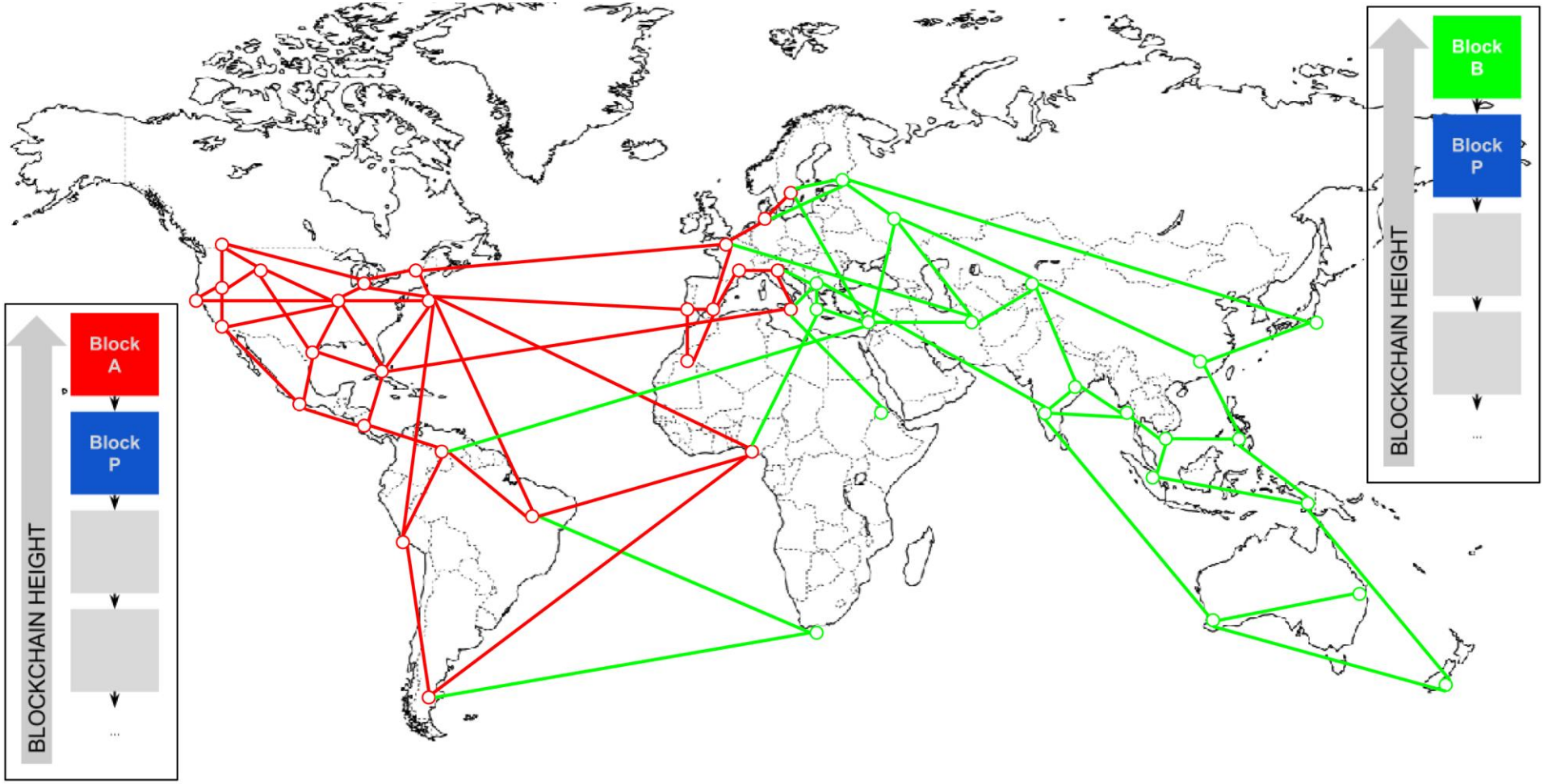
blockchain fork event—before the fork



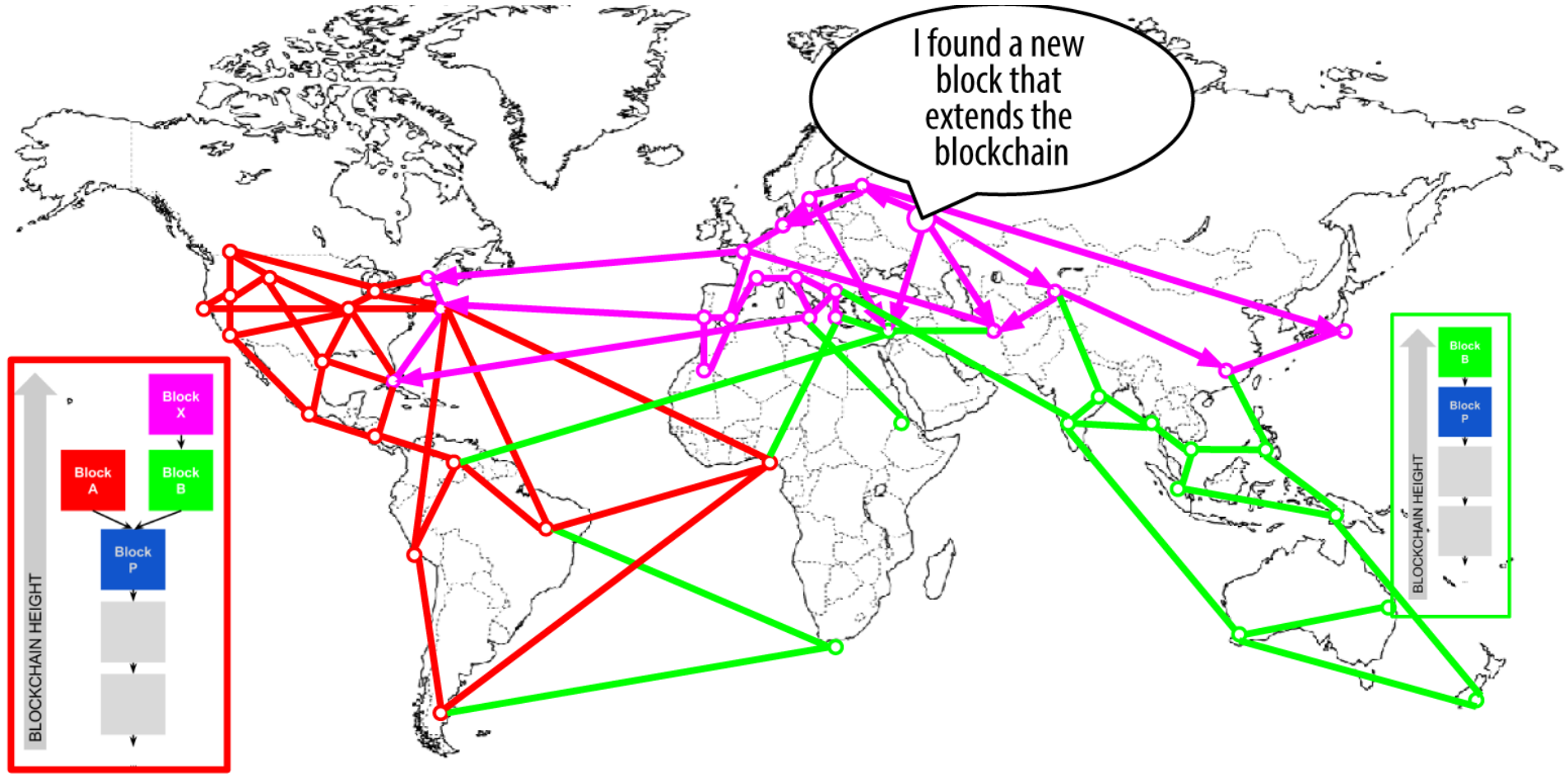
blockchain fork event—two blocks found simultaneously



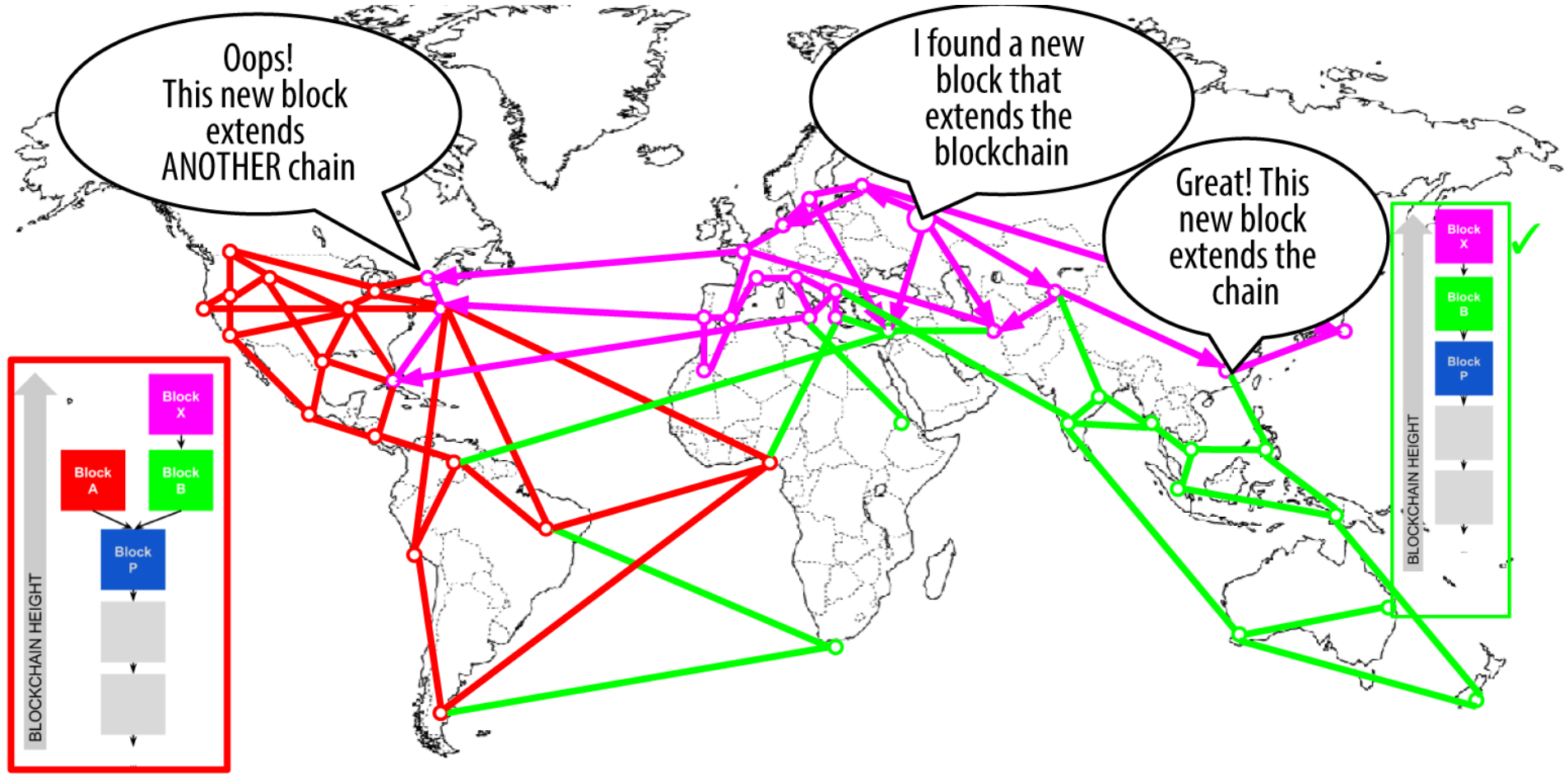
two blocks propagate, splitting the network



a new block extends one fork



the network reconverges on a new longest chain



Block Time – A Design Compromise

- It is possible for a fork to extend to two blocks, if two blocks are found almost simultaneously.
- Whereas a one-block fork might occur every week, a two-block fork is very rare.
- Bitcoin's block interval of 10 minutes is a design compromise between fast confirmation times (settlement of transactions) and the probability of a fork.
  - Faster block time makes transactions clear faster but lead to more frequent blockchain forks
  - Slower block time decreases the number of forks but make settlement slower

SPV - Simplified Payment Verification

Current size of Bitcoin blockchain: 205 GB and growing

- **2 times of 2 years ago**
- **Storing the full Bitcoin blockchain will likely not be feasible for the average user, so how do we address this?**

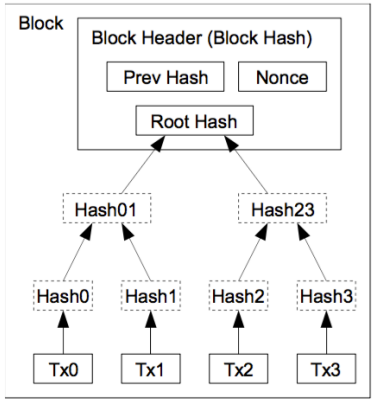
SPV - Simplified Payment Verification

- **"SPV nodes" or "thin" client, as opposed to "full nodes"**
  - Don't store the full blockchain
  - Only store the pieces of data needed to verify transactions that concern them
- **Nearly all nodes on network are SPV nodes, ex wallet software**

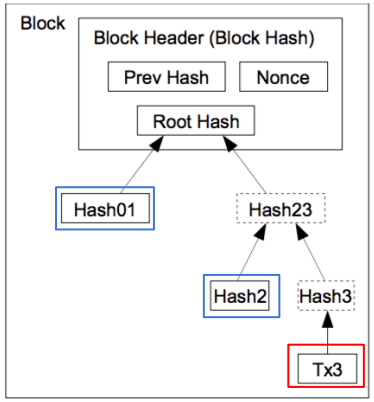
SPV - Structure

SPV only keep block headers

- Obtained by querying different full nodes until the node is convinced it has the longest chain
- block headers have PoW info stored - very hard to create



Transactions Hashed in a Merkle Tree

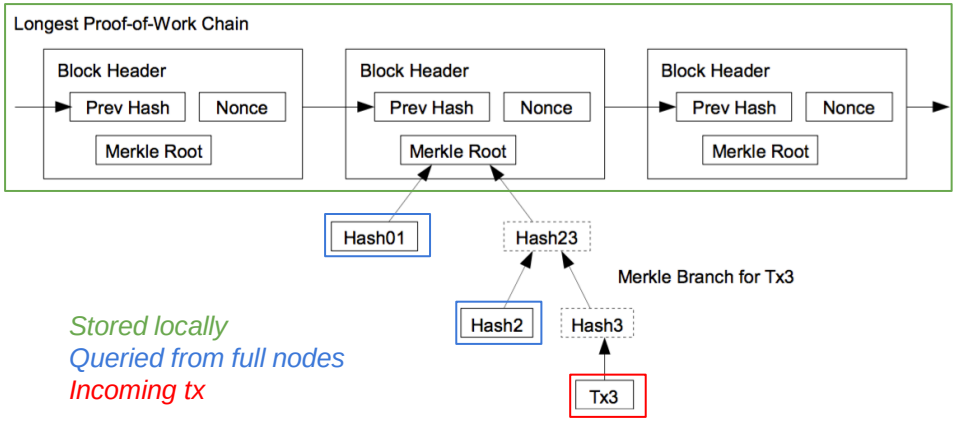


After Pruning Tx0-2 from the Block

Nakamoto, 2009

How to validate an incoming tx?

- Query full nodes to get the Merkle branch for that transaction
- Hash tx together with intermediate hashes to obtain a Merkle root, check that it matches the one saved locally.
- Wait for this tx to have enough confirmations before delivering good



SPV - Security Analysis

SPV nodes:

- **Don't have full tx history, don't know UTXO set**
- **Don't have same level of security of full nodes**
  - Can't check if every tx included in a block is actually valid

SPV nodes assume:

- **...that incoming block headers aren't a false chain**
  - Very expensive for attacks (or anyone) to create blocks
  - Not sustainable over the long term
- **...that there ARE full nodes out there validating all transactions**
  - There are efficiency benefits and incentives to doing so
- **...that miners ensure that the transactions they include in their blocks are valid**
  - Otherwise their blocks would be rejected by full nodes (very expensive mistake!)

SPV - Cost savings

Huge cost savings

- **Block headers are only ~1/1000 the size of the full blockchain**
  - 83 MB vs. 331 GB (up from 266G a year ago)

SPV nodes obtain data for verifying transactions lazily

- **Could be an issue for big merchants who must verify many txs**

For most consumers and users of Bitcoin, SPV is a decent tradeoff.

IPFS - Interplanetary File Systems

IPFS

IPFS is a decentralized model for file transfer in contrast to the centralized namespace and transfer provided by the http family of protocols.

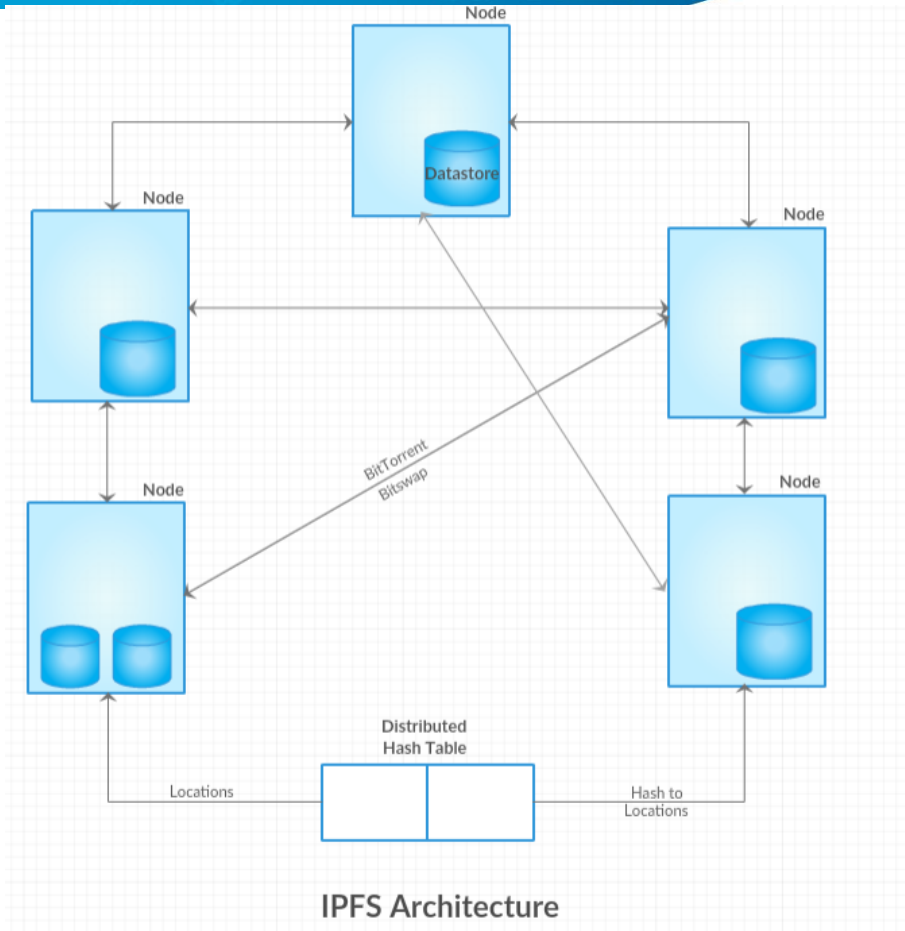
HTTP operates in a centralized, hierarchical namespace: a web site is identified by http://www.ust.hk where each item is resolved hierarchically by the domain name service (DNS).

Examples of peer-peer transfer of data include Napster and Gnutella media sharing and bittorrent services that is underlying many of our current data services.

Juan Benet in the IPFS white paper describes it as a "Content Addressed, Versioned, P2P Filesystem".

IPFS leverages many successful p2p ideas

- Global distributed file system: IPFS is about “distribution” decentralization.
- Content-based identification with secure hash of contents - Resolving locations using Distributed Hash Table (DHT)
- Block exchanges using popular Bittorrent peer-to-peer file distribution protocol
- Incentivized block exchange using Bitswap protocol
- Merkle DAG (Directed Acyclic Graph) version-based organization of files, similar to Git version control system
- Self-certification servers for the storage nodes for security



IPFS Architecture

Files in distributed storage, and distributed hash table, uses the hash of the file as a key to return the location of the file. Once the location is determined, the transfer takes place p2p as a decentralized transfer.

Distributed Nodes

Nodes are identified by cryptographic hashes of public key. (Similar to bitcoin blockchain nodes).

They hold the objects that form the files to be exchanged. Objects are identified by a secure hash and an object may contain sub objects each with its own hash that is used in the creation of the root hash of the object. Like Merkle tree.

Content Addressable

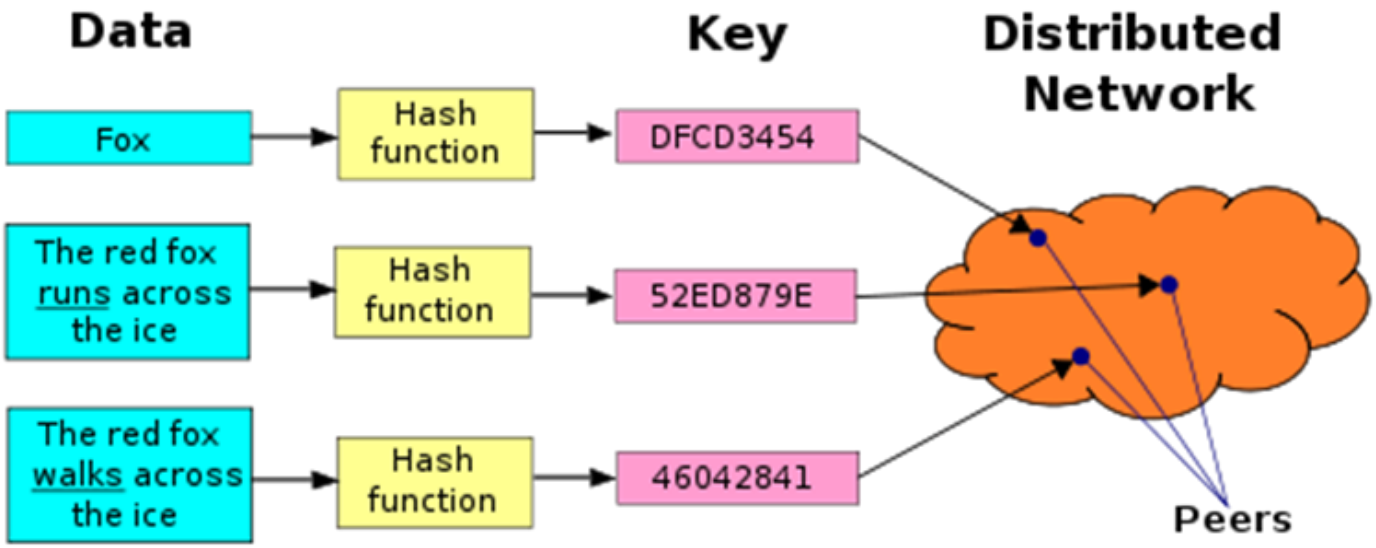
IPFS identifies the resources by secure hash of its content - universally unique identifier , instead of location as in HTTP.

How to resolve the location? start with the hash identifier of the resource. Send around a request for anyone with a resource with this identifier; on a successful response, access it peer-to-peer.

How to resolve location of objects?

Routing part of the IPFS protocol maintains a DHT (Distributed Hash Table) for the locating the nodes as well as for file objects.

A simple DHT hold the hash as the key and location as the value. Key can directly hash into the location.



Exchange the blocks of the file - BitSwap

DHT resolves to the closest location to the key value. The peer nodes holding the data blocks are incentivized by a protocol called BitSwap.

Peer nodes have a want_list and have_list and some form of a barter system is formed.

Any imbalance is noted in form of a BitSwap credit and debt; Bitswap protocol manages the block exchanges involving the nodes accordingly.

The nodes in the network thus have to provide value in the form of blocks - ideal use case for a “digital token?”; if you send a block you get a IPFS token that can be used when you need a block.

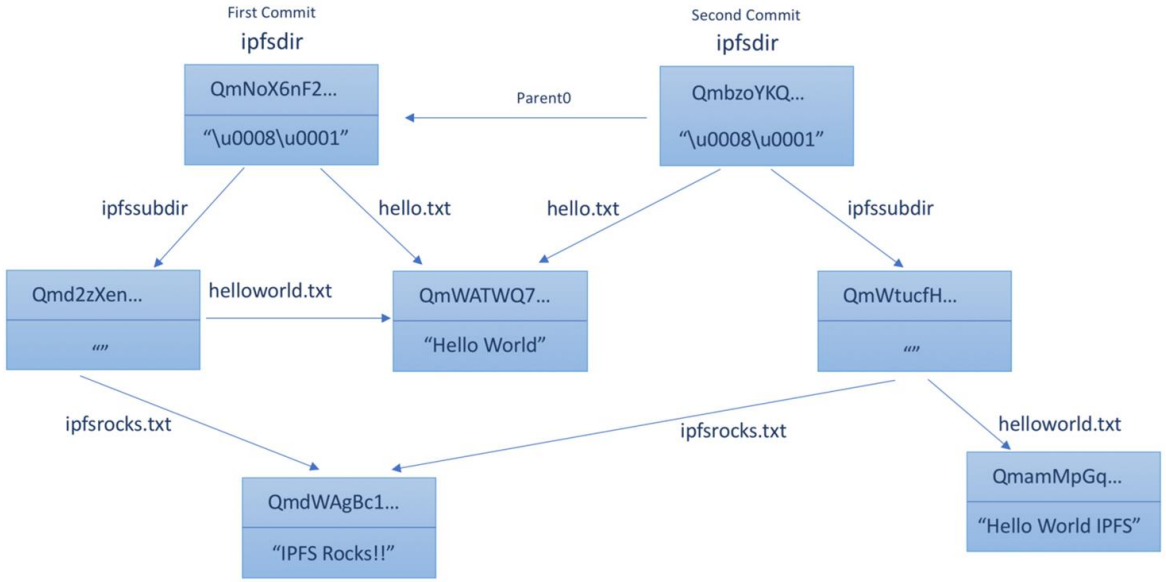
The Bitswap protocol has provisions for handling exceptions such as free loading node, node wanting nothing, node having nothing.

Merkle DAG and Deduplication

Multiple versions of a file are maintained using a Merkle Directed Acyclic Graph data structure on top of the file system.

The basic elements of the block, list of blocks, tree of block representing an instance, and commit that is snapshot of the tree.

This Merkle DAG helps in checking any tampering and deduplication.

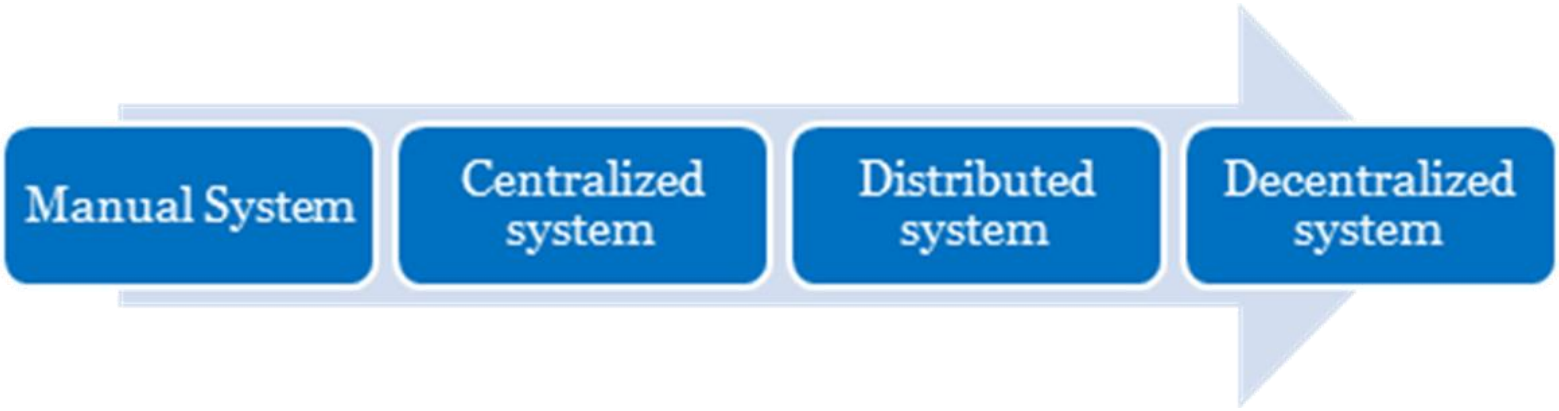


Two commits of the file directories: four nodes on the left form the first commit and three nodes on the right the second commit

Motivation for FileCoin

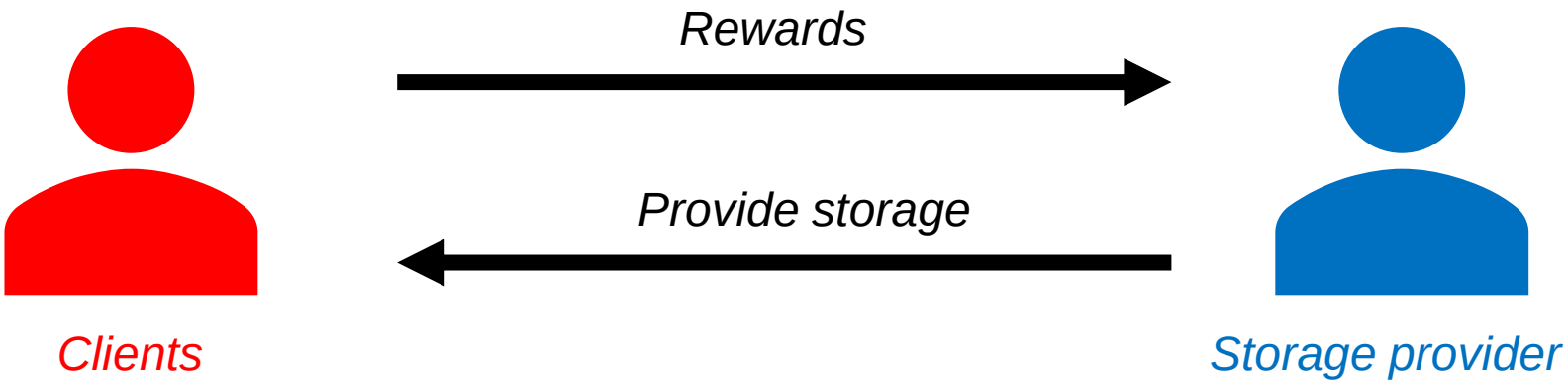
IPFS can serve an important role of decentralized storage for blockchain application that have a lot of data, but will store only the hash on the blockchain.

Instead of a centralized store, IPFS can be the decentralized store that work in tandem with the decentralized ledger technology of the blockchain to create a powerful solution for many storage-rich business use cases.



Motivation for FileCoin

- Decentralized storage network (runs on a blockchain system)
- Miners earn rewards by providing storage to clients



- User-managed storage network
- Robust
- Secure (e.g. end-to-end encryption)

- ”Incentive layer” on top of IPFS → constantly expanding/regenerating network

Main Components of Filecoin

1. **Decentralized Storage Network (DSN)**
   - network of independent storage providers to offer storage and retrieval services
2. **Proofs of Storage (PoS)**
   - Proof of Replication
   - Proof of Spacetime
3. **Verifiable Markets**
   - Store and process storage requests and retrieval requests
   - Storage Market + Retrieval Market
4. **Proof of Work (PoW)**
   - Miners store “useful” data to run consensus protocols

Proof of Replication & Proof of Spacetime

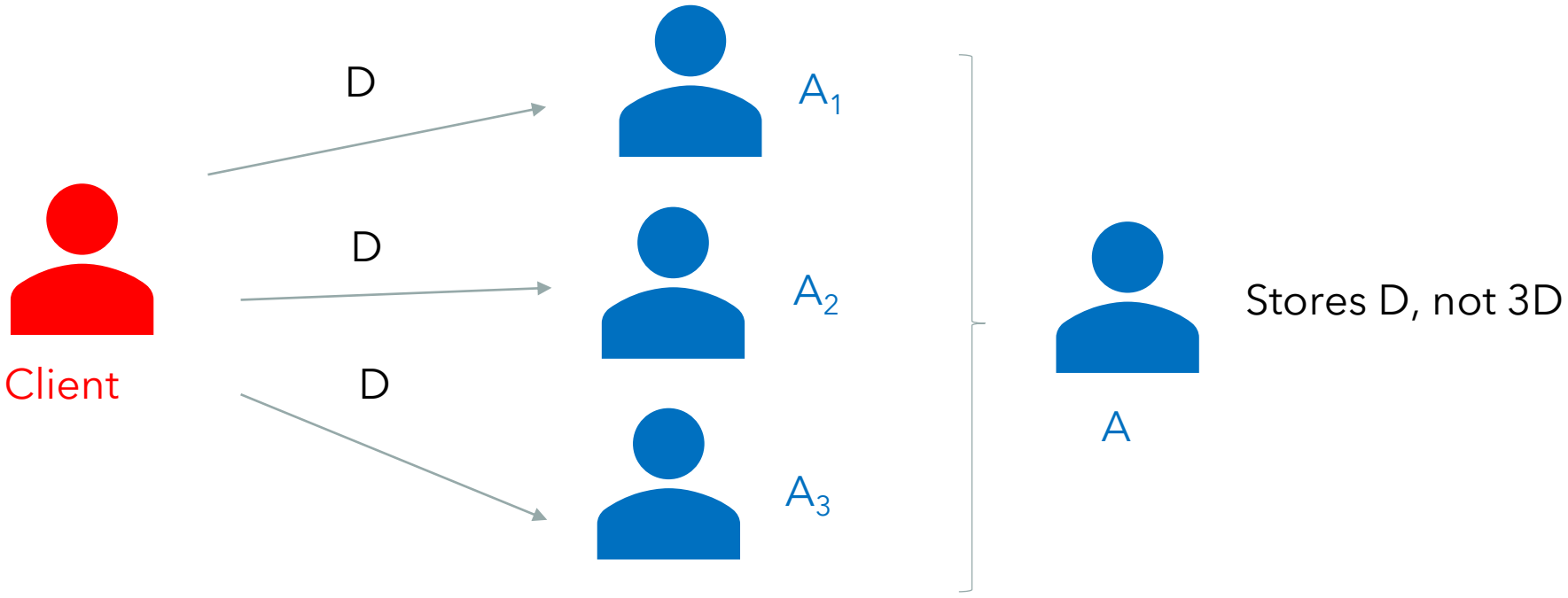
- Proof of Storage (PoS) definition: “allow a user who distributes data D to a server (i.e. the prover P) to repeatedly check if the server is still storing D.”
- PoS can only guarantee that the prover P is storing data AT THE TIME OF THE CHALLENGE

We do not know if the prover P is honestly storing data ALL THE TIME!!! (i.e. when the challenge is not issued)

Therefore, we need STRONGER proof schemes to prevent malicious behaviour.

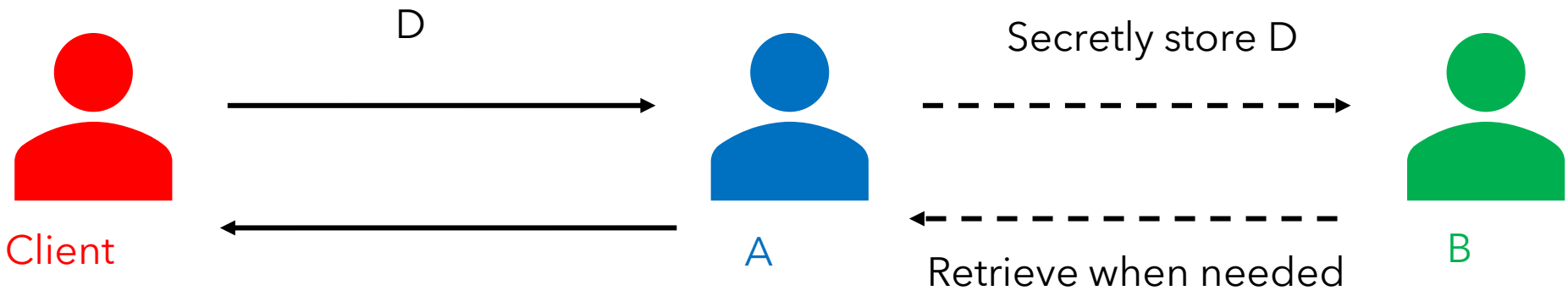
Types of malicious attacks

- **Sybil Attacks:** creation of multiple Sybil identities



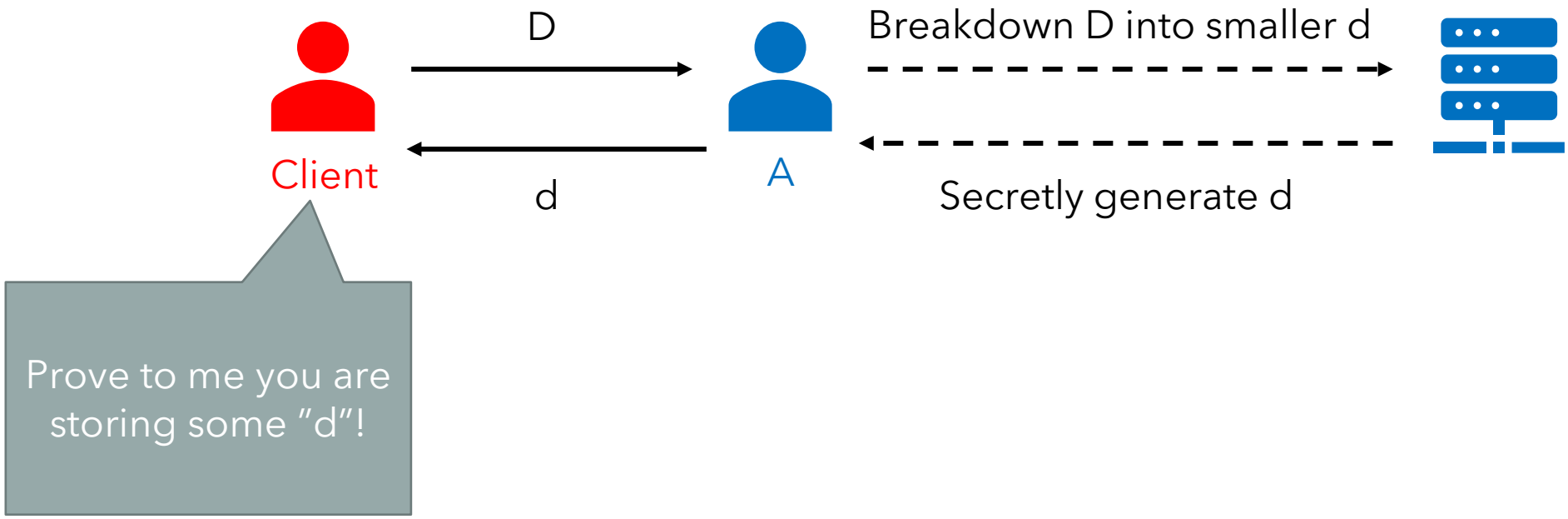
Types of malicious attacks

- **Outsourcing Attacks:** fetch data from other storage providers



Types of malicious attacks

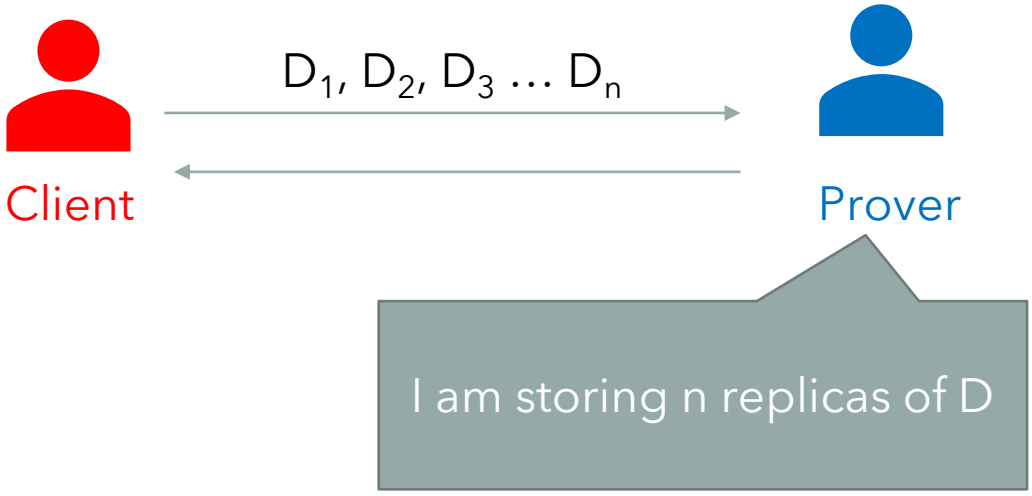
- **Generation Attacks:** generating data on-demand



Proof of Replication

"allows a server (i.e. the prover P) to convince a user (i.e. the verifier V) that some data D has been replicated to its own uniquely dedicated physical storage"

store n distinct replicas (physically independent copies) of data D



Sybil attacks
Outsourcing attacks
Generation attacks

Proof of Spacetime

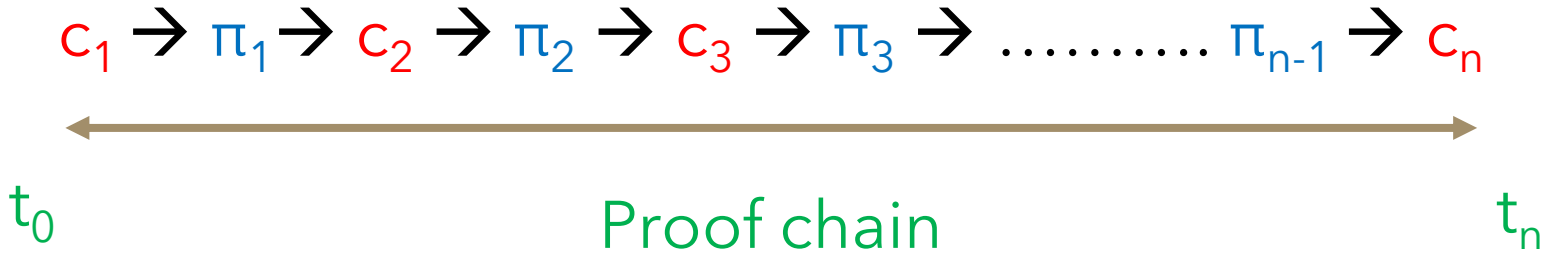
"Allows a verifier to check if a prover is storing his data for a range of time"

1) generate sequential Proofs-of-Replication, as a way to determine time

2) recursively compose the executions to generate a short proof

How does it work:

A verifier can construct a series of challenges and proofs so that the **challenge at iteration n (c_n) can be derived deterministically from the proof at iteration n-1 (π_{n-1}).**



Useful Work Consensus

- PoW schemes are non-reusable and wasteful
  - Non reusable: only has one purpose – to secure the blockchain
  - Wasteful: expensive in machinery & energy consumption

Proof schemes that are

- Reusable: e.g. Primecoin – find new prime numbers
- Non-wasteful e.g. Spacemint – require users to store useful data instead of computational power

Filecoin's Useful Work Consensus

- "the probability that the network elects a miner to create a new block (**voting power** of the miner) is proportional to their storage currently in use in relation to the rest of the network"
  - Greater the storage → more likely to create/mine a new block → more rewards
  - Motivates users to invest in storage space rather than computational power

Benefits:

- Miners sacrifice storage space to participate in consensus
- DSN storage expands ("useful work")

Network power & Expected Consensus

- Upon the creation of every block → miners submit PoSt → added to blockchain if majority of network power considers them valid

Using power to achieve consensus: **Expected Consensus (EC)**

1. Elect a leader at a particular epoch
   - Probability of winning this "election" is proportional to a miner's storage
2. At each epoch, the chain is extended & leaders distribute the new chain to the network
   - If there is a leaderless epoch, an empty block is added instead
3. Each epoch introduces more certainty of the existing chain

Expected Consensus

A miner can be chosen as leader given that they satisfy:

- H – secure cryptographic hash function
- t – epoch/timepoint
- rand(t) – public randomizer variable
- M_i – miner
- L – size of cryptographic hash function
- ${}^{pt}_{i}$ – power of miner M_i at time t
- $\sum jp^t{}_j$ = total network power at time t

$$\mathcal{H}\left(\langle t||\text{rand}(t)\rangle_{\mathcal{M}_i}\right)/2^L \leq \frac{p_i^t}{\sum_j p_j^t}$$

Thank you