Blockchain: Smart Contracts

Lecture 3-4

What is Blockchain?

- A Platform for executing transactional services
- Spanned over multiple organizations or individuals who may not (need not) **trust** each other
- An append-only shared ledger of digitally signed and encrypted transactions replicated across a network of peer nodes

The Block in a Blockchain – Securing Data Cryptographically

- Digitally signed and encrypted transactions "**verified" by peers**
- **Cryptographic security** – Ensures that participants can only view information on the ledger that they are authorized to see

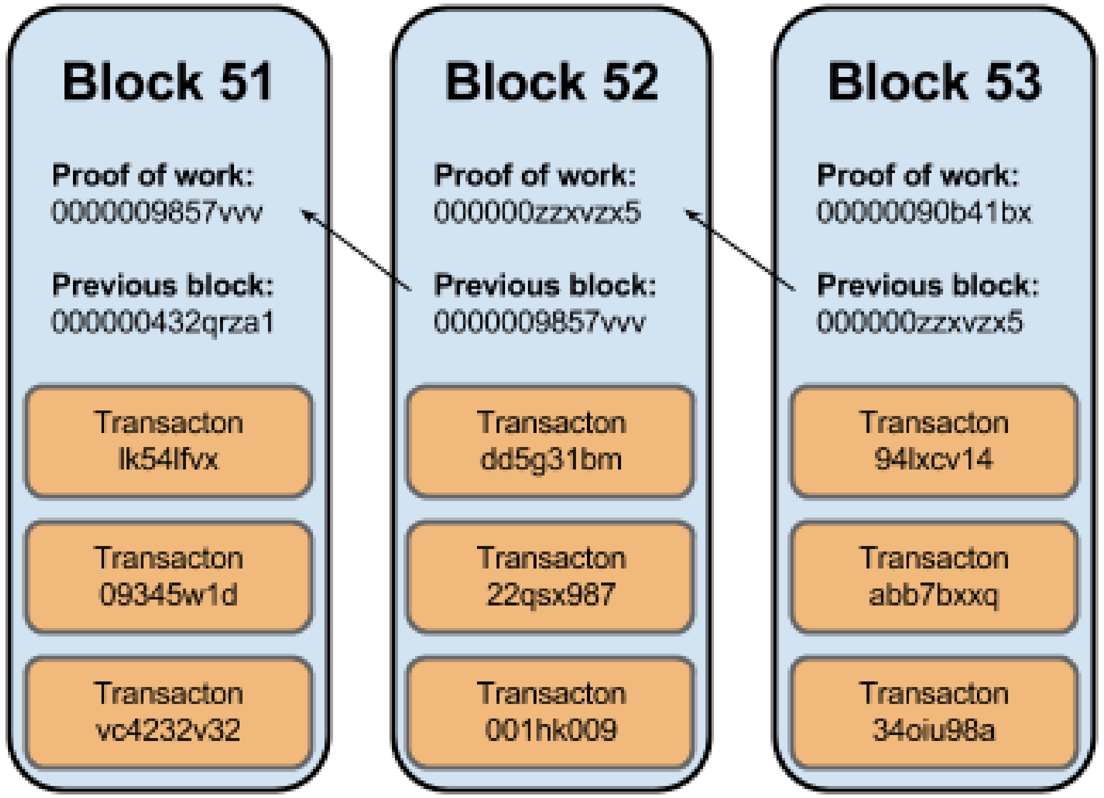


Image source: http://dataconomy.com/

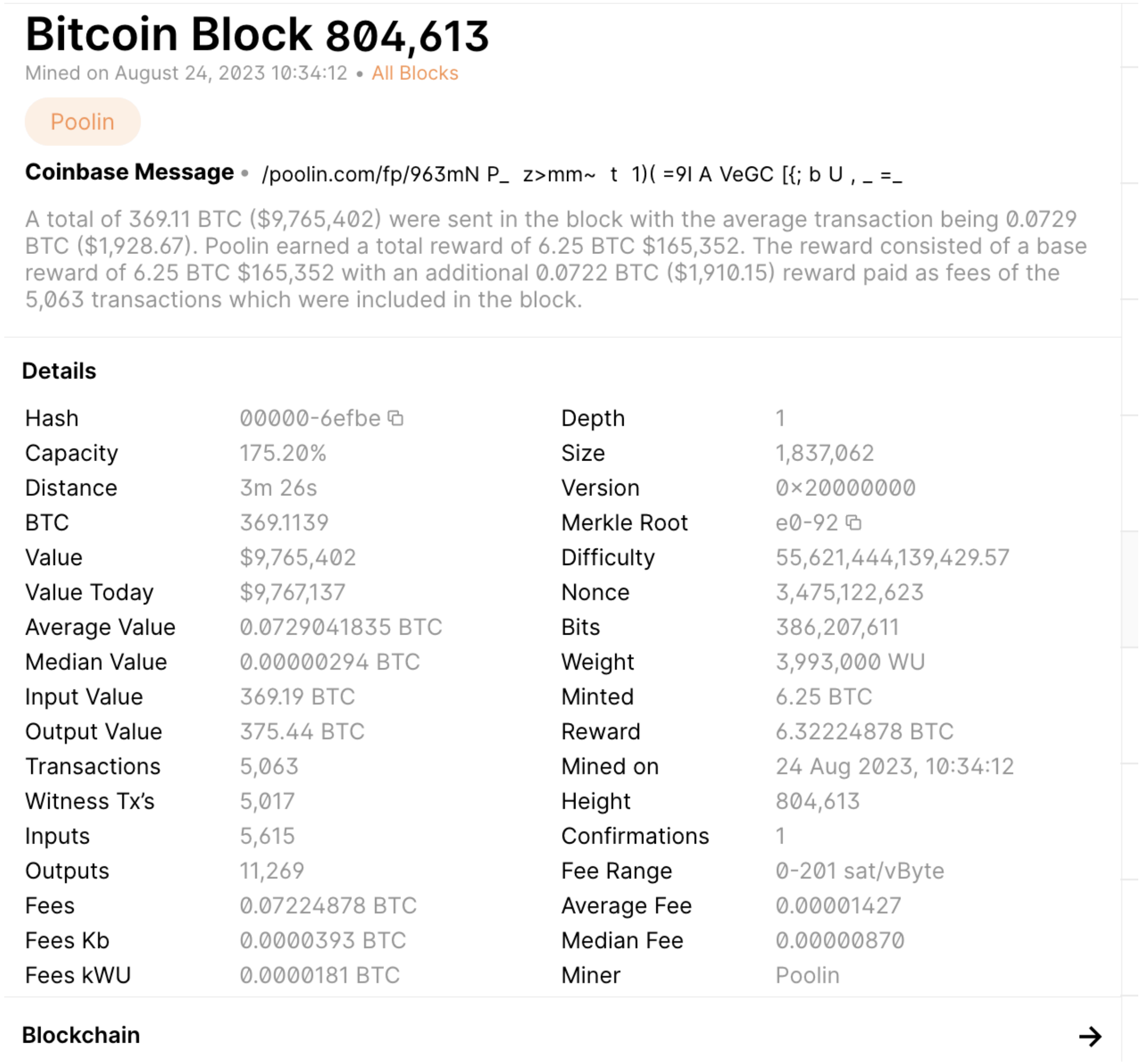
Structure of a Block

- A block is a **container data structure** that contains a series of transactions

- **In Bitcoin:** A block may contain more than 500 transactions on average, the average size of a block is around 1 MB (an upper bound proposed by Satoshi Nakamoto in 2010)
  - May grow up to 8 MB or sometime higher (several conflicting views on this!!)
  - Larger blocks can help in processing large number of transactions in one go.
  - But longer time for verification and propagation

Structure of a Block (Reference: Bitcoin)

- Two components:
  - **Block Header**
  - **List of Transactions**



Bitcoin Block 804,613

Mined on August 24, 2023 10:34:12 • All Blocks

Poolin

Coinbase Message • /poolin.com/fp/963mN P_ z>mm~ t 1)(=9I A VeGC [{; b U , _ =_

A total of 369.11 BTC ($9,765,402) were sent in the block with the average transaction being 0.0729 BTC ($1,928.67). Poolin earned a total reward of 6.25 BTC $165,352. The reward consisted of a base reward of 6.25 BTC $165,352 with an additional 0.0722 BTC ($1,910.15) reward paid as fees of the 5,063 transactions which were included in the block.

Details

Hash	00000-6efbe	Depth	1
Capacity	175.20%	Size	1,837,062
Distance	3m 26s	Version	0x20000000
BTC	369.1139	Merkle Root	e0-92
Value	$9,765,402	Difficulty	55,621,444,139,429.57
Value Today	$9,767,137	Nonce	3,475,122,623
Average Value	0.0729041835 BTC	Bits	386,207,611
Median Value	0.00000294 BTC	Weight	3,993,000 WU
Input Value	369.19 BTC	Minted	6.25 BTC
Output Value	375.44 BTC	Reward	6.32224878 BTC
Transactions	5,063	Mined on	24 Aug 2023, 10:34:12
Witness Tx's	5,017	Height	804,613
Inputs	5,615	Confirmations	1
Outputs	11,269	Fee Range	0-201 sat/vByte
Fees	0.07224878 BTC	Average Fee	0.00001427
Fees Kb	0.0000393 BTC	Median Fee	0.00000870
Fees kWU	0.0000181 BTC	Miner	Poolin

Blockchain →

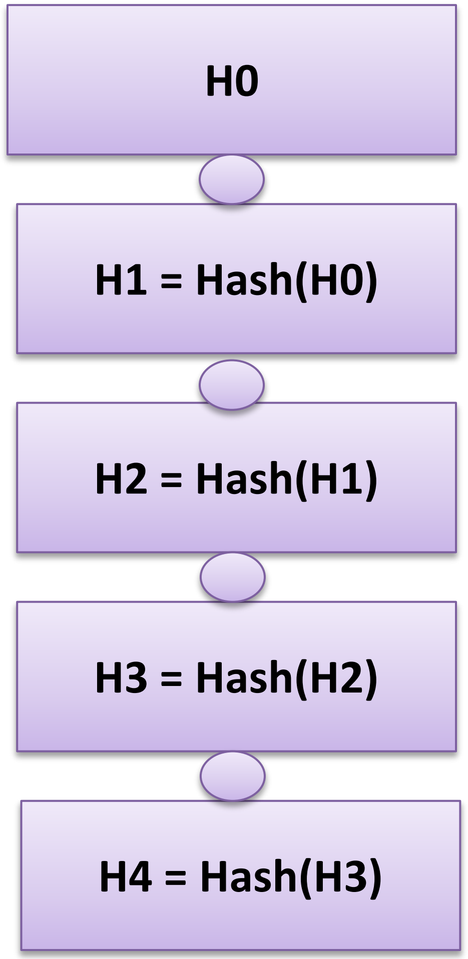
Block Source: https://www.blockchain.com/explorer/blocks/btc/804613

Block Header (Reference: Bitcoin)

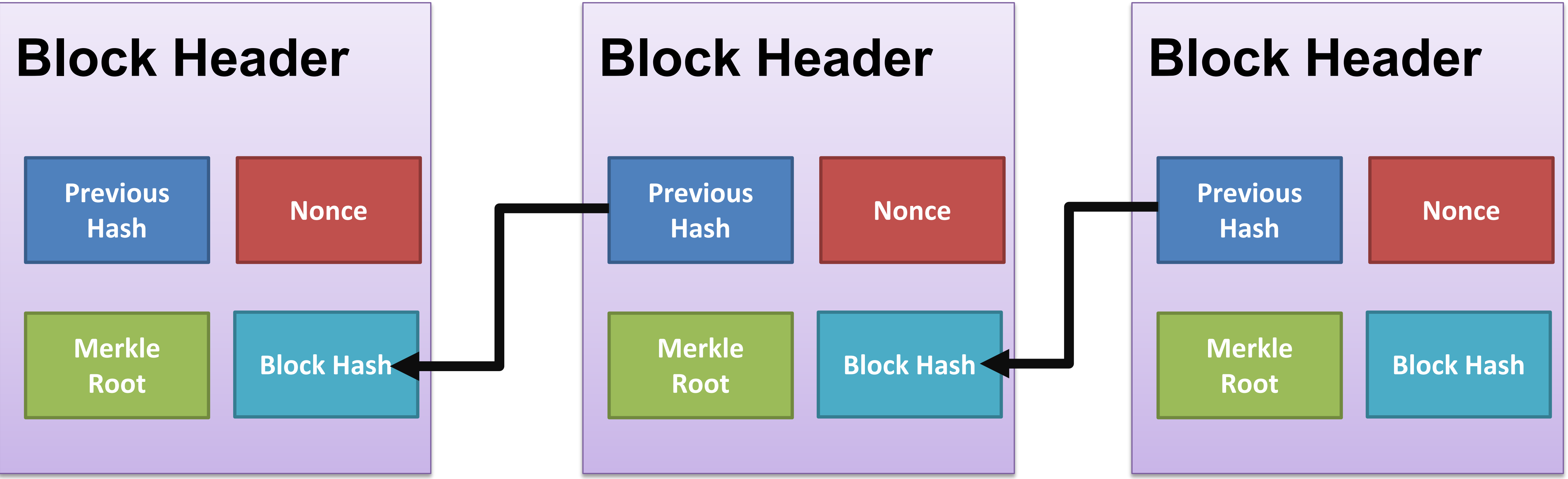
- Metadata about a block –

1. Previous block hash,
2. Mining statistics used to construct the block,
3. Merkle tree root

- **Previous block hash:** Every block inherits from the previous block – we use previous block's hash to create the new block's hash – make the blockchain **tamper proof.**



Block Generation Puzzle



- **Find out the nonce which generates the desired hash (certain zero bits at the prefix - 0000000000000000004a2b84f93a285b7a7………)**

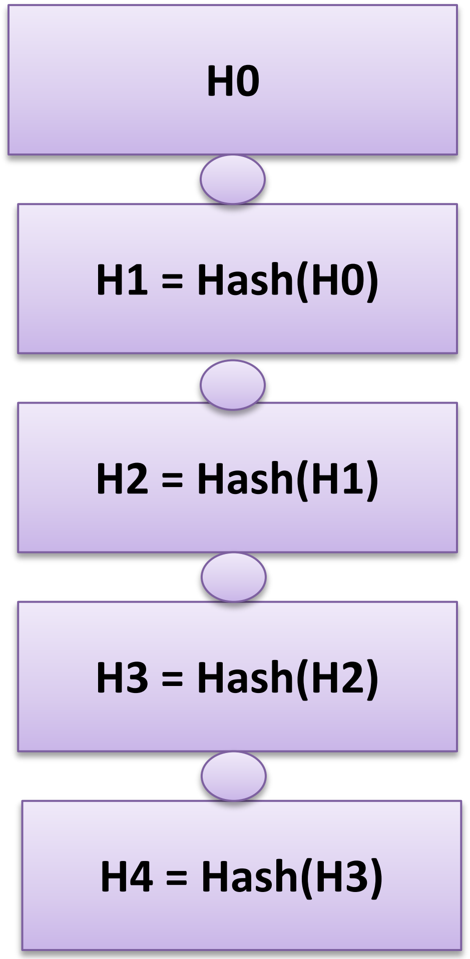
Block Header (Reference: Bitcoin)

- **Mining** – the mechanism to generate the hash
  - The mechanism needs to be complicated enough, to make the blockchain **tamper proof**
  - **Bitcoin Mining:** $H_k = Hash(H_{k-1} \,||\, T \,||\, Nonce \,||\, \textit{Something more})$
  - Find the nonce such that H_k has certain predefined **complexity** (number of zeros at the prefix)

- The header contains mining statistics – timestamp, nonce and difficulty
- Understanding Difficulty and Bits https://medium.com/@dongha.sohn/bitcoin-6-target-and-difficulty-ee3bc9cc5962

- Difficulty is the largest possible target (0x00000000FFFF0000000000000000000000000000000000000000000000000000) divided by the current target , e.g., (0x00000000000000000012180B00000000000000000000071522A353A529800000).

- Remember: "Cost of Mining" – Pretty High (Computing Power and Energy)



The Hashes in a Block Header (Reference: Bitcoin)

- Block identifier – the hash of the current block header (Hash algorithm: Double SHA256)
- Merkle Root
- Previous block hash is used to compute the current block hash
- Timestamp, Previous hash, Merkle root, Difficulty Bits, Nonce and Version used to compute current hash

Block Source: https://www.blockchain.com/explorer/blocks/btc

Block Generation Cost

- Energy efficiency ~0.098 J/GH = ~100 J/TH
- ASIC Hardware for bitcoin can perform about 750 TH/s
- Hash rate of the Bitcoin network ~400M TH/s (as of 24 August 2023)!! Many actually go waste ☹ https://www.blockchain.com/charts/hash-rate
- Bitcoin network consumes about 80 TW-hours of electricity annually. These figures vary between sources and are all some form of estimates
- Average household in Germany of four people consumes approx. 4,000 KW-hours of electricity per year.
- Can power about 20,000 households
- Concept of Pooling is used
- What ensures tamperproof operation in terms of honest nodes

The Blockchain Replicas

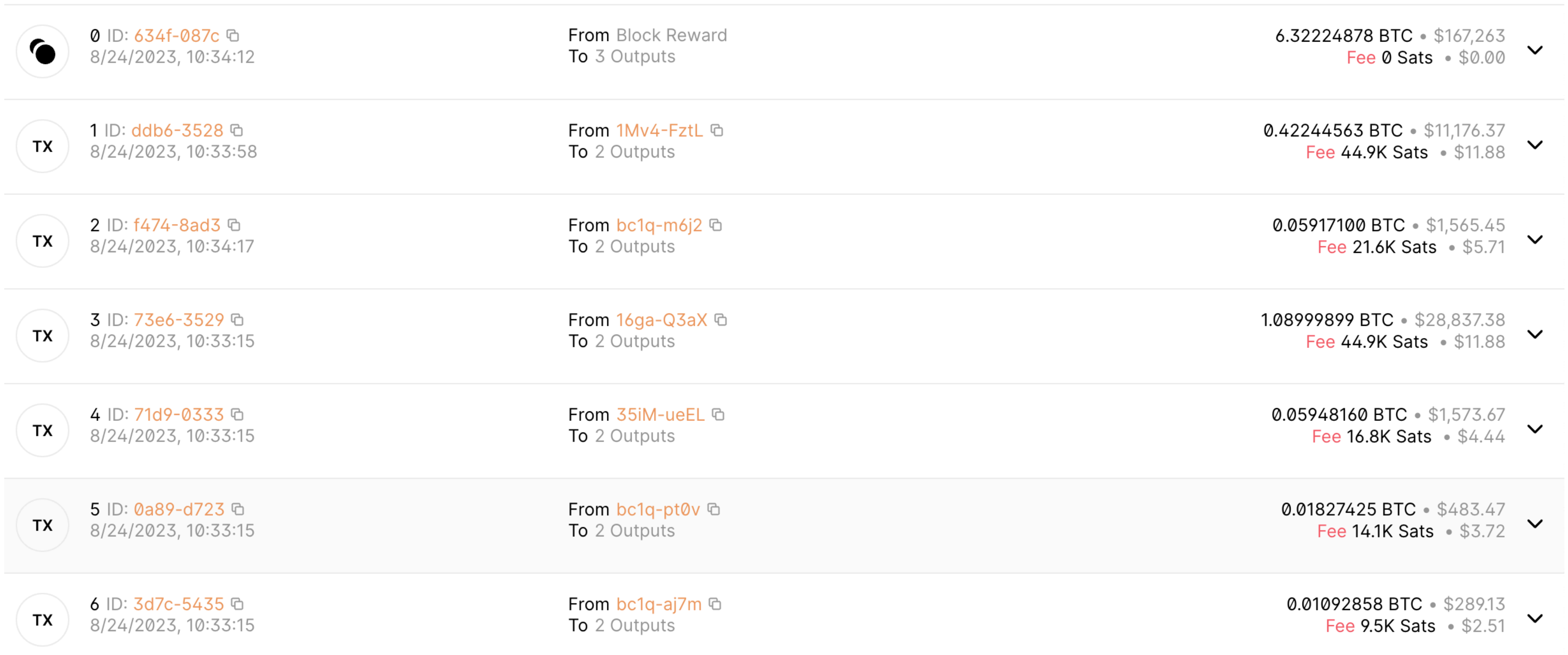
- Every peer in a Blockchain network maintains a local copy of the Blockchain.
- Size is just about 505.64 GB for Aug 23 2023 ☺
  https://www.blockchain.com/explorer/charts/blocks-size
- As a new user joins the network, they can get the whole copy

- **Requirements**
  - All the replicas need to be **updated** with the last mined block
  - All the replicas need to be **consistent** – the copies of the Blockchain at different peers need to be **exactly similar**

Transactions in a Block (Reference: Bitcoin)

- Transactions are organized as a Merkle Tree. The Merkle Root is used to construct the block hash
- If you change a transaction, you need to change all the subsequent block hashes
- The **difficulty** of the mining algorithm determines the **toughness** of tampering with a block in a blockchain

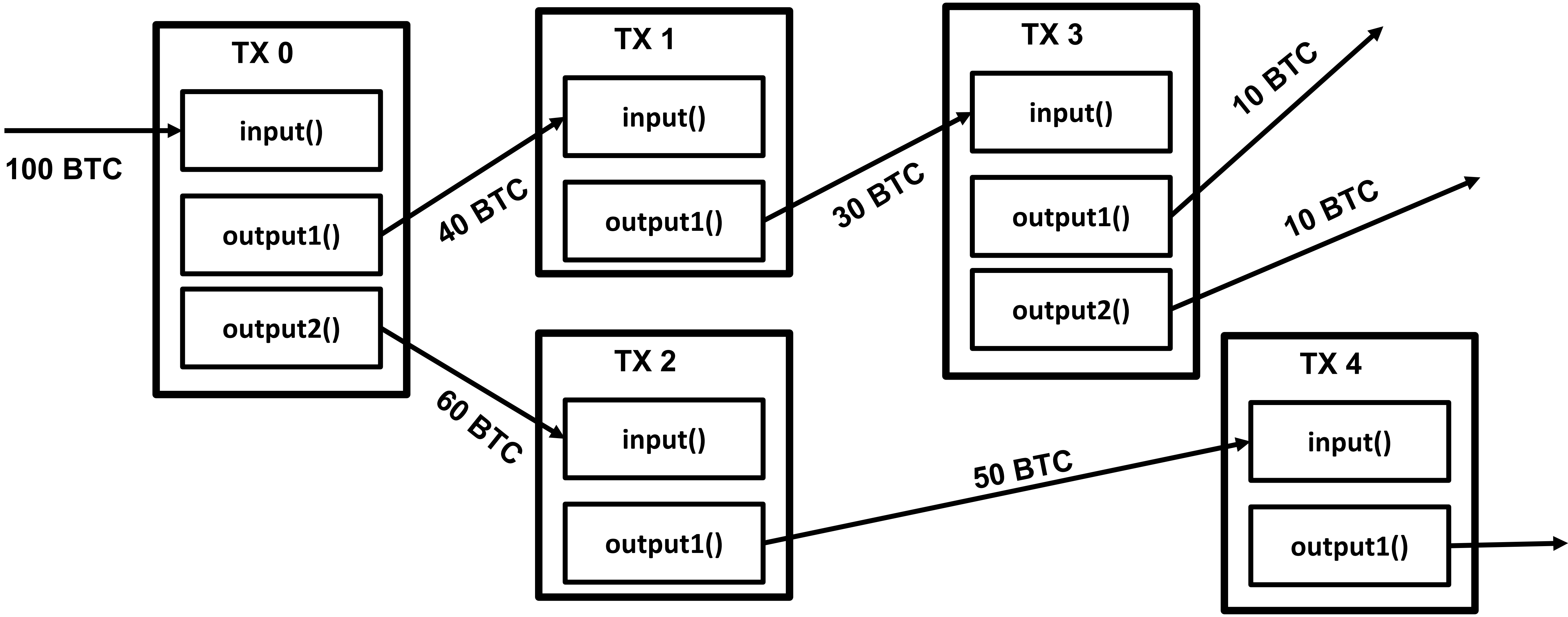
Transactions in a Block (Reference: Bitcoin)



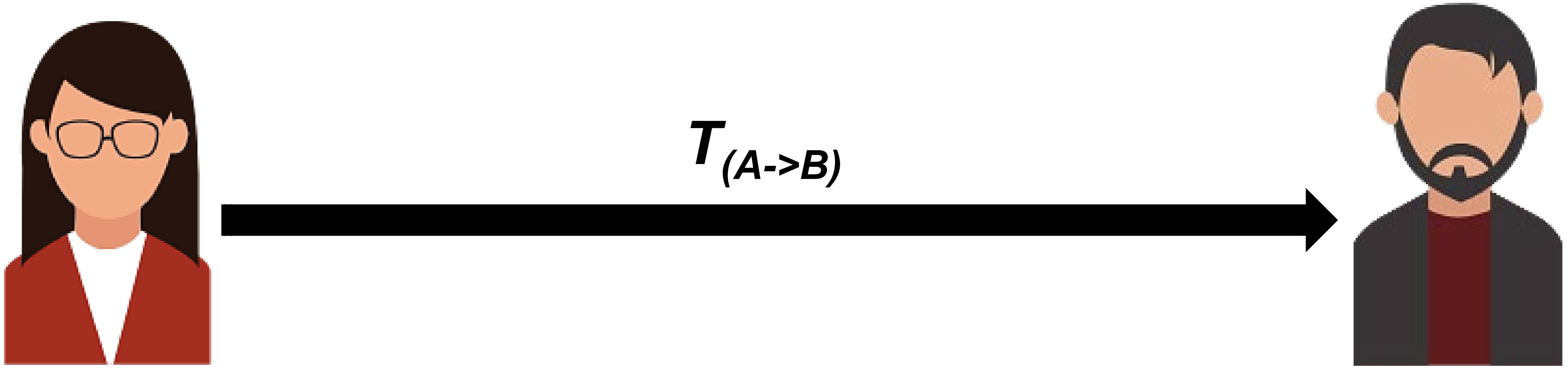
●	0 ID: 634f-087c 8/24/2023, 10:34:12	From Block Reward To 3 Outputs	6.32224878 BTC • $167,263 Fee 0 Sats • $0.00
TX	1 ID: ddb6-3528 8/24/2023, 10:33:58	From 1Mv4-FztL To 2 Outputs	0.42244563 BTC • $11,176.37 Fee 44.9K Sats • $11.88
TX	2 ID: f474-8ad3 8/24/2023, 10:34:17	From bc1q-m6j2 To 2 Outputs	0.05917100 BTC • $1,565.45 Fee 21.6K Sats • $5.71
TX	3 ID: 73e6-3529 8/24/2023, 10:33:15	From 16ga-Q3aX To 2 Outputs	1.08999899 BTC • $28,837.38 Fee 44.9K Sats • $11.88
TX	4 ID: 71d9-0333 8/24/2023, 10:33:15	From 35iM-ueEL To 2 Outputs	0.05948160 BTC • $1,573.67 Fee 16.8K Sats • $4.44
TX	5 ID: 0a89-d723 8/24/2023, 10:33:15	From bc1q-pt0v To 2 Outputs	0.01827425 BTC • $483.47 Fee 14.1K Sats • $3.72
TX	6 ID: 3d7c-5435 8/24/2023, 10:33:15	From bc1q-aj7m To 2 Outputs	0.01092858 BTC • $289.13 Fee 9.5K Sats • $2.51

Block Source: https://www.blockchain.com/explorer/blocks/btc/804613

Bitcoin Transactions and Input and Output

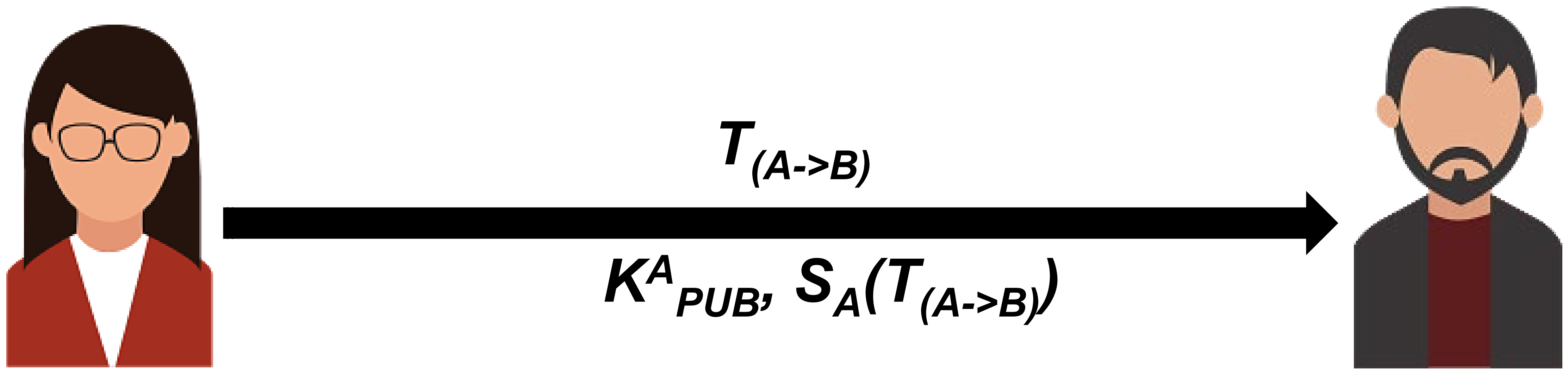


Bitcoin Scripts – A Simple Example



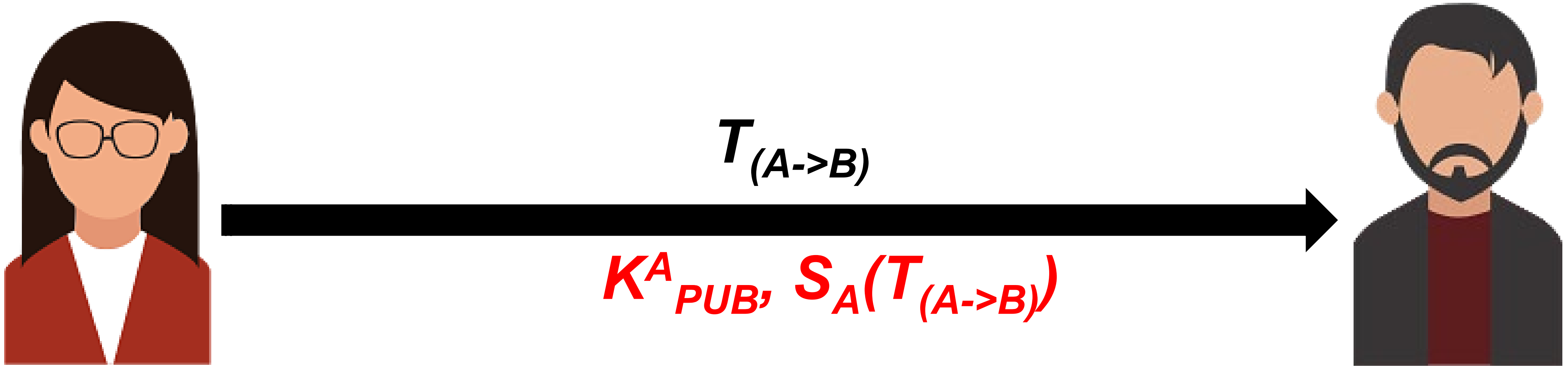
How Bob will verify that the transaction is actually originated from Alice?

Bitcoin Scripts – A Simple Example



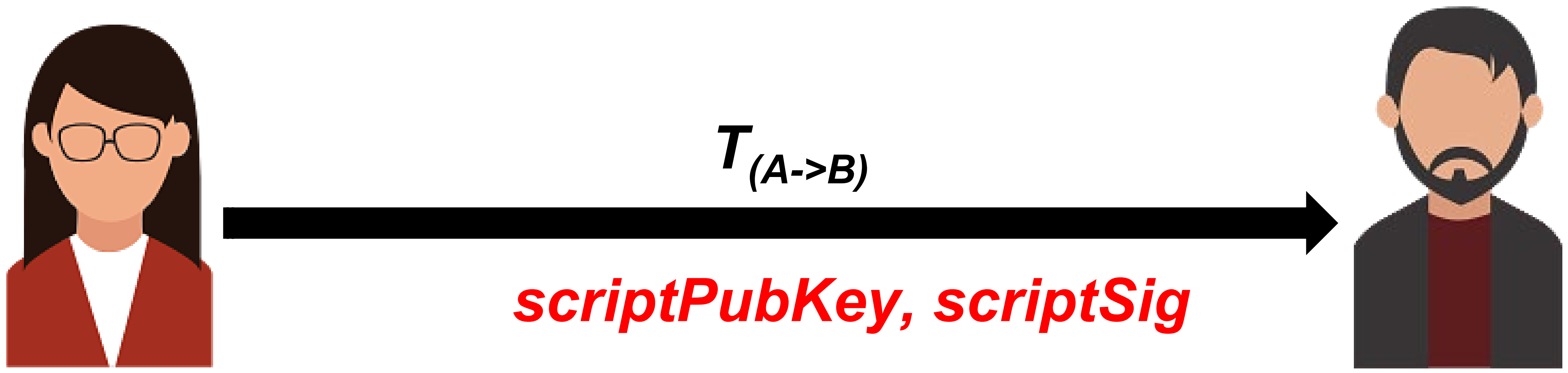
Send the public key of Alice along with the signature -> Bob can verify this

Bitcoin Scripts – A Simple Example



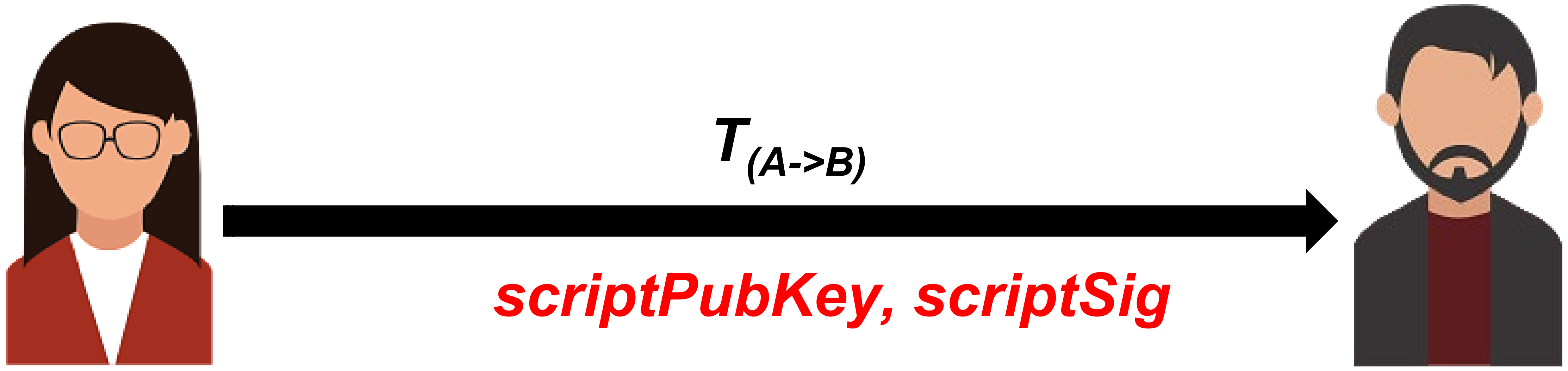
Bitcoin indeed transfers scripts instead of the signature and the public key

Bitcoin Scripts – A Simple Example



Bitcoin indeed transfers scripts instead of the signature and the public key

Bitcoin Scripts – A Simple Example



Bob can spend the bitcoins only if both the scripts return `true` after execution

Bitcoin Scripts

- With every transaction Bob must provide
  - A public key that, when hashed, yields the address of Bob embedded in the script
  - A signature to provide ownership of the private key corresponding to the public key of Bob

Bitcoin Scripts

Transaction Input

scriptSig:

18E14A7B6A30…
D61967F63C7DD…

Transaction Output

scriptPubKey:

OP_DUP
OP_HASH160
16UwLL9Risc3QfPqBUvKof…
OP_EQUALVERIFY
OP_CHECKSIG

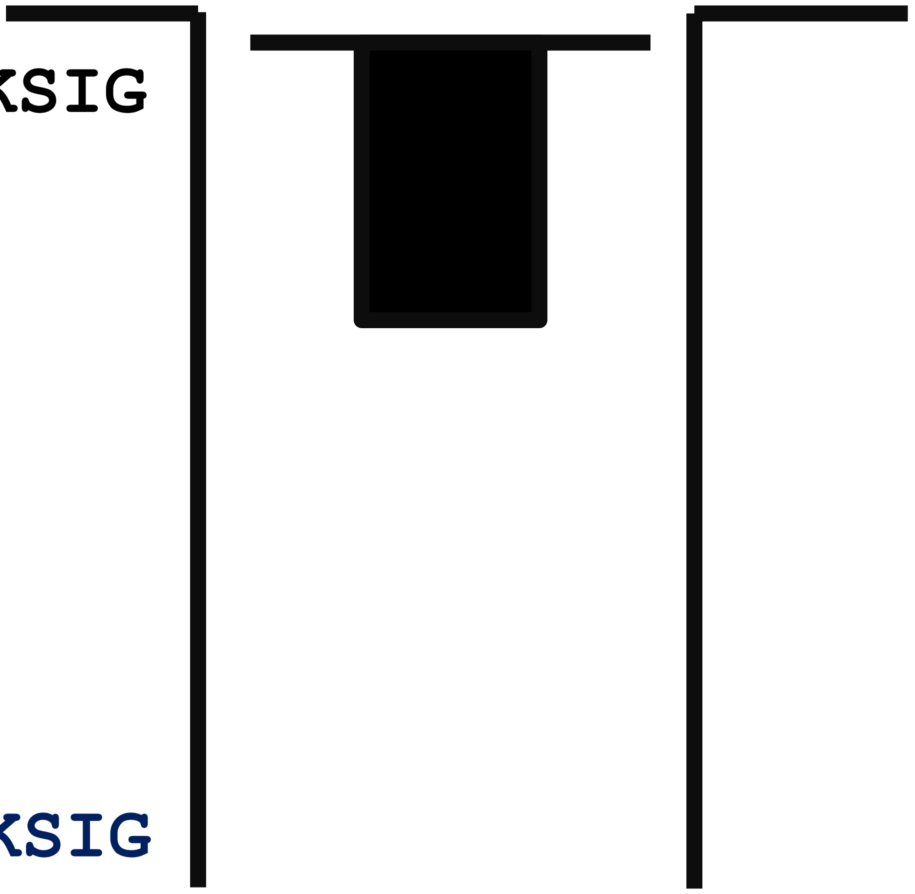
See for detailed steps: https://developer.bitcoin.org/devguide/transactions.html
Simple Example: https://medium.com/@aalim.khan/bitcoin-transactions-scripts-and-digital-signatures-506688e1630a

Bitcoin Scripts

```
scriptPubKey: OP_DUP OP_HASH160
<pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG

scriptSig: <sig> <pubKey>
```

- The stack is initially empty. Both the scripts are combined – input followed by output

```
<sig> <pubKey> OP_DUP OP_HASH160
<pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG
```

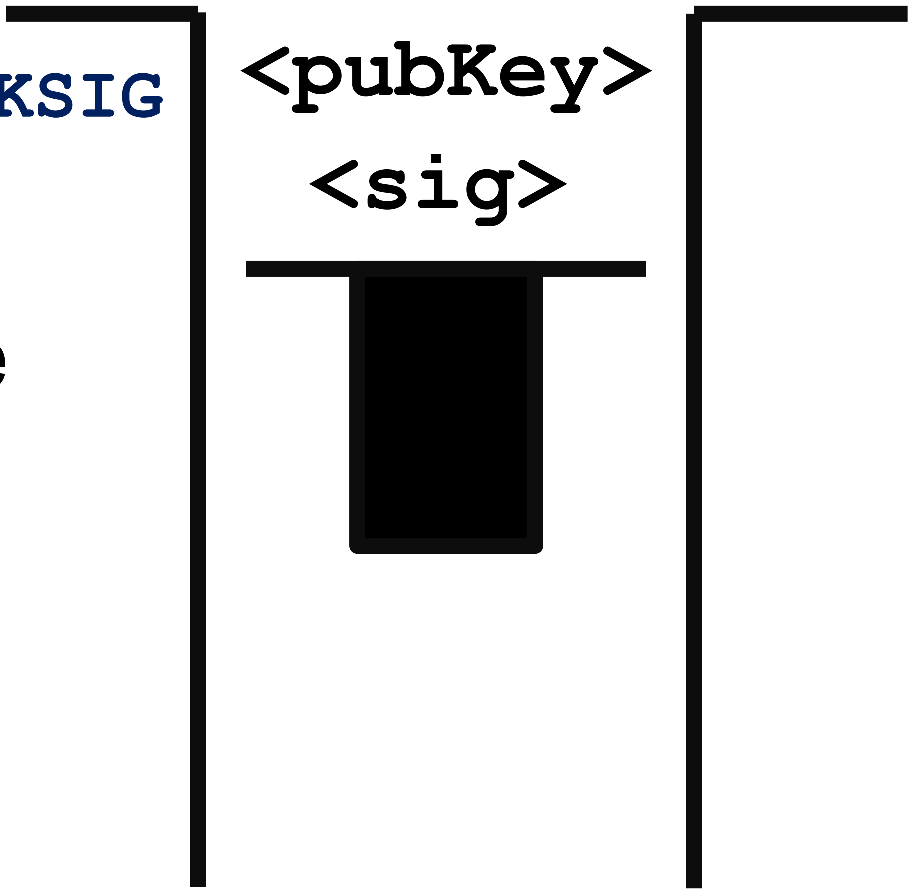


Bitcoin Scripts

```
<sig> <pubKey> OP_DUP OP_HASH160
<pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG
```

- The stack is initially empty. Both the scripts are combined

```
OP_DUP OP_HASH160 <pubKeyHash>
OP_EQUALVERIFY OP_CHECKSIG
```

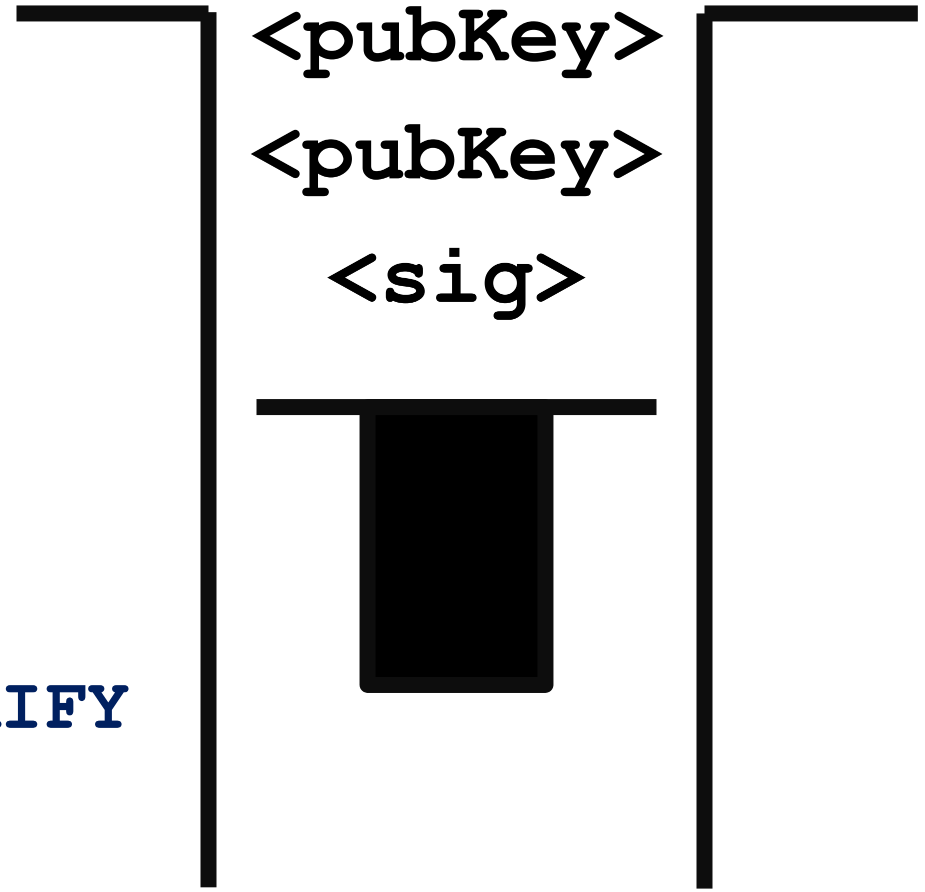


Bitcoin Scripts

```
OP_DUP OP_HASH160 <pubKeyHash>
OP_EQUALVERIFY OP_CHECKSIG
```

- Top stack item is duplicated

```
OP_HASH160 <pubKeyHash> OP_EQUALVERIFY
OP_CHECKSIG
```

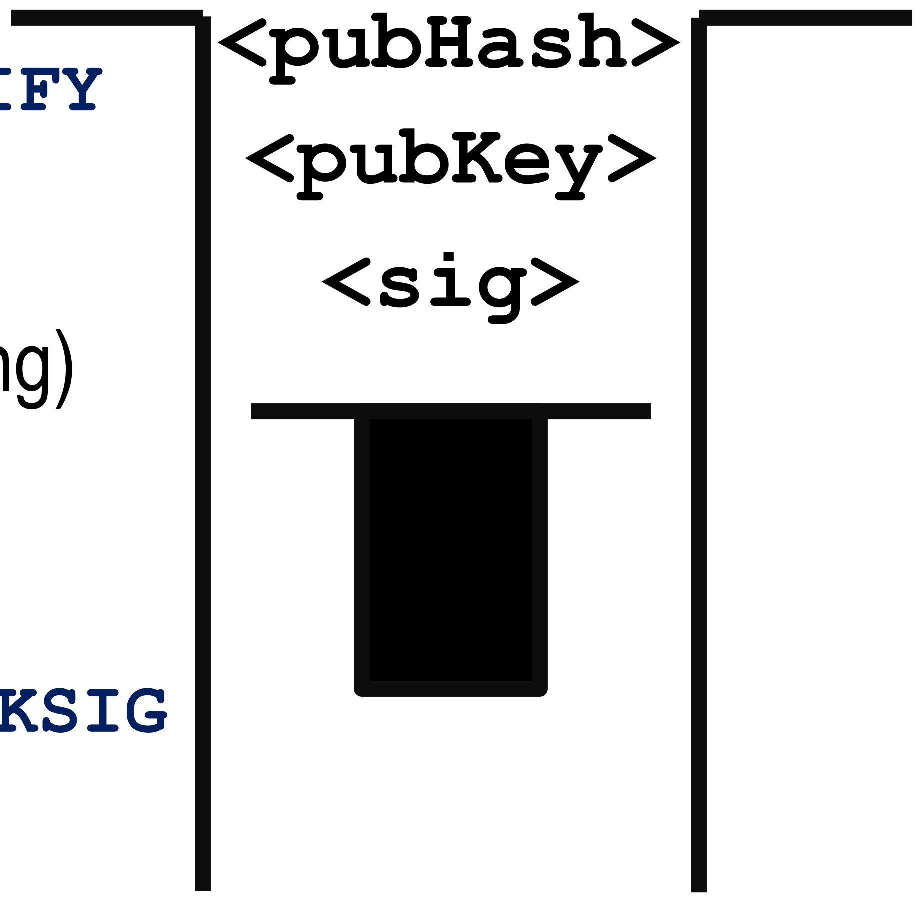


Bitcoin Scripts

```
OP_HASH160 <pubKeyHash> OP_EQUALVERIFY
OP_CHECKSIG
```

- Top stack item is hashed (RIPEMD-160 hashing)

```
<pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG
```

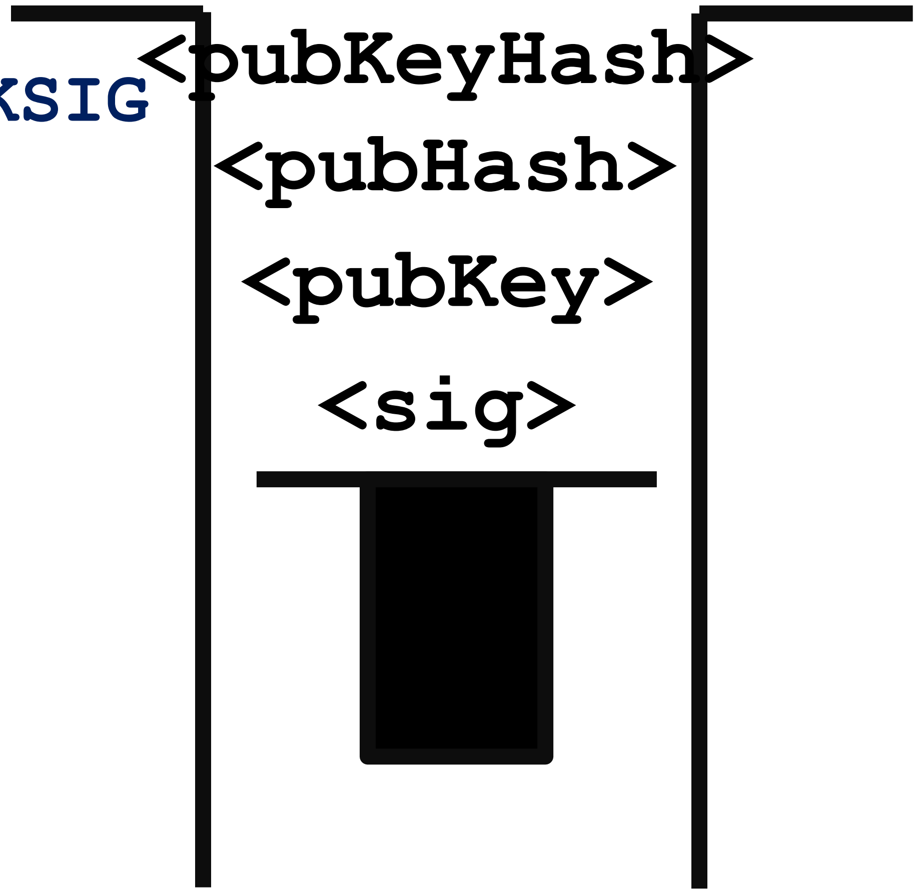


Bitcoin Scripts

```
<pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG
```

- The constant is pushed in the stack

```
OP_EQUALVERIFY OP_CHECKSIG
```

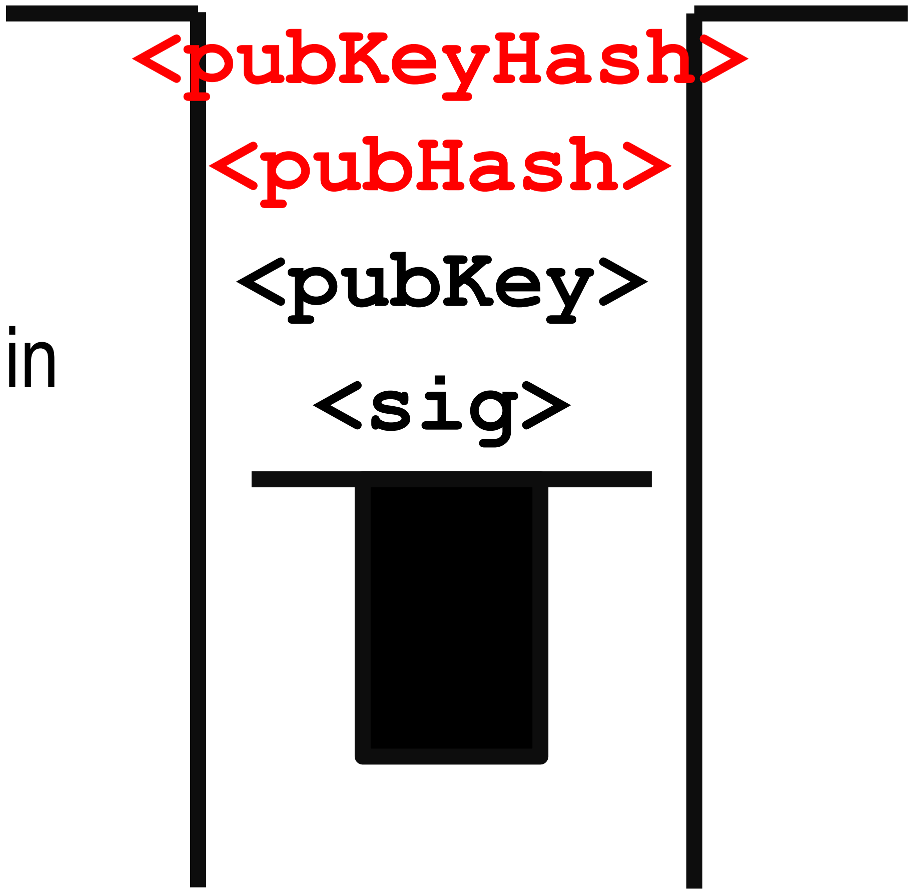


Bitcoin Scripts

```
OP_EQUALVERIFY OP_CHECKSIG
```

- Equality is checked between the top two items in the stack

```
OP_CHECKSIG
```

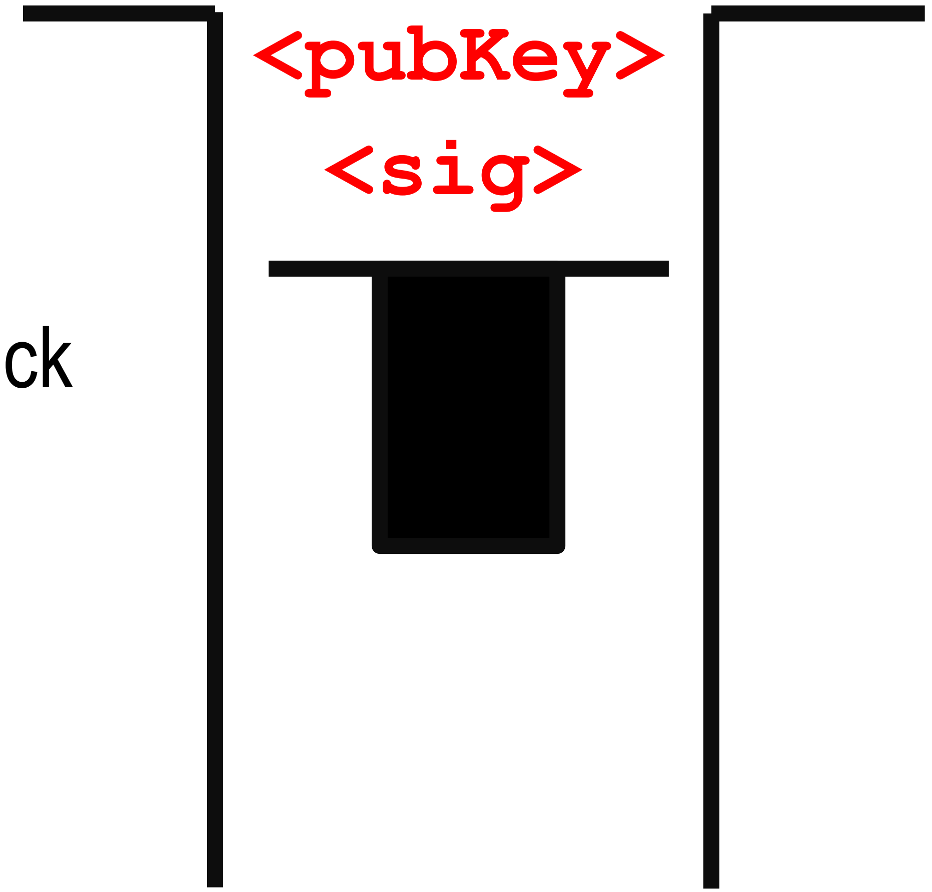


Bitcoin Scripts

OP_CHECKSIG

- Signature is checked based on the top two stack item

TRUE



Bitcoin Script Instructions

- Total 256 opcodes (15 disabled, 75 reserved)
    - Arithmetic operations
    - if-then conditions
    - Logical operators
    - Data handling (like OP_DUP)
    - Cryptographic operations
        - Hash functions
        - Signature verification
        - Multi-signature verification

Interesting Bitcoin Scripts

Provably un-spendable or prunable outputs

```
scriptPubKey: OP_RETURN {zero or more ops}
```

Anyone-can-spend outputs

```
scriptPubKey: {empty}
 scriptSig: OP_TRUE
```

Source: https://en.bitcoin.it/wiki/Script

Interesting Bitcoin Scripts

Freezing funds until a time in the future

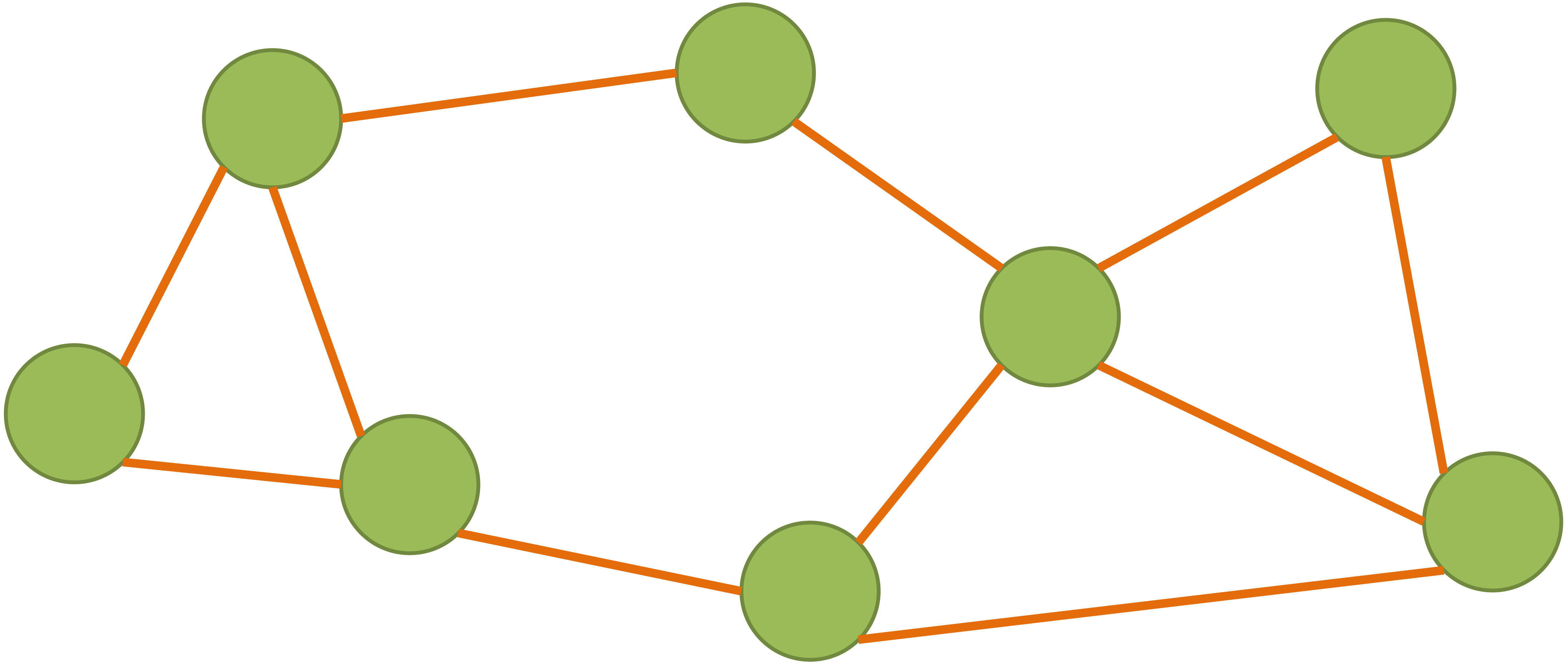
```
scriptPubKey: <expiry_time> OP_CHECKLOCKTIMEVERIFY OP_DROP
 OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG

  scriptSig: <sig> <pubKey>
```

Source: https://en.bitcoin.it/wiki/Script

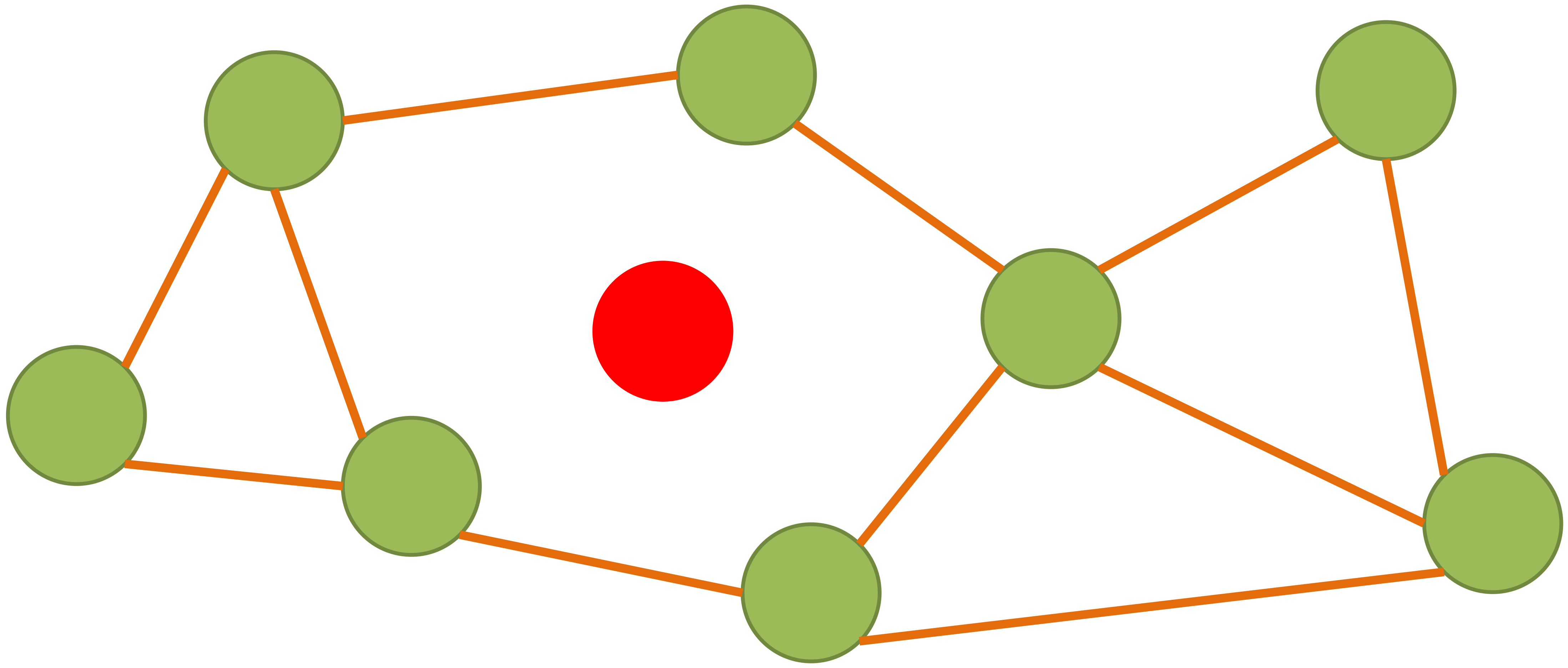
Bitcoin P2P Network

- An ad-hoc network with random topology, Bitcoin protocol runs on TCP port 8333
- All nodes (users) in the bitcoin network are treated equally
- New nodes can join any time, non-responding nodes are removed after 3 hours

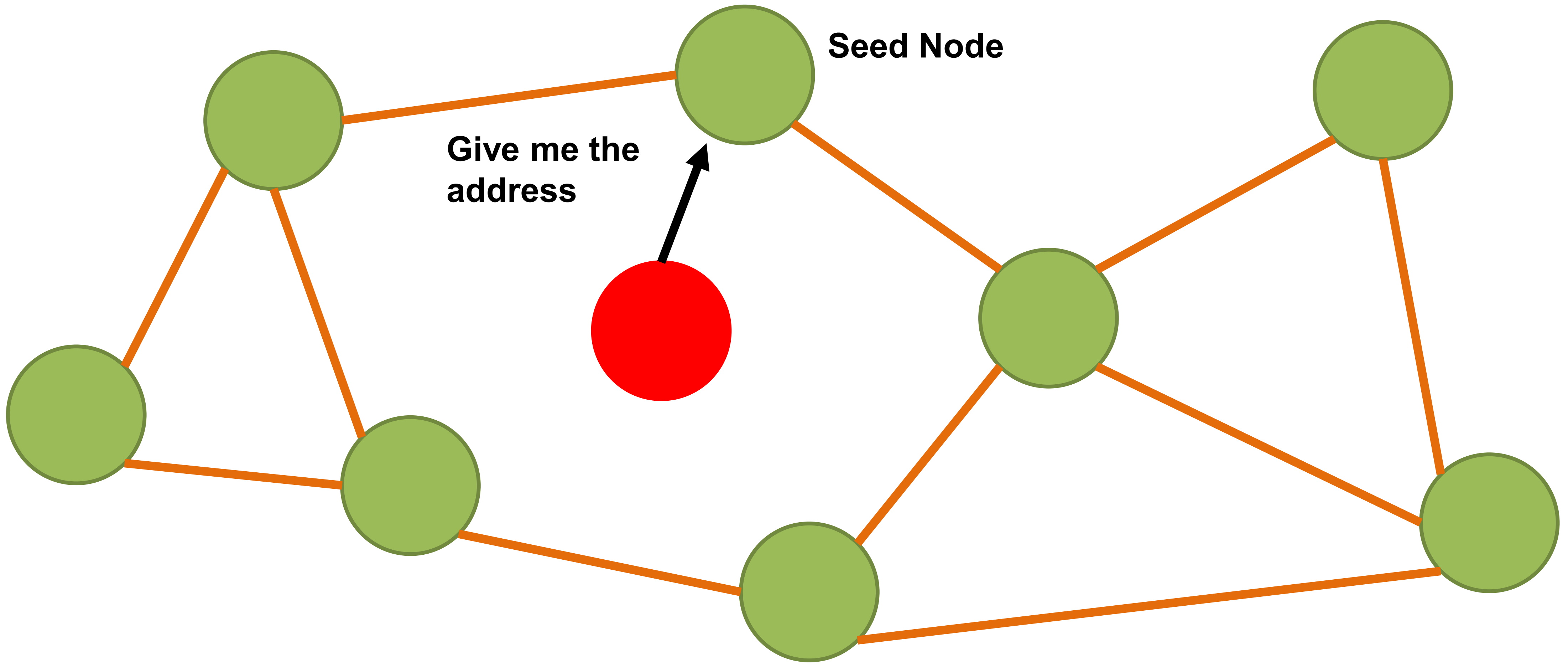
Joining in a Bitcoin P2P Network



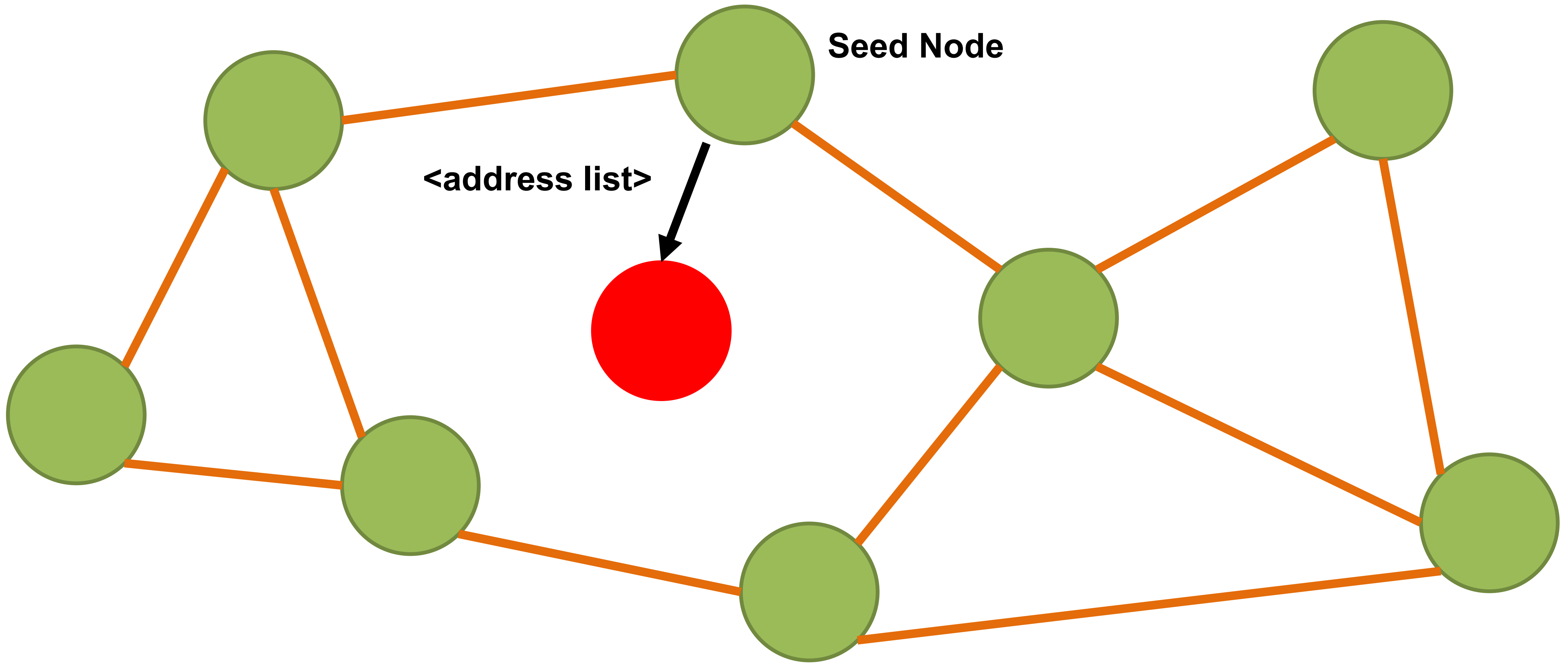
Joining in a Bitcoin P2P Network



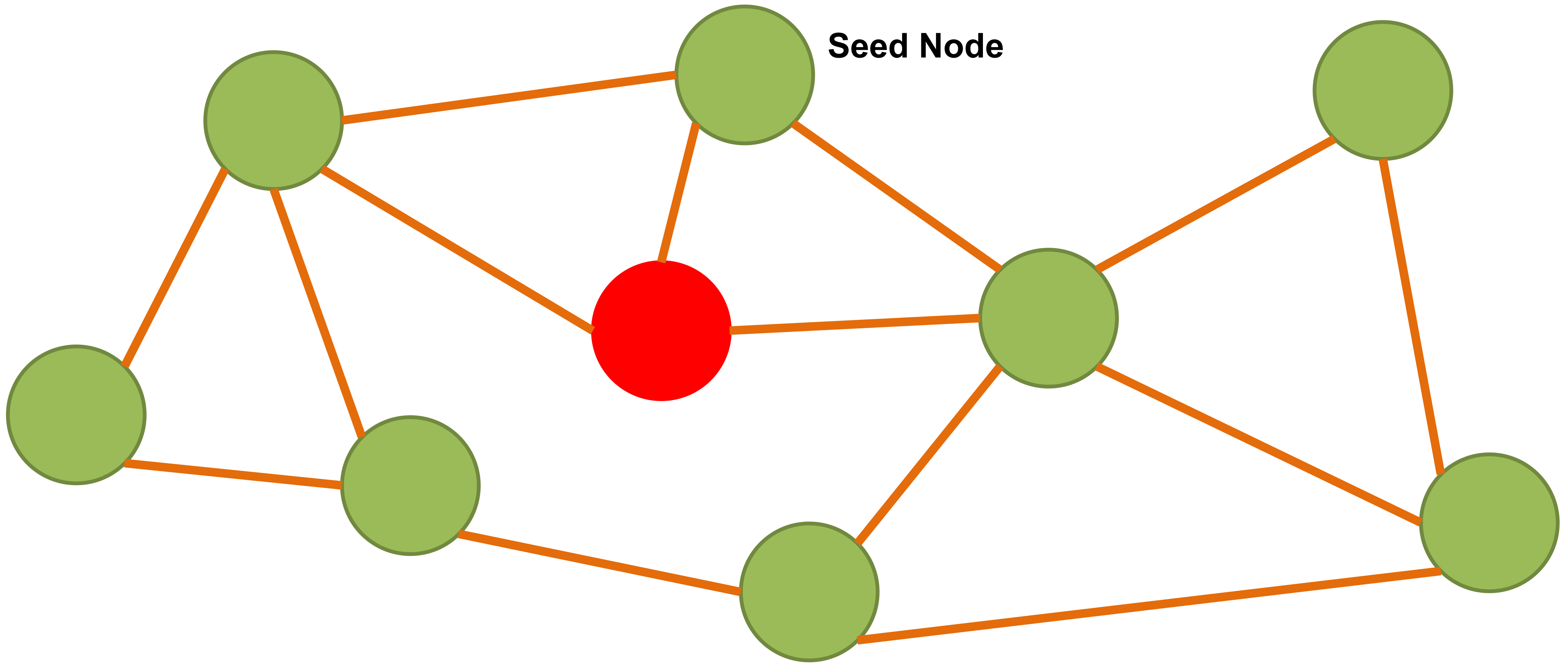
Joining in a Bitcoin P2P Network



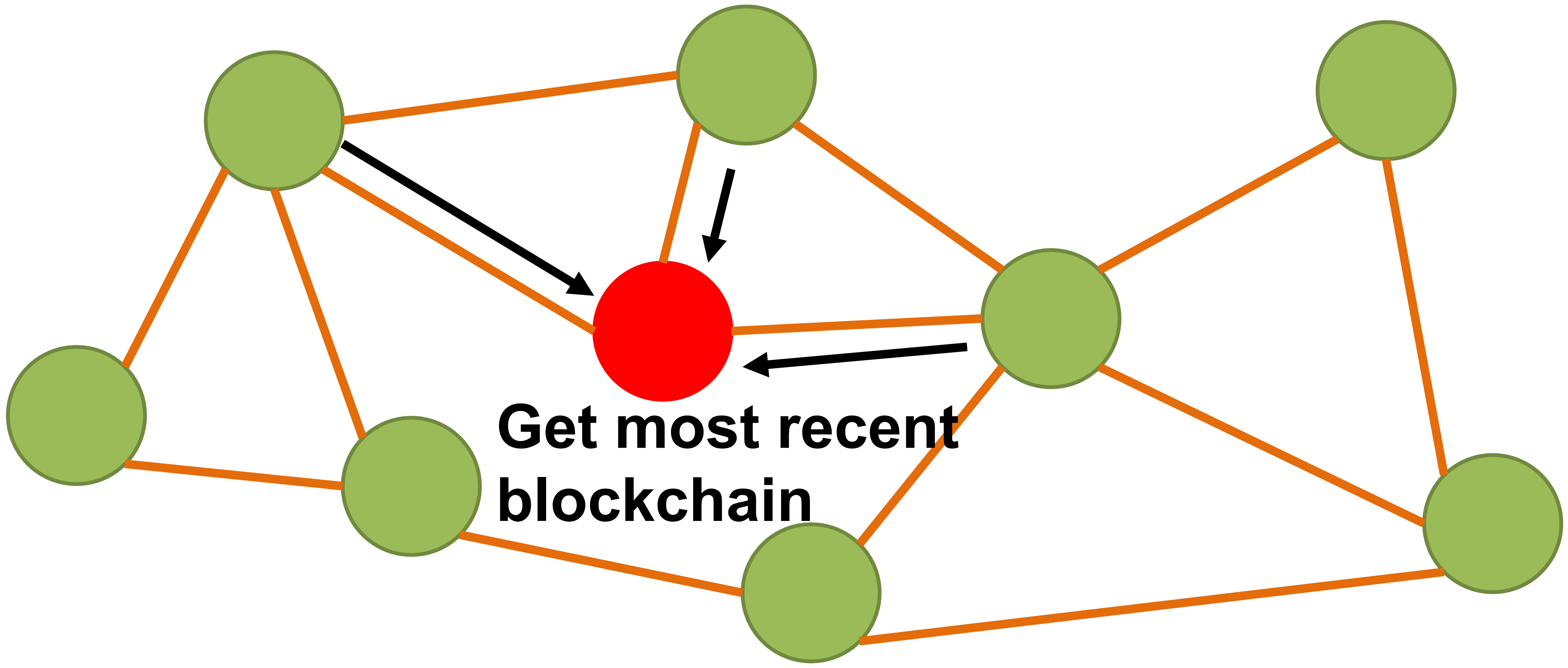
Joining in a Bitcoin P2P Network



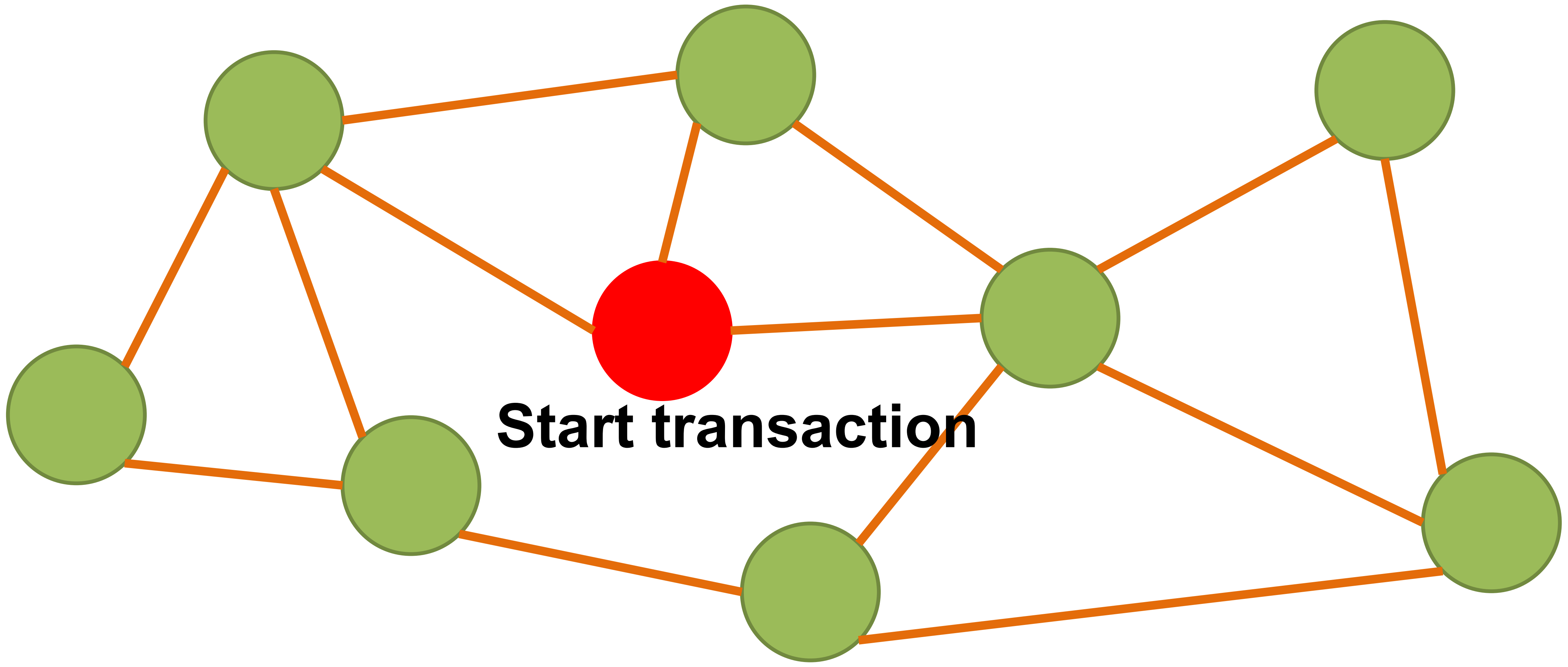
Joining in a Bitcoin P2P Network



Joining in a Bitcoin P2P Network

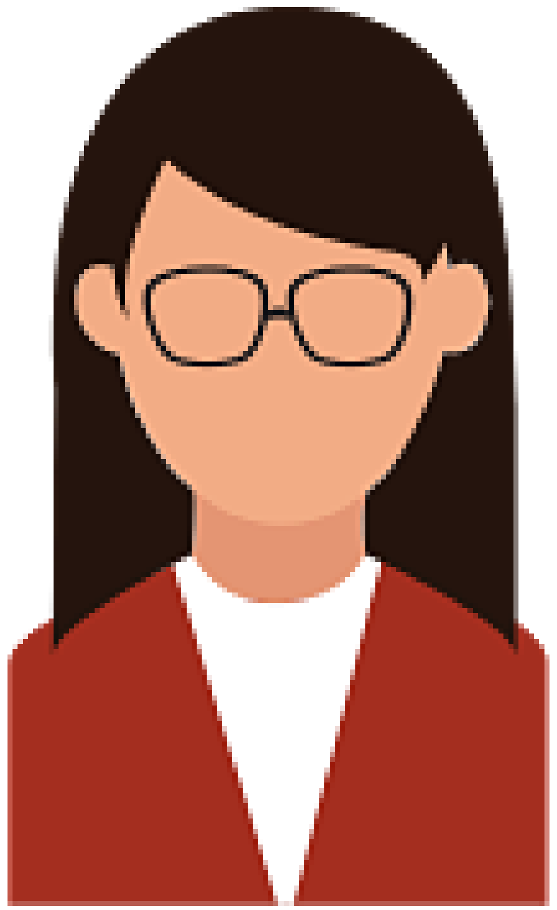


Joining in a Bitcoin P2P Network

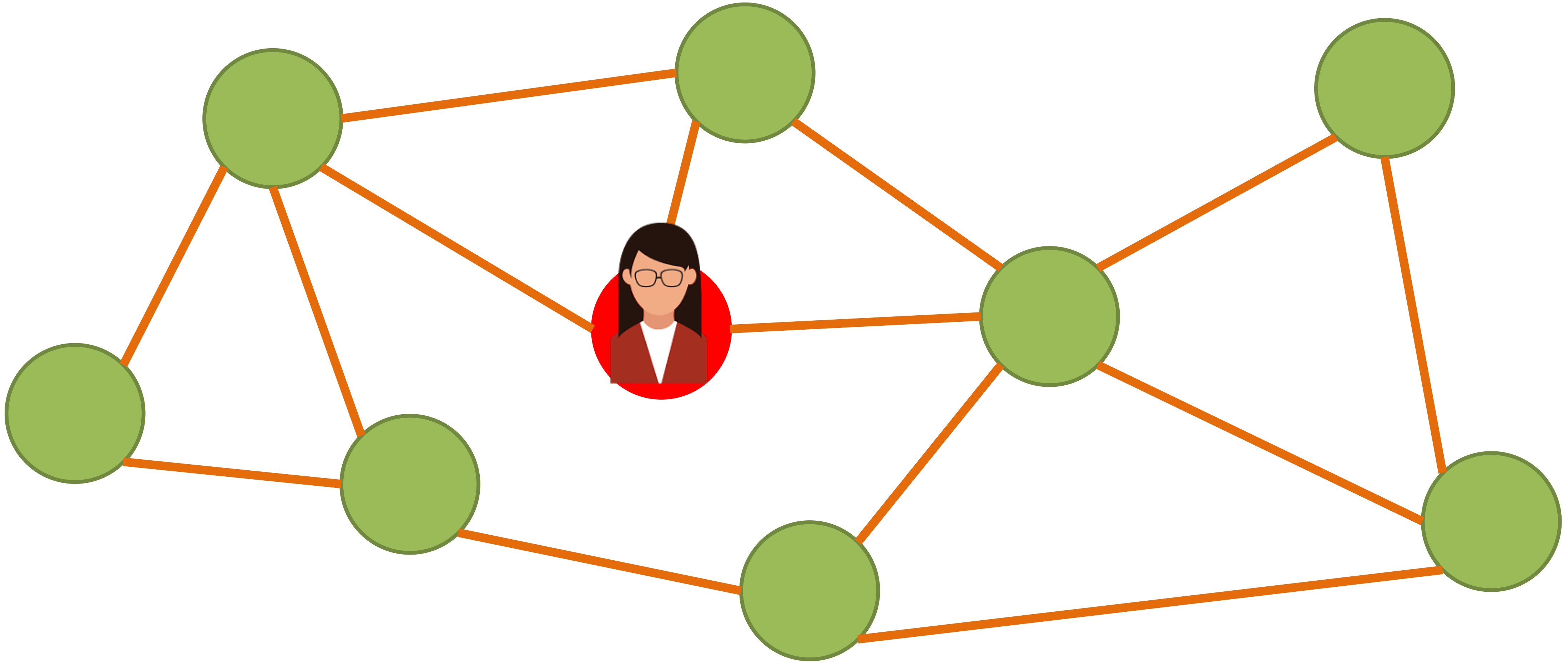


Transaction in a Bitcoin Network

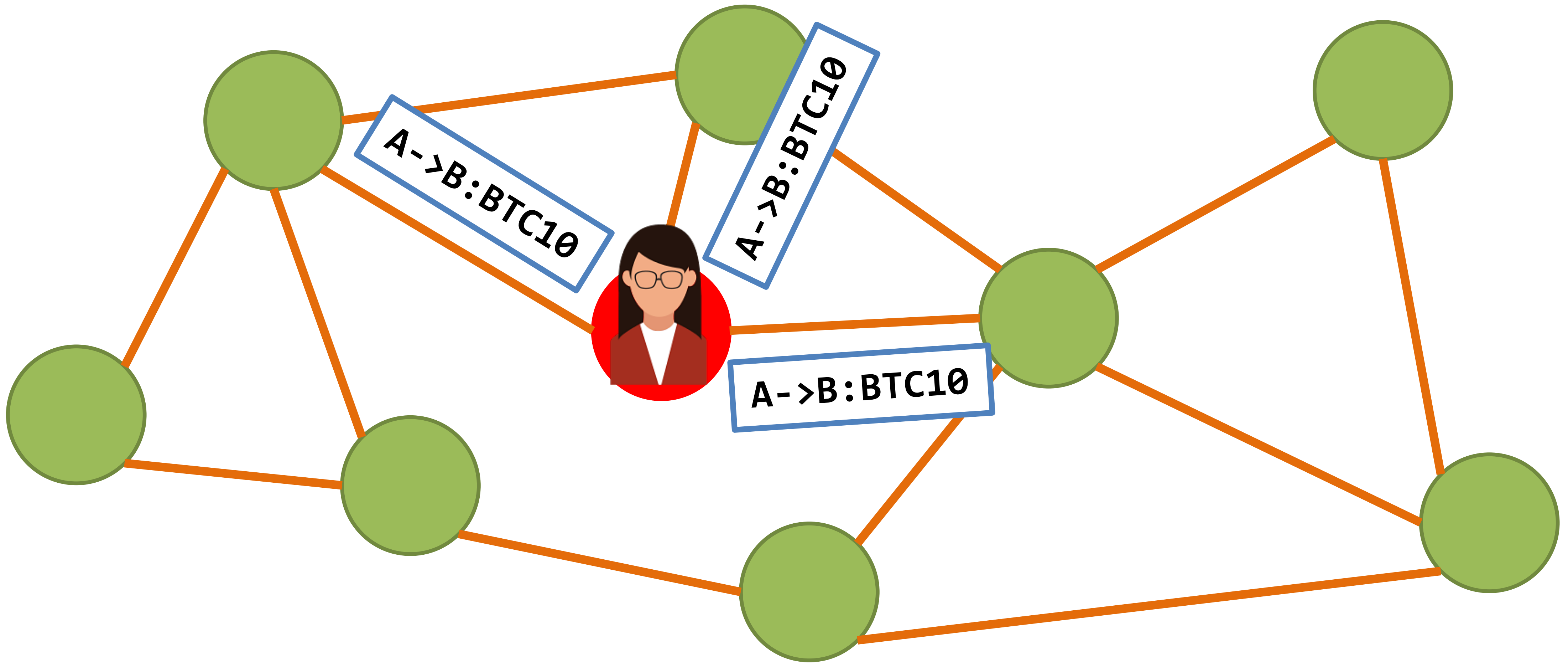
- Alice joins the Bitcoin network by opening her applet
- Alice makes a transaction to Bob: **`A->B: BTC 10`**
- Alice includes the scripts with the transactions
- Alice broadcasts this transaction in the Bitcoin network



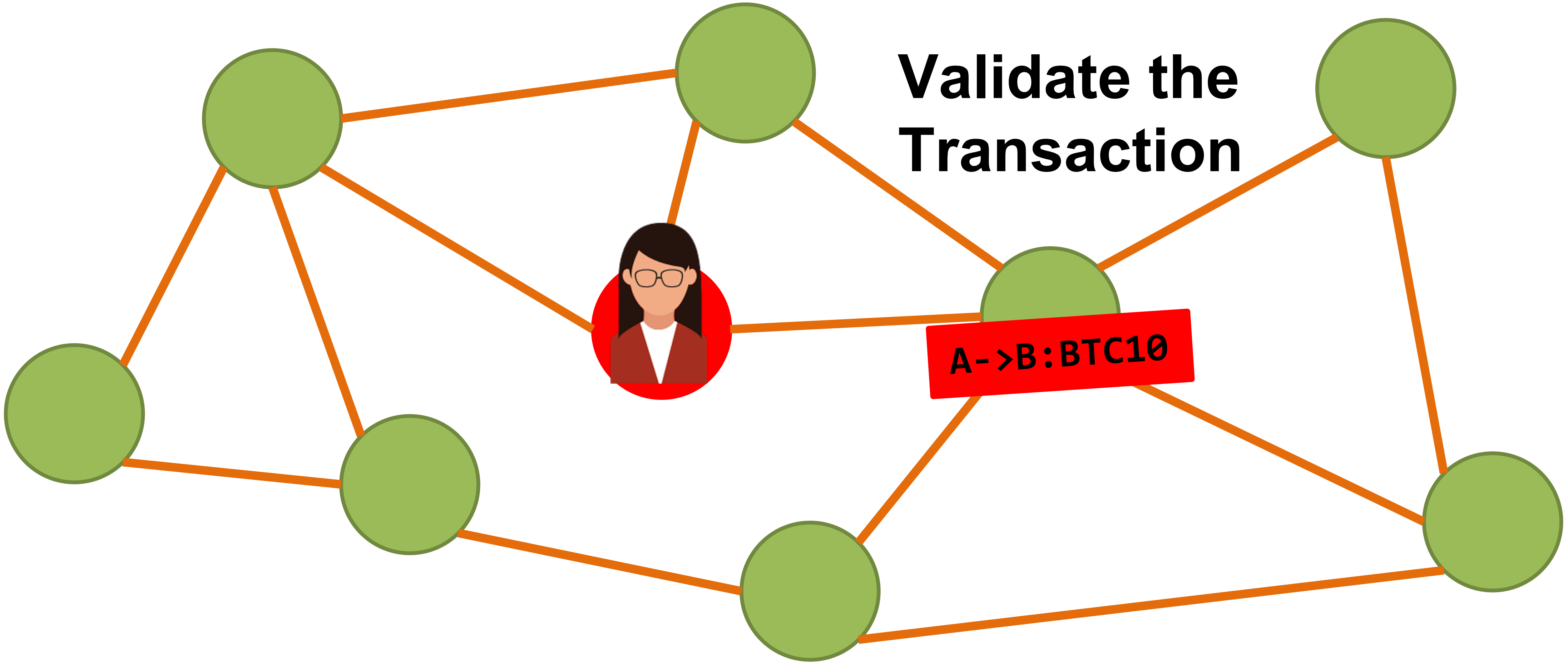
Transaction Flooding in a Bitcoin Network



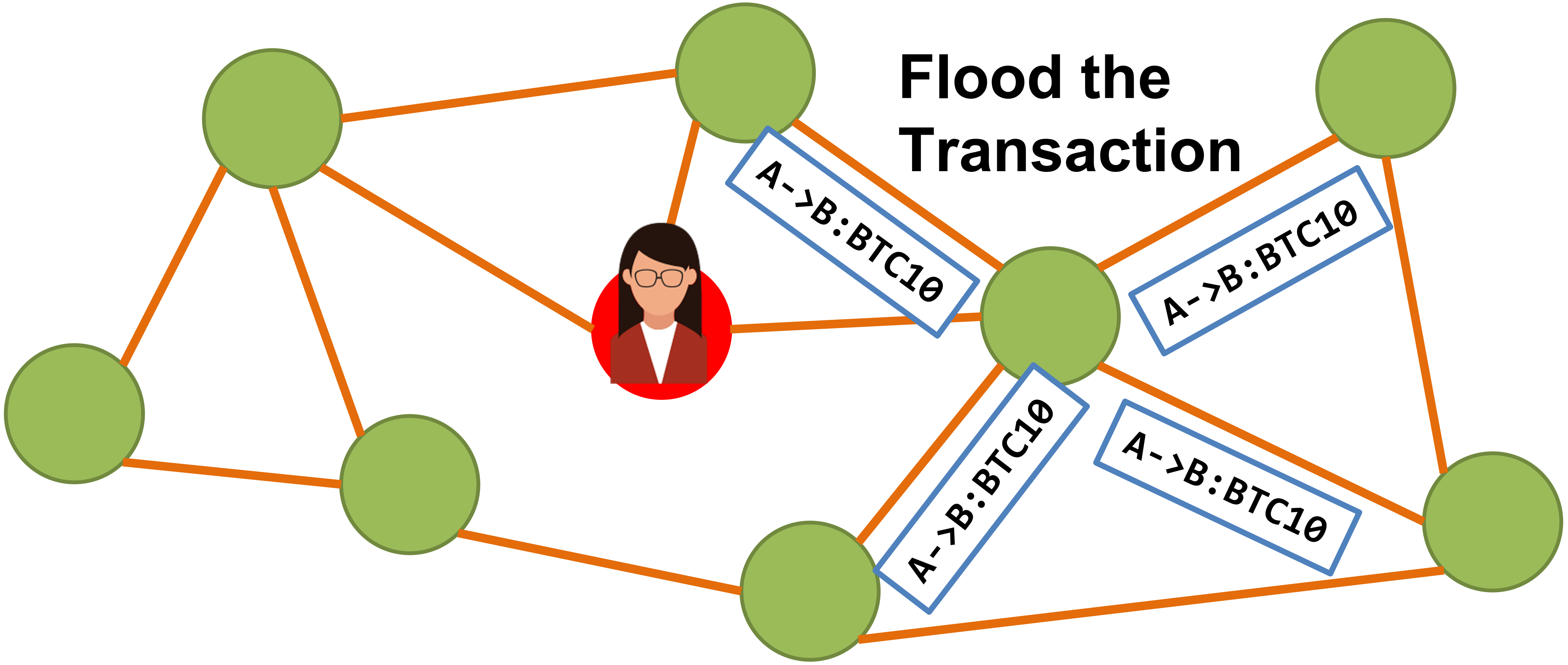
Transaction Flooding in a Bitcoin Network



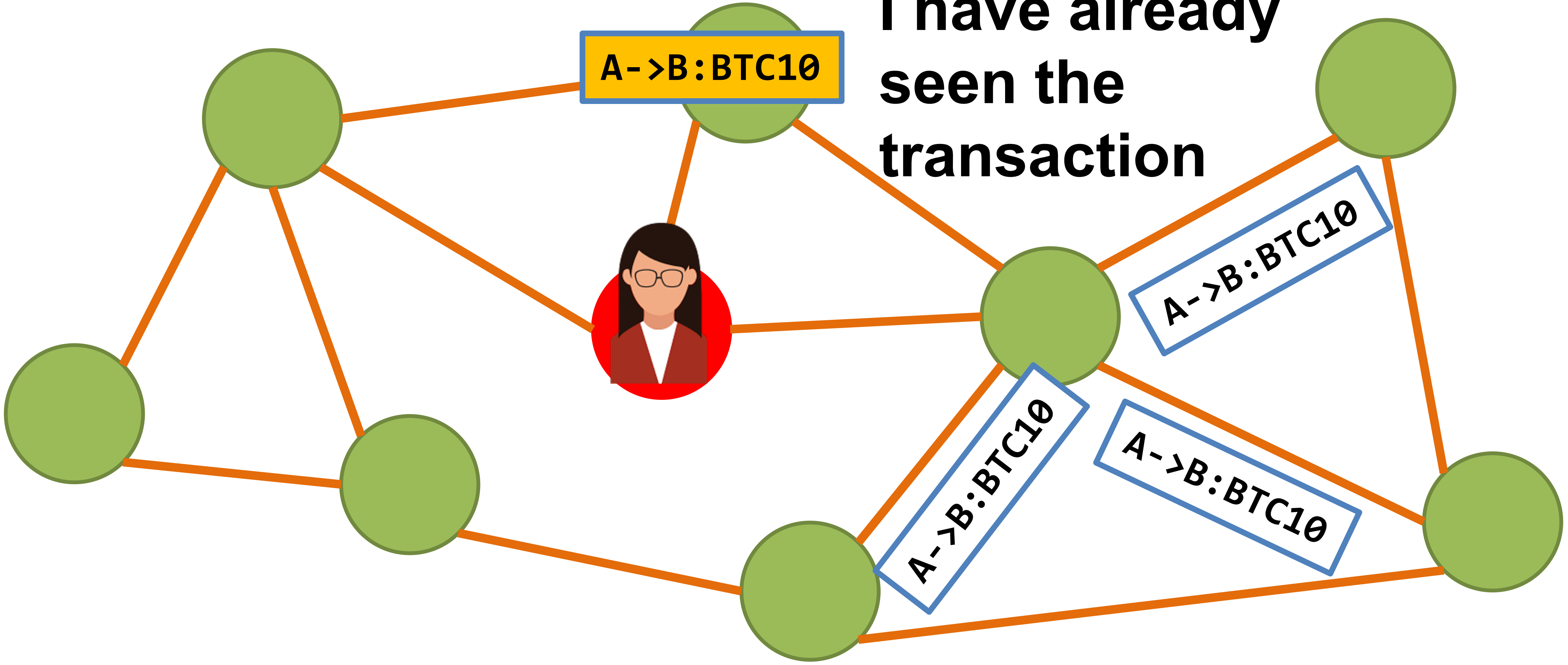
Transaction Flooding in a Bitcoin Network



Transaction Flooding in a Bitcoin Network



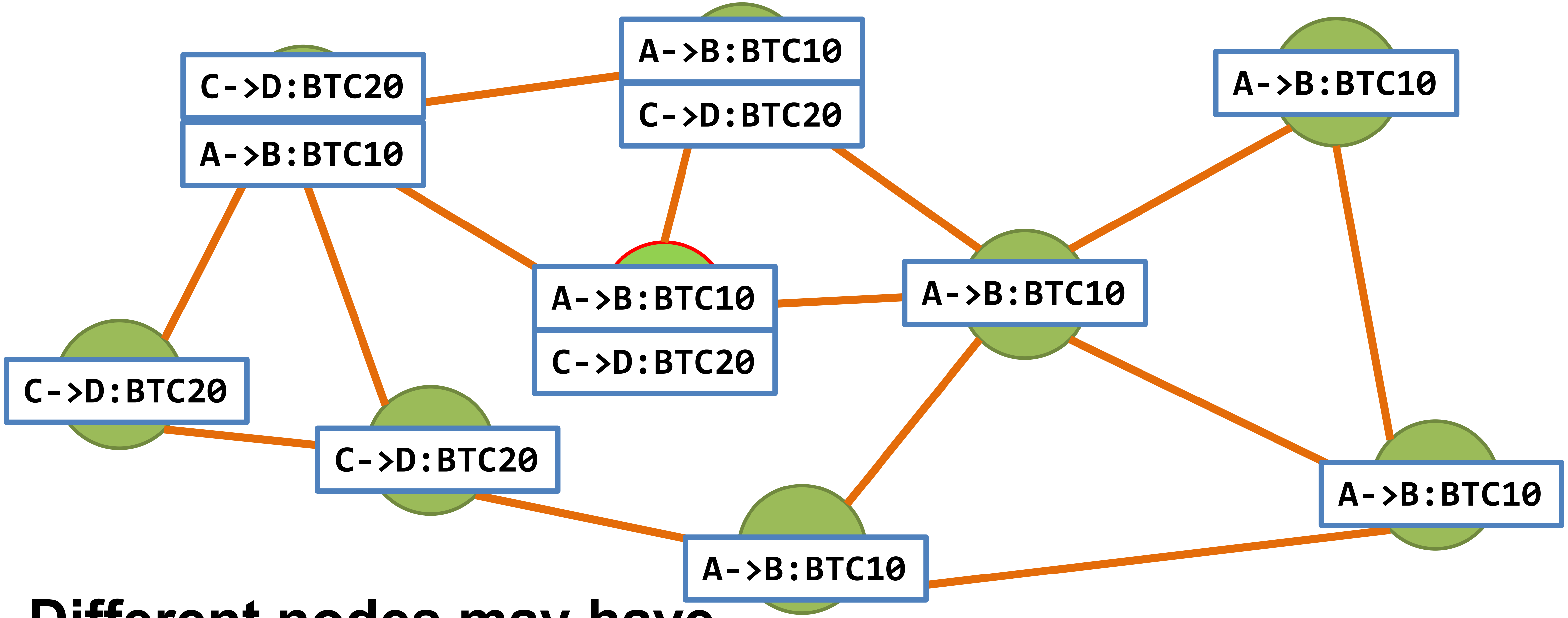
Transaction Flooding in a Bitcoin Network



Which Transactions Should You Relay?

- The transaction is valid with current blockchain
  - No conflict
  - No double spending
- The script matches with a pre-given set of whitelist scripts – avoid unusual scripts, avoid infinite loops
- Does not conflict with other transactions that I have relayed after getting the blockchain updated – avoid double spending

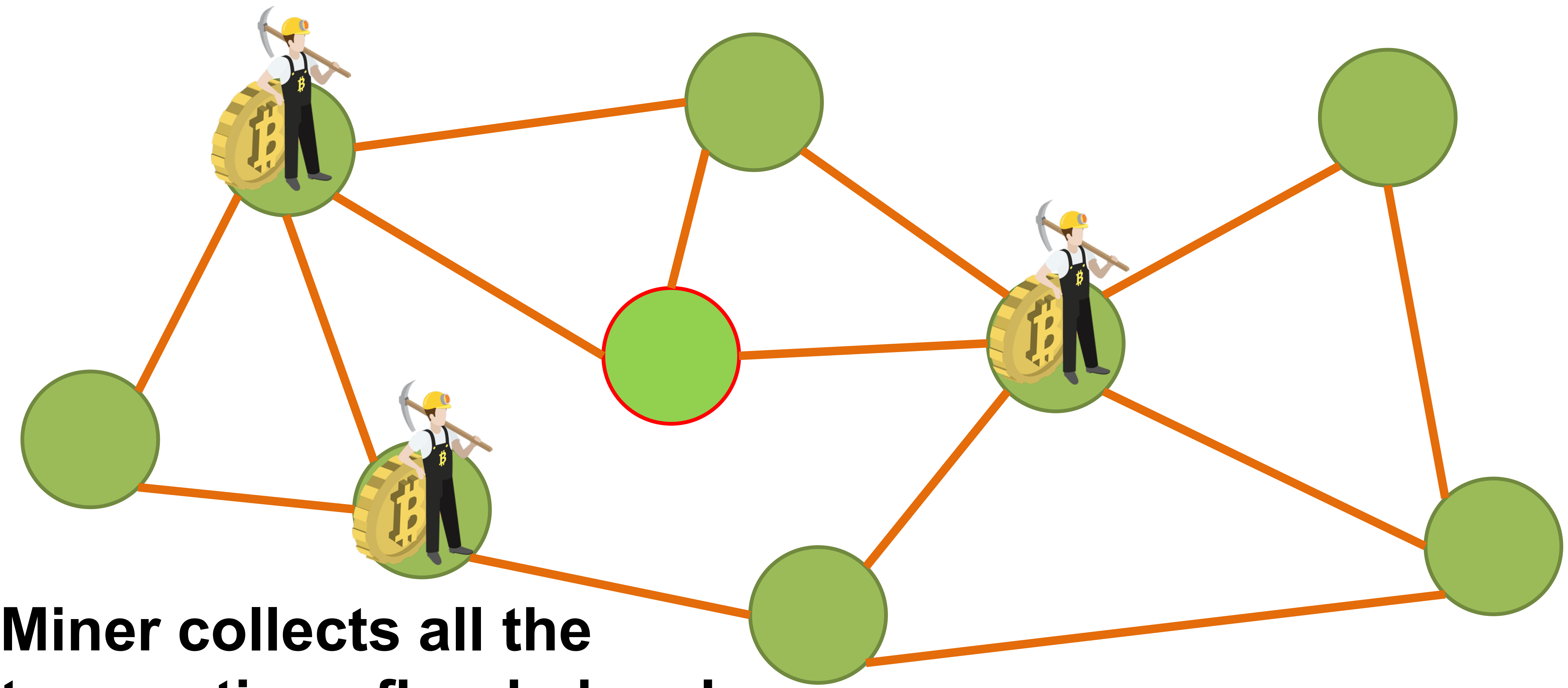
Transaction Flooding in a Bitcoin Network



Different nodes may have different transaction pools

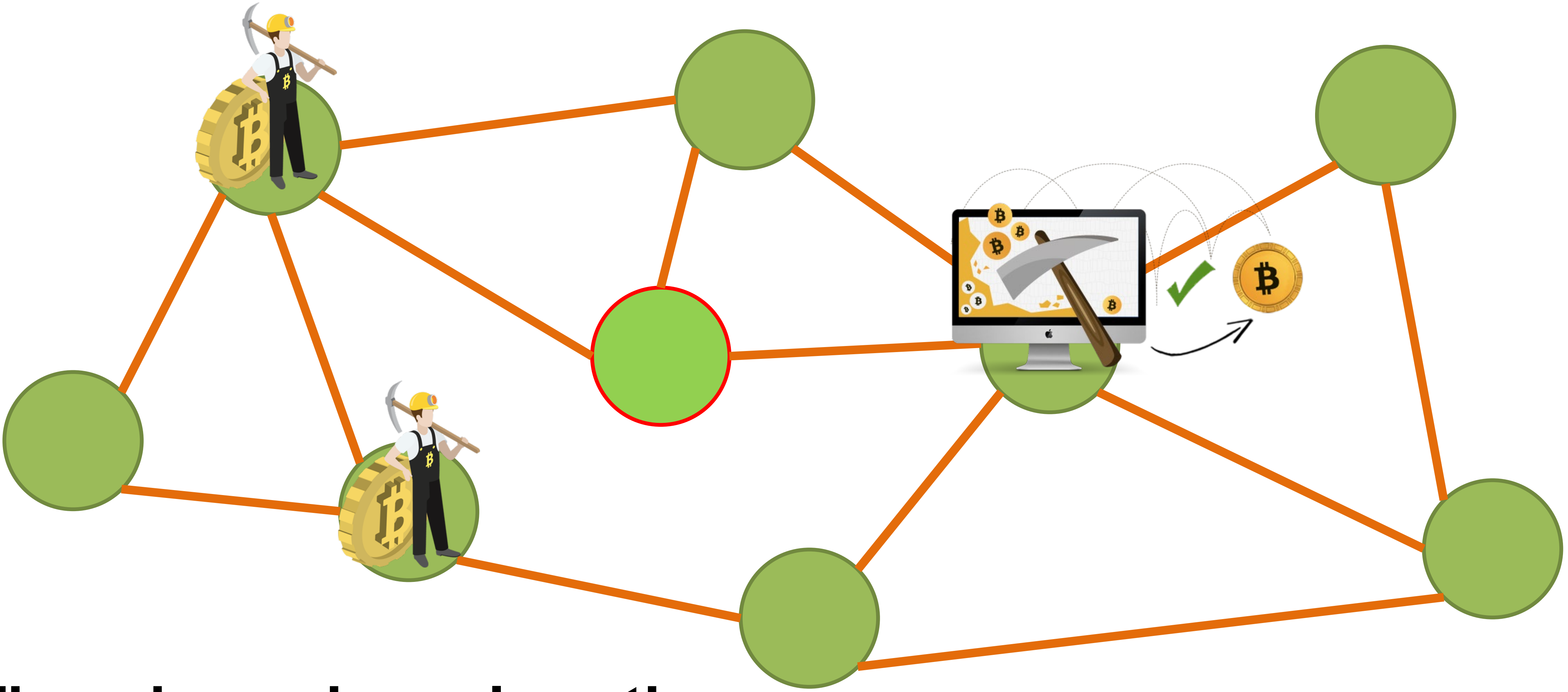
Accept the first set of transactions that you have heard

Mining in a Bitcoin Network



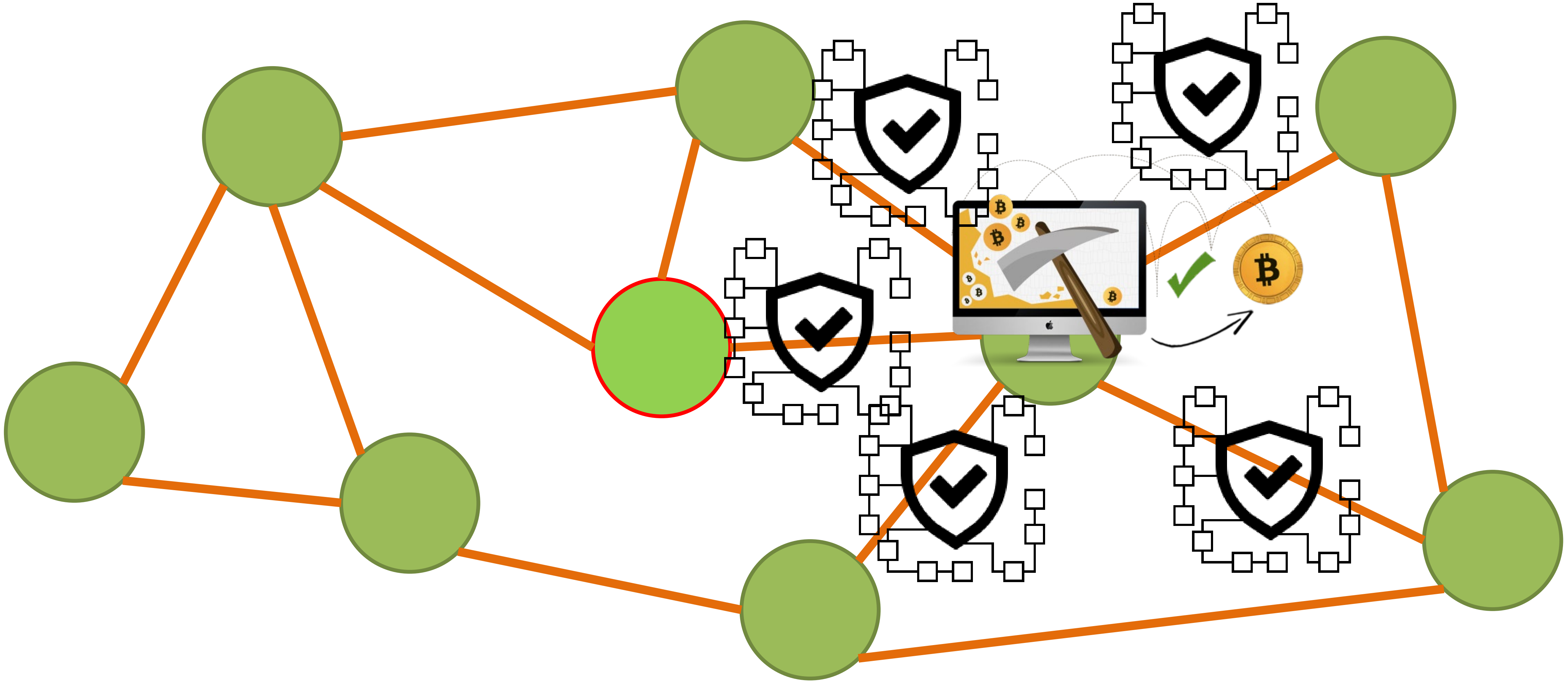
Miner collects all the transactions flooded and starts Mining

Block Generation



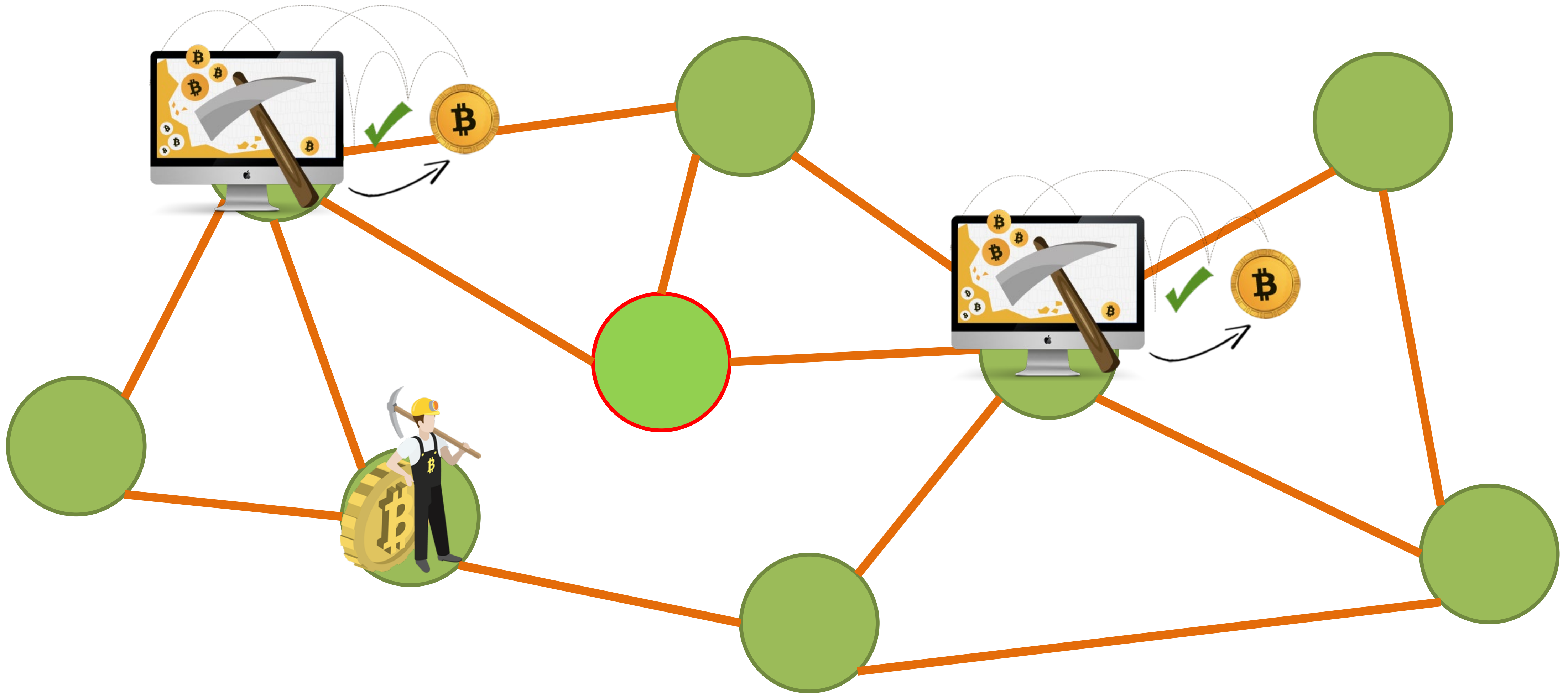
The miner who solves the puzzle first, generates a new block

Block Flooding



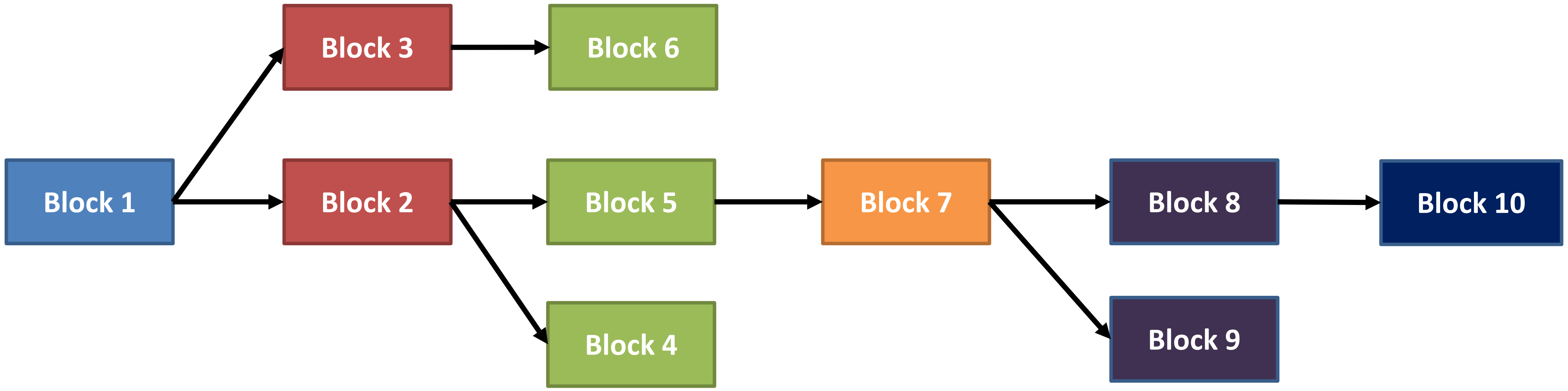
Flood the blockchain with the new block included

Block Propagation

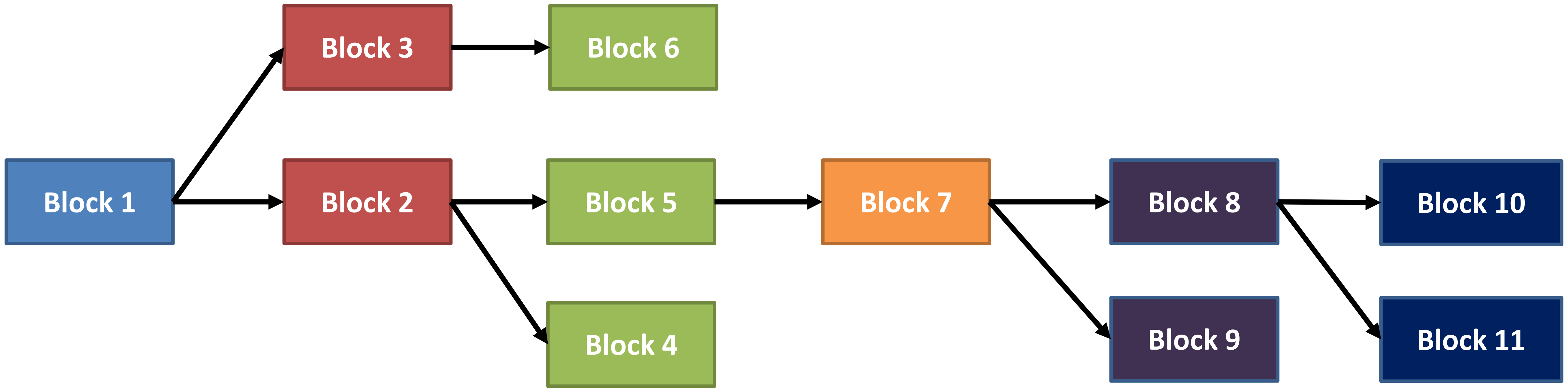


Multiple miners can mine a new block simultaneously or in a near identical time

Block Propagation – Accept the Longest Chain



Block Propagation – Accept One of the Longest Chains

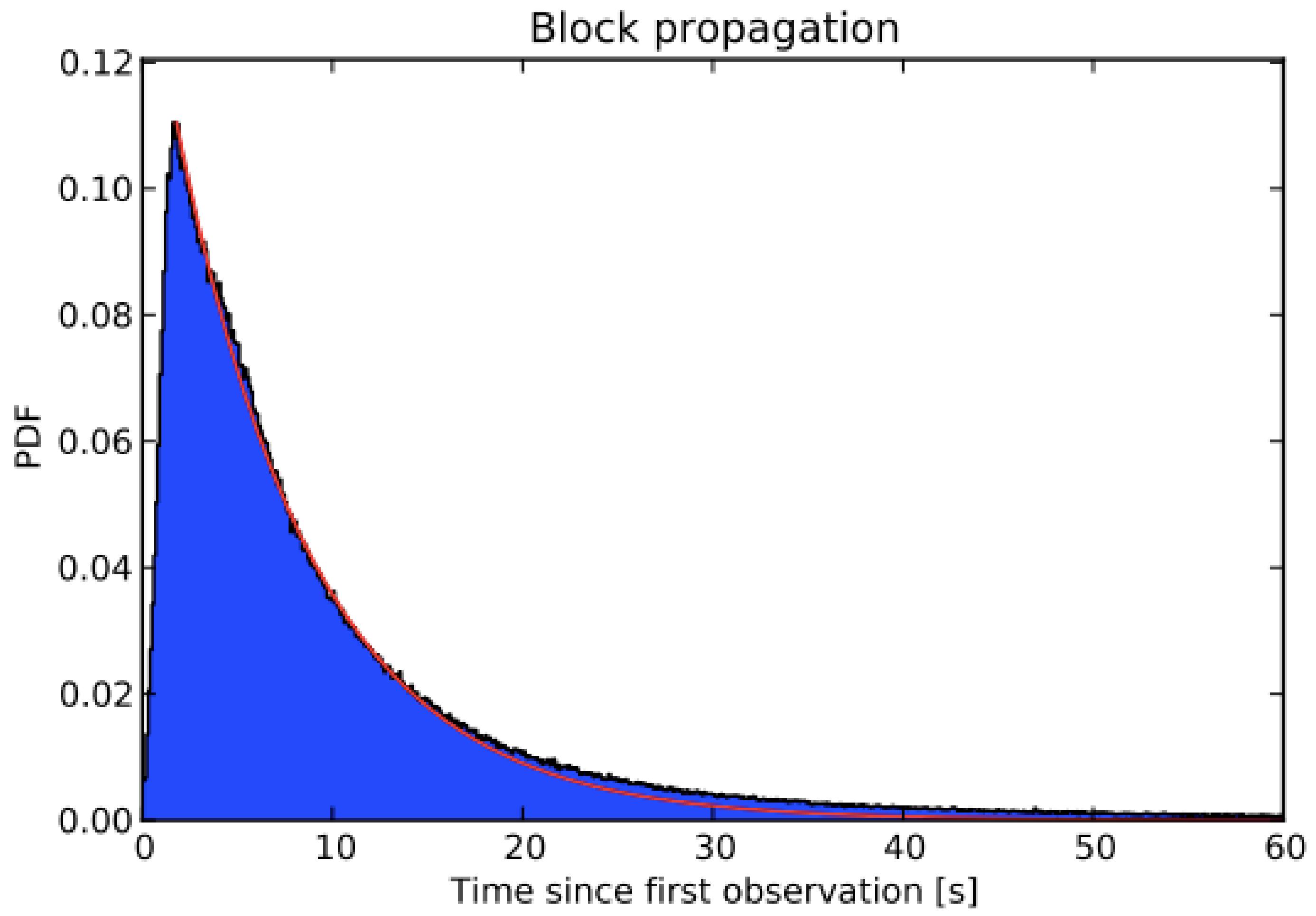


- "Accidental" forks occur rarely. Even if they occur, eventually only one becomes part of the longest chain
- There are "intentional" forks of two type: hard forks and soft forks to come up with new versions like Bitcoin Cash, etc., or to upgrade software versions
- Will be further discussed during our lectures on consensus

Which Block to Relay

- Block contains the correct hash based on the existing blockchain
- All the transactions inside the block are valid
  - Check the scripts
  - Validate with the existing blockchain
- The block is included in the current longest chain
  - Do not relay the forks

Block Propagation Latency



Mean time = 12.6 Seconds

95% of the nodes can see the block within 40 seconds

Decker, Christian, and Roger Wattenhofer. "Information propagation in the bitcoin network." ***2013 IEEE Thirteenth International Conference on Peer-to-Peer Computing (P2P)*****. IEEE, 2013.**

Bitcoin – The Beginning

- “A **decentralized** digital currency enables instant payments to anyone, anywhere in the world” – en.bitcoin.it

- No central authority, uses peer-to-peer technology

- Two broad operations
  - **Transaction Management** – transfer of bitcoins from one user to another
  - **Money Issuance** – regulate the monetary base

Bitcoin Basics – Creation of Coins

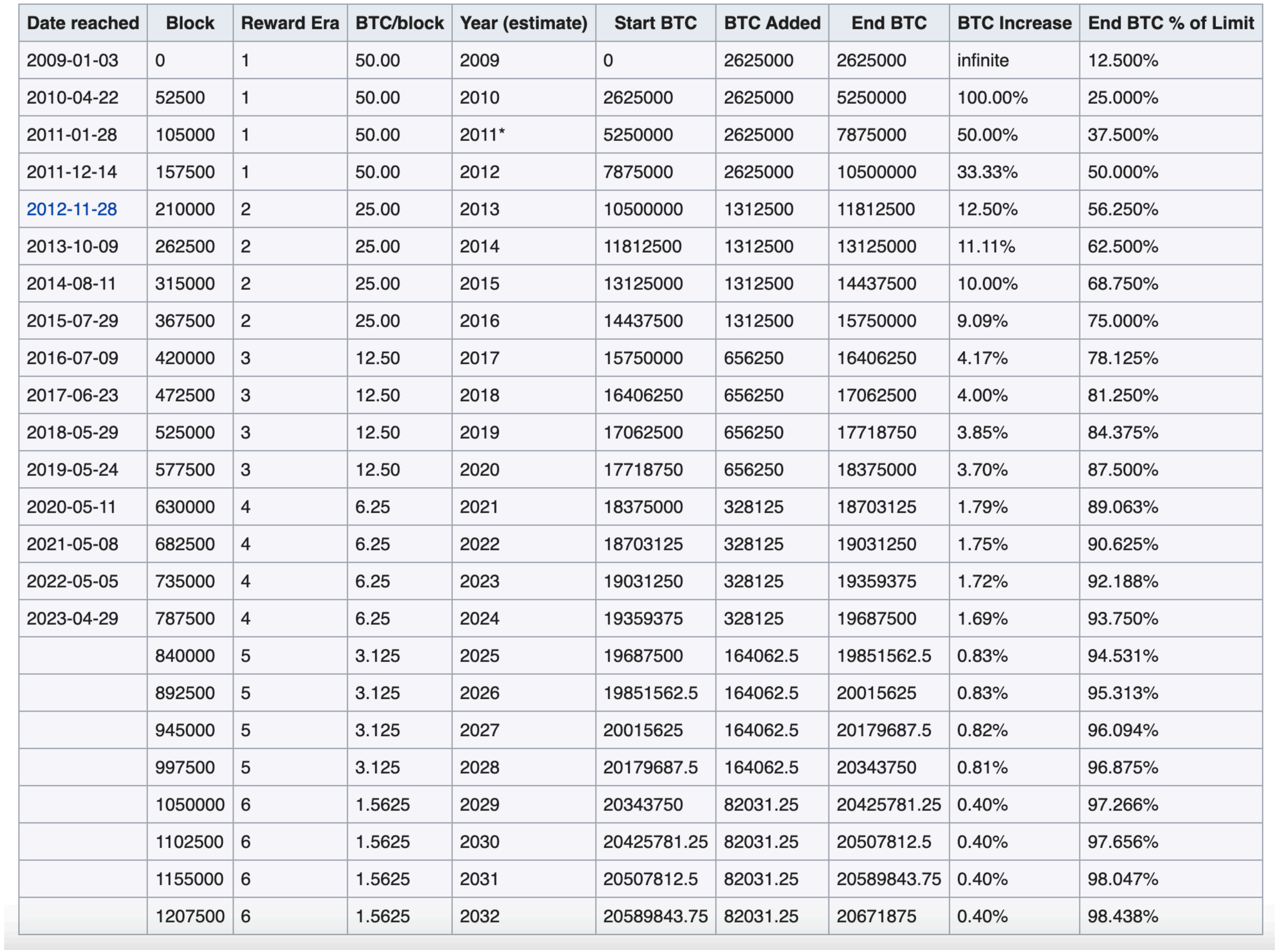
- **Controlled Supply:** Must be limited for the currency to have value – any maliciously generated currency needs to be rejected by the network
- Bitcoins are generated **during the mining** – each time a user discovers a new block
- The rate of block creation is adjusted every 2016 blocks to aim for **a constant two week adjustment period**
- The last bitcoin will be mined in 2140 (estimated and unless changed)

Bitcoin Basics – Creation of Coins

- The number of bitcoins generated per block is set to decrease **geometrically**, with a 50% reduction for every 210,000 blocks, or approximately 4 years

- This reduces, with time, the amount of bitcoins generated per block
  - Theoretical limit for total bitcoins: Slightly less than *21 million*
  - Miners will get less reward as time progresses
  - How to pay the mining fee – increase the transaction fee

Projected Bitcoins



Date reached	Block	Reward Era	BTC/block	Year (estimate)	Start BTC	BTC Added	End BTC	BTC Increase	End BTC % of Limit
2009-01-03	0	1	50.00	2009	0	2625000	2625000	infinite	12.500%
2010-04-22	52500	1	50.00	2010	2625000	2625000	5250000	100.00%	25.000%
2011-01-28	105000	1	50.00	2011*	5250000	2625000	7875000	50.00%	37.500%
2011-12-14	157500	1	50.00	2012	7875000	2625000	10500000	33.33%	50.000%
2012-11-28	210000	2	25.00	2013	10500000	1312500	11812500	12.50%	56.250%
2013-10-09	262500	2	25.00	2014	11812500	1312500	13125000	11.11%	62.500%
2014-08-11	315000	2	25.00	2015	13125000	1312500	14437500	10.00%	68.750%
2015-07-29	367500	2	25.00	2016	14437500	1312500	15750000	9.09%	75.000%
2016-07-09	420000	3	12.50	2017	15750000	656250	16406250	4.17%	78.125%
2017-06-23	472500	3	12.50	2018	16406250	656250	17062500	4.00%	81.250%
2018-05-29	525000	3	12.50	2019	17062500	656250	17718750	3.85%	84.375%
2019-05-24	577500	3	12.50	2020	17718750	656250	18375000	3.70%	87.500%
2020-05-11	630000	4	6.25	2021	18375000	328125	18703125	1.79%	89.063%
2021-05-08	682500	4	6.25	2022	18703125	328125	19031250	1.75%	90.625%
2022-05-05	735000	4	6.25	2023	19031250	328125	19359375	1.72%	92.188%
2023-04-29	787500	4	6.25	2024	19359375	328125	19687500	1.69%	93.750%
	840000	5	3.125	2025	19687500	164062.5	19851562.5	0.83%	94.531%
	892500	5	3.125	2026	19851562.5	164062.5	20015625	0.83%	95.313%
	945000	5	3.125	2027	20015625	164062.5	20179687.5	0.82%	96.094%
	997500	5	3.125	2028	20179687.5	164062.5	20343750	0.81%	96.875%
	1050000	6	1.5625	2029	20343750	82031.25	20425781.25	0.40%	97.266%
	1102500	6	1.5625	2030	20425781.25	82031.25	20507812.5	0.40%	97.656%
	1155000	6	1.5625	2031	20507812.5	82031.25	20589843.75	0.40%	98.047%
	1207500	6	1.5625	2032	20589843.75	82031.25	20671875	0.40%	98.438%

Information Source: https://en.bitcoin.it/wiki/Controlled_supply

Bitcoin Basics – Sending Payments

- Alice wants to send bitcoin to Bob
  - Bob sends his address to Alice
  - Alice adds Bob's address and the amount of bitcoins to transfer in a "transaction" message
  - Alice signs the transaction with her private key, and announces her public key for signature verification
  - Alice broadcasts the transaction on the Bitcoin network for all to see

Handle Double Spending using Blockchain

- When multiple valid continuation to this chain appear, only the longest such branch is accepted and it is then extended further **(longest chain)**
- Once a transaction is committed in the blockchain, everyone in the network can validate all the transactions by using Alice’s public address
- The validation prevents double spending in bitcoin

Bitcoin Anonymity

- Bitcoin is permission-less, you do not need to setup any “account”, or required any e-mail address, user name or password to login to the wallet

- The public and the private keys do not need to be registered, the wallet can generate them for the users

- The **bitcoin address** is used for transaction, not the user name or identity
- A **bitcoin address** mathematically corresponds to a public key based on ECDSA – the digital signature algorithm used in bitcoin
- A sample bitcoin address: 1PHYrmdJ22MKbJevpb3MBNpVckjZHt89hz
- Each person can have many such addresses, each with its own balance
  - Difficult to know which person owns what amount

To Sum it All Up!!

- Bitcoins do not really “exist” as any tangible or electronic object.
- There is no bit”coin” as you see in its logo

- Owning a bitcoin simply means you have access to a key pair that includes
  - A public key to which somebody else had sent some bitcoin
  - A matching private key that gives you the authority to send the previously received bitcoin to another address

- If you lose your private key, you lose the corresponding bitcoin(s)

Physical Payment using Bitcoin

- All that is needed is a (set of) private key(s) – Public key can be generated from the private key.
- Safely store the private key – in your desktop, on the web, mobile phone, special hardware attachment, printed on a piece of paper as QR
- For online payment, you can use the wallet and an appropriate mode of applying the private key
- For offline payments like in store payments or paying to your friend, you can use your mobile phone to present the private key or use the hardcopy!! As simple as using PayTM, Google Pay and so on.

Bitcoin Exchange

- Trading bitcoin as commodity
- Centralized exchanges – (In India: WazirX, CoinDCX, Zebpay, CoinSwitch Kuber, etc.)
  - Identity verification using KYC documents
  - Maintain your balance in Bitcoin and another currency like INR.
  - You set the buying and selling prices and quantities
  - If necessary, you can take the money out in a preferred currency
  - Some exchanges provide the payout option in anonymous prepaid cards
- There can also be decentralized exchanges with appropriate procedures for handling similar requirements

Consensus in Permissionless Settings

Permissionless Model

- Open network
  - Anyone can join in the network and initiate transactions
  - Participants are free to leave the network, and can join later again

Permissionless Model

Open network

- Anyone can join in the network and initiate transactions
- Participants are free to leave the network, and can join later again

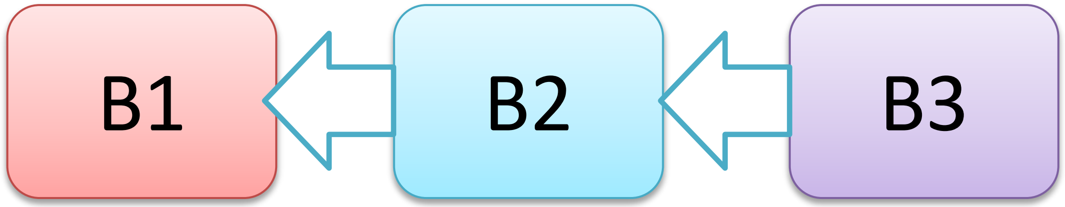
Assumption: More than 50% of the participants are honest

- A society cannot run if majority of its participants are dishonest !!

Consensus in a Permissionless Model - Challenges

- Participants do not know others
  - Cannot use message passing !!
- Anyone can propose a new block
  - Who is going to add the next block in the blockchain?
- The network is asynchronous
  - We do not have any global clock
  - Theoretically, a node may see the blocks in different orders

Consensus in a Permissionless Model - Challenges



Participants do not know others

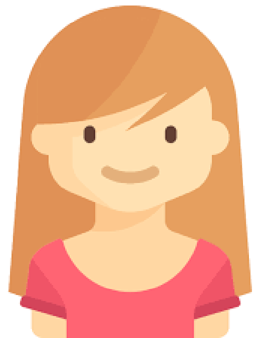
- Cannot use message passing !!

Anyone can propose a new block

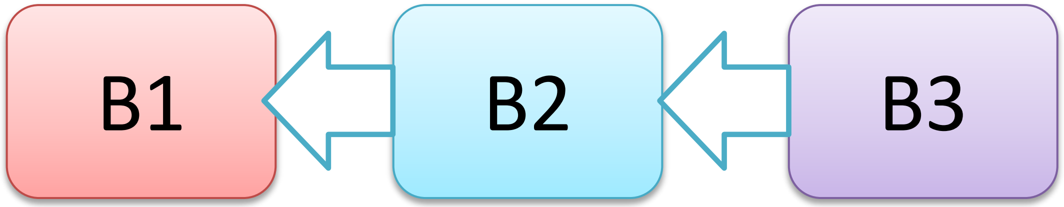
- Who is going to add the next block in the blockchain?

The network is asynchronous

- We do not have any global clock
- Theoretically, a node may see the messages in different orders



Consensus in a Permissionless Model - Challenges



Participants do not know others

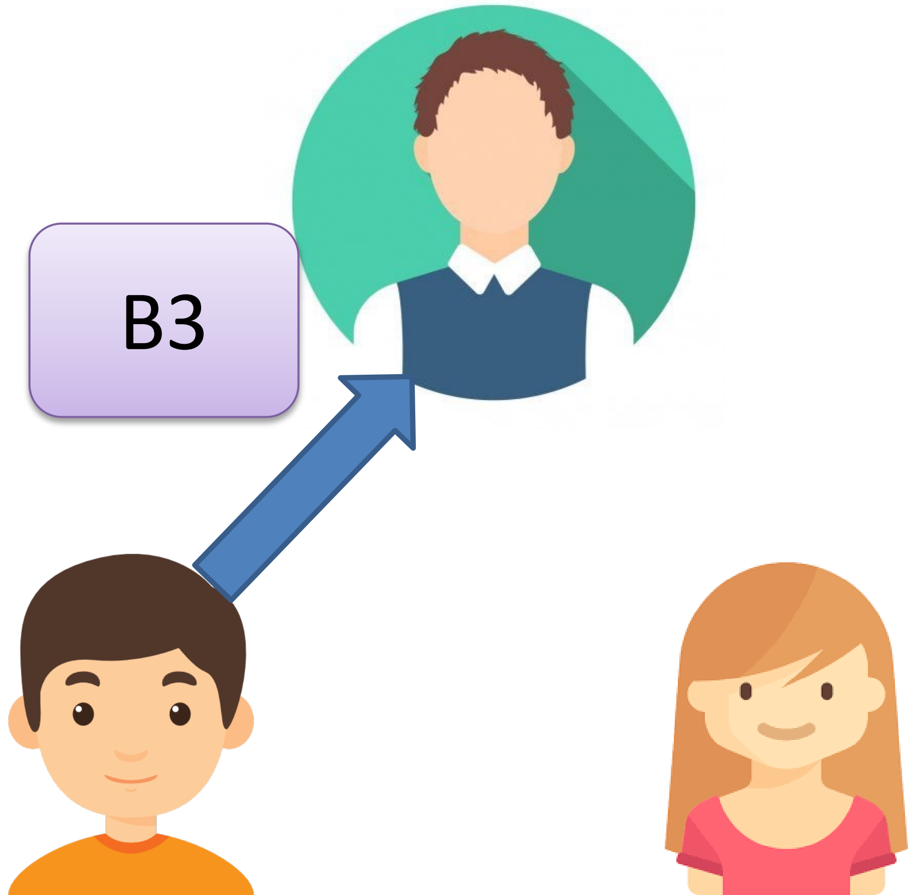
- Cannot use message passing !!

Anyone can propose a new block

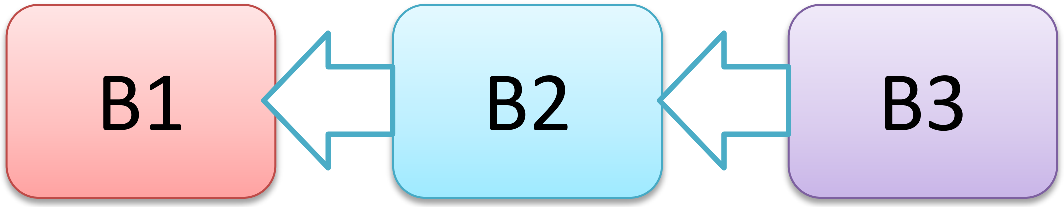
- Who is going to add the next block in the blockchain?

The network is asynchronous

- We do not have any global clock
- Theoretically, a node may see the messages in different orders



Consensus in a Permissionless Model - Challenges



Participants do not know others

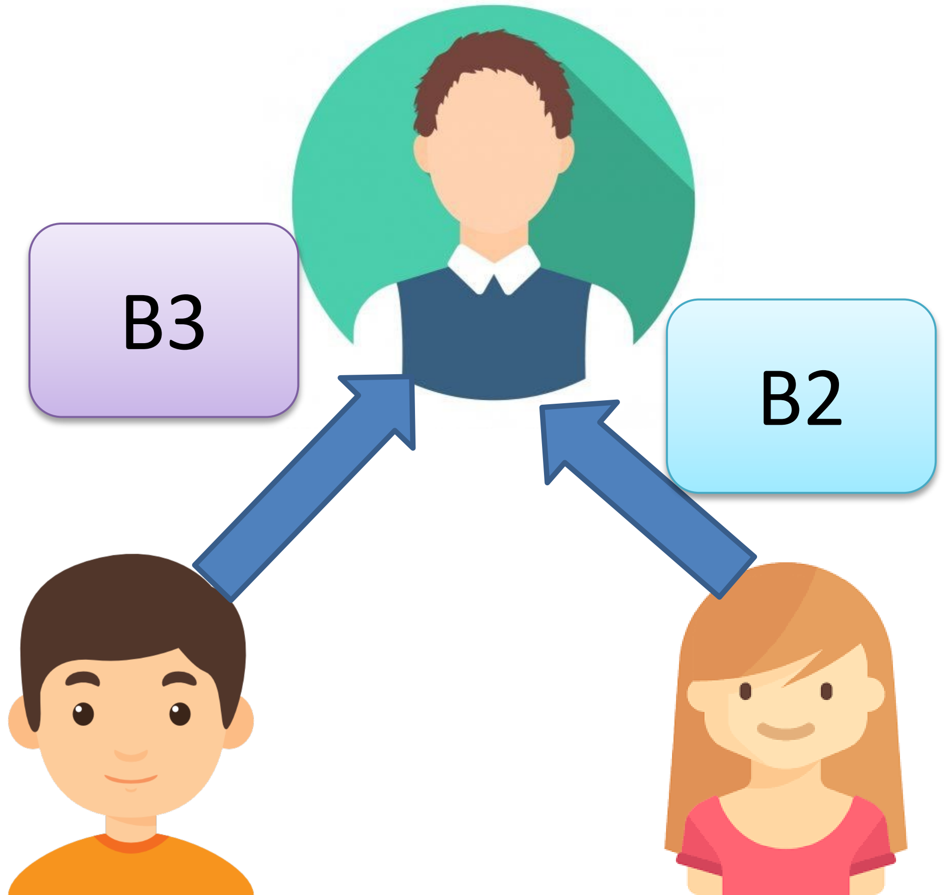
- Cannot use message passing !!

Anyone can propose a new block

- Who is going to add the next block in the blockchain?

The network is asynchronous

- We do not have any global clock
- Theoretically, a node may see the messages in different orders



Consensus in a Permissionless Model - Challenges

- Participants do not know others
  - Cannot use message passing !!
- Anyone can propose a new block
  - Who is going to add the next block in the blockchain?
- The network is asynchronous
  - We do not have any global clock
  - Theoretically, a node may see the messages in different orders
- Any types of monopoly needs to be prevented
  - A single user or a group of users should not gain the control – we don't trust anyone

Remember the FLP Impossibility?

Synchronous vs Asynchronous Networks

- **Synchronous**: I am sure that I'll get the message within a predefined time threshold
- **Asynchronous**: I am not sure whether and when the message will arrive

Failures in a network --

- **Crash Fault**: A node stops responding
- **Link Fault** (or Network Fault): A link fails to deliver the message
- **Byzantine Fault**: A node starts behaving maliciously

The Impossibility Theorem: Consensus is not possible in a perfect asynchronous network even with a single crash failure

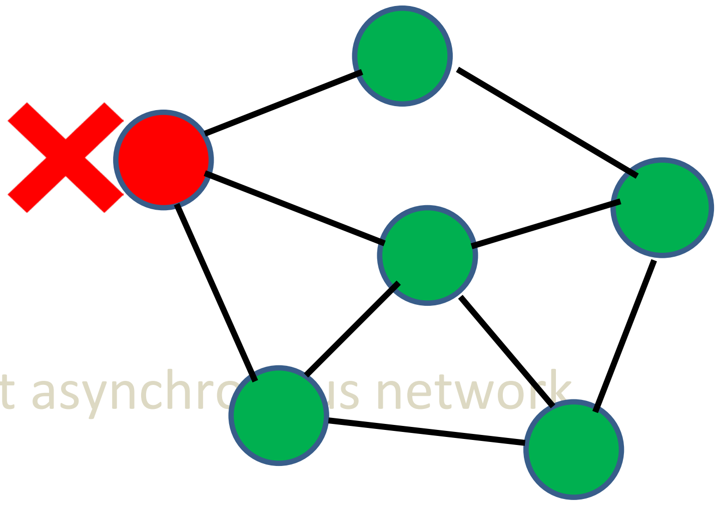
Remember the FLP Impossibility?

Synchronous vs Asynchronous Networks

- **Synchronous**: I am sure that I'll get the message within a predefined time threshold
- **Asynchronous**: I am not sure whether and when the message will arrive

Failures in a network --

- **Crash Fault**: A node stops responding
- **Link Fault** (or Network Fault): A link fails to deliver the message
- **Byzantine Fault**: A node starts behaving maliciously



The Impossibility Theorem: Consensus is not possible in a perfect asynchronous network even with a single crash failure

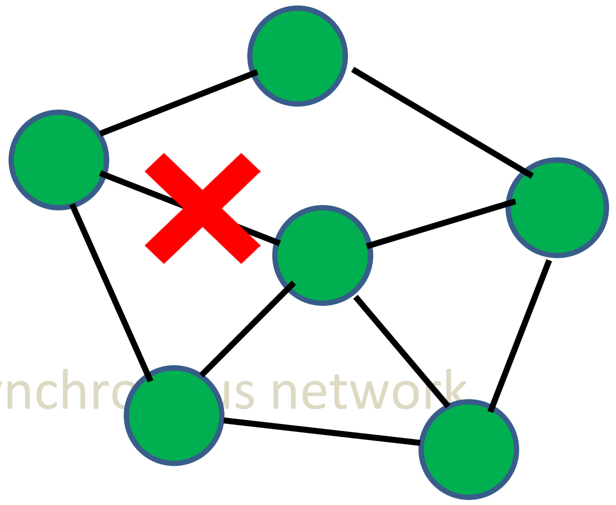
Remember the FLP Impossibility?

Synchronous vs Asynchronous Networks

- **Synchronous**: I am sure that I'll get the message within a predefined time threshold
- **Asynchronous**: I am not sure whether and when the message will arrive

Failures in a network --

- **Crash Fault**: A node stops responding
- **Link Fault** (or Network Fault): A link fails to deliver the message
- **Byzantine Fault**: A node starts behaving maliciously



The Impossibility Theorem: Consensus is not possible in a perfect asynchronous network even with a single crash failure

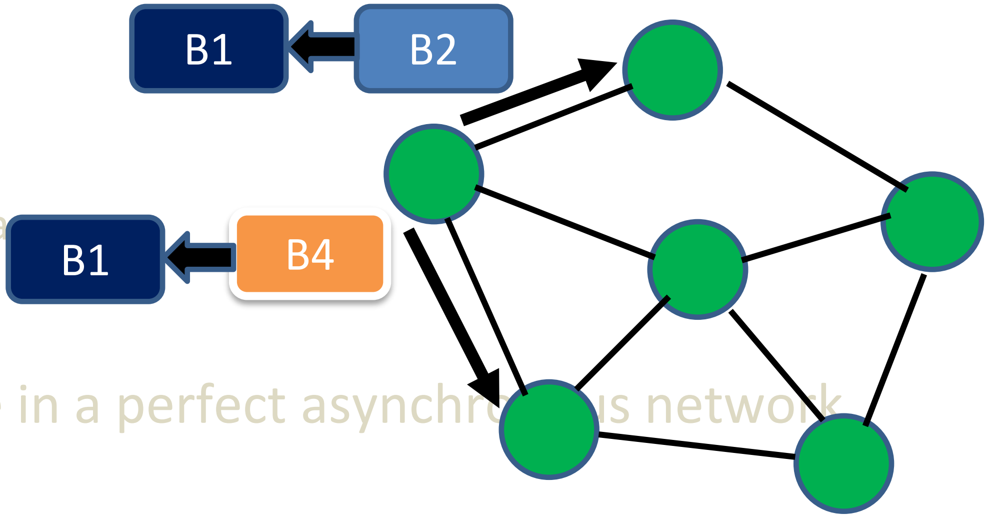
Remember the FLP Impossibility?

Synchronous vs Asynchronous Networks

- **Synchronous**: I am sure that I'll get the message within a predefined time threshold
- **Asynchronous**: I am not sure whether and when the message will arrive

Failures in a network --

- **Crash Fault**: A node stops responding
- **Link Fault** (or Network Fault): A link fails to deliver the messa
- **Byzantine Fault**: A node starts behaving maliciously



The Impossibility Theorem: Consensus is not possible in a perfect asynchr us network even with a single crash failure

Remember the FLP Impossibility?

Synchronous vs Asynchronous Networks

- **Synchronous**: I am sure that I'll get the message within a predefined time threshold
- **Asynchronous**: I am not sure whether and when the message will arrive

Failures in a network --

- **Crash Fault**: A node stops responding
- **Link Fault** (or Network Fault): A link fails to deliver the message
- **Byzantine Fault**: A node starts behaving maliciously

The Impossibility Theorem: Consensus is not possible in a perfect asynchronous network even with a single crash failure

- Cannot ensure safety and liveness simultaneously

Safety vs Liveness Dilemma

Synchronous vs Asynchronous Networks

- **Synchronous:** I am sure that I'll get the message within a predefined time threshold
- **Asynchrono**

Failures in a ne

- **Crash Fault:**
- **Link Fault** (o
- **Byzantine Fa**

The Impossibi hronous network even with a single crash failure

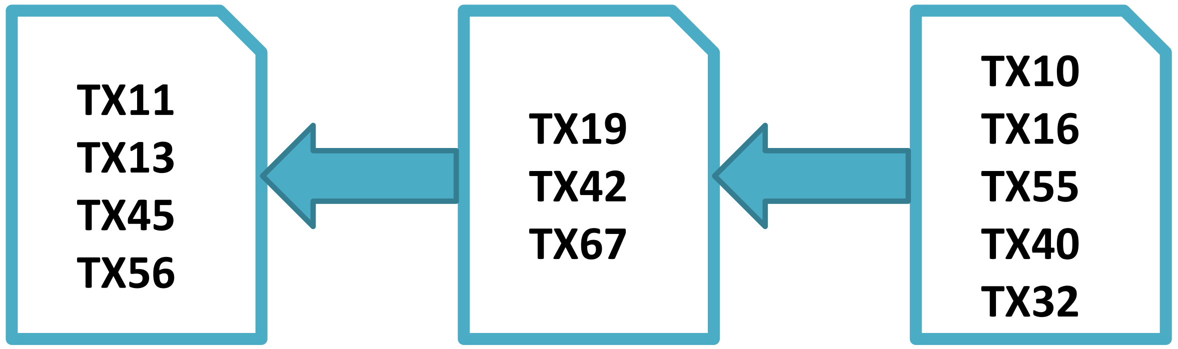
- Cannot ensure safety and liveness simultaneously

The Nakamoto Consensus (Proof of Work)

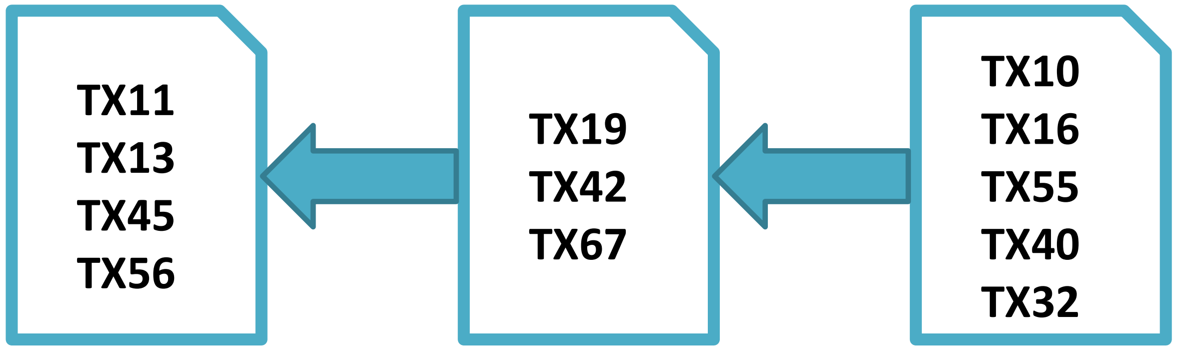
Liveness is more important than Safety

Immediate focus is on liveness with a minimum safety guarantee, full safety will be ensured eventually

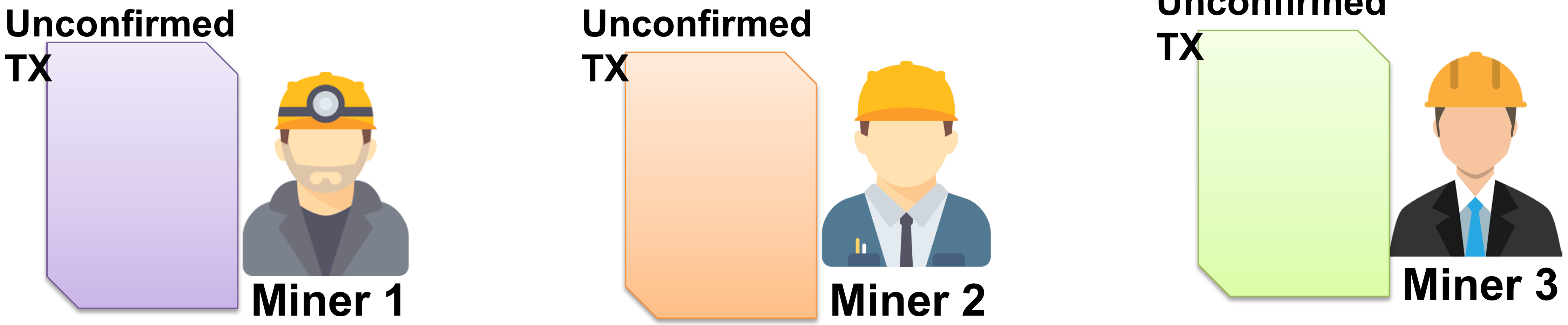
Breaking the "Safety vs Liveness" Dilemma



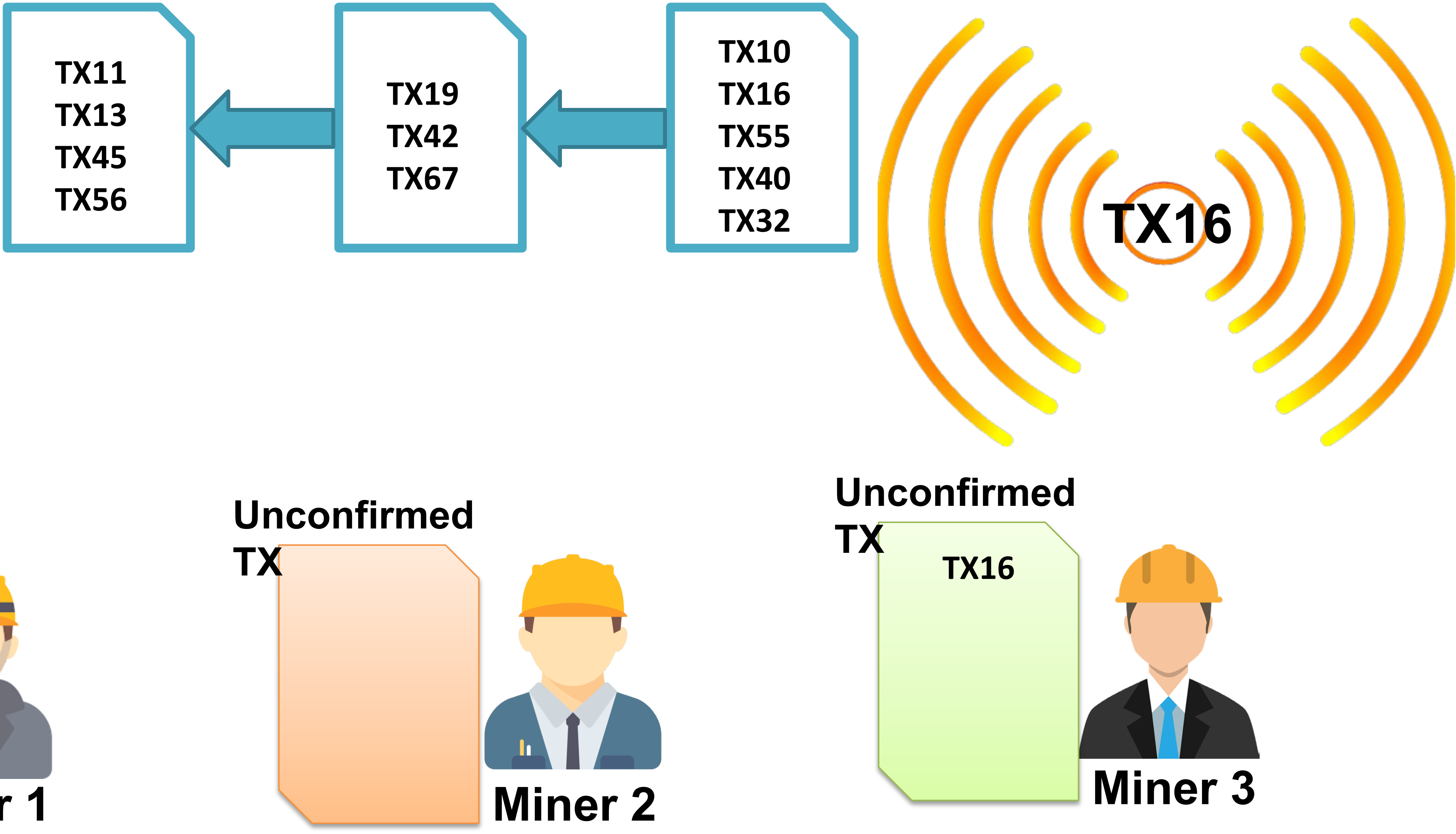
Breaking the "Safety vs Liveness" Dilemma



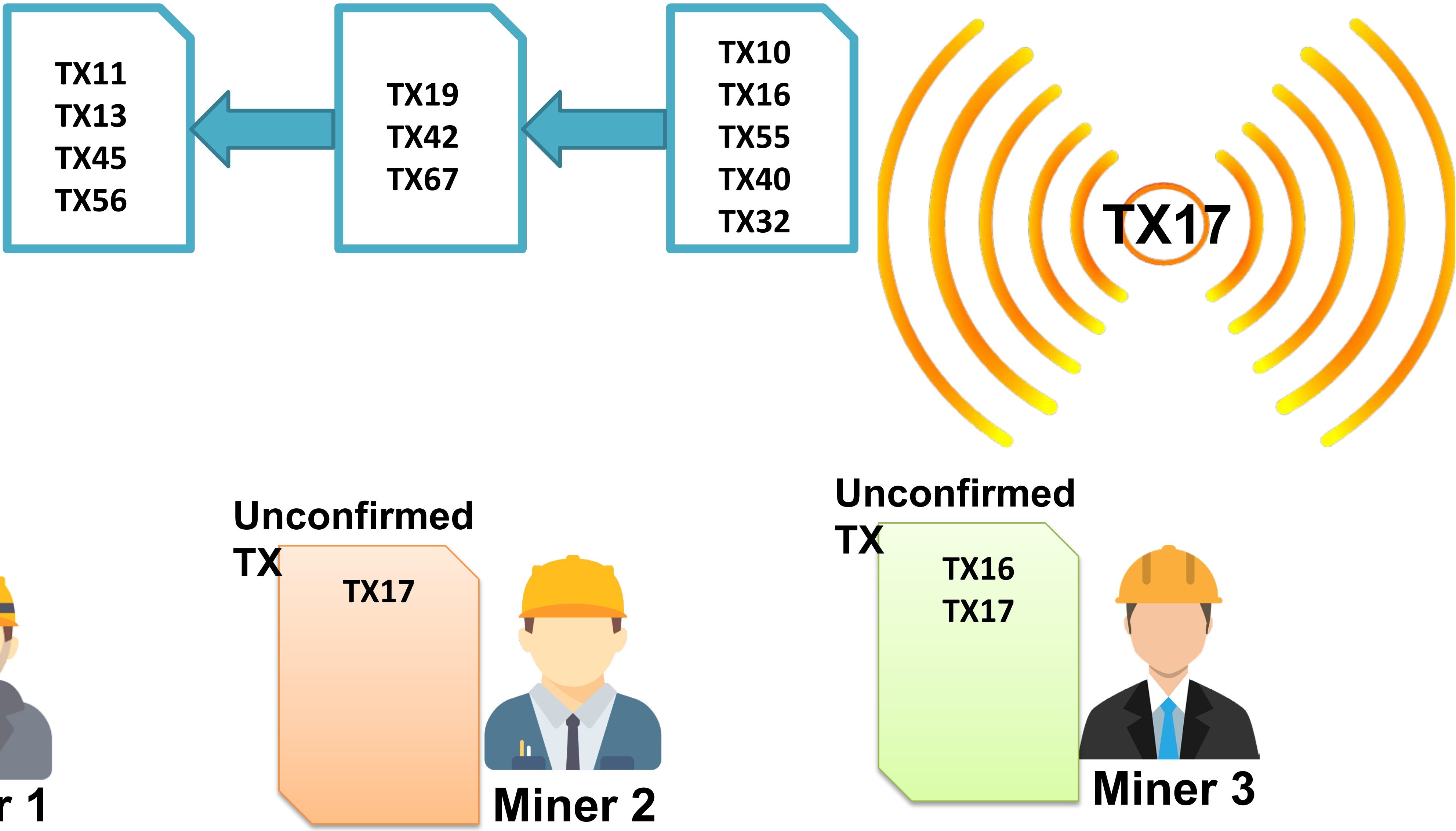
Bitcoin Unconfirmed TX (mempool) : https://www.blockchain.com/btc/unconfirmed-transactions



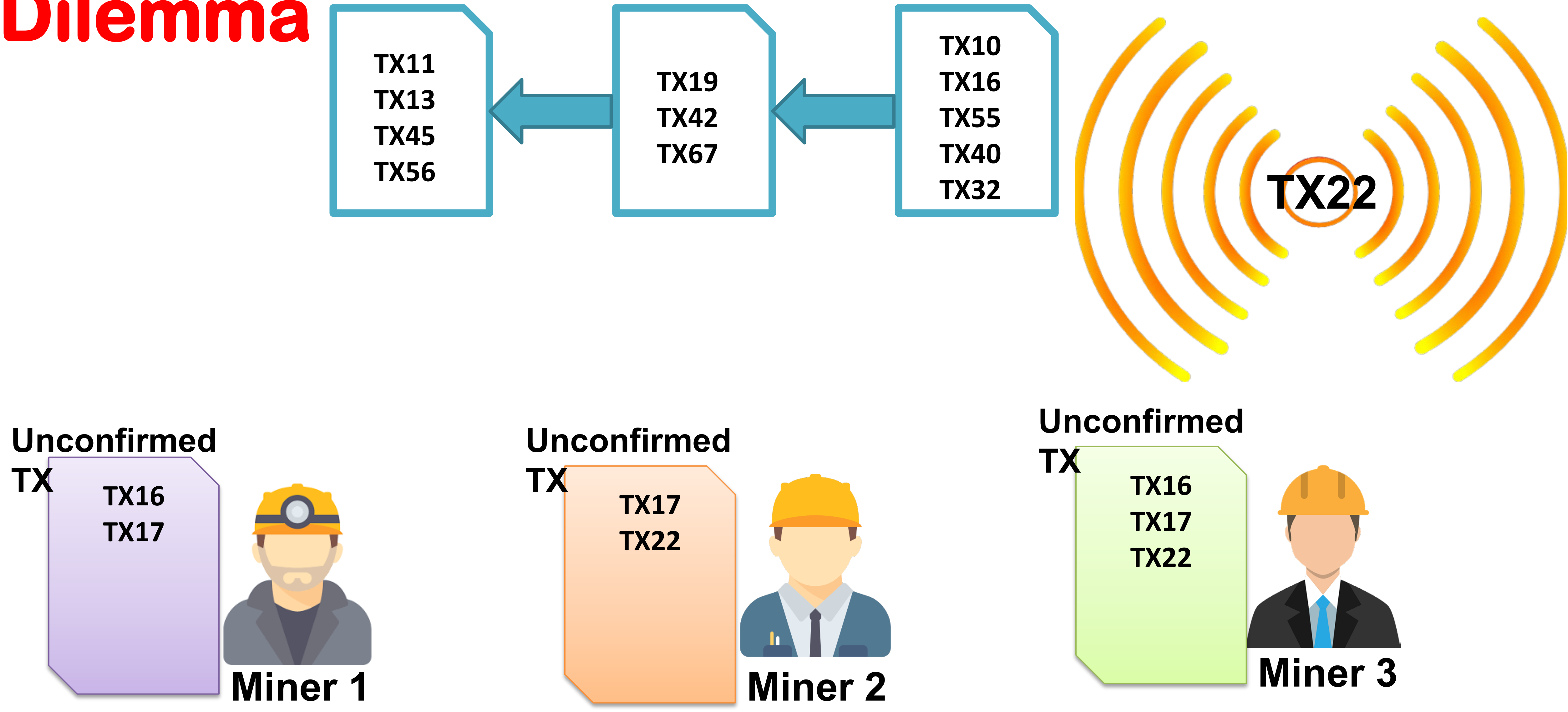
Breaking the "Safety vs Liveness" Dilemma



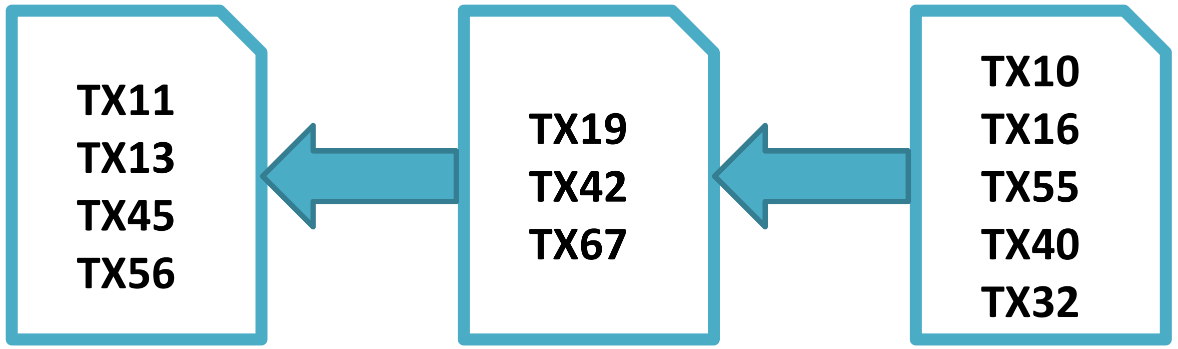
Breaking the "Safety vs Liveness" Dilemma

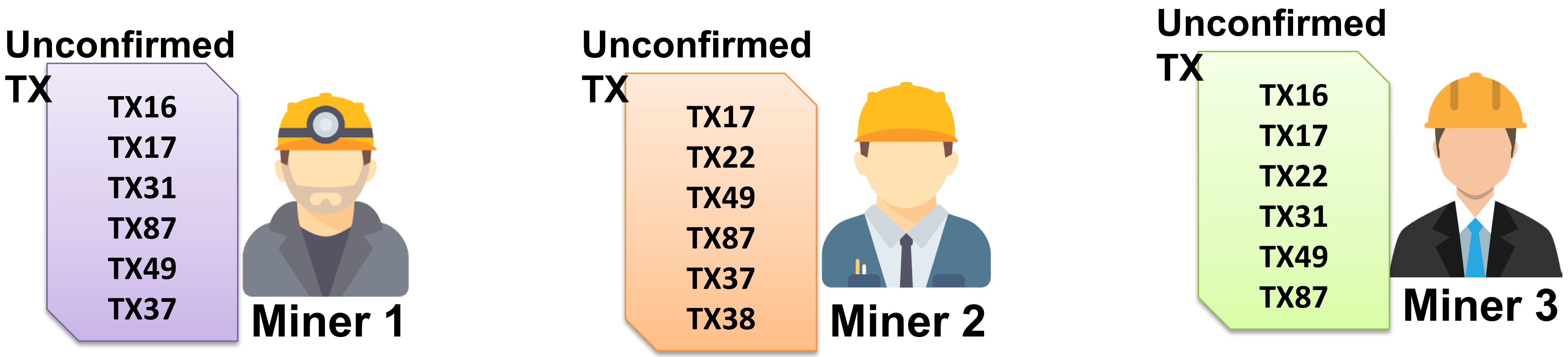


Breaking the "Safety vs Liveness" Dilemma

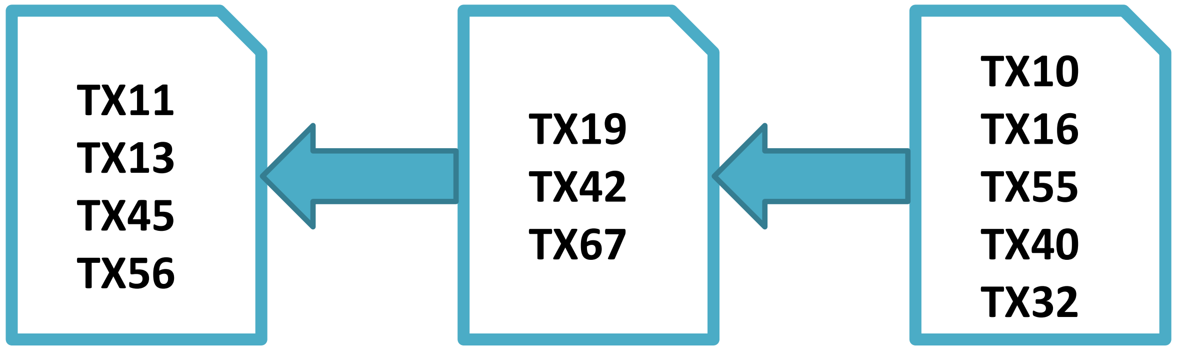


Breaking the "Safety vs Liveness" Dilemma



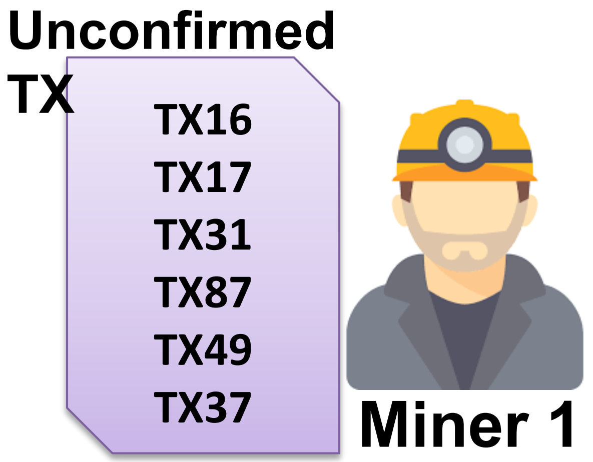


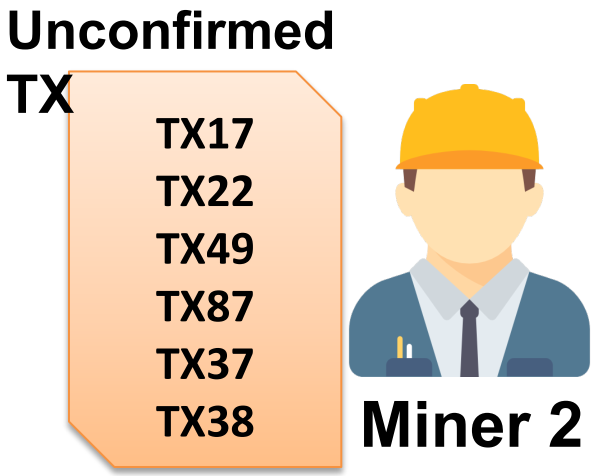
Breaking the "Safety vs Liveness" Dilemma

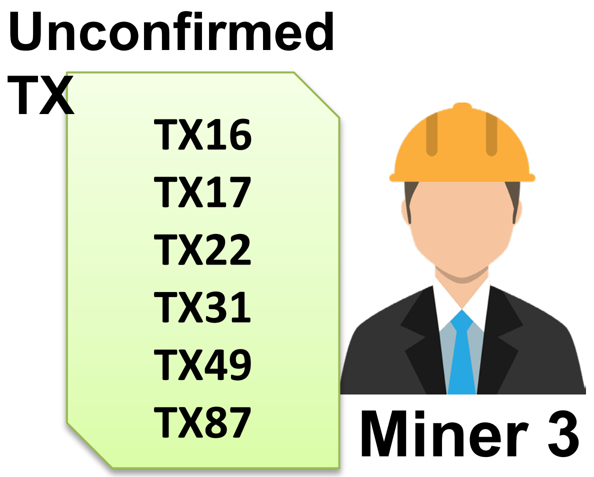


Safety-1: The next block should be "correct" in practice

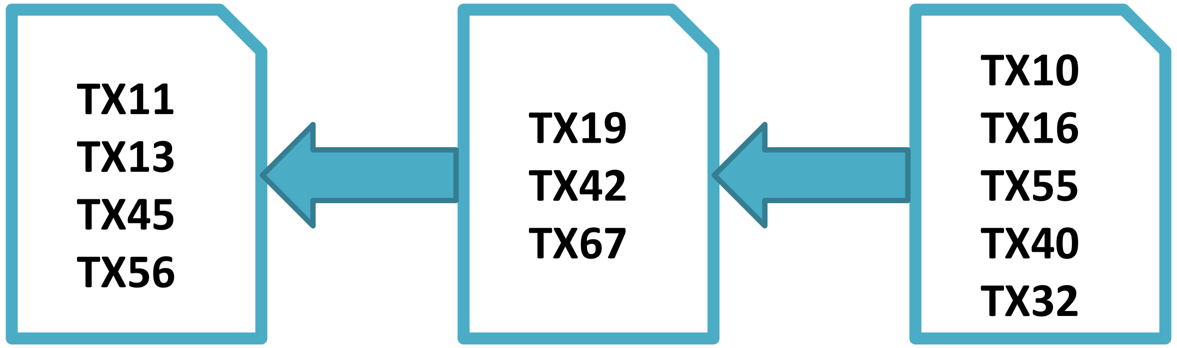
- Transactions are verified, the block contains the correct Hash and Nonce







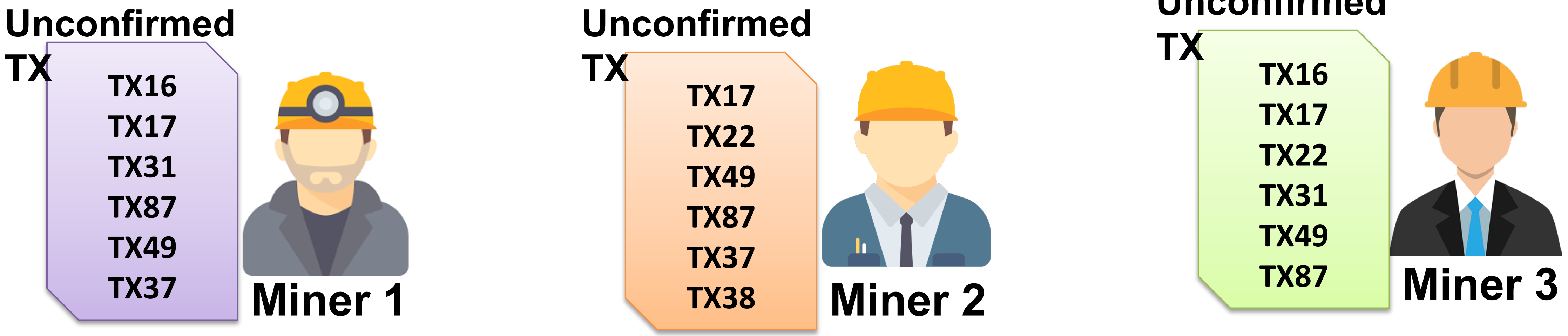
Breaking the "Safety vs Liveness" Dilemma



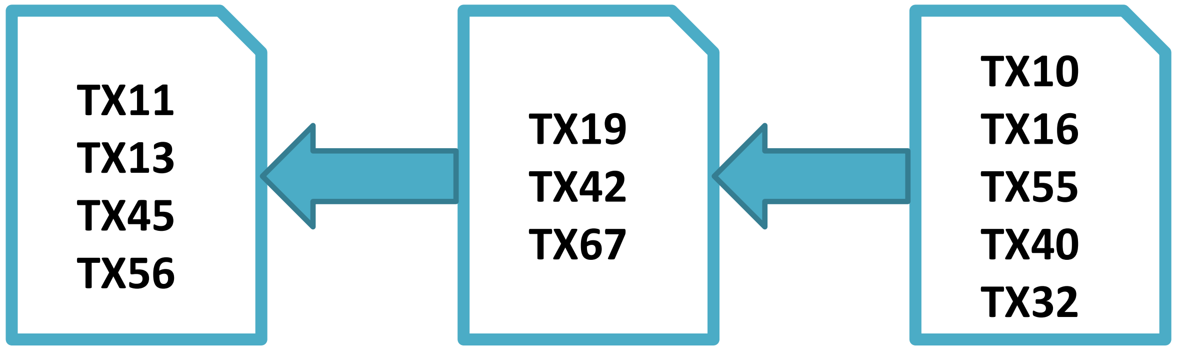
This can be ensured – the block mined by a miner is verified by all

Safety-1: The next block should be "correct" in practice

- Transactions are verified, the block contains the correct Hash and Nonce

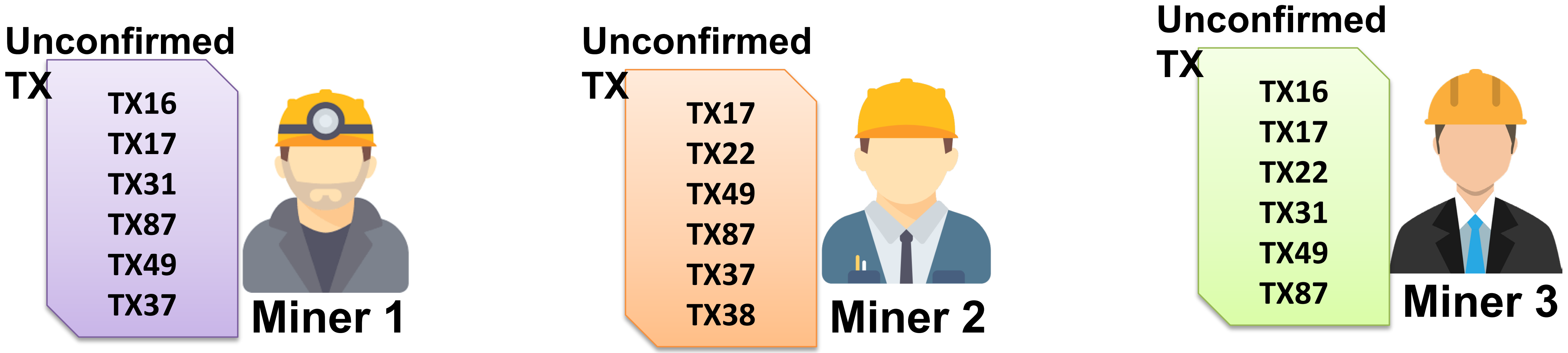


Breaking the "Safety vs Liveness" Dilemma

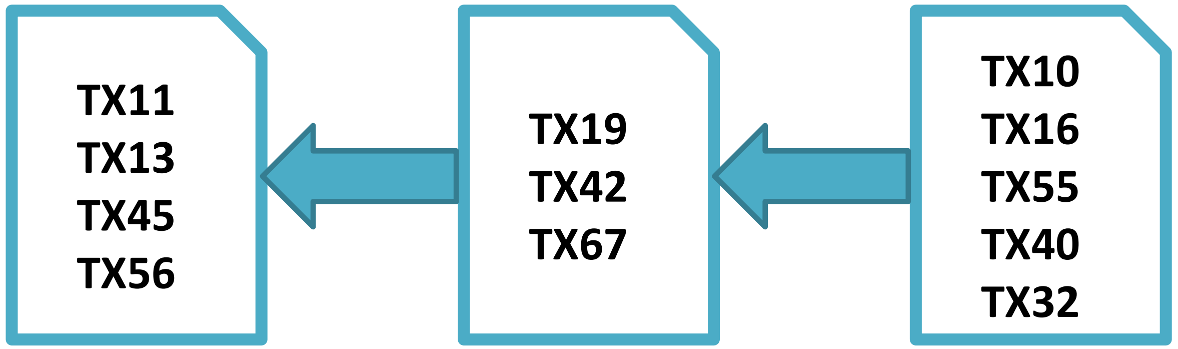


Safety-2: All the miners should agree on a single block

- The next block of the blockchain should be selected unanimously



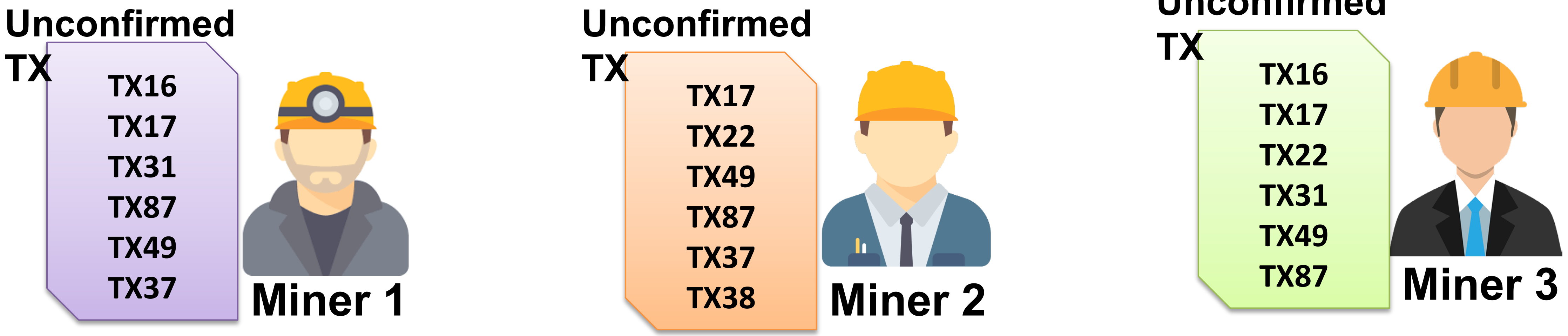
Breaking the "Safety vs Liveness" Dilemma



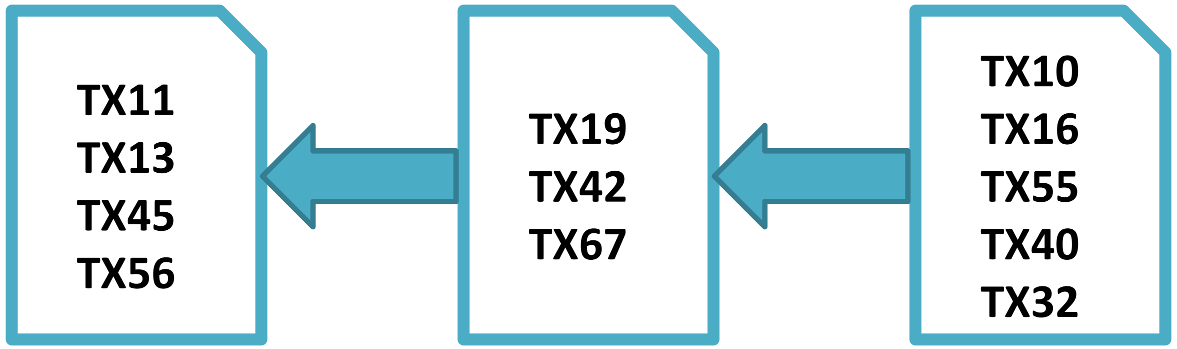
Miners do not know each other – how can they agree on the same block?

Safety-2: All the miners should agree on a single block

- The next block of the blockchain should be selected unanimously



Breaking the "Safety vs Liveness" Dilemma



We compromise here!

Safety-2: All the miners should agree on a single block

- The next block of the blockchain should be selected unanimously

Unconfirmed TX

TX16
TX17
TX31
TX87
TX49
TX37

Miner 1

Unconfirmed TX

TX17
TX22
TX49
TX87
TX37
TX38

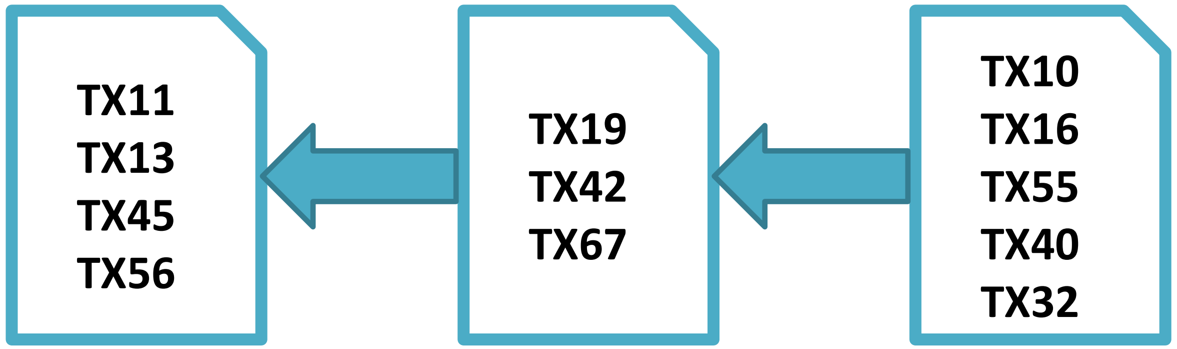
Miner 2

Unconfirmed TX

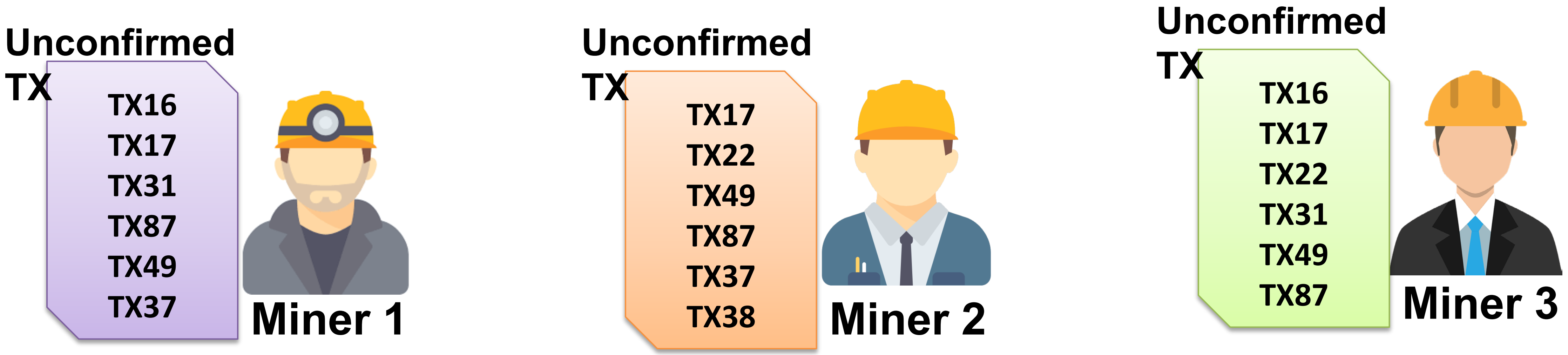
TX16
TX17
TX22
TX31
TX49
TX87

Miner 3

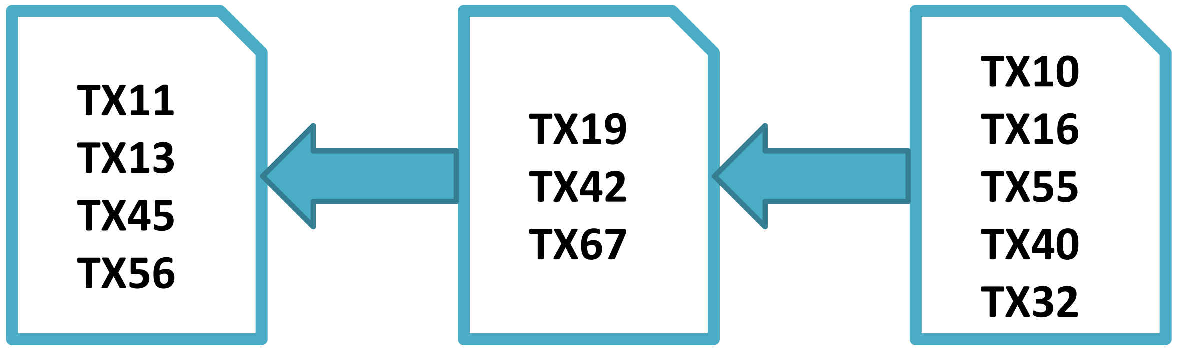
Breaking the "Safety vs Liveness" Dilemma



Liveness: Add a block as long as it is correct (contains valid transactions from the unconfirmed TX list) and move further

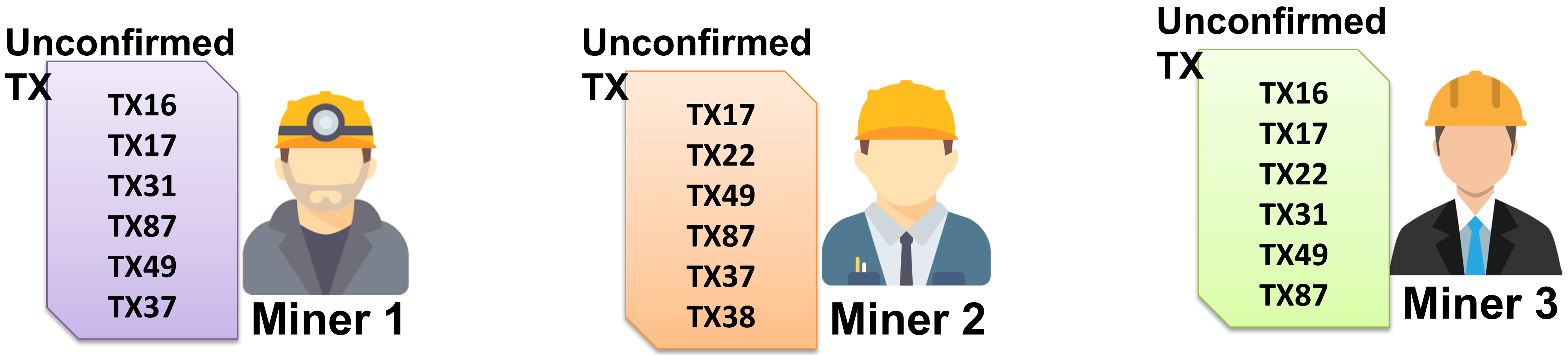


Breaking the "Safety vs Liveness" Dilemma

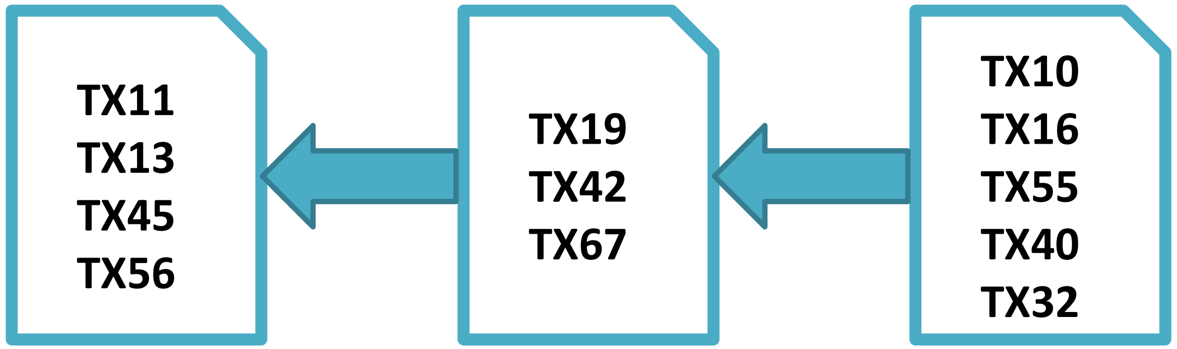


Two (or more) different miners may add two (or more) different blocks

Liveness: Add a block as long as it is correct (contains valid transactions from the unconfirmed TX list) and move further



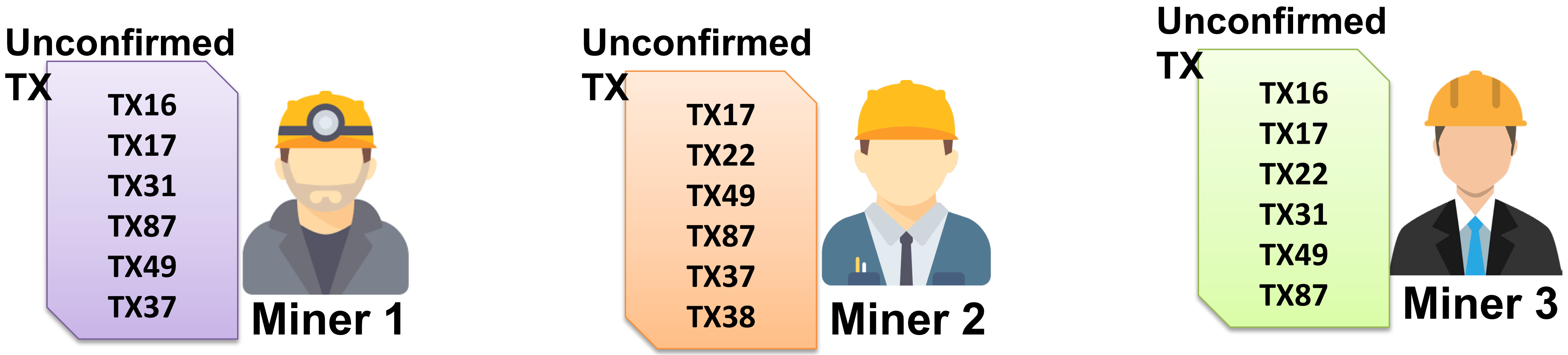
Breaking the "Safety vs Liveness" Dilemma



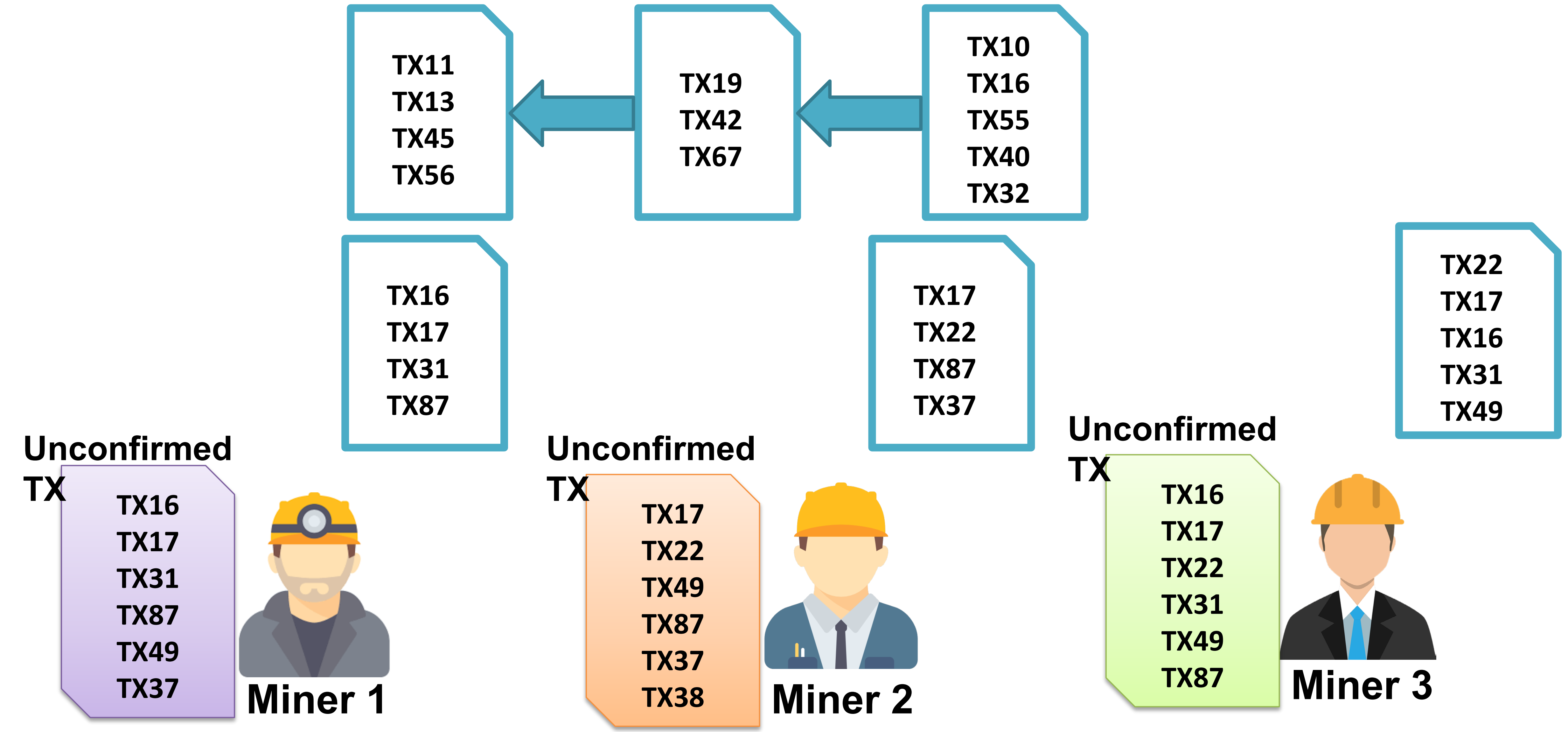
Two (or more) different miners may add two (or more) different blocks

Will resolve this later!

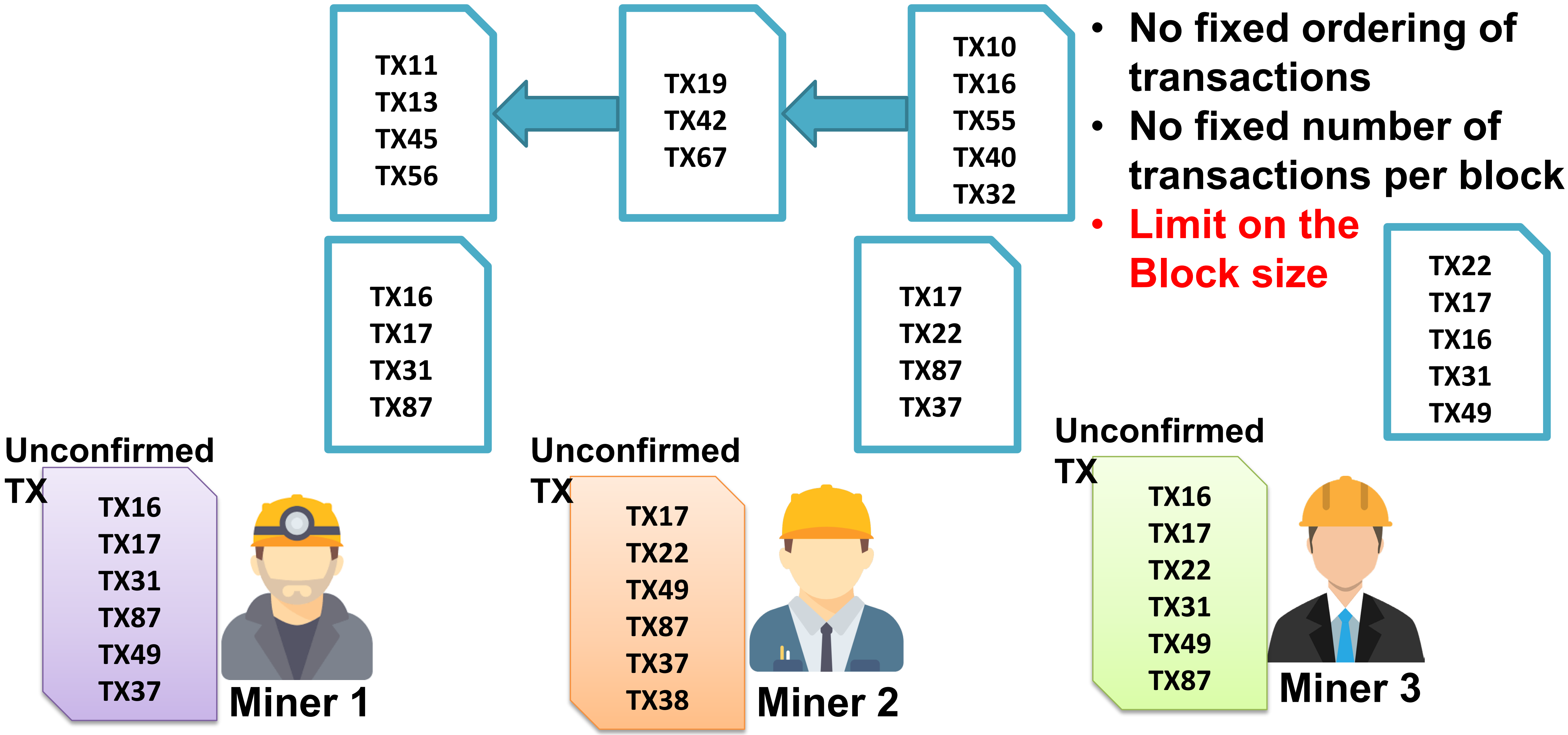
Liveness: Add a block as long as it is correct (contains valid transactions from the unconfirmed TX list) and move further



Nakamoto Consensus (Proof of Work)

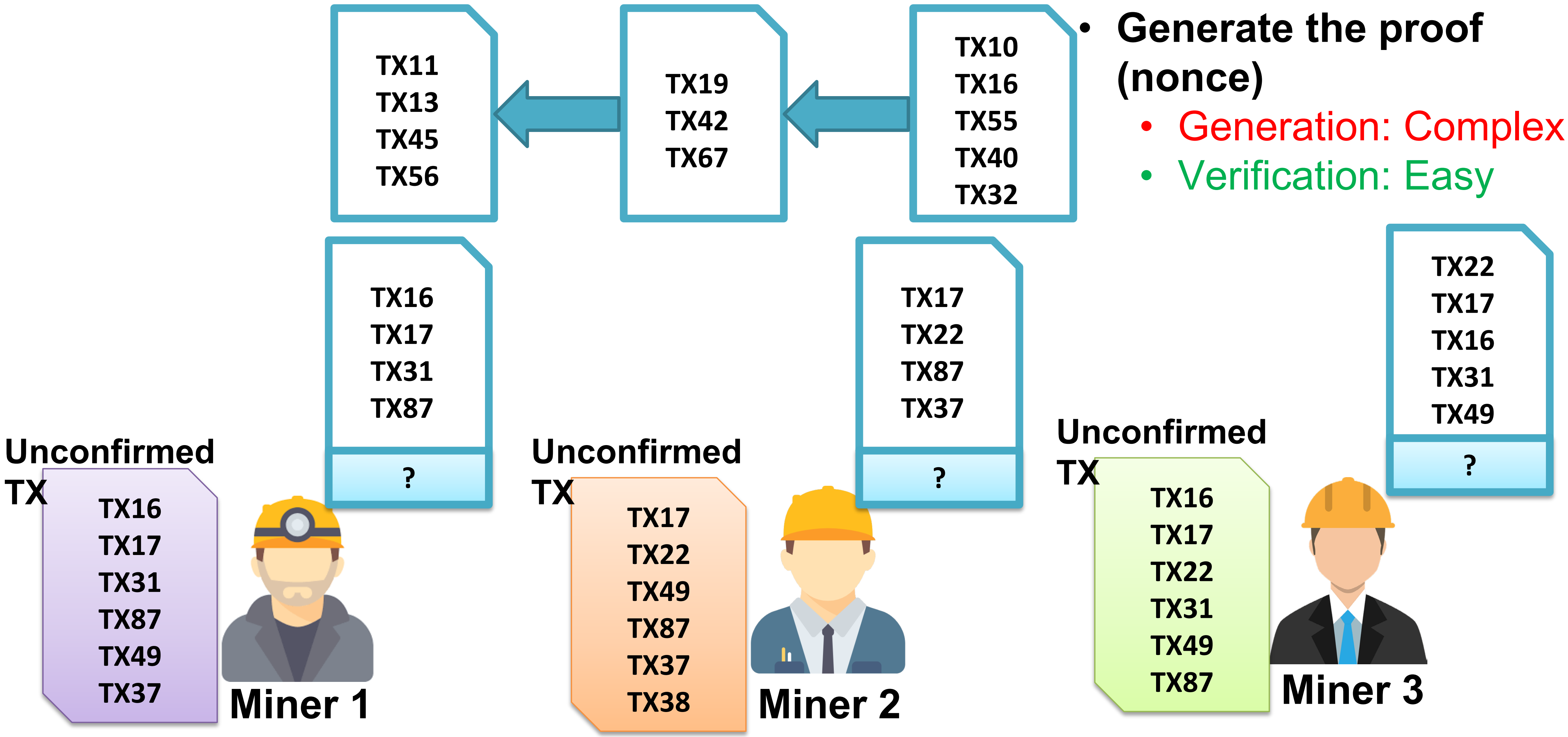


Nakamoto Consensus (Proof of Work)



- No fixed ordering of transactions
- No fixed number of transactions per block
- Limit on the Block size

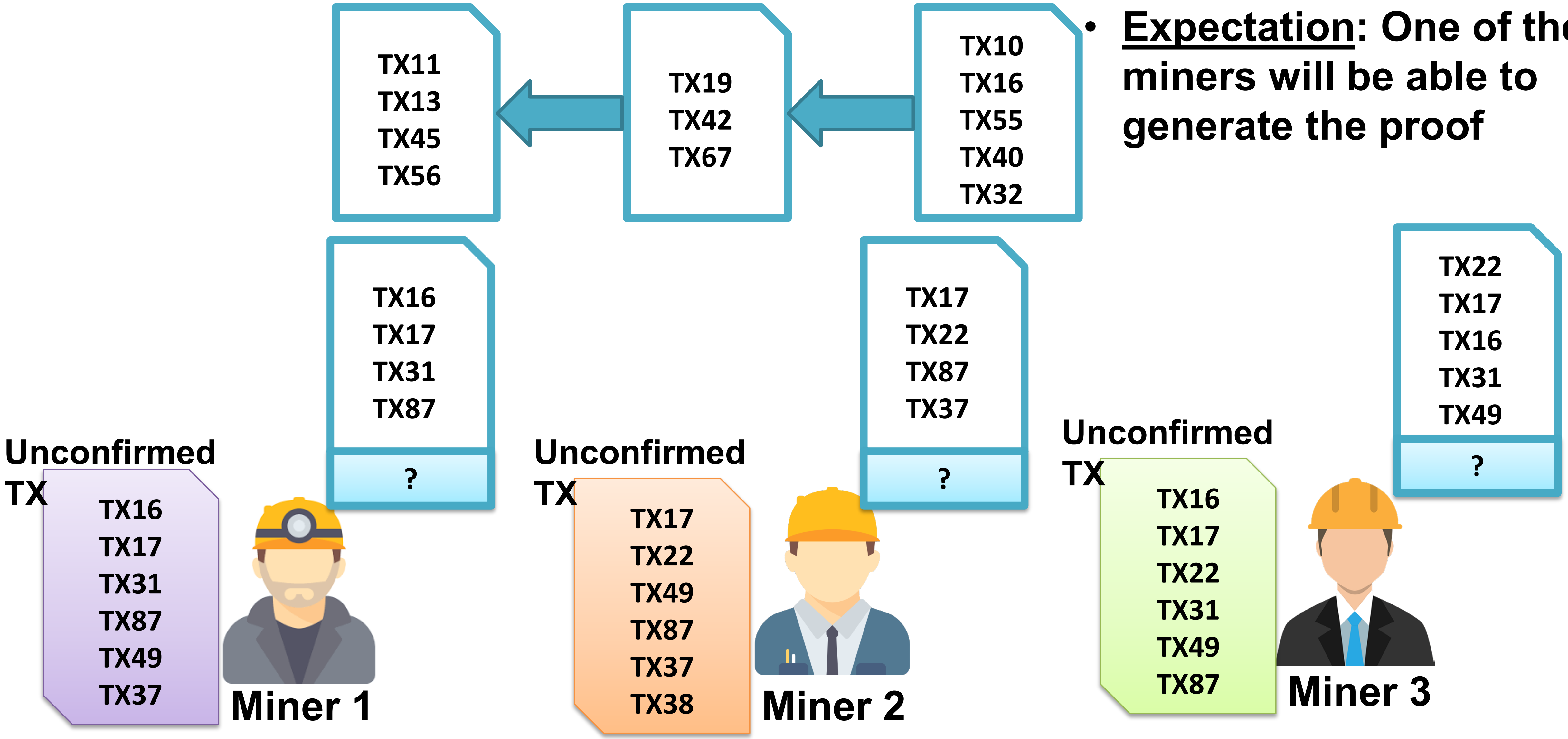
Nakamoto Consensus (Proof of Work)



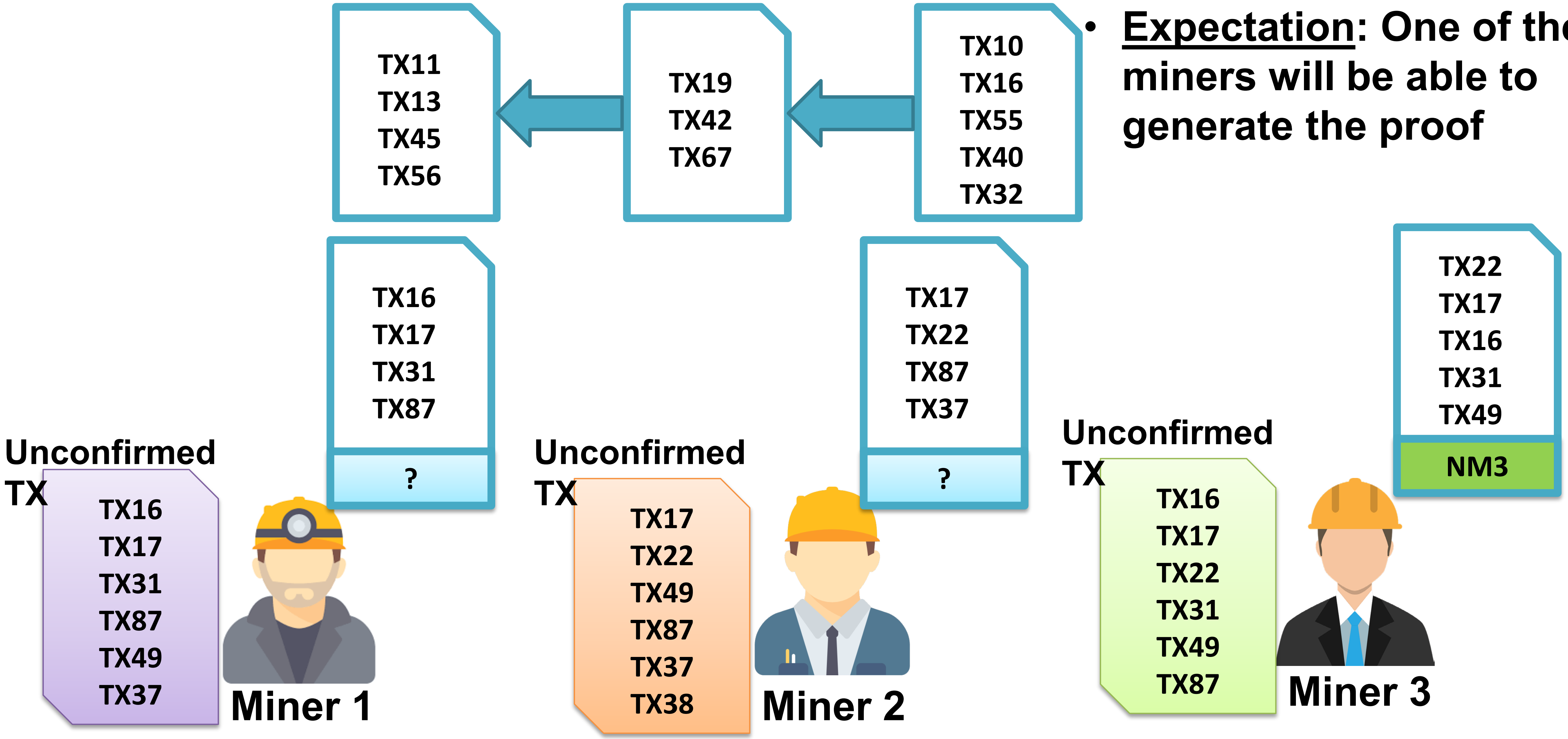
- **Generate the proof (nonce)**
  - Generation: Complex
  - Verification: Easy

Nakamoto Consensus (Proof of Work)

- **Expectation**: One of the miners will be able to generate the proof



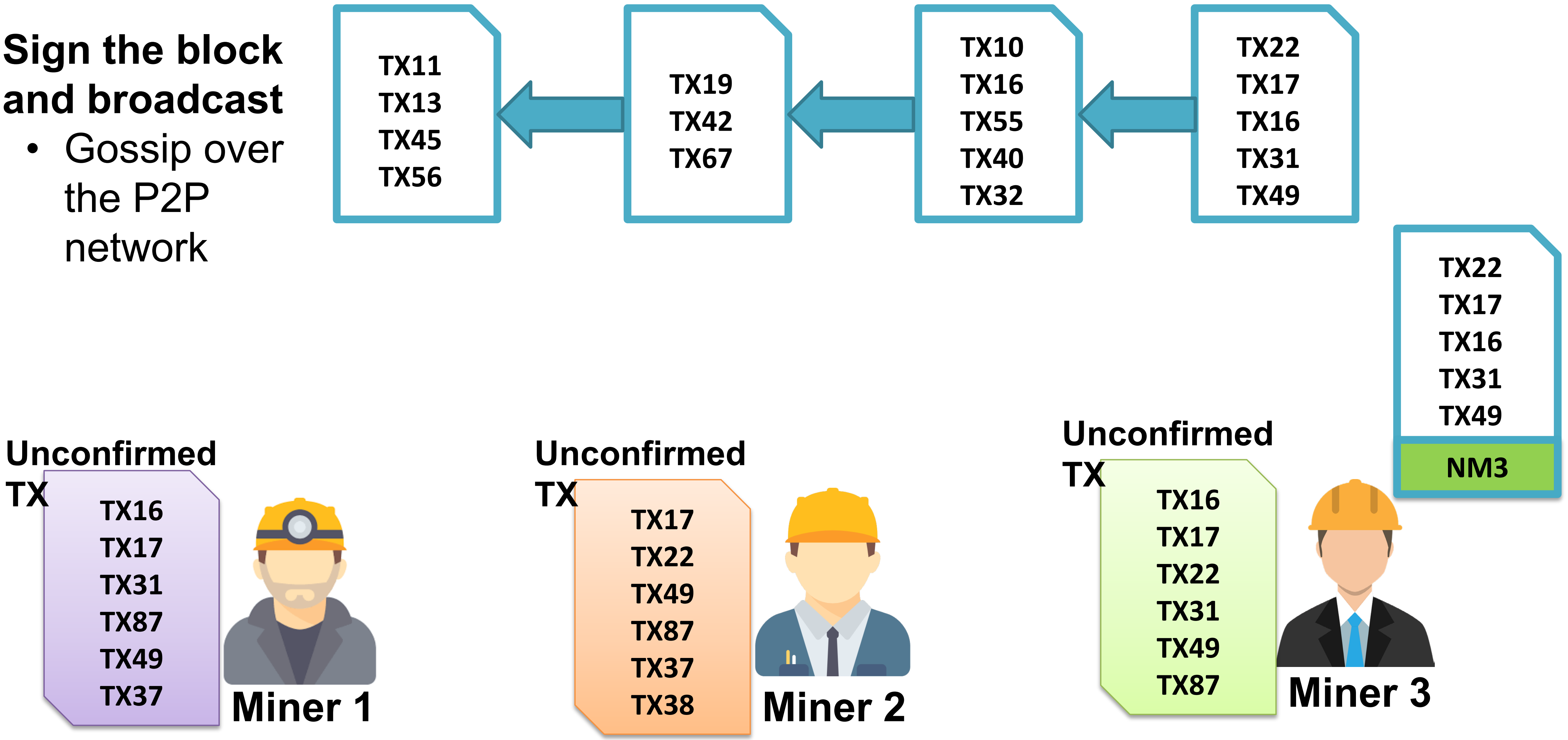
Nakamoto Consensus (Proof of Work)



- **Expectation**: One of the miners will be able to generate the proof

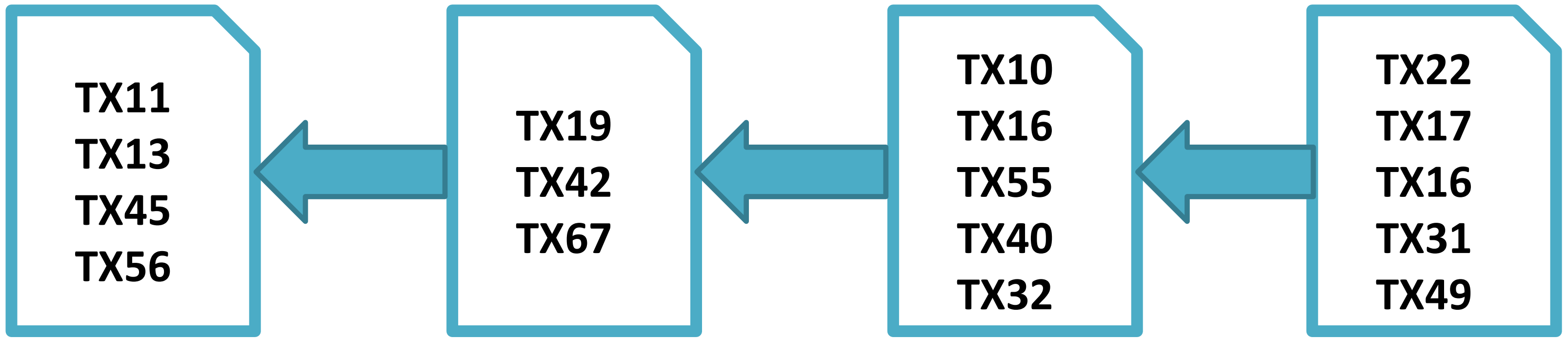
Nakamoto Consensus (Proof of Work)

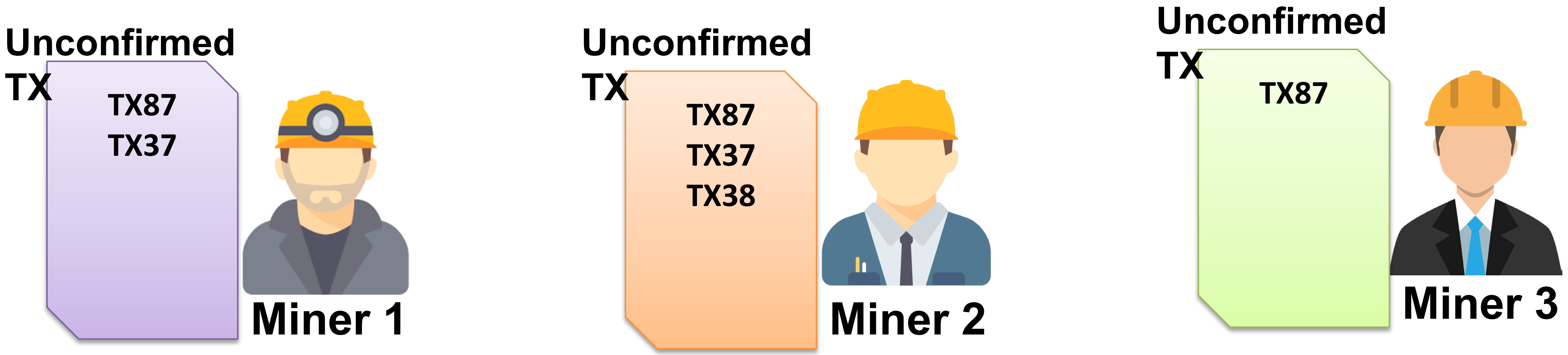
- **Sign the block and broadcast**
  - Gossip over the P2P network



Nakamoto Consensus (Proof of Work)

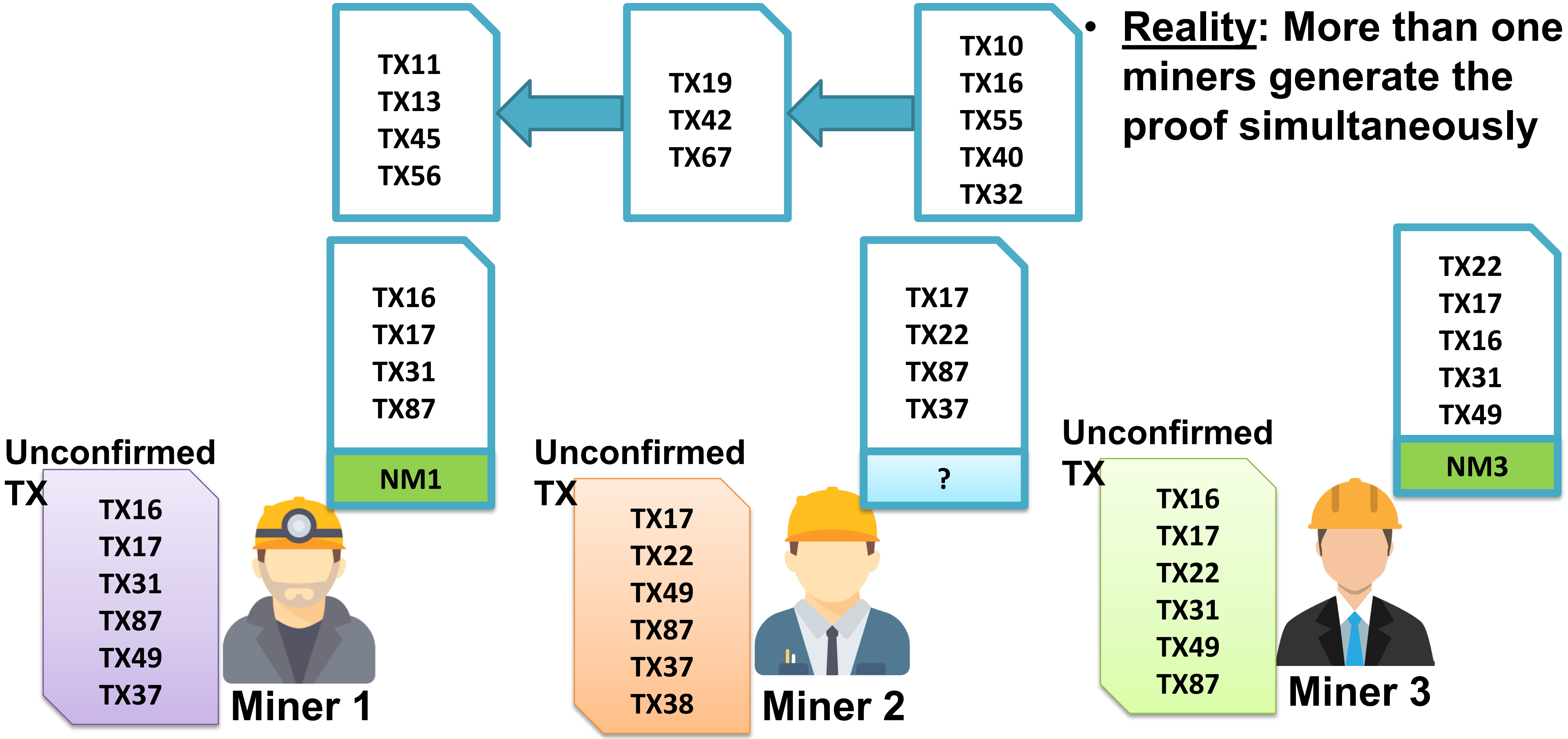
- Remove the transactions from unconfirmed TX list



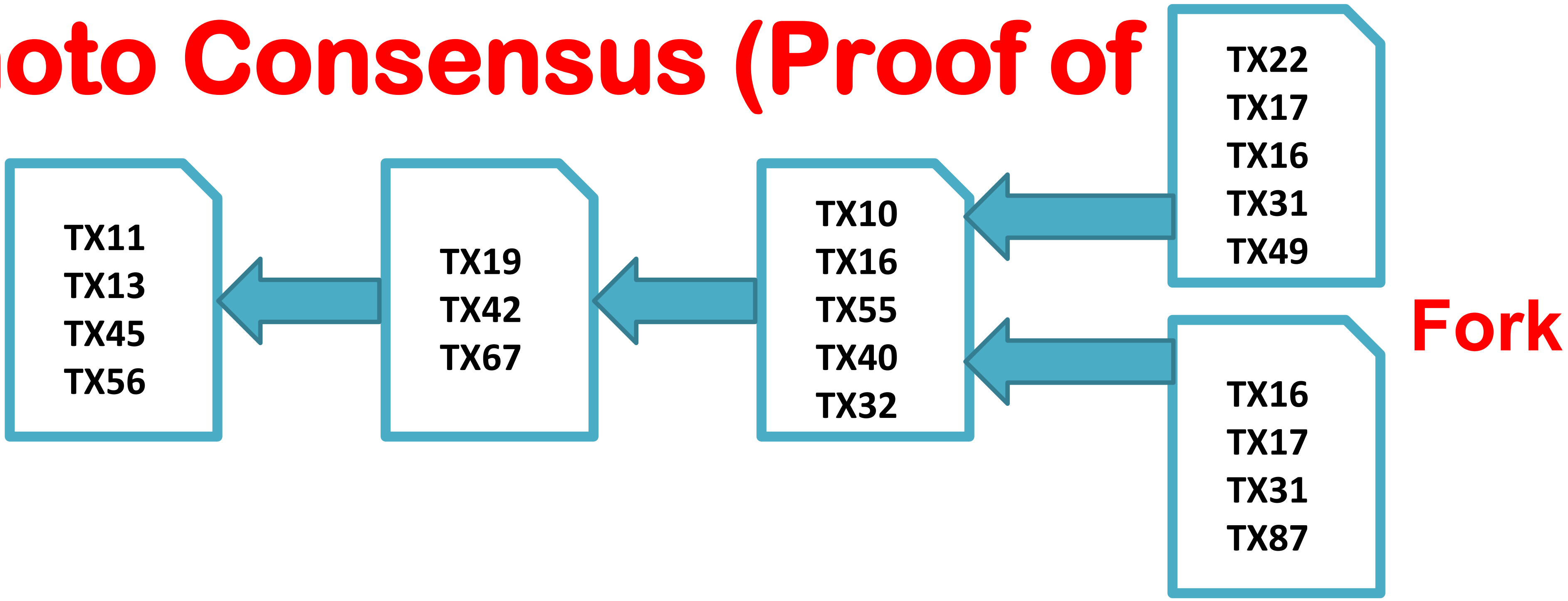


Nakamoto Consensus (Proof of Work)

- <u>Reality</u>: More than one miners generate the proof simultaneously



Nakamoto Consensus (Proof of Work)



Unconfirmed TX

TX16
TX17
TX31
TX87
TX49
TX37

Miner 1

Unconfirmed TX

TX17
TX22
TX49
TX87
TX37
TX38

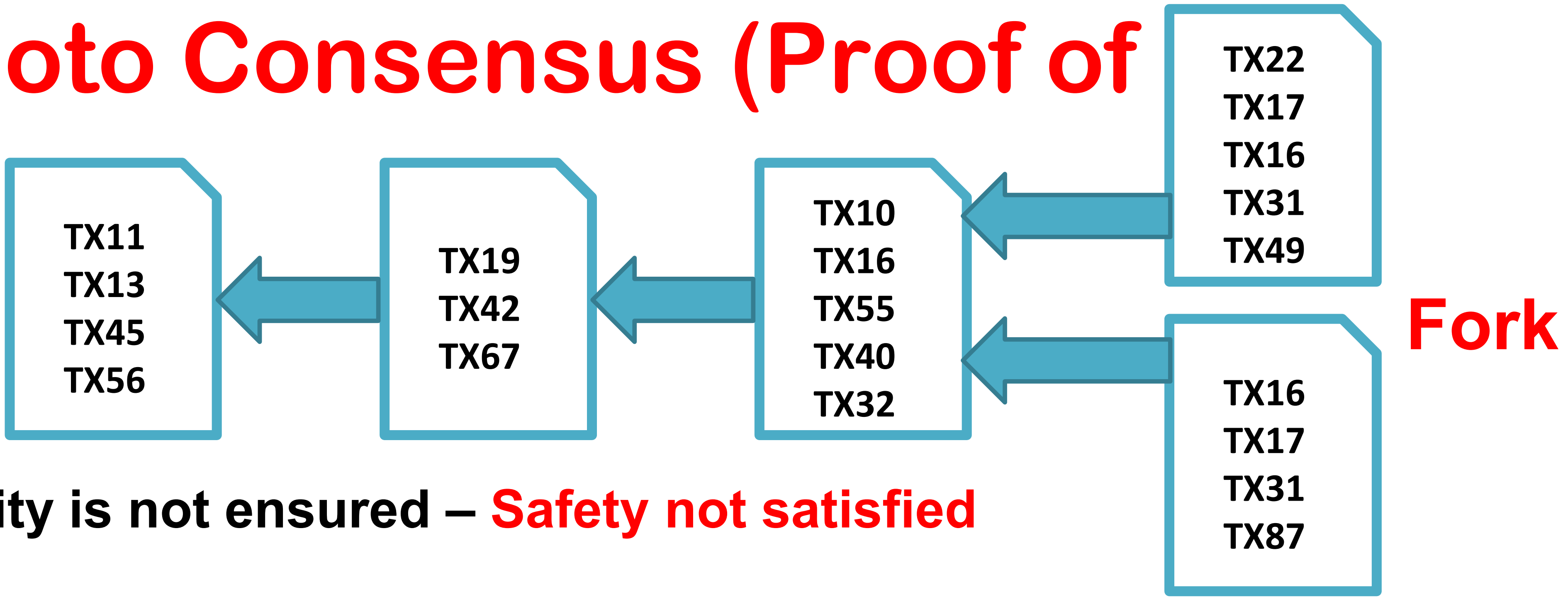
Miner 2

Unconfirmed TX

TX16
TX17
TX22
TX31
TX49
TX87

Miner 3

Nakamoto Consensus (Proof of Work)



Consensus finality is not ensured – Safety not satisfied immediately

- **The network remains partitioned for some amount of time**

Unconfirmed TX

TX16
TX17
TX31
TX87
TX49
TX37

Miner 1

Unconfirmed TX

TX17
TX22
TX49
TX87
TX37
TX38

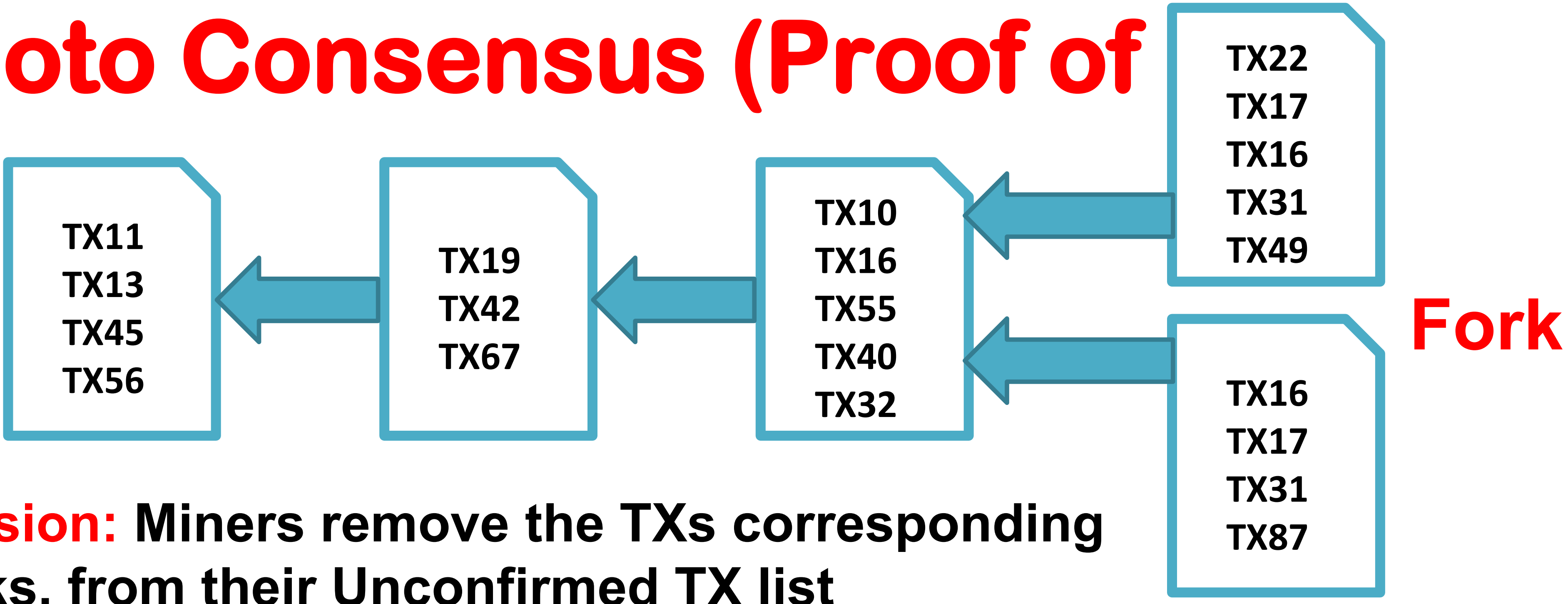
Miner 2

Unconfirmed TX

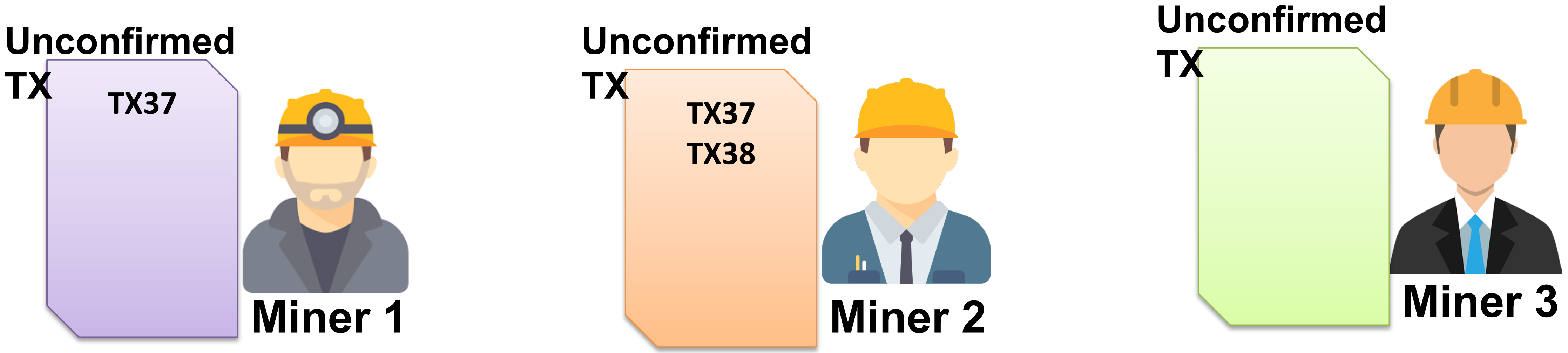
TX16
TX17
TX22
TX31
TX49
TX87

Miner 3

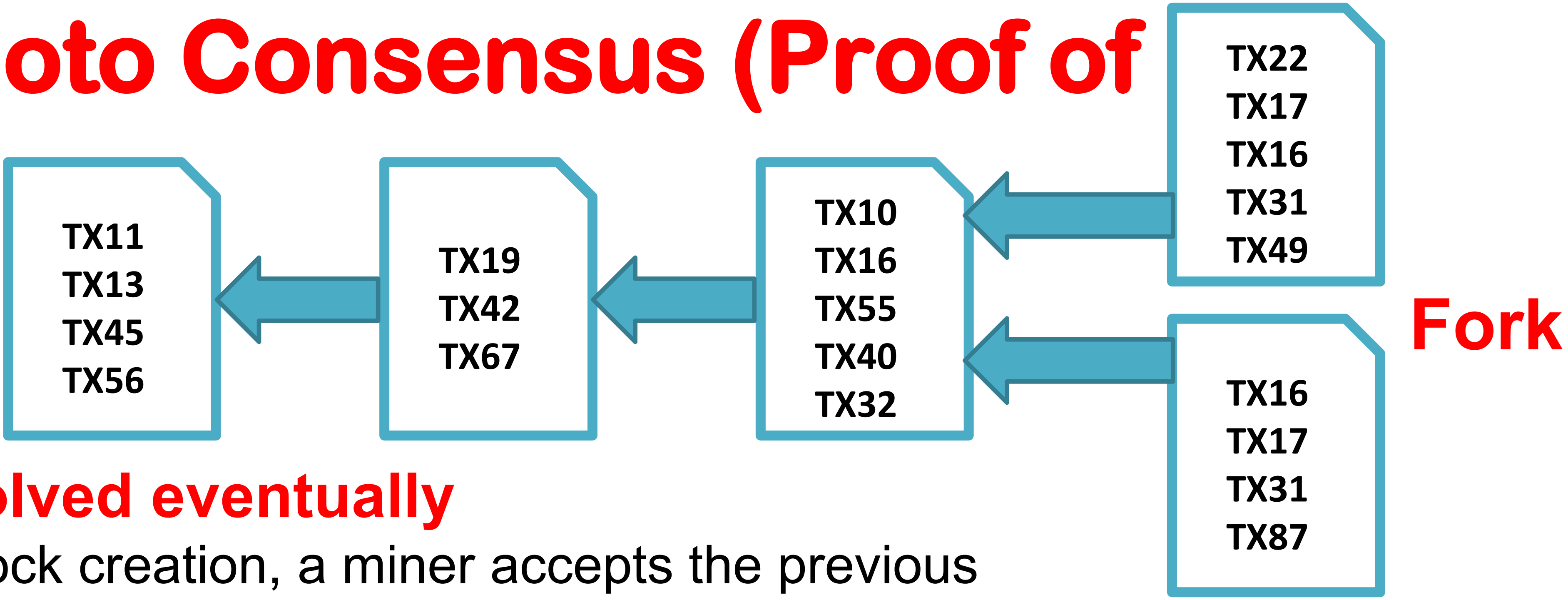
Nakamoto Consensus (Proof of Work)



Momentary Decision: Miners remove the TXs corresponding to both the blocks, from their Unconfirmed TX list

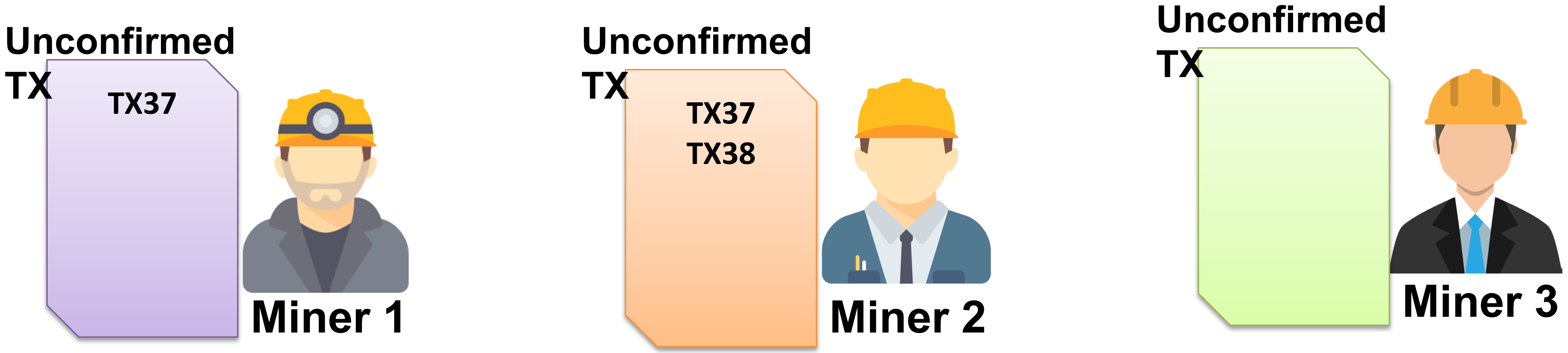


Nakamoto Consensus (Proof of Work)

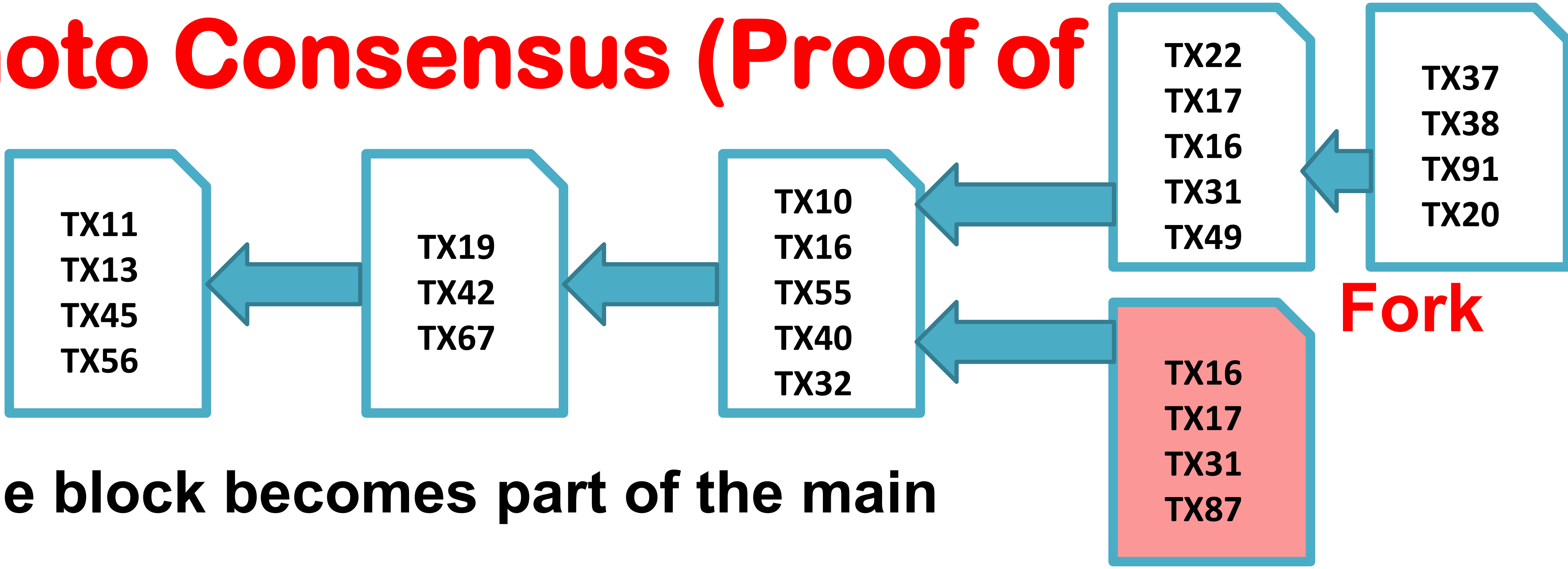


Forks are resolved eventually

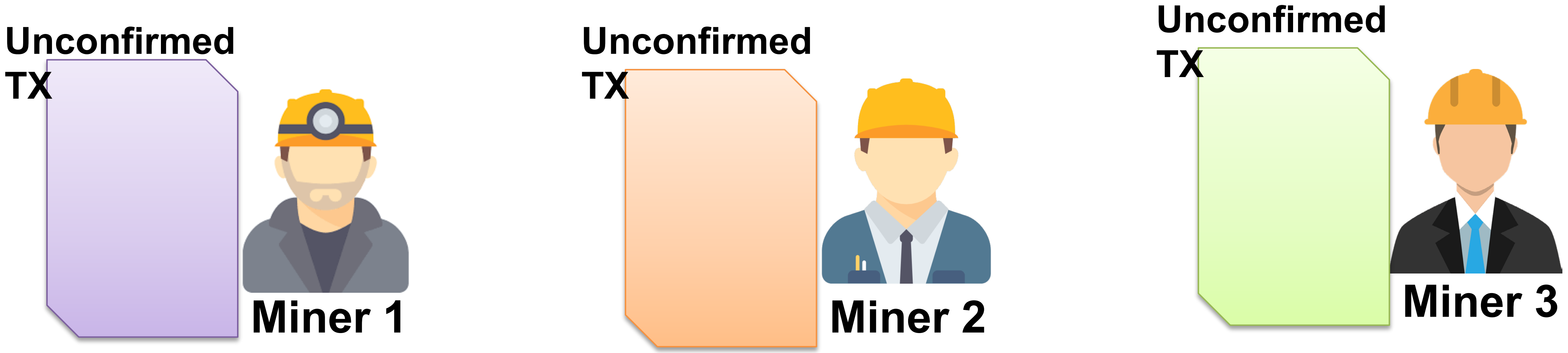
- For the next block creation, a miner accepts the previous block that it hears from the majority of the neighbor



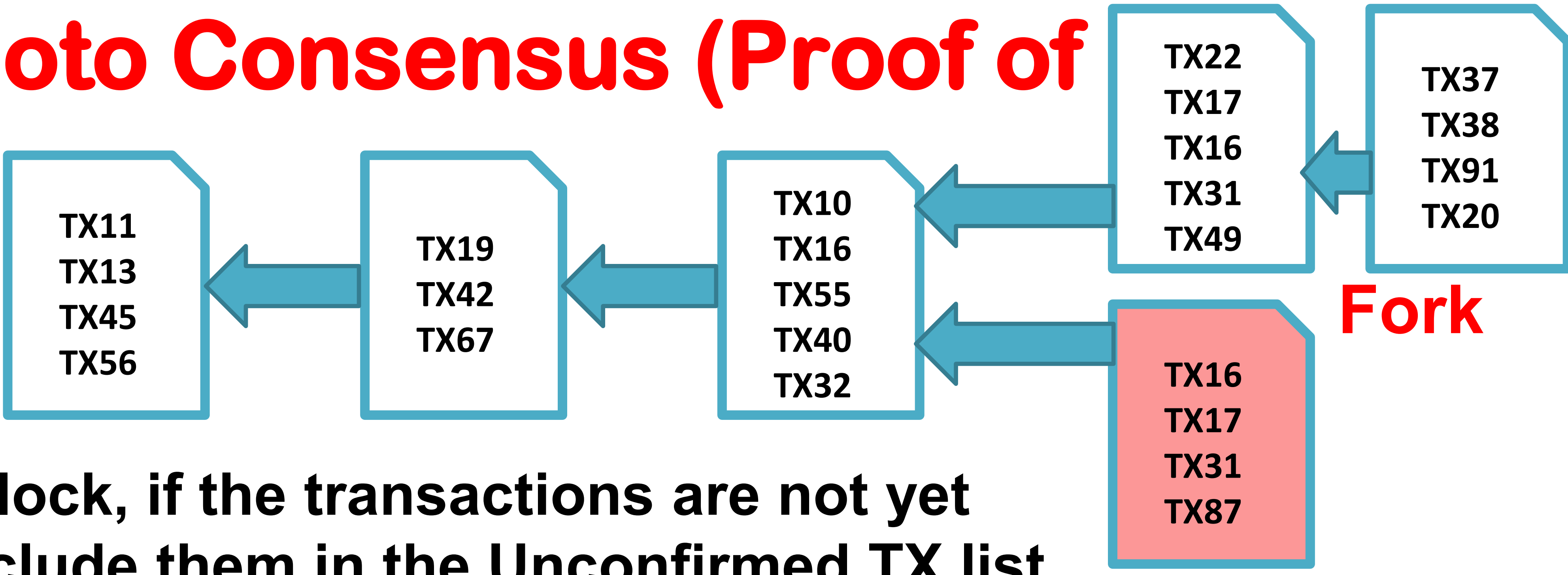
Nakamoto Consensus (Proof of Work)



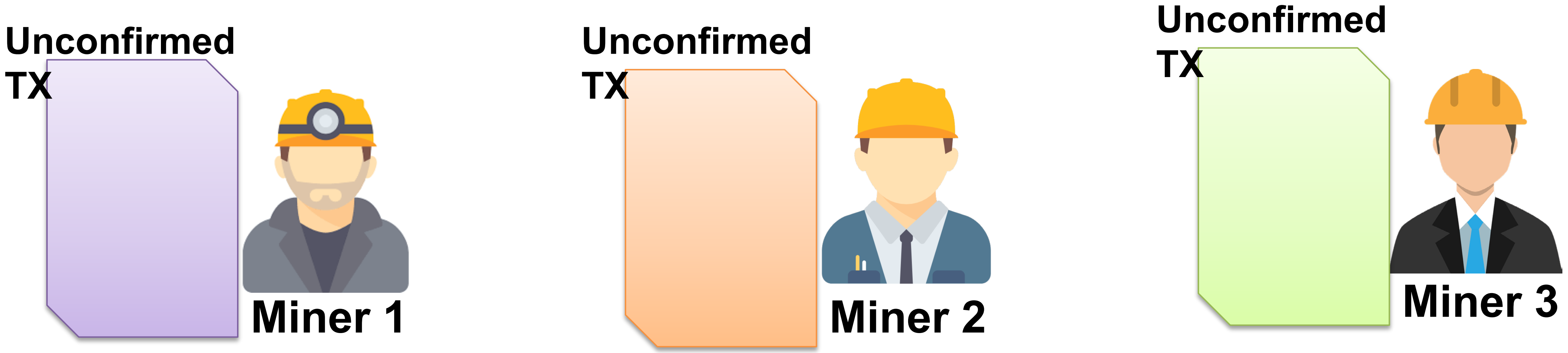
Eventually, one block becomes part of the main chain



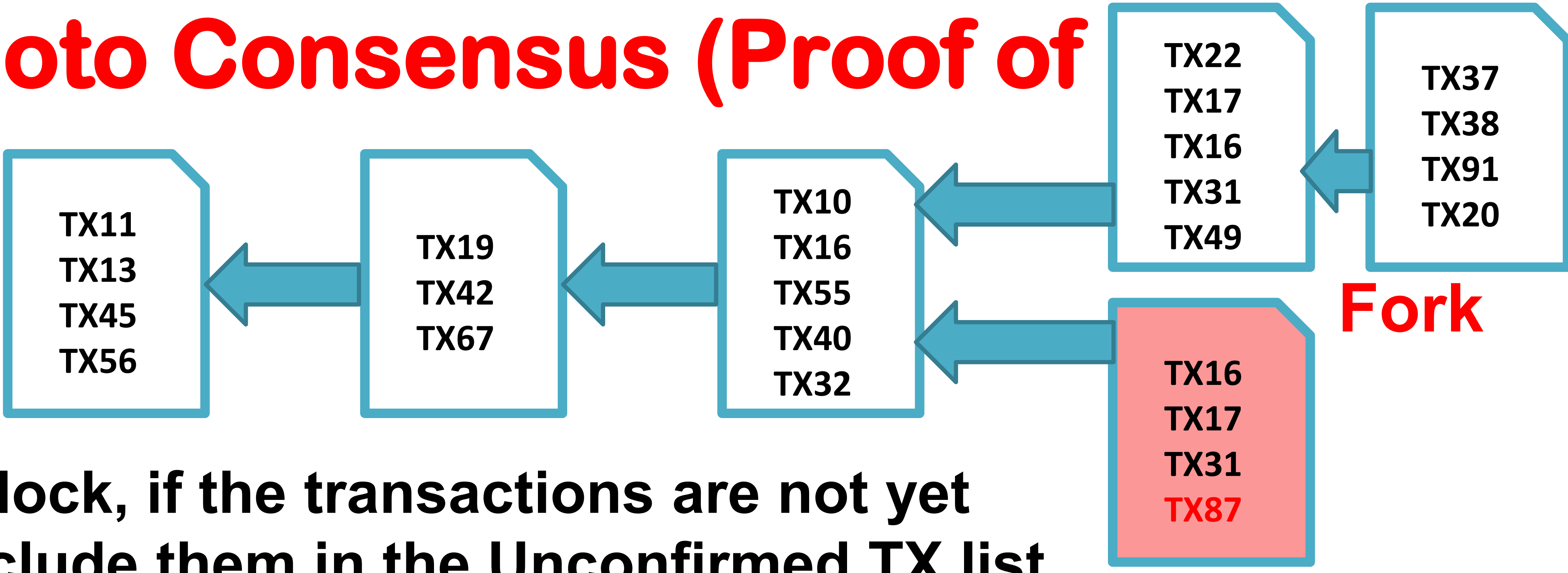
Nakamoto Consensus (Proof of Work)



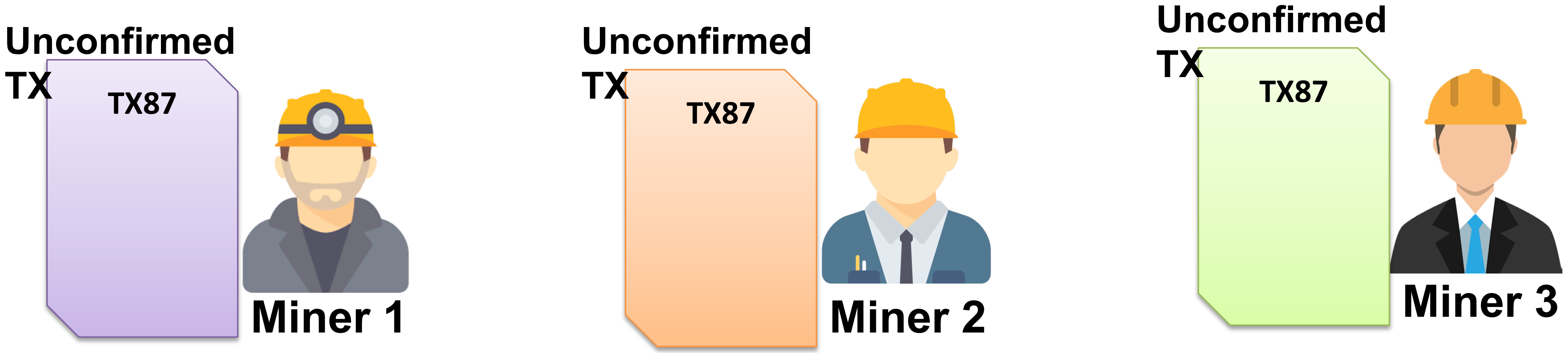
For a forked block, if the transactions are not yet committed, include them in the Unconfirmed TX list



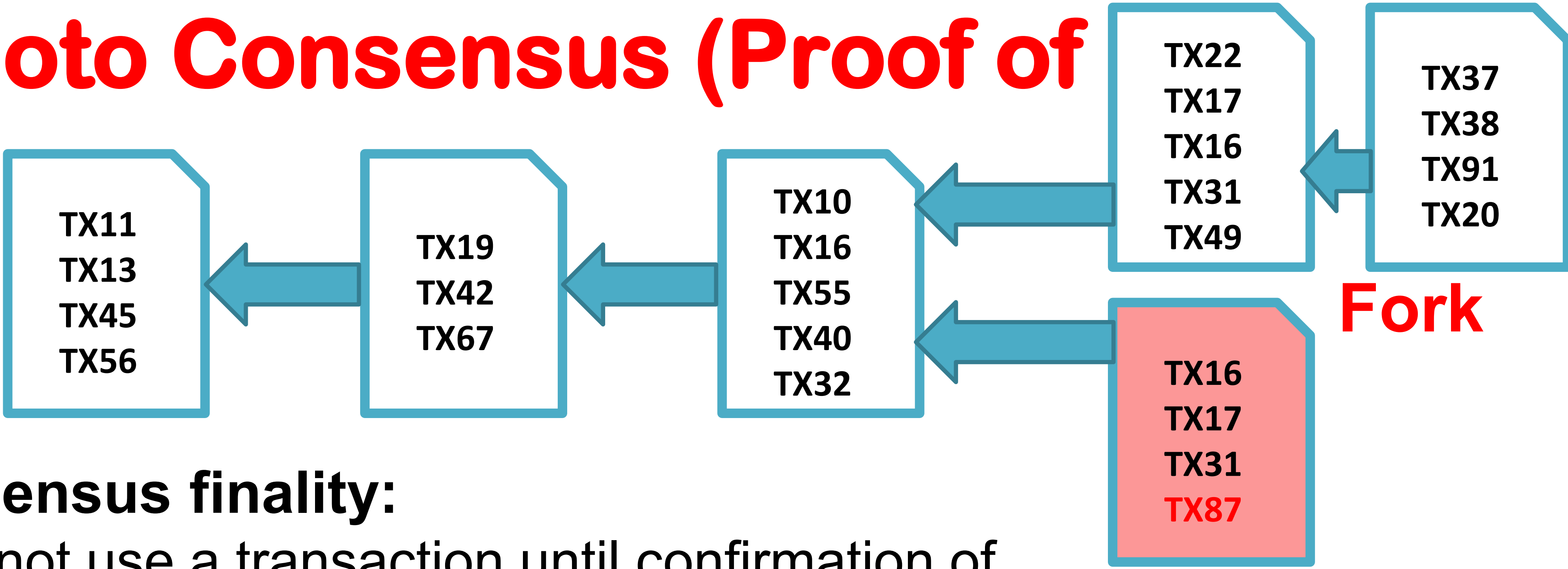
Nakamoto Consensus (Proof of Work)



For a forked block, if the transactions are not yet committed, include them in the Unconfirmed TX list

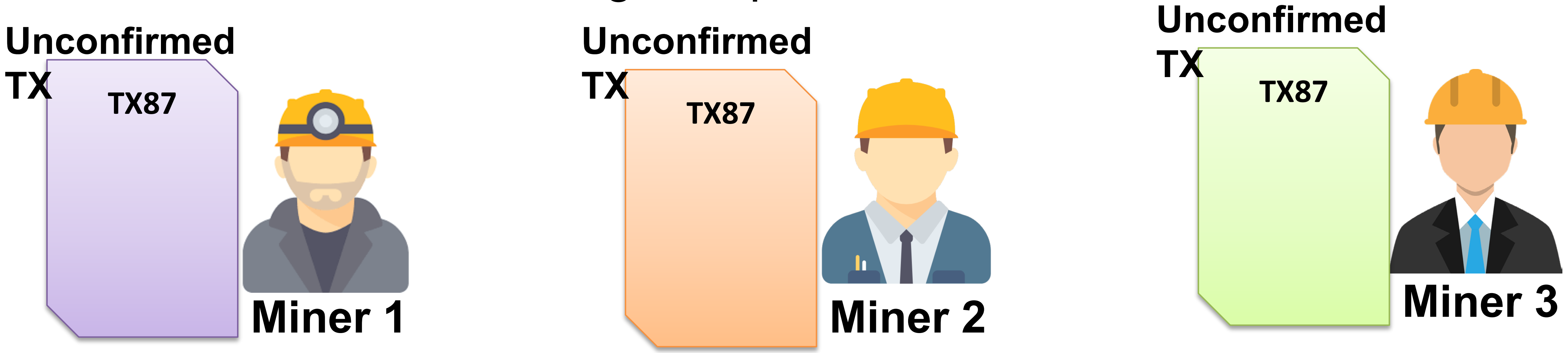


Nakamoto Consensus (Proof of Work)



Eventual consensus finality:

- (Bitcoin) Cannot use a transaction until confirmation of 6 blocks – ensured through scripts



Security Measures for PoW

- **Sybil Attacks**
  - Attacker attempts to fill the network with the clients under its control
  - Create multiple identities (multiple public key addresses) to control the network – refuse to relay valid blocks or relay attacked blocks
  - **Solution:** Diversify the connections – Bitcoin allows one outbound connection to per /16 block of IP addresses – cannot make both 202.141.81.2/16 and 202.141.80.18/16 as the peers

Security Measures for PoW

- **Sybil Attacks**
  - Attacker attempts to fill the network with the clients under its control
  - Create multiple identities (multiple public key addresses) to control the network – refuse to relay valid blocks or relay attacked blocks
  - **Solution:** Diversify the connections – Bitcoin allows one outbound connection to per /16 block of IP addresses – cannot make both 202.141.81.2/16 and 202.141.80.18/16 as the peers

- **Denial of Service (DoS)**
  - Send a lot of data to a node – block the processing power
  - **Solution:** Limit forwarding of blocks, disconnect a peer that sends too many transactions

Breaking PoW

- Bitcoin PoW is **computationally difficult** to break, but not **impossible**
- Attackers can deploy high power servers to do more work than the total work of the blockchain
- A known case of successful double-spending
  - (November 2013) "it was discovered that the GHash.io mining pool appeared to be engaging in repeated payment fraud against *BetCoin Dice*, a gambling site" [Source: https://en.bitcoin.it/]

The Monopoly Problem

- PoW depends on the computing resources available to a miner
  - Miners having more resources have more probability to complete the work

- Monopoly can increase over time (*Tragedy of the Commons*)
  - Miners will get less reward over time
  - Users will get discouraged to join as the miner
  - Few miners with large computing resources may get control over the network

- **51% Attack:** A group of miners control more than 50% of the hash rate of the network
  - Hypothetical as of now for Bitcoin (as the network is large), but not impossible (happened for Kryptom – Ethereum based blockchain, in August, 2016)

The Limit of PoW

- **The Good:** A fully decentralized consensus for permissionless models
  - works good for cryptocurrencies – serves its purposes

The Limit of PoW

- **The Good:** A fully decentralized consensus for permissionless models
  - works good for cryptocurrencies – serves its purposes

- **The Bad:** Do not trust the individuals, but trust the society as a whole
  - You need a real large network to prevent the 51% attack – **not at all suitable for enterprise applications**

The Limit of PoW

- **The Good:** A fully decentralized consensus for permissionless models
  - works good for cryptocurrencies – serves its purposes
- **The Bad:** Do not trust the individuals, but trust the society as a whole
  - You need a real large network to prevent the 51% attack – **not at all suitable for enterprise applications**
- **The Ugly**: Low transaction throughput, Overuse of computing power !!
  - (Bitcoin) 3.3 to 7 transactions per second, (Ethereum) ~15 transactions per second
  - Millions of miners – thousands tries, but only one gets the success

Bitcoin Energy Consumption

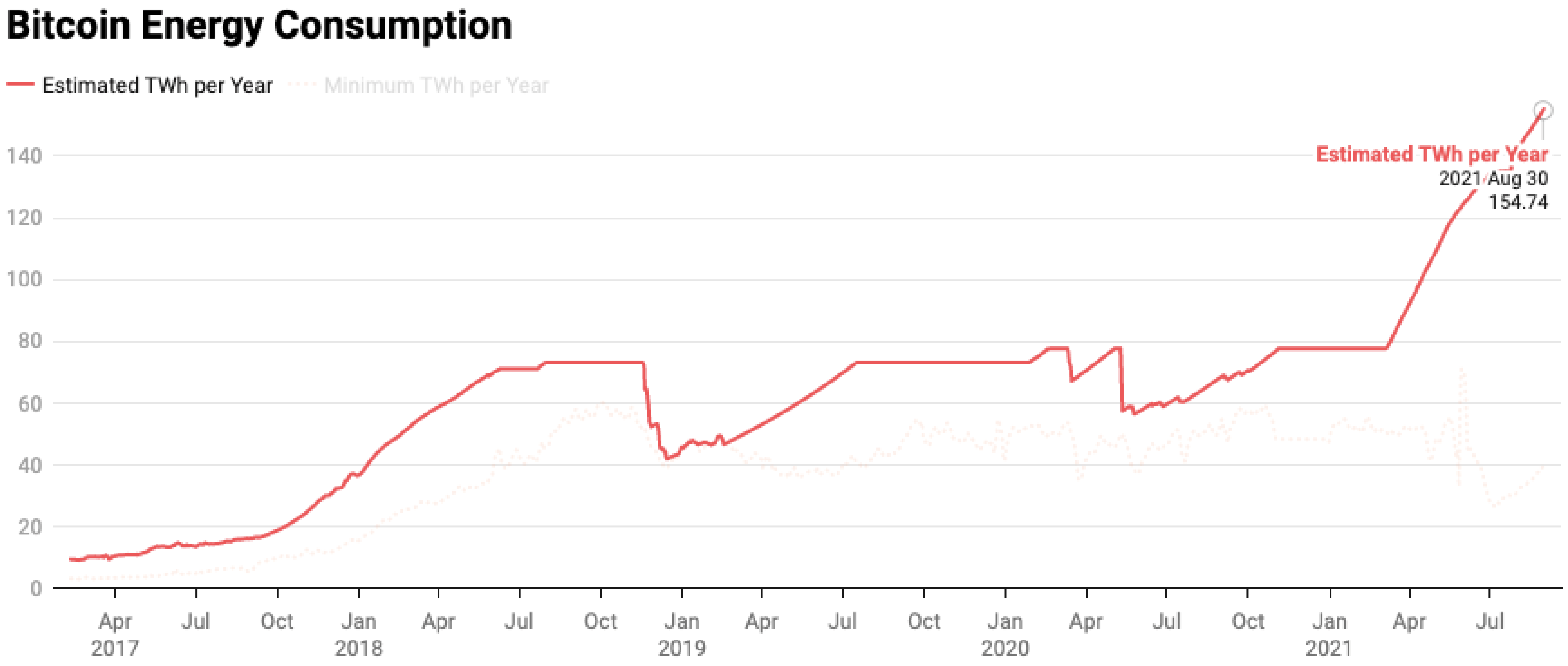
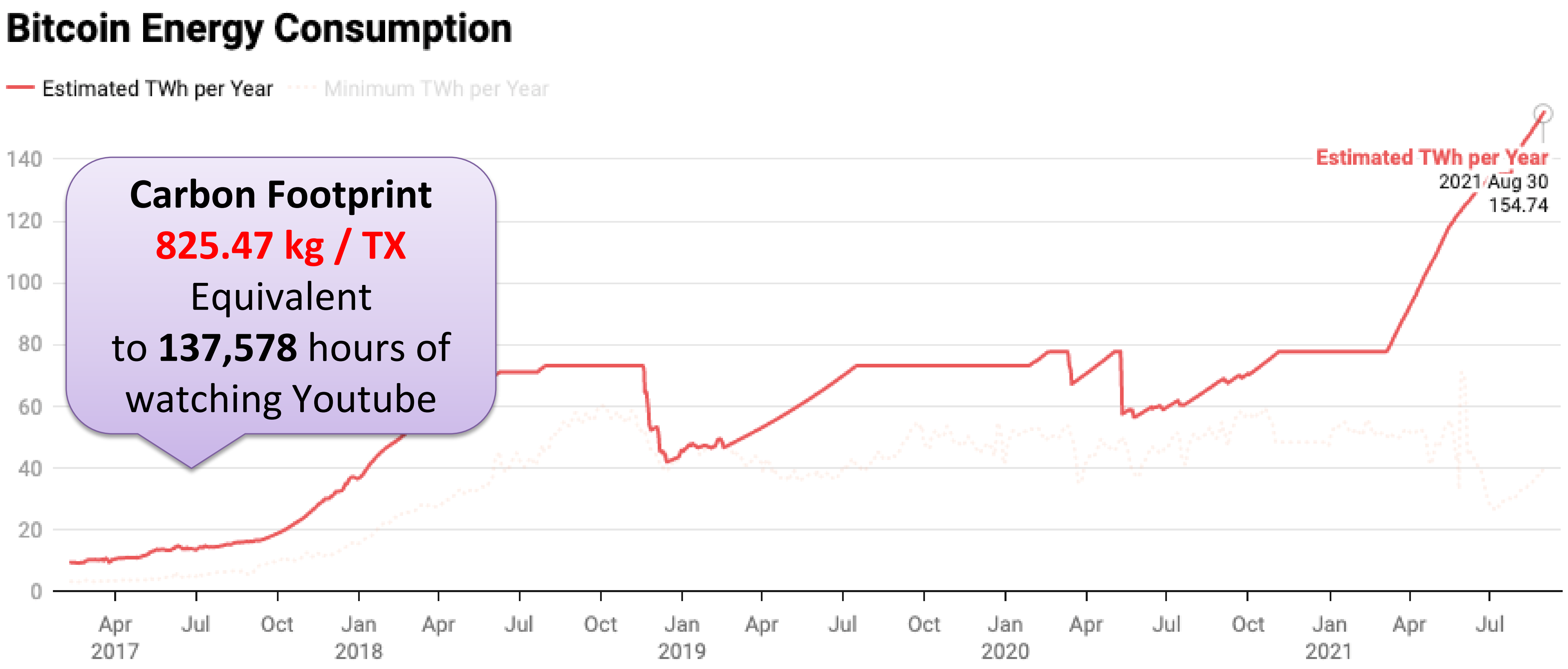


Image Source: Digiconomist Bitcoin Energy Consumption Index

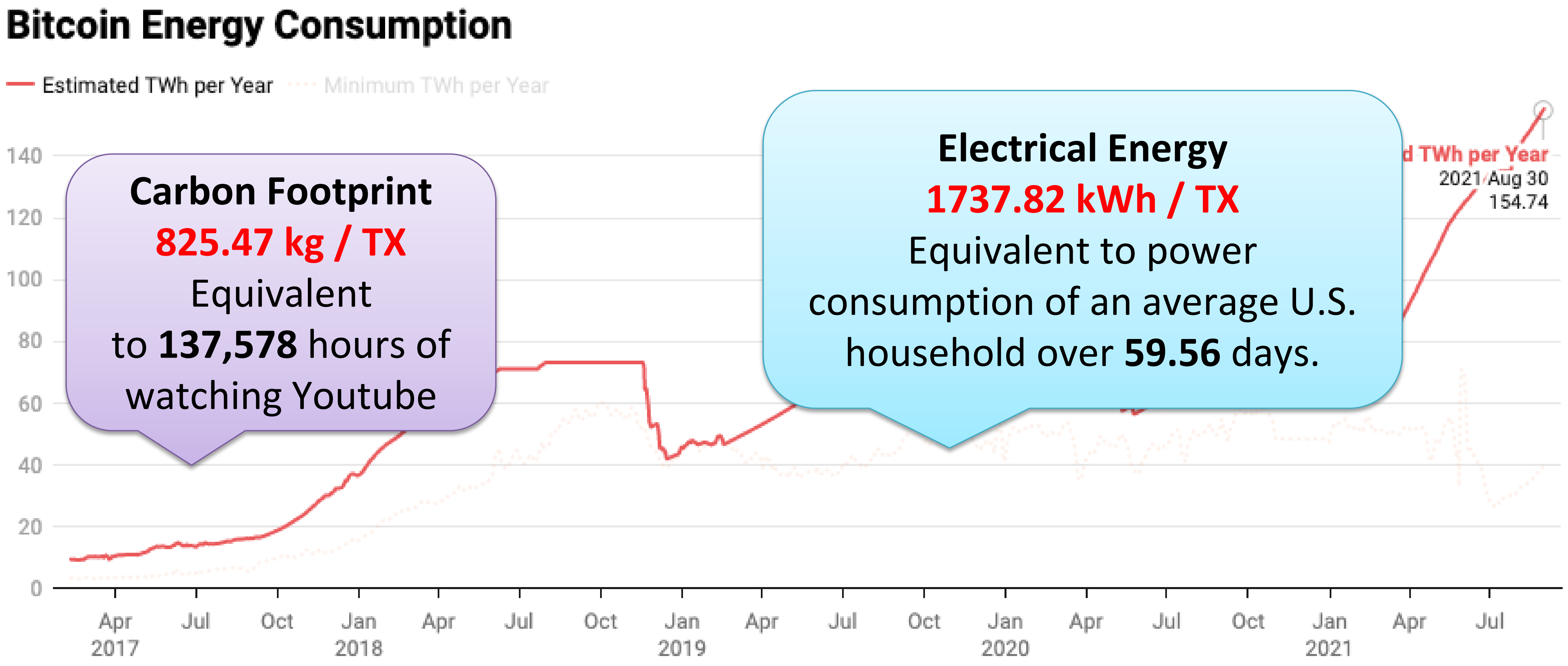
Bitcoin Energy Consumption



Carbon Footprint
825.47 kg / TX
Equivalent
to **137,578** hours of
watching Youtube

Image Source: Digiconomist Bitcoin Energy Consumption Index

Bitcoin Energy Consumption



Carbon Footprint
825.47 kg / TX
Equivalent
to **137,578** hours of
watching Youtube

Electrical Energy
1737.82 kWh / TX
Equivalent to power
consumption of an average U.S.
household over **59.56** days.

Image Source: Digiconomist Bitcoin Energy Consumption Index

Proof of Stake (PoS)

- Possibly proposed in 2011 by a Member in Bitcoin Forum - https://bitcointalk.org/index.php?topic=27787.0
  - Make a transition from PoW to PoS when bitcoins are widely distributed
  -

- PoW vs PoS
  - **PoW**: Probability of mining a block depends on the work done by the miner
  - **PoS**: Amount of bitcoin that the miner holds – Miner holding 1% of the Bitcoin can mine 1% of the PoS blocks.

Proof of Stake (PoS)

- Provides increased protection
  - Executing an attack is expensive, you need more Bitcoins
  - **Reduced incentive for attack** – the attacker needs to own a majority of bitcoins – an attack will have more affect on the attacker

- Variants of “stake”
  - Randomization in combination of the stake (*used in Nxt and BlackCoin*)
  - **Coin-age**: Number of coins multiplied by the number of days the coins have been held (*used in Peercoin*)

Ethereum PoS

- The default consensus mechanism in Ethereum is PoS
  - Switched to PoS from PoW in 2022

- **Validators:** The users who responsible for checking that new blocks propagated over the network are valid
  - Occasionally creates and propagates new blocks themselves.

- **Ethereum PoS: P**rove that validators have put something of value into the network that can be destroyed if they act dishonestly.
  - Validators explicitly stake capital in the form of ETH into a smart contract on Ethereum
  - If they try to defraud the network (*for example by proposing multiple blocks when they ought to send one or sending conflicting attestations*), some or all of their staked ETH can be destroyed.

Ethereum PoS: Validators

- To participate as a validator, a user must
  - Deposit 32 ETH into the deposit contract
  - Run three separate pieces of software: an execution client, a consensus client, and a validator.
- On depositing their ETH, the user joins an activation queue that limits the rate of new validators joining the network.
  - Once activated, validators receive new blocks from peers on the Ethereum network.
- The transactions delivered in the block are re-executed to check that the proposed changes to Ethereum's state are valid, and the block signature is checked.
  - The validator then sends a vote (called an attestation) in favor of that block across the network.

Ethereum PoS: Validators

- Time is divided into slots (12 seconds) and epochs (32 slots).
  - One validator is randomly selected (selected pseudo-randomly using RANDAO -- https://www.randao.org/) to be a block proposer in every slot.
  - This validator is responsible for creating a new block and sending it out to other nodes on the network.

- In every slot, a committee of validators is randomly chosen, whose votes are used to determine the validity of the block being proposed.
  - Committees divide up the validator set so that every active validator attests in every epoch, but not in every slot.

The Need for Randomness

- Two fundamental questions --
  - How do you decide who is going to be the proposer for a new block?
  - How do you decide the committee members?

- The selection process must be **fair** to all validators

The Need for Randomness

- Two fundamental questions --
  - How do you decide who is going to be the proposer for a new block?
  - How do you decide the committee members?

- The selection process must be fair to all validators

- **Naïve solution:** Make a sorted list of validators (may be, based on the stake), and then allow each validator to produce a block based on "round-robin" on the list
  - **What can be a possible issue with this solution?**

The Need for Randomness

- Two fundamental questions --
  - How do you decide who is going to be the proposer for a new block?
  - How do you decide the committee members?

- The selection process must be fair to all validators

- **Naïve solution:** Make a sorted list of validators (may be, based on the stake), and then allow each validator to produce a block based on "round-robin" on the list
  - **What can be a possible issue with this solution?** Attackers know whose turn is the next – launch a DoS on that validator, or validators can conspire among their Neighbours

The Need for Randomness

- **Solution:** Use random numbers to produce block producers in each epoch
  - **But, how do you get on-chain randomness?**

The Need for Randomness

- **Solution:** Use random numbers to produce block producers in each epoch
  - **But, how do you get on-chain randomness?**

- One of the validators propose a random number – **Does this solution work?**

The Need for Randomness

- **Solution:** Use random numbers to produce block producers in each epoch
  - **But, how do you get on-chain randomness?**

- One of the validators propose a random number – **Does this solution work?**
  - **What if the proposed value is not a random value generated by the computer, but something that the proposer decide? How do you validate this?**

The Need for Randomness

- **Solution:** Use random numbers to produce block producers in each epoch
  - **But, how do you get on-chain randomness?**

- One of the validators propose a random number – **Does this solution work?**
  - **What if the proposed value is not a random value generated by the computer, but something that the proposer decide? How do you validate this?**

- Use a **group of validators** to propose their own random number – use a mixing function to generate the final output number
  - `Output → mix (input-1, input-2, input-3, …, input-n)`
  - **Condition:** Use a mixing function that gives equal weight to all the inputs

The Need for Randomness

- Example of a mixing function: The XOR function
  - $(1111 \oplus 1001) \oplus 0101 \rightarrow 0011$
  - **Do you foresee any issue with this mixing function?**

The Need for Randomness

- Example of a mixing function: The XOR function
  - $(1111 \oplus 1001) \oplus 0101 \rightarrow 0011$
  - **Do you foresee any issue with this mixing function?** The last validator has an advantage to propose a value that makes the output of their choice
  - The attack is possible with other mixing functions as well, although might be a little hard; for example, using cryptographic hash as the mixing function

RANDAO

- Example of a mixing function: The XOR function
  - $(1111 \oplus 1001) \oplus 0101 \rightarrow 0011$
  - **Do you foresee any issue with this mixing function?** The last validator has an advantage to propose a value that makes the output of their choice
  - The attack is possible with other mixing functions as well, although might be a little hard; for example, using cryptographic hash as the mixing function

- **Solution:** Let other participants do not see the inputs until all inputs are revealed → Hide the proposed input through cryptographic hash until all the inputs are revealed **(Cryptographic commitments)**
  - Propose the cryptographic hash of the secret (commit phase)
  - Once everyone makes the hash values public, then reveal the original secret value (Reveal phase)

RANDAO

- RANDAO uses cryptographic commitments along with a mixing function to generate the random number
  - Ethereum uses BLS signatures (earlier used hash onions)

RANDAO

- RANDAO uses cryptographic commitments along with a mixing function to generate the random number
  - Ethereum uses BLS signatures (earlier used hash onions)
- **One possible issue with RANDAO** -- while they prevent a participant from changing their number, they don't actually force the participant to reveal their number.
  - Solution: Verifiable Delay Function (VDF) -- Feed the RANDAO in a function that delay the output – and use the delayed output (the output is guaranteed to be produced after a delay – you cannot arbitrarily delay to reveal the output)
  - Ethereum does not incorporate such solutions yet, is an interesting research topic

RANDAO in Ethereum

- Every block contains a field `randao_reveal` which is the RANDAO contribution for the block proposer

- The RANDAO contributions for each block is used to compute the chain's RANDAO till the last block
  - The chain's RANDAO (along with the stake contributions from the validators) is used to select the next block proposer and the committee members to validate that block

- Check this for a detailed description of Ethereum PoS: https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/

Proof of Burn (PoB)

- Miners should show proof that they have *burned* some coins
  - Sent them to a verifiably un-spendable address
  - Expensive just like PoW, but no external resources are used other than the burned coins
- PoW vs PoB – Real resource vs virtual/digital resource
- PoB works by burning PoW mined cryptocurrencies

Proof of Burn (PoB)

- Miners should show proof that they have *burned* some coins
  - Sent them to a verifiably un-spendable address
  - Expensive just like PoW, but no external resources are used other than the burned coins
- PoW vs PoB – Real resource vs virtual/digital resou
- PoB works by burning PoW mined cryptocurrenci

PoS and PoB

Ultimately depends on PoW mined cryptocurrencies

You cannot use them to bootstrap a new blockchain

Proof of Elapsed Time (PoET)

- Proposed by Intel, as a part of Hyperledger Sawtooth – a blockchain platform for building distributed ledger applications

- **Basic idea:**
  - Each participant in the blockchain network waits a random amount of time
  - The first participant to finish becomes the leader for the new block

- How will one verify that the proposer has **really waited** ?
  - Utilize special CPU instruction set – *Intel Software Guard Extension* (SGX) – a trusted execution platform
  - The trusted code is private to the rest of the application
  - The specialized hardware provides an attestation that the trusted code has been set up correctly

Proof of Elapsed Time (PoET)

- Proposed by Intel, as a part of Hyperledger Sawtooth – a blockchain platform for building distributed ledger applications

- **Basic idea:**
  - Each participant in the blockchain network waits a random amount of time
  - The first participant to finish becomes the leader for the new block

- How will one verify that the proposer has **really waited** ?
  - Utilize special CPU instruction set – *Intel Software Guard Extension* (SGX) – a trusted execution platform
  - The trusted code is private to the rest of the application
  - The specialized hardware provides an attestation that the trusted code has been set up correctly

What Next?

- Enterprise blockchains and Consensus thereafter
- Consensus scalability
- A decade of research on Blockchain (Distributed System?) consensus …