CS 5594: BLOCKCHAIN TECHNOLOGIES

Spring 2024

THANG HOANG, PhD

BITCOIN

Bitcoin Overview

Recap: Bitcoin is not equivalent to blockchain

Bitcoin is the foundation of all BC technologies today

Understanding Bitcoin will mostly understand how BC technologies are built

Bitcoin Components

P2P Network

Address and Wallet

Transactions

Blocks

Consensus

Mining

Bitcoin Components

Bitcoin Network

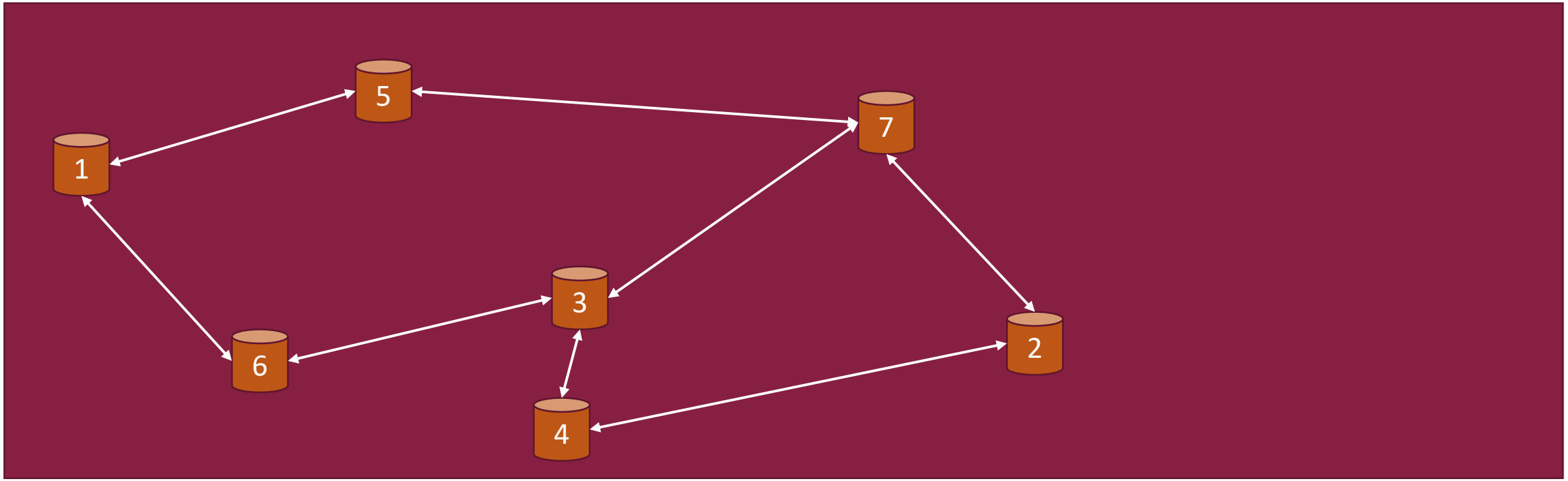
Bitcoin Network

Ad-hoc network (TCP port 8333) with random topology

All nodes are equal – no special/master nodes

Nodes can join and leave at any time (permissionless/public blockchain)

Forget non-responding nodes after 3 hours



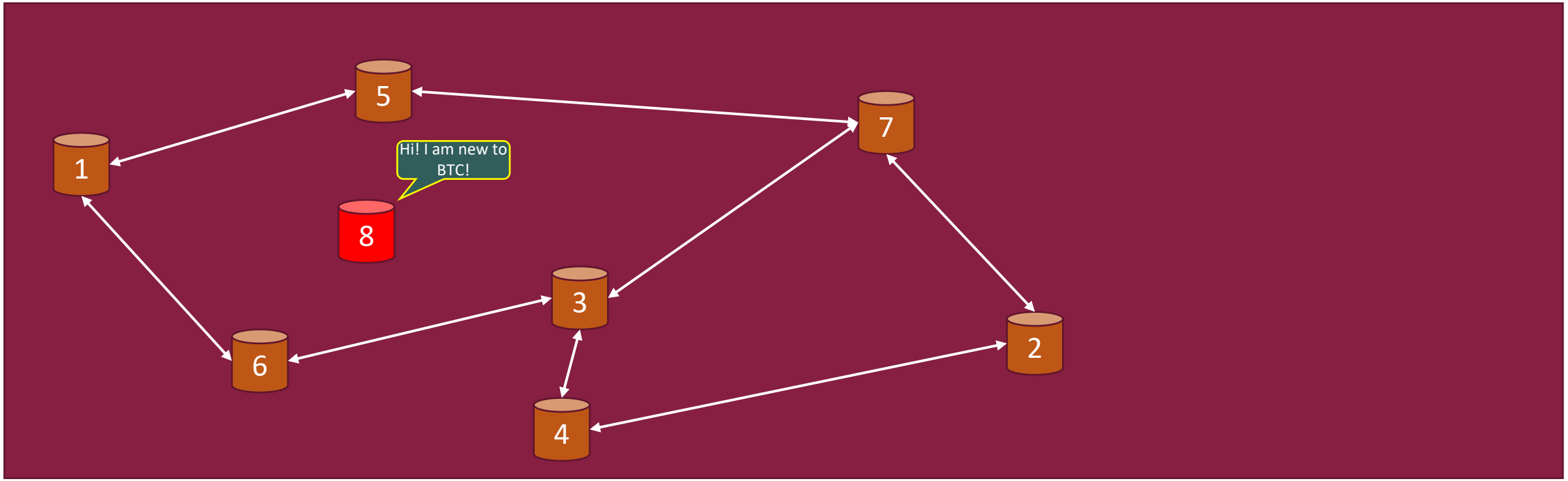
Bitcoin Network

Ad-hoc network (TCP port 8333) with random topology

All nodes are equal – no special/master nodes

Nodes can join and leave at any time (permissionless/public blockchain)

Forget non-responding nodes after 3 hours



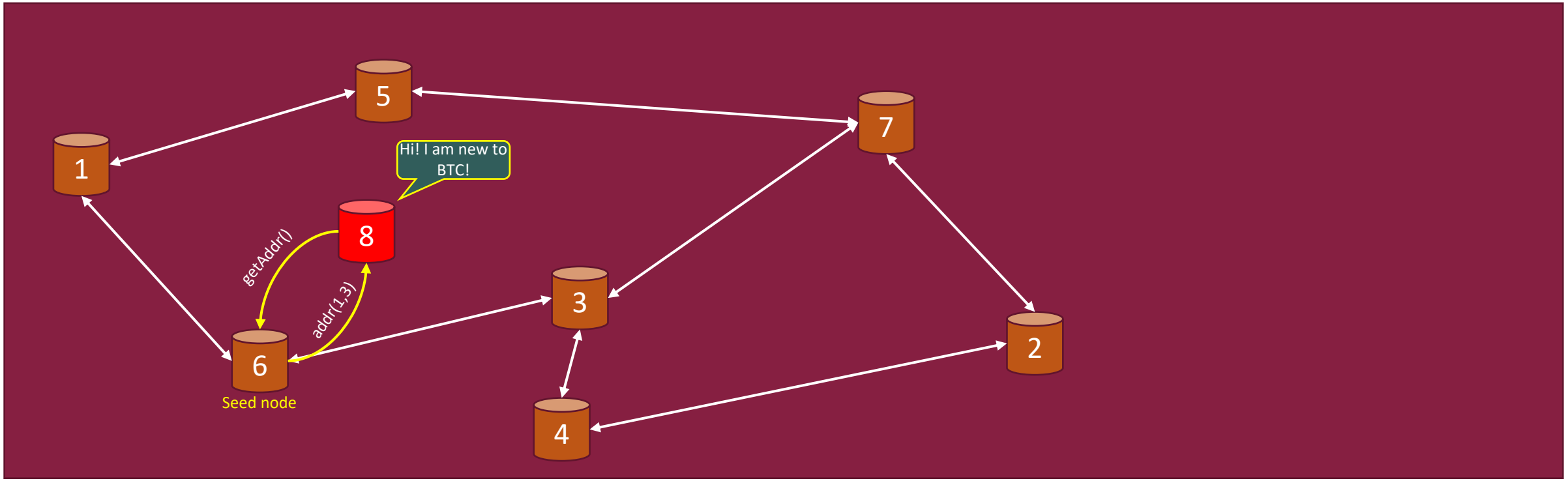
Bitcoin Network

Ad-hoc network (TCP port 8333) with random topology

All nodes are equal – no special/master nodes

Nodes can join and leave at any time (permissionless/public blockchain)

Forget non-responding nodes after 3 hours



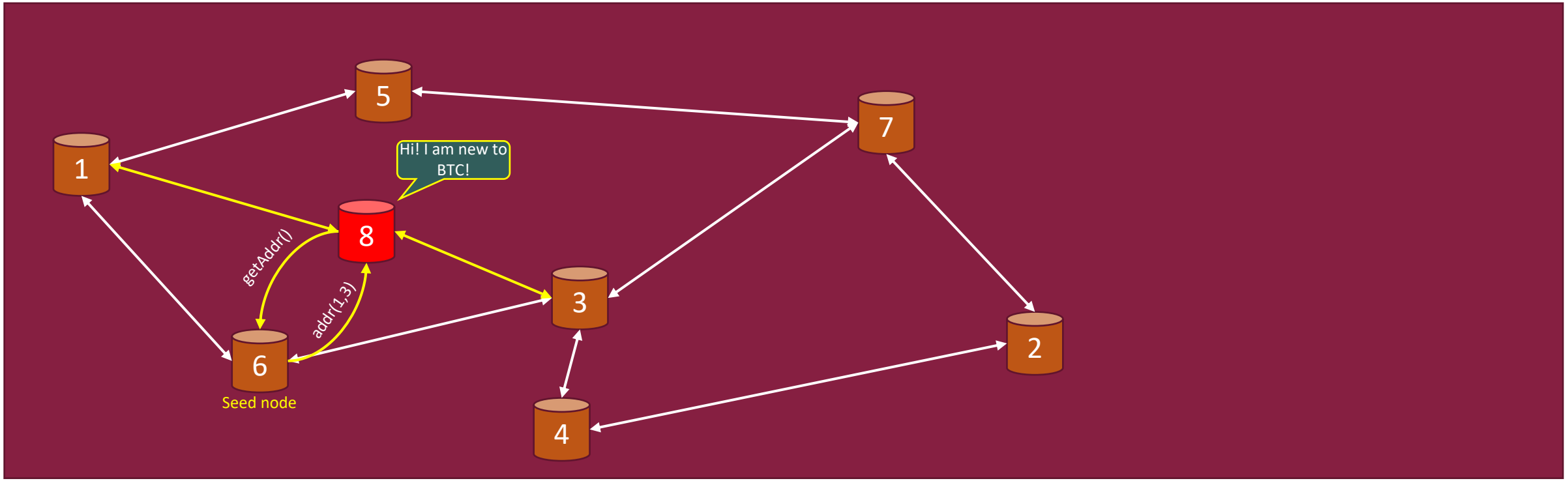
Bitcoin Network

Ad-hoc network (TCP port 8333) with random topology

All nodes are equal – no special/master nodes

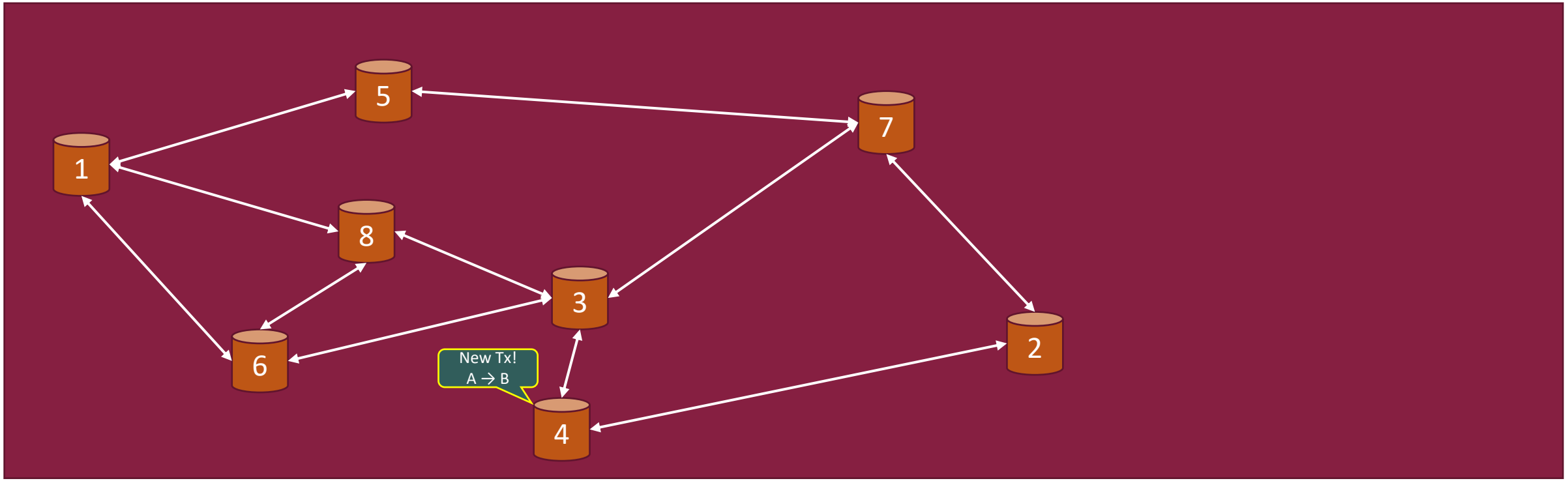
Nodes can join and leave at any time (permissionless/ public blockchain)

Forget non-responding nodes after 3 hours



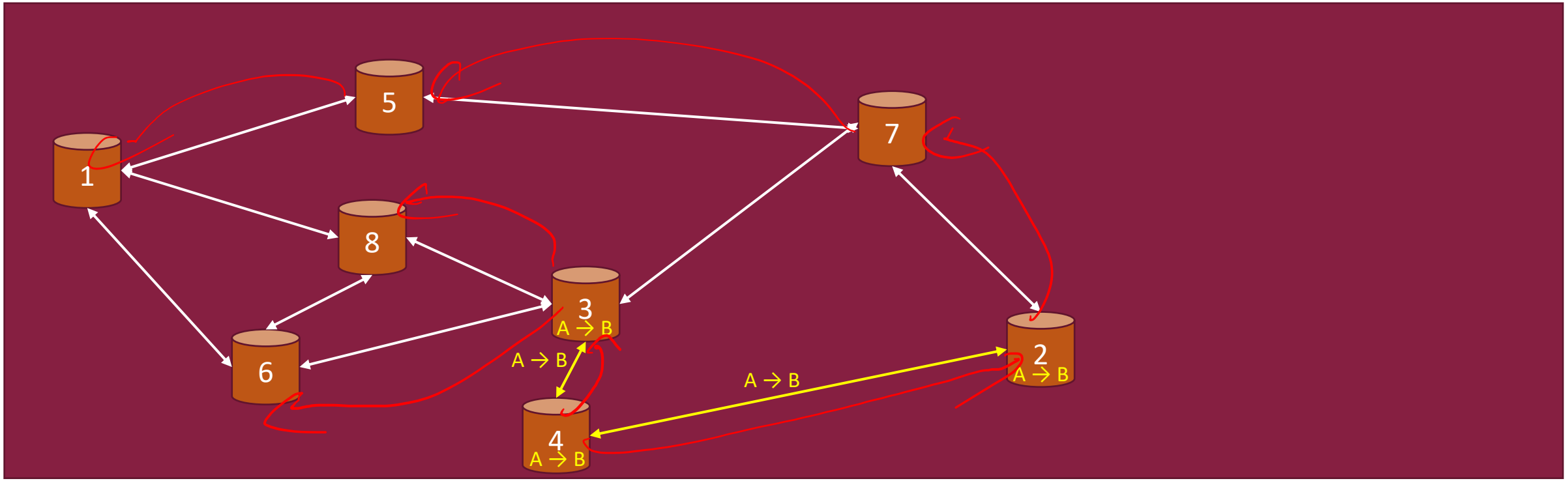
Transaction Propagation

Nodes listen and relay new transactions to other nodes via flooding (gossip) protocol



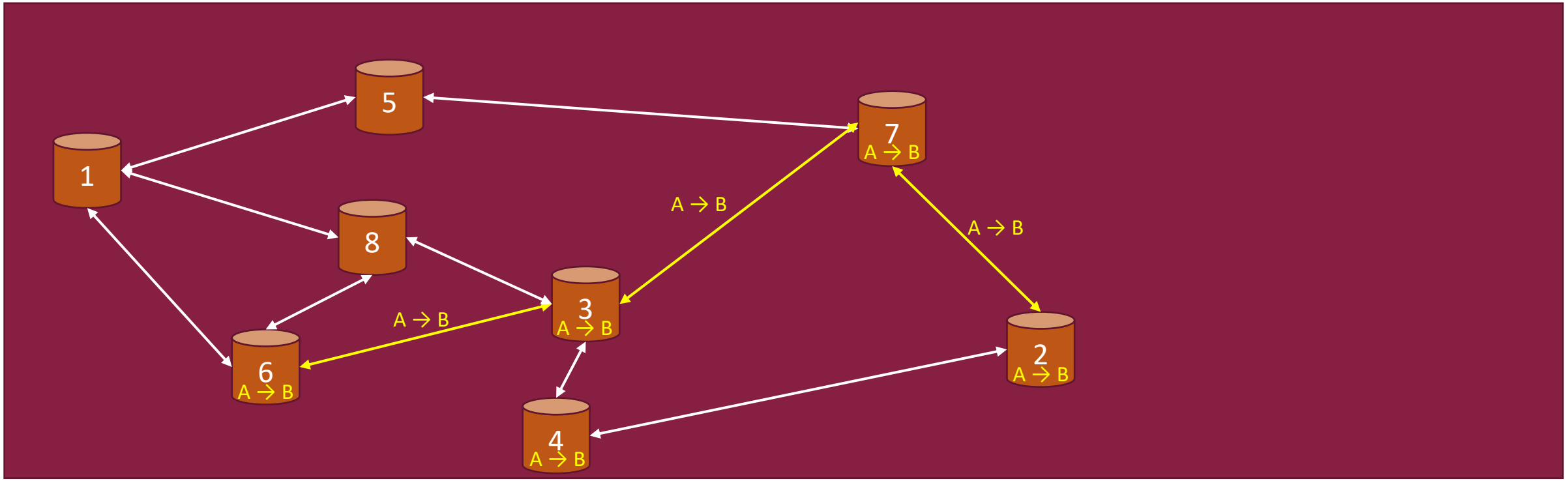
Transaction Propagation

Nodes listen and relay new transactions to other nodes via flooding (gossip) protocol



Transaction Propagation

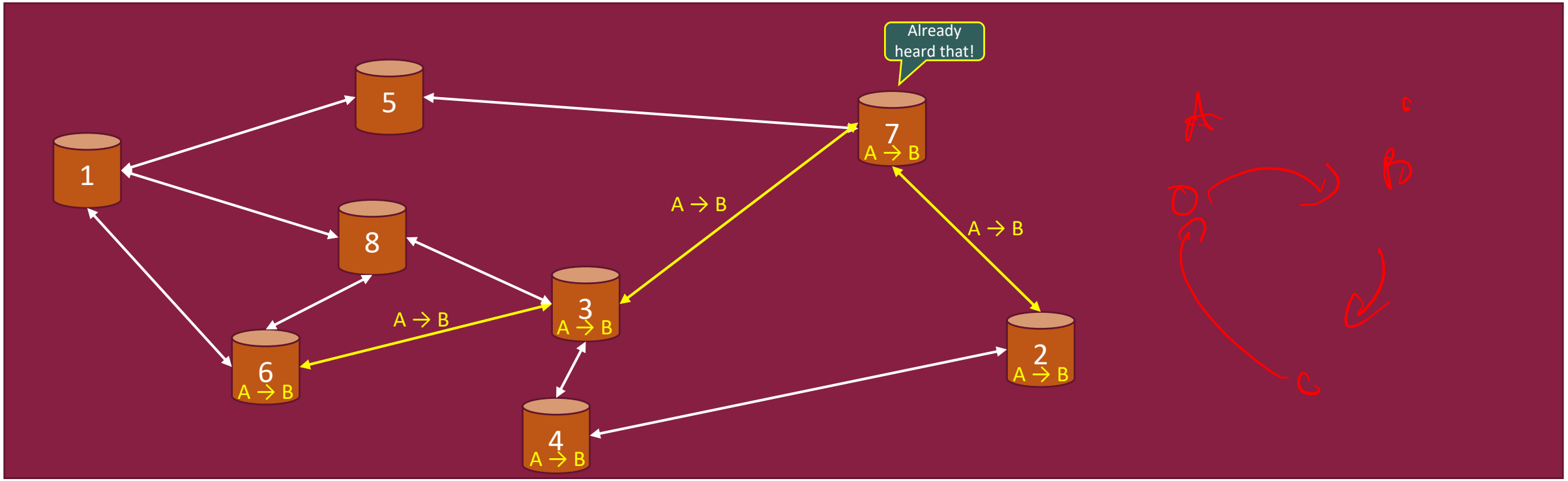
Nodes listen and relay new transactions to other nodes via flooding (gossip) protocol



Transaction Propagation

Node relays a transaction when

- Transaction is valid with current blockchain (e.g., coin not redeemed elsewhere)
- Have not seen the transaction before
  - Avoid infinite loops
- Does not conflict with other transactions it has relayed
  - Avoid double-spend attacks



Transaction Propagation

Four conditions for a node to relay a transaction:

1. Valid transaction with current blockchain (e.g., coin not redeemed elsewhere)
2. Outputs redeemed were not spent elsewhere

   Avoid double-spend attacks (to be explained later)
3. Have not seen the transaction before

   Avoid infinite loops
4. (default) Script matches a whitelist

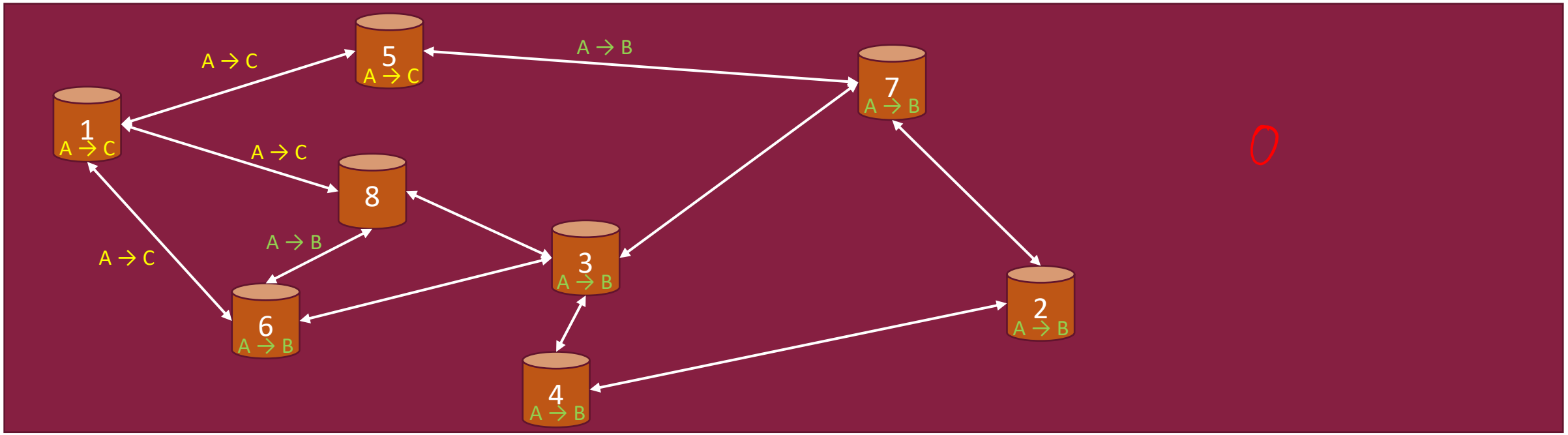
   Avoid unusual script

Sanity checks only!
Some (malicious) nodes may not follow these rules

Transaction Propagation

Race condition may arise since nodes may have different transaction pool

- Transactions may get conflict
- <u>Default behavior:</u> retain whatever the node hear first
- Network position matters
- Implement some logics to handle race condition

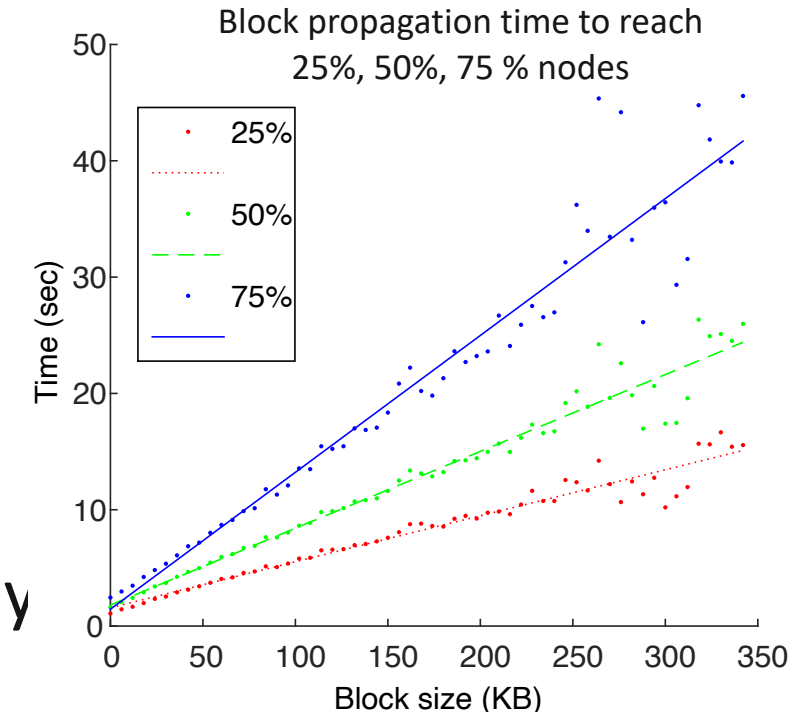


Block Propagation

Almost identical with transaction propagation

Node relays a new block when

- Block meets the hash target
- Block has all valid transactions
  - Run all scripts, even if you would not relay
- Block builds on current longest chain
  - Avoid forks
- Sanity check only, may be ignored by malicious node



Source: Yonatan Sompolinsky and Aviv Zohar: "Accelerating Bitcoin's Transaction Processing", 2014

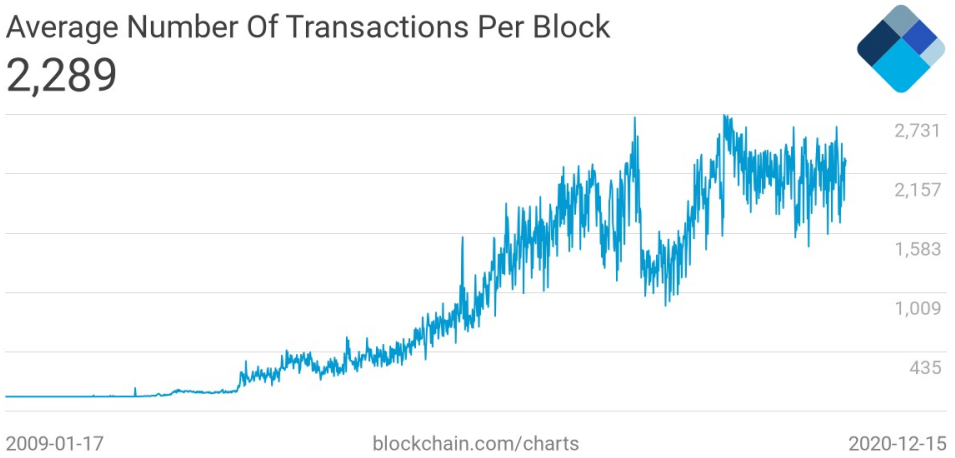
How does a bitcoin transaction/block look like?

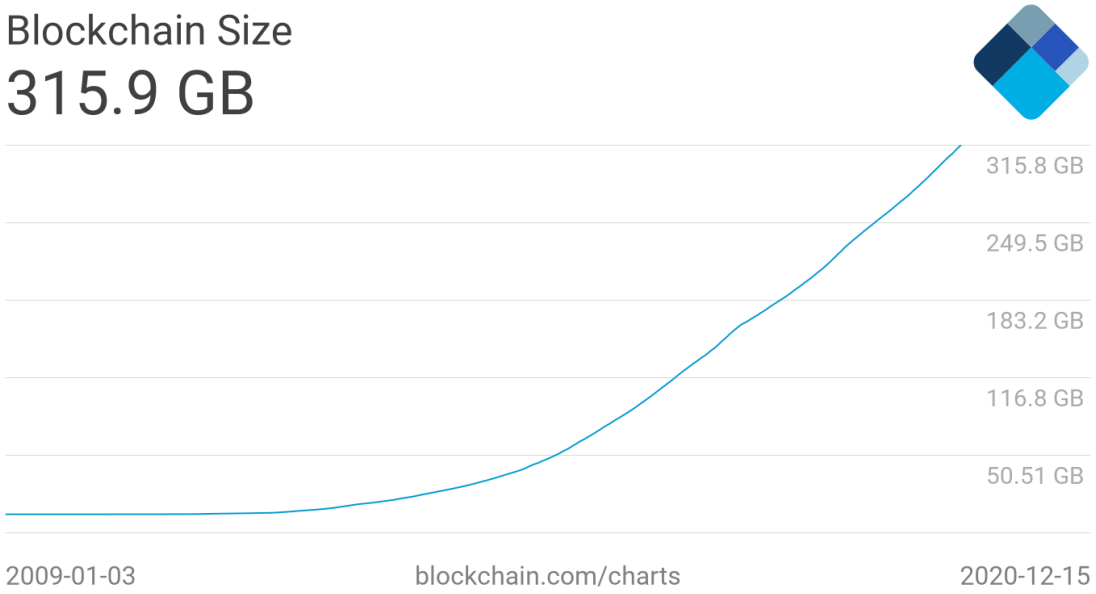
Bitcoin Network Stats

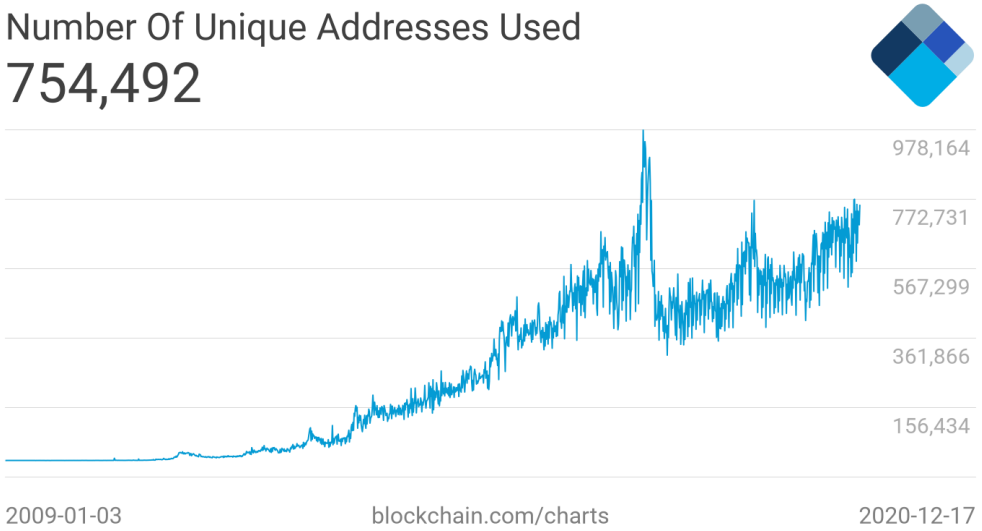
Exponential growth

Hard to measure precisely due to dynamism

Nodes join and leave frequently







Bitcoin Network Stats

Two common types of nodes in bitcoin network

- **Full** nodes (~10K, maybe dropping!)
  - Permanently connected
  - Store entire block chain (~316 GB as of 2020)
  - Listen every transaction and forward to every node

- **Thin/SPV** nodes
  - Do not store everything → save storage cost
  - Only store block headers (~100 MB)
  - Request transactions as needed
  - Trust full nodes

Bitcoin Network Stats

90% nodes run Core Bitcoin (C++)

- Some use out-of-date version

Other implementations adapted and integrated to Bitcoin network successfully

- BitcoinJ (Java)
- Libbitcoin (C++)
- Btcd (Go)

Original Satoshi client

Bitcoin Address

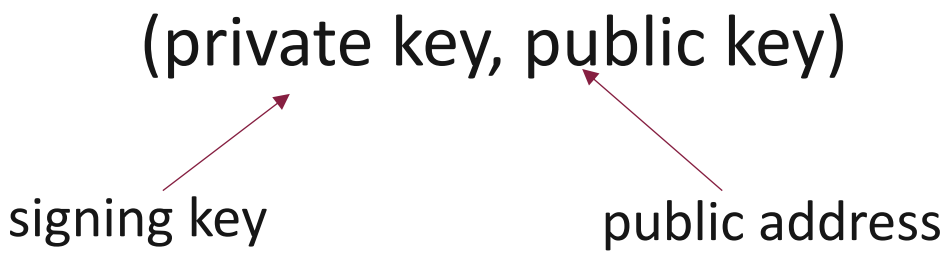
Bitcoin Address

To make a transaction, nodes need to get some information:

- Their secret signing key
- Recipient's address
- Some info from the public blockchain

Pseudonymity is the main goal of bitcoin

Recap: public-key cryptography



Key management

Bitcoin Public Address

Hash of public key

Why hash?

Encode as text string: base-58 notation

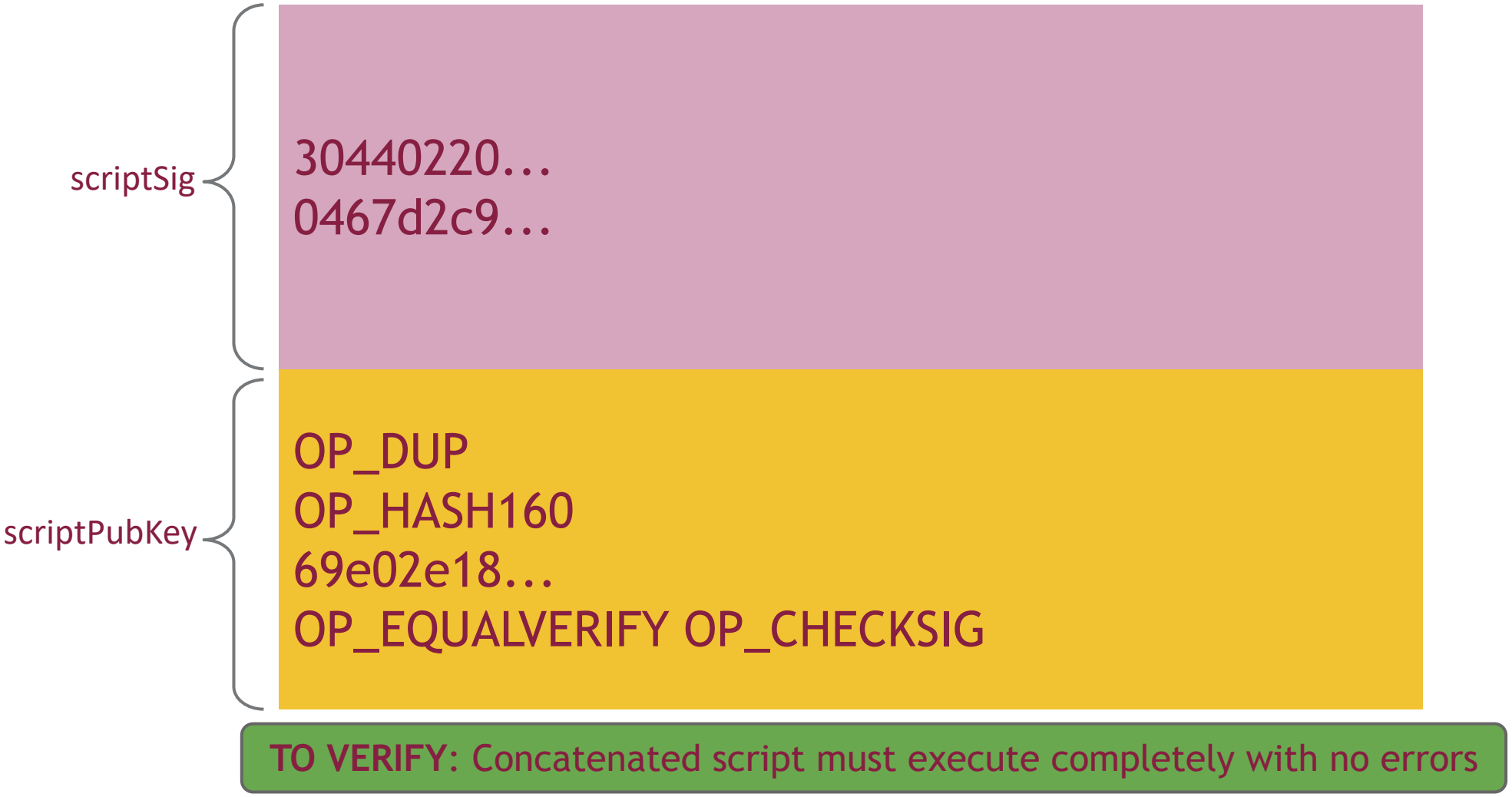
Why 58?

123456789~~0~~ABCDEFGHJKLMN~~O~~PQRSTUVWXYZabcdefghijkmnopqrstuvwxyz

or use QR code

Bitcoin Address

Sender and recipient “addresses” are indicated in **scripts** (will be discussed later)



Bitcoin Wallet

Manage your private keys

Three main goals:

- Availability: You can spend your coins
- Security: Nobody else can spend your coins
- Convenience: easy to manage

Easiest (and convenient) way:

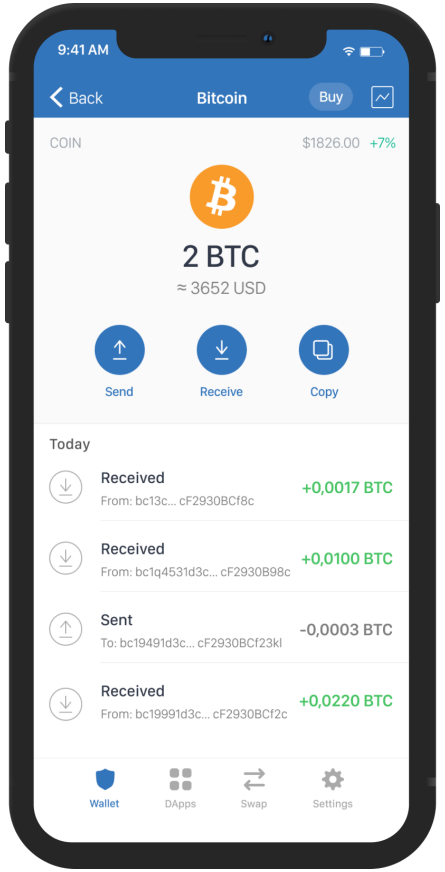
- Store key in a file on your computer or phone
- As available as your device
  - Device lost/wiped ⇒ key lost ⇒ coins lost
- As secure as your device
  - Device compromised ⇒ key leaked ⇒ coins stolen

Wallet Software

Keep track of your coins

Manage your keys

Nice user interface

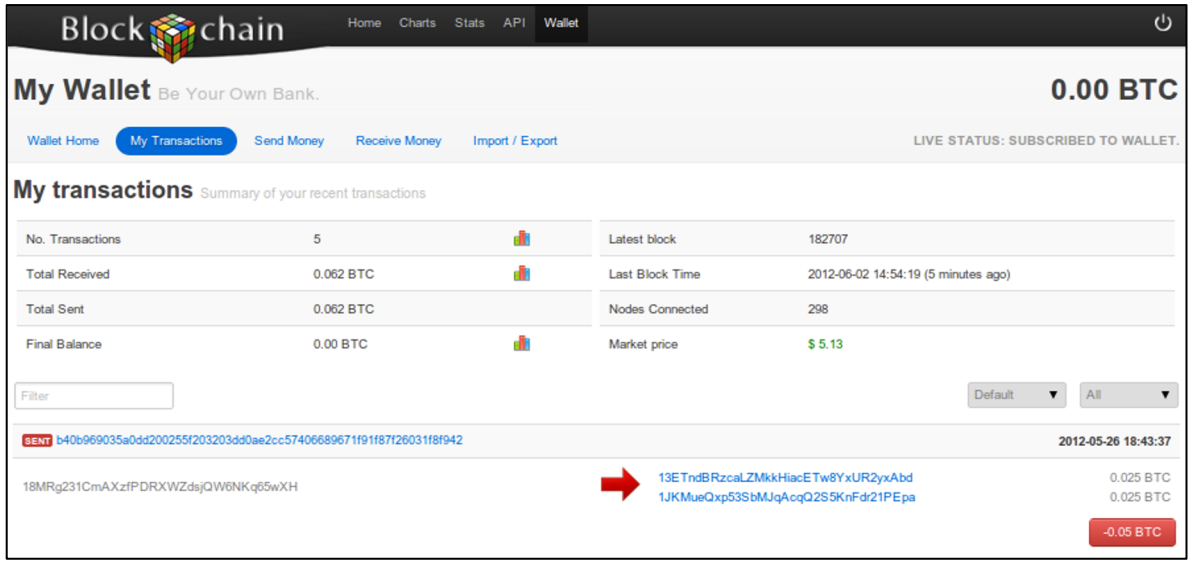


<u>Nice trick:</u> use a separate address/key for each coin

- Benefits privacy (looks like separate owners)
- Wallet can do the bookkeeping, user needn’t know

Online Wallet

Like a local wallet but in the cloud



Runs in your browser

- Site sends code and stores keys
- Log in to access wallet

<u>Pros:</u>

- Convenient, nothing to install
- Availability, works on multiple devices

<u>Cons:</u>

- Security vulnerability
- Malicious site

Should be maintained by security professionals

Hot vs. Cold Storage

Hot storage

online

convenient but risky

Cold storage

offline

archival but safer

separate keys

Hot vs. Cold Storage

Hot storage

Cold storage

hot secret key(s)

cold address(es)

payments

cold secret key(s)

hot address(es)

Hot vs. Cold Storage

Hot storage

online

hot secret key(s)

cold address(es)

payments

Cold storage

offline!!

New (address-key) pair per coin sent to cold

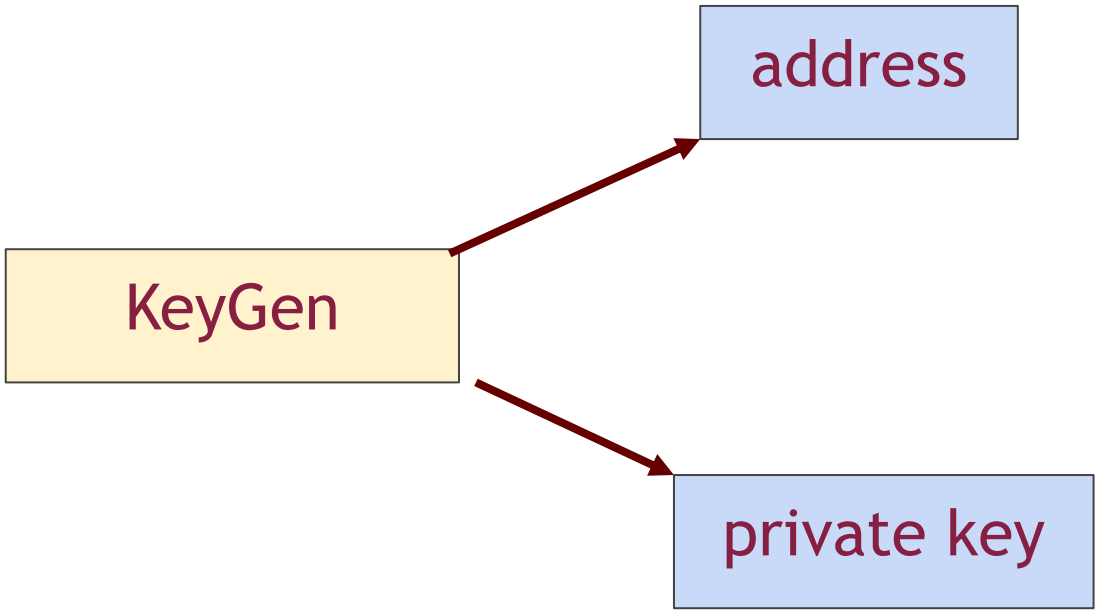
What if cold wallet goes dark?

Generate many addresses/keys beforehand

Periodical connection w/ online wallet

Hierarchical wallet

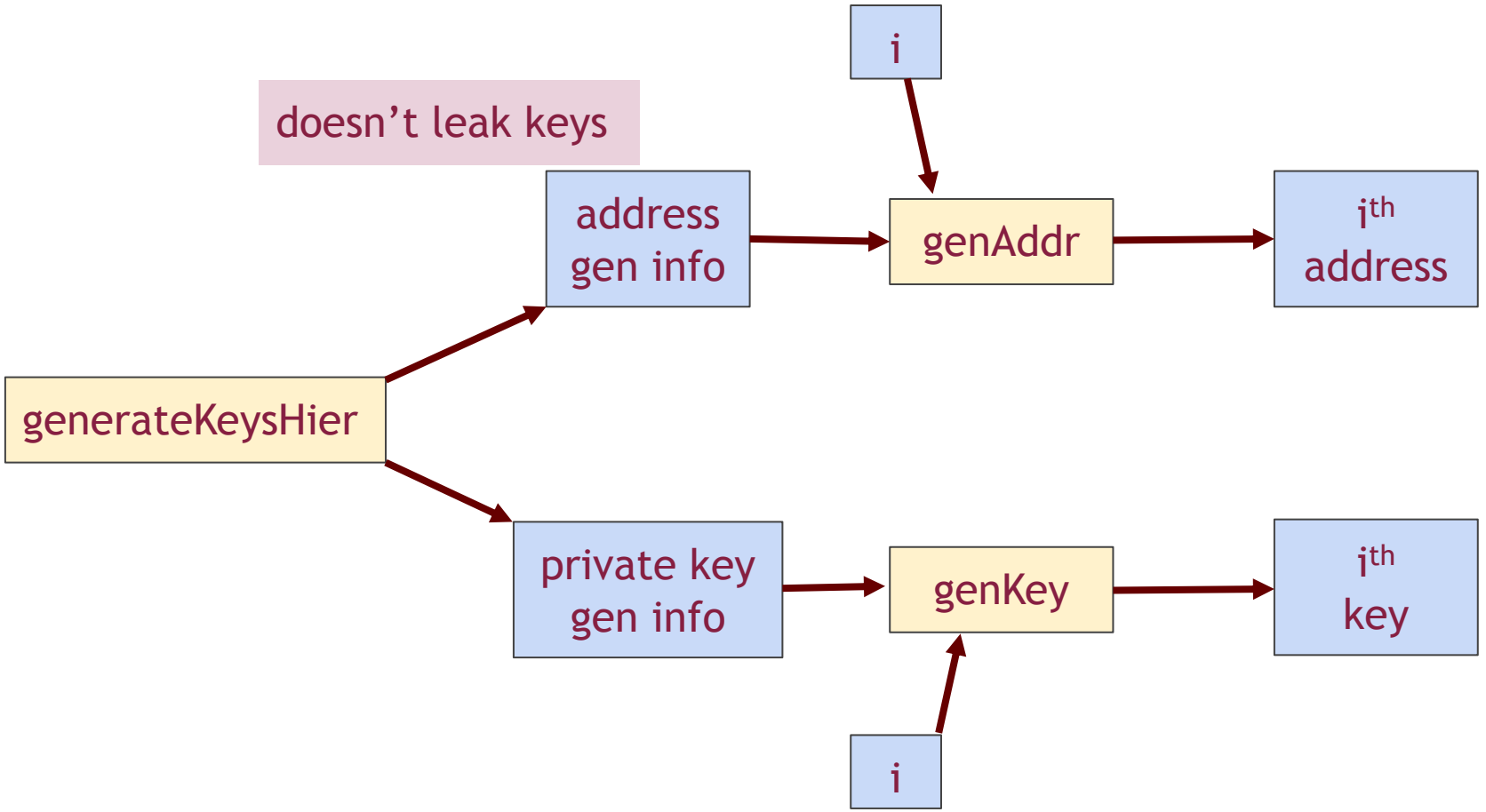
Regular Key Generation



Hierarchical Key Generation

Allow cold side to generate unlimited number of addresses

Hot side can know all these address with only one-time communication



Example

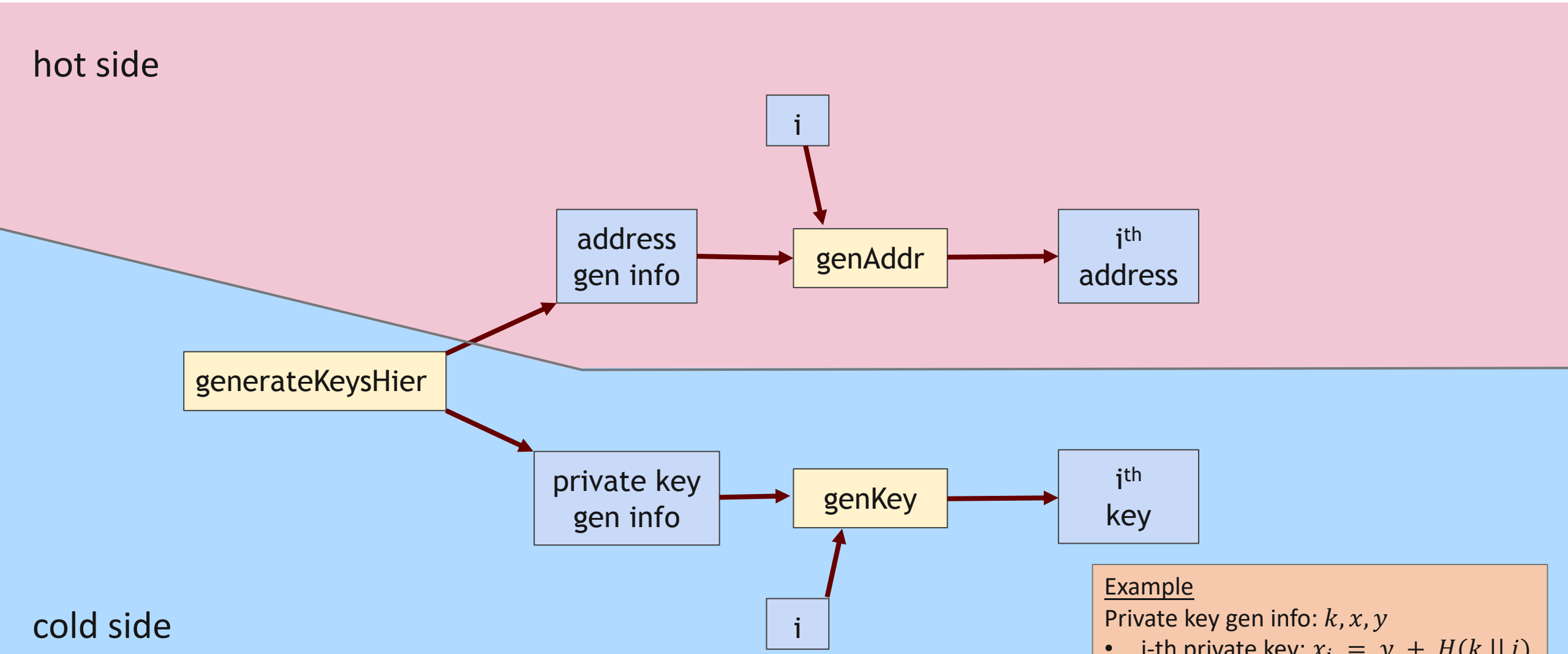
Private key gen info: k, x, y

- i-th private key: $x_i = y + H(k \,||\, i)$

Address gen info: k, g^y

- i-th public key $g^{x_i} = g^y \cdot g^{H(k||i)}$
- i-th address: $H(g^{x_i})$

Regular Key Generation



Example

Private key gen info: k, x, y

- i-th private key: $x_i = y + H(k \,||\, i)$

Address gen info: k, g^y

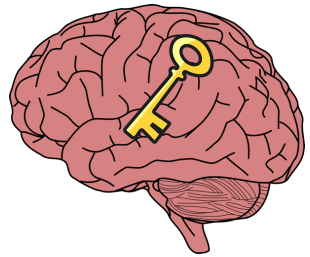
- i-th public key $g^{x_i} = g^y \cdot g^{H(k||i)}$
- i-th address: $H(g^{x_i})$

Other Cold Info Storage Mechanism

Stored in a device locked in a safe

“Brain” wallet

Encrypt info with passphrase that we remember



Paper wallet

Print info on paper

Lock up the paper

Tamper-resistant device

Device will sign things with keys inside, but won’t divulge keys

Combination of multiple methods above

Splitting and Sharing Keys

Split and store key in multiple locations

Distributed Key Storage

Crypto Tools: Secret Sharing

Split secret into N pieces, such that

Given any K pieces, can reconstruct the secret

Less than K pieces, don't learn anything

Example: N=2, K=2 P = a large prime

S = secret in [0, P)

R = random in [0, P)

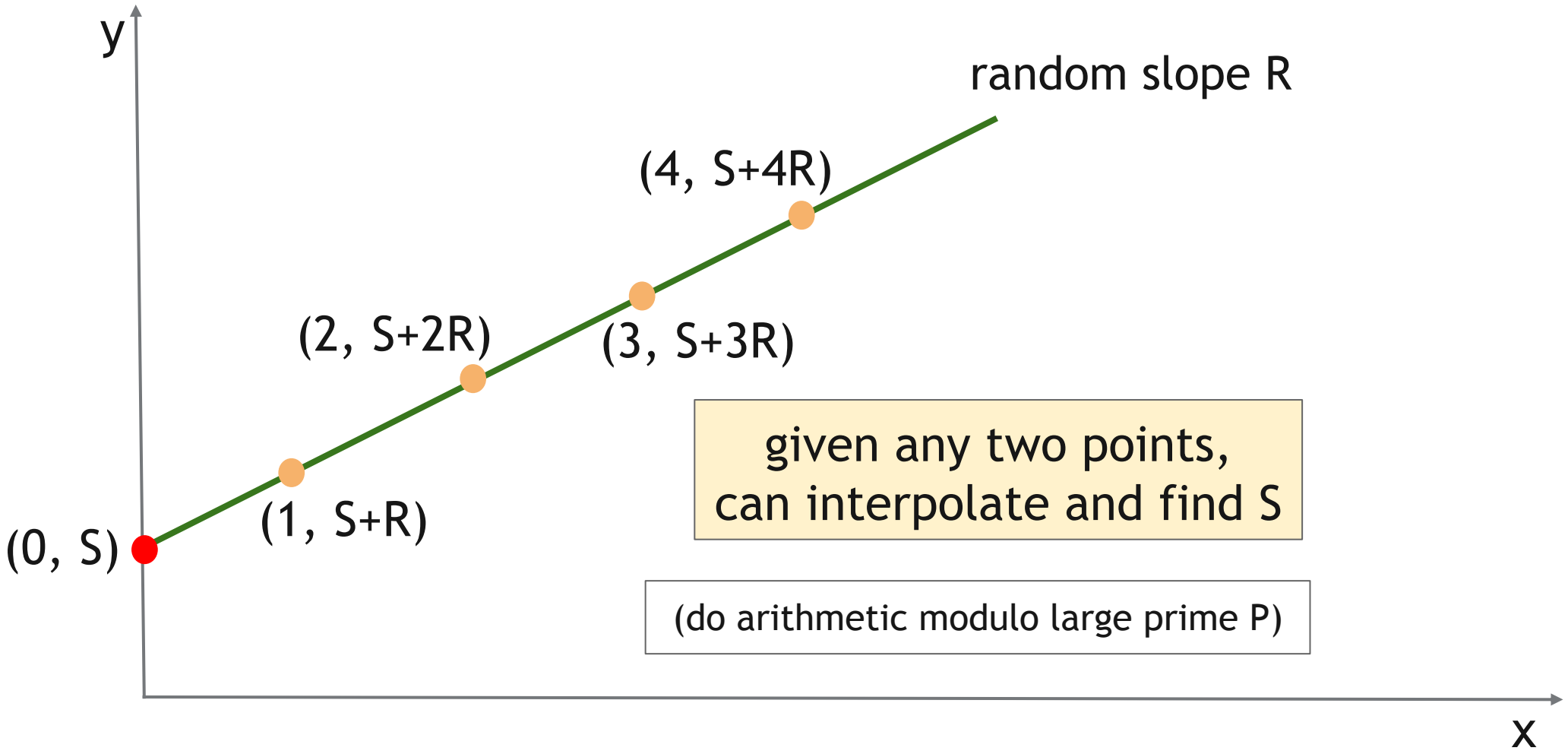
split:

$X_1 = (S+R) \bmod P$ $X_2 = (S+2R) \bmod P$

reconstruct:

$(2X_1-X_2) \bmod P = S$

Splitting and Sharing Keys



Splitting and Sharing Keys

Pros:

- Improved resiliency (distributed key storage)
- Adversary must compromise several shares to get the key

Cons:

- To sign, need to bring some shares altogether
- Reconstruct the key before signing ⇐ vulnerable

Threshold signature

- Sign (in distributed manner) without reconstructing the key
- Complex math behind (won't discuss here)

Multi-signature

- Address directly split among multiple independent keys
- Key stored in different locations, signatures produced separately
- *k*-out-of-*n* valid signature to create valid transaction

Bitcoin Transaction

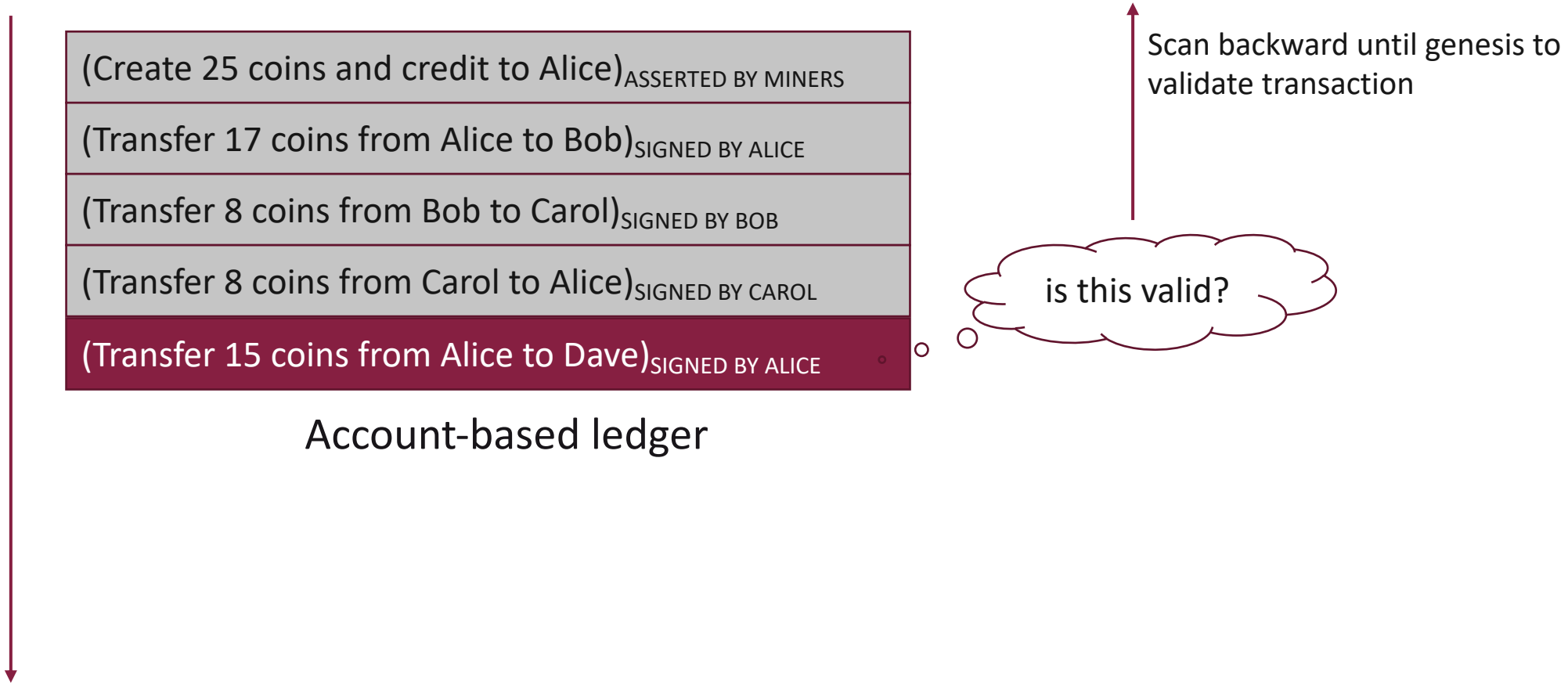
Bitcoin Transactions

Nodes now have all necessary information to make a transaction

How does a bitcoin transaction look like?

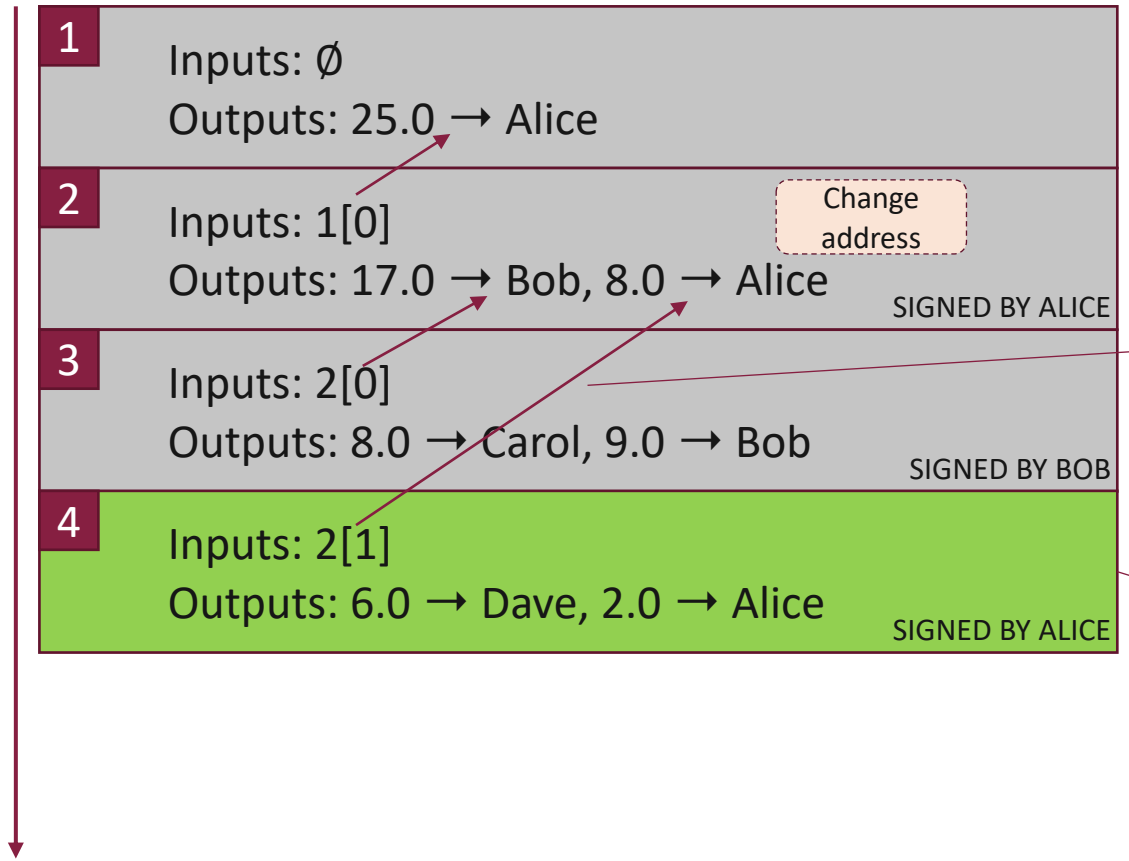
Bitcoin Transaction

Fundamental building block in bitcoin network



Bitcoin Transaction

Transaction-based ledger



hash pointer

Check total input = total output

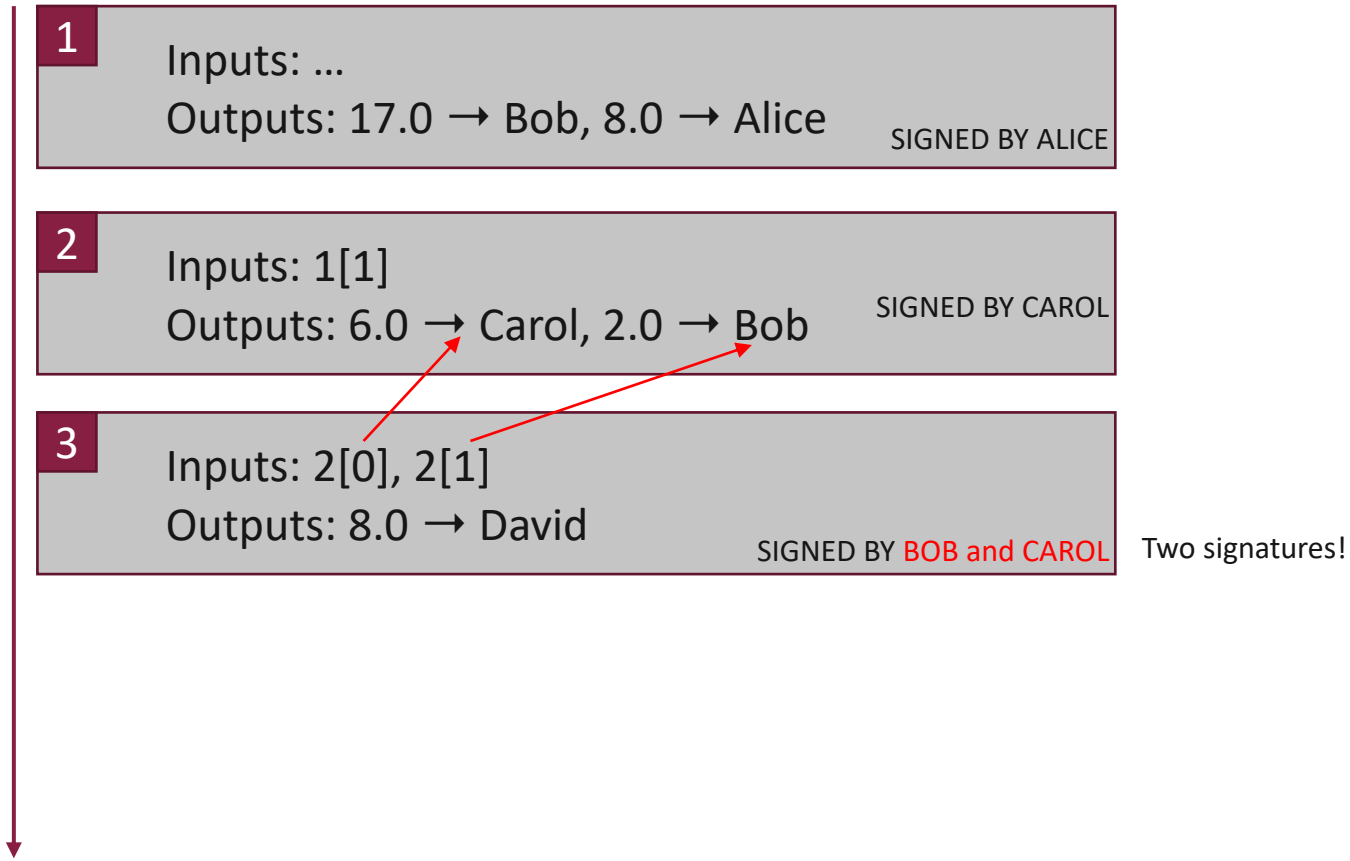
Efficient verification:

No need to go all to the beginning of the chain

Bitcoin Transaction

Transaction-based ledger

Joint payment

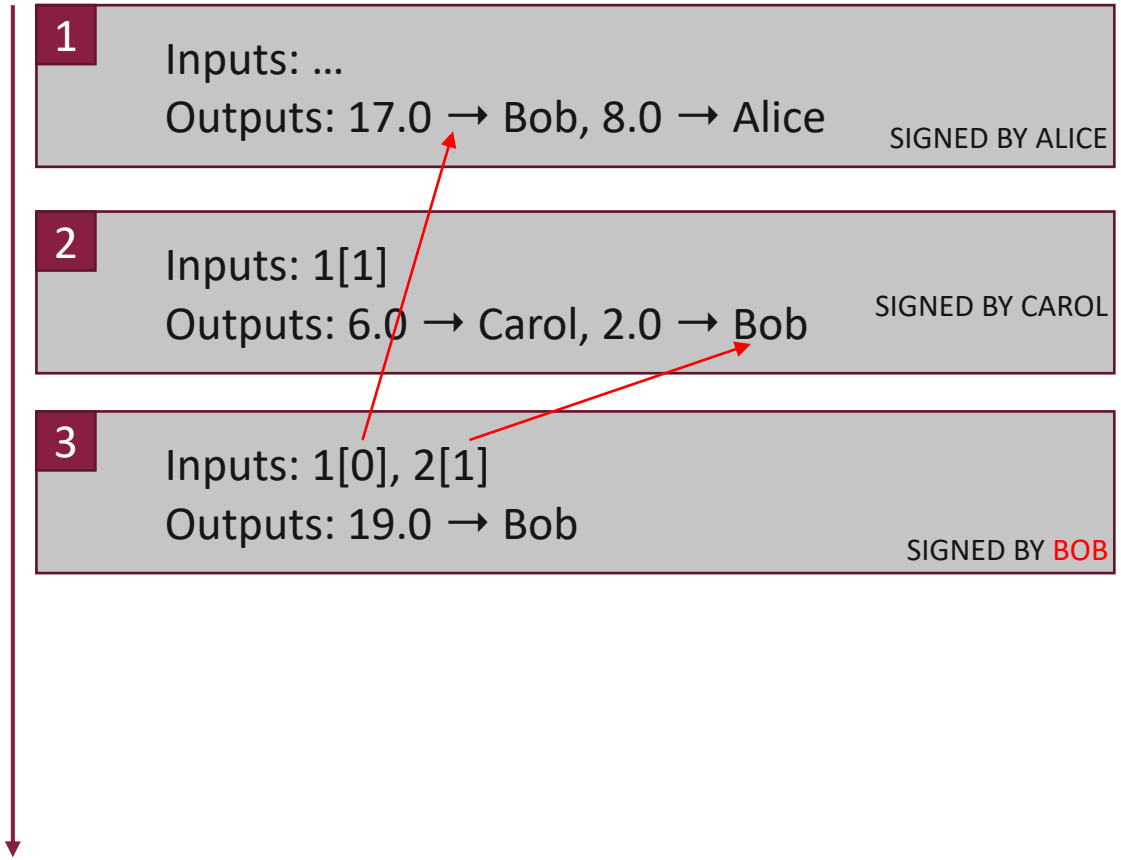


Two signatures!

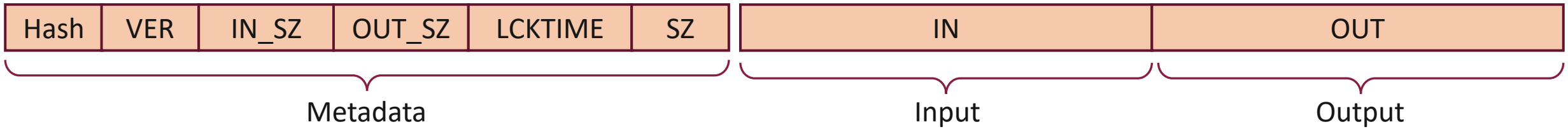
Bitcoin Transaction

Transaction-based ledger

Consolidate fund



Bitcoin Transaction Structure



Field	Description
Hash	Hash of the entire transaction (unique ID for pointer)
VER	Version being used (for some rules to be applied)
IN_SZ	Number of inputs
OUT_SZ	Number of outputs
LCKTIME	Unix timestamp or block number
SZ	Total size of transaction (in bytes)
IN	List of transaction inputs
OUT	List of transaction outputs

Bitcoin Transaction Script

```
{
“hash”: “5a2590fbe0a90ee8e…”   transaction hash
”ver”: 1, version
“in_sz”: 1,  # of inputs
“out_sz”:1,  # of outputs
“lcktime”:0,  lock time (TBD)
“sz”:404, size
“in”:[
    {
        “prev_out”:{                                  } previous transaction
            “hash”: “3be4ac9728a0823…”
            “n”:0 index
         },
        “scriptSig”:”30440…” } Signature
    }
    ],
“out”:[
    {                                          recipient address?
        “value”: “10.122877097”  value
        “scriptPubKey”:”OP_DUP OP_HASH160 69a02e18b… OP_EQUALVERIFY OP_CHECKSIG”
    }
}
```

Bitcoin Script

A language that contain instructions to be executed

Concatenated script must be executed completely without errors

Design Goals for Script

- Built for Bitcoin
- Simple, compact
- Support cryptography
- Stack-based language
- Limits on time/memory
- No looping

Bitcoin Script Instructions

256 opcodes total (15 disabled, 75 reserved)

- Arithmetic
- If/then
- Logic/data handling

Crypto!

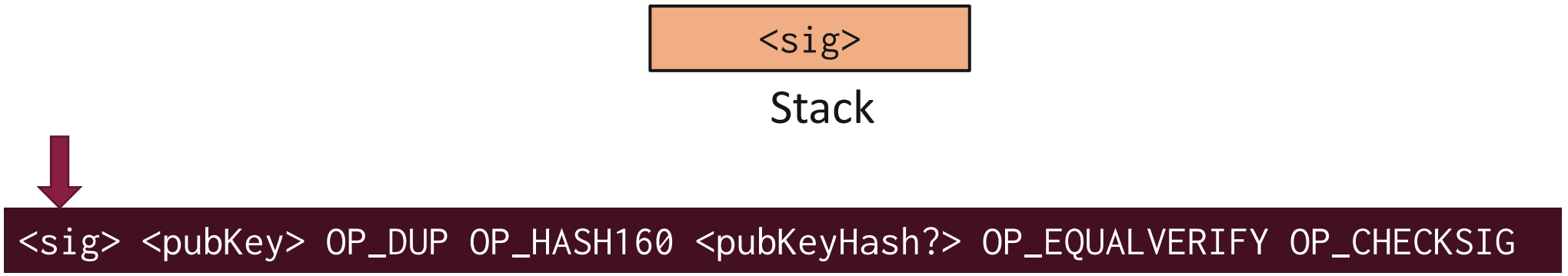
- Hashes
- Signature verification
- Multi-signature verification

OP_CHECKMULTISIG

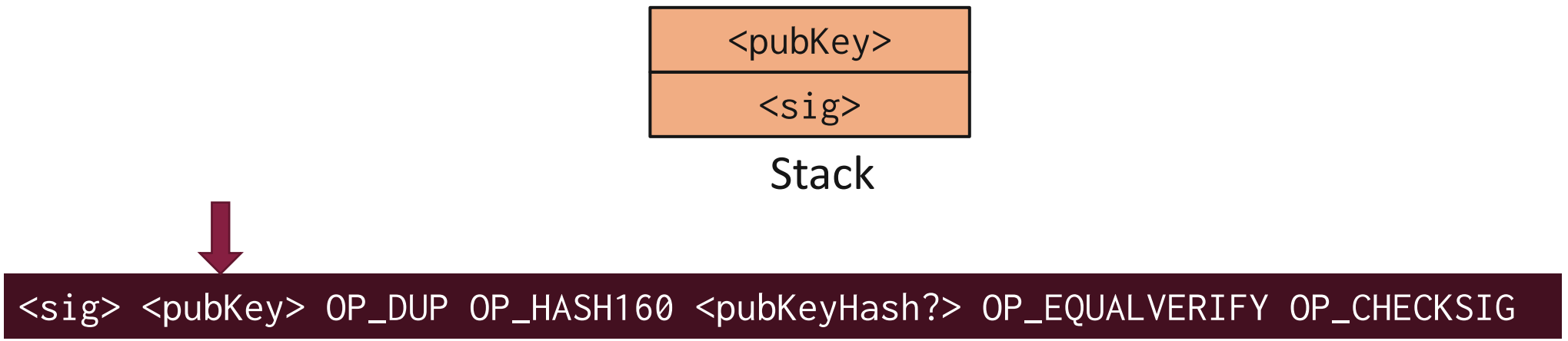
- Built-in support for joint signatures
- Specify n public keys
- Specify t (threshold)
- Verification requires t signatures

BUG ALERT: Extra data value popped from the stack and ignored

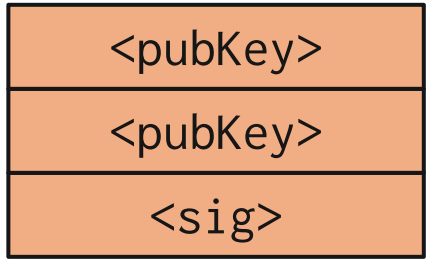
Bitcoin Script Execution



Bitcoin Script Execution



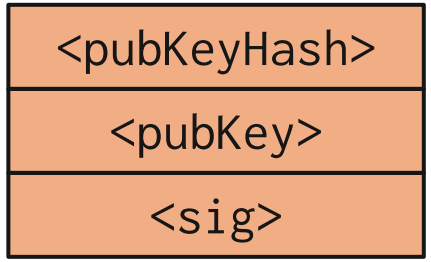
Bitcoin Script Execution



Stack

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

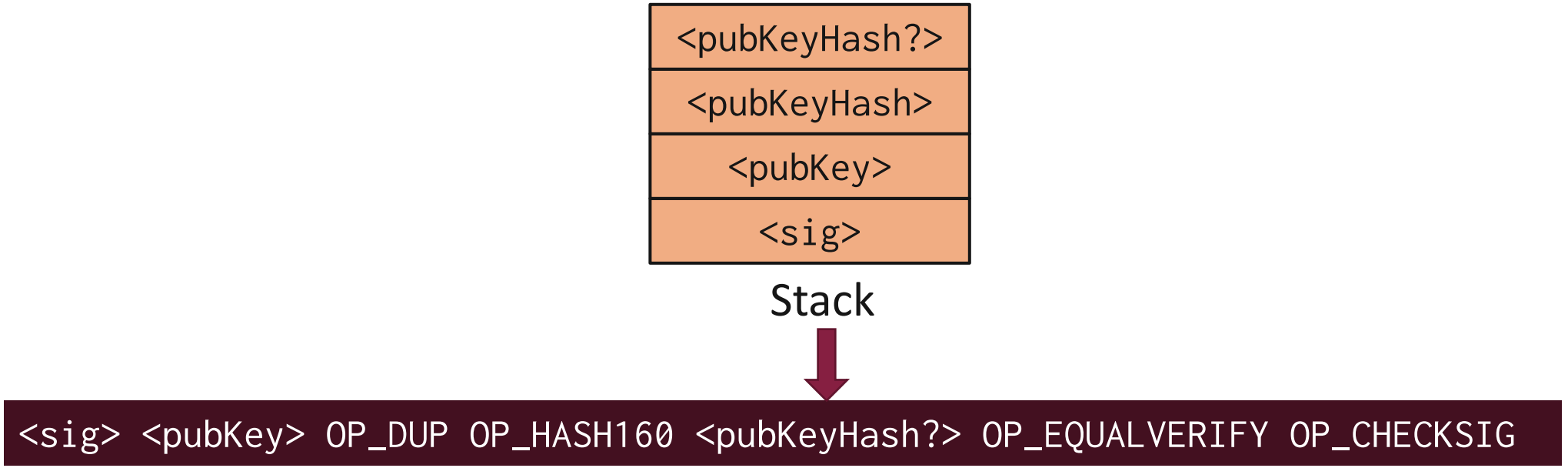
Bitcoin Script Execution



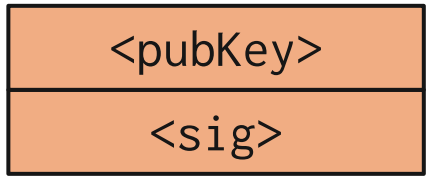
Stack

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

Bitcoin Script Execution



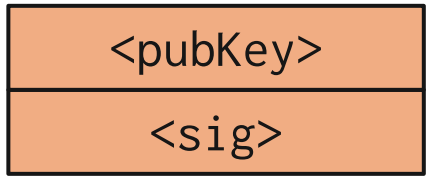
Bitcoin Script Execution



Stack

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

Bitcoin Script Execution



Stack

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

Bitcoin Script Execution

- If no error, transaction is validated

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

Bitcoin Scripts in Practice

Most nodes whitelist known scripts

99.9% are simple signature checks

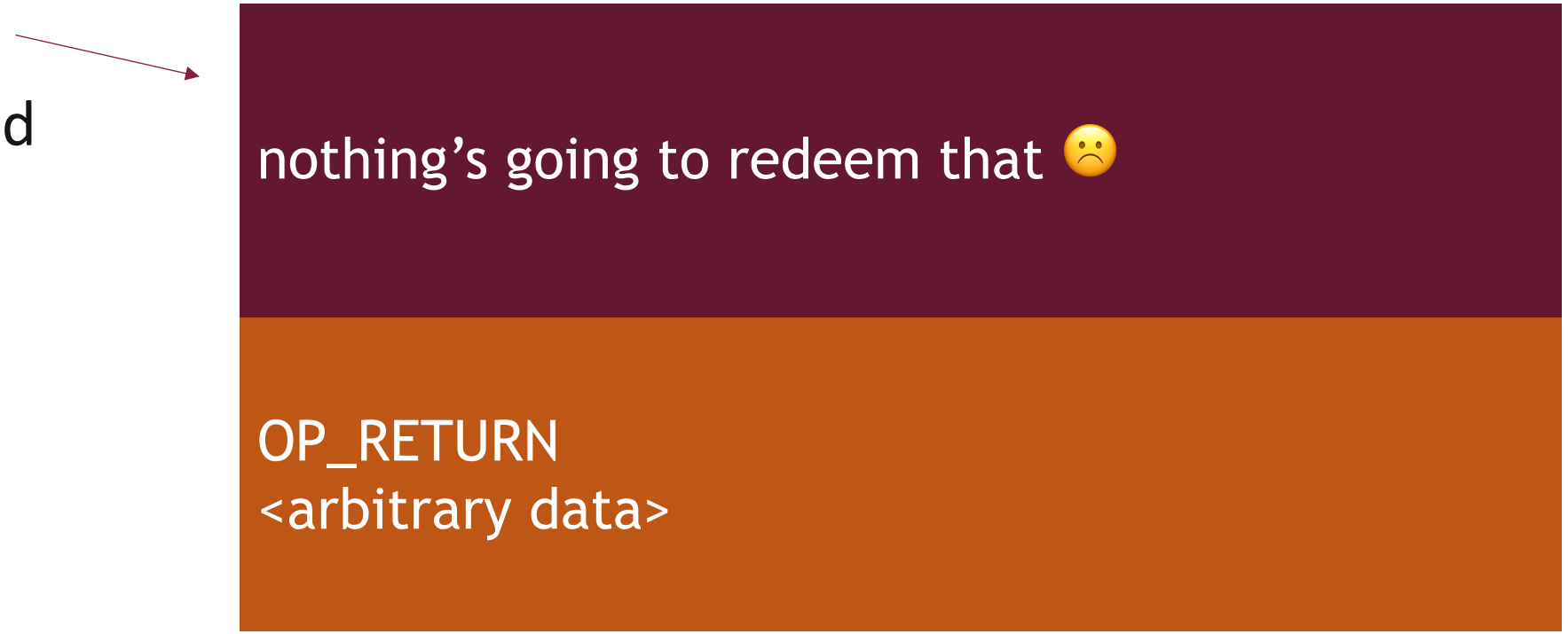
~0.01% are MULTISIG

~0.01% are Pay-to-Script-Hash (newly added to original BTC)

Remainders are errors

Proof-of-burn script

- Cannot be redeemed
- Destroy coin



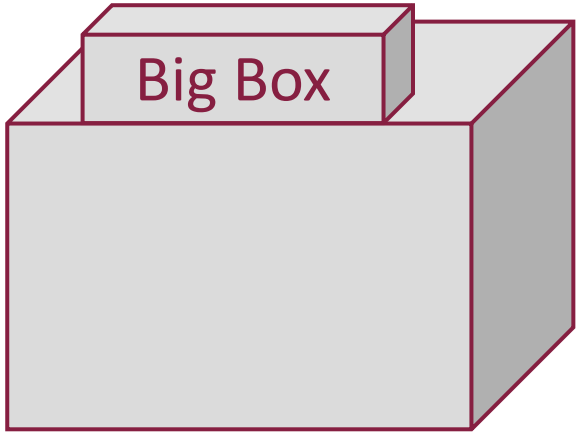
Pay-to-Script-Hash

Should senders specify scripts?

Complicate things to sender

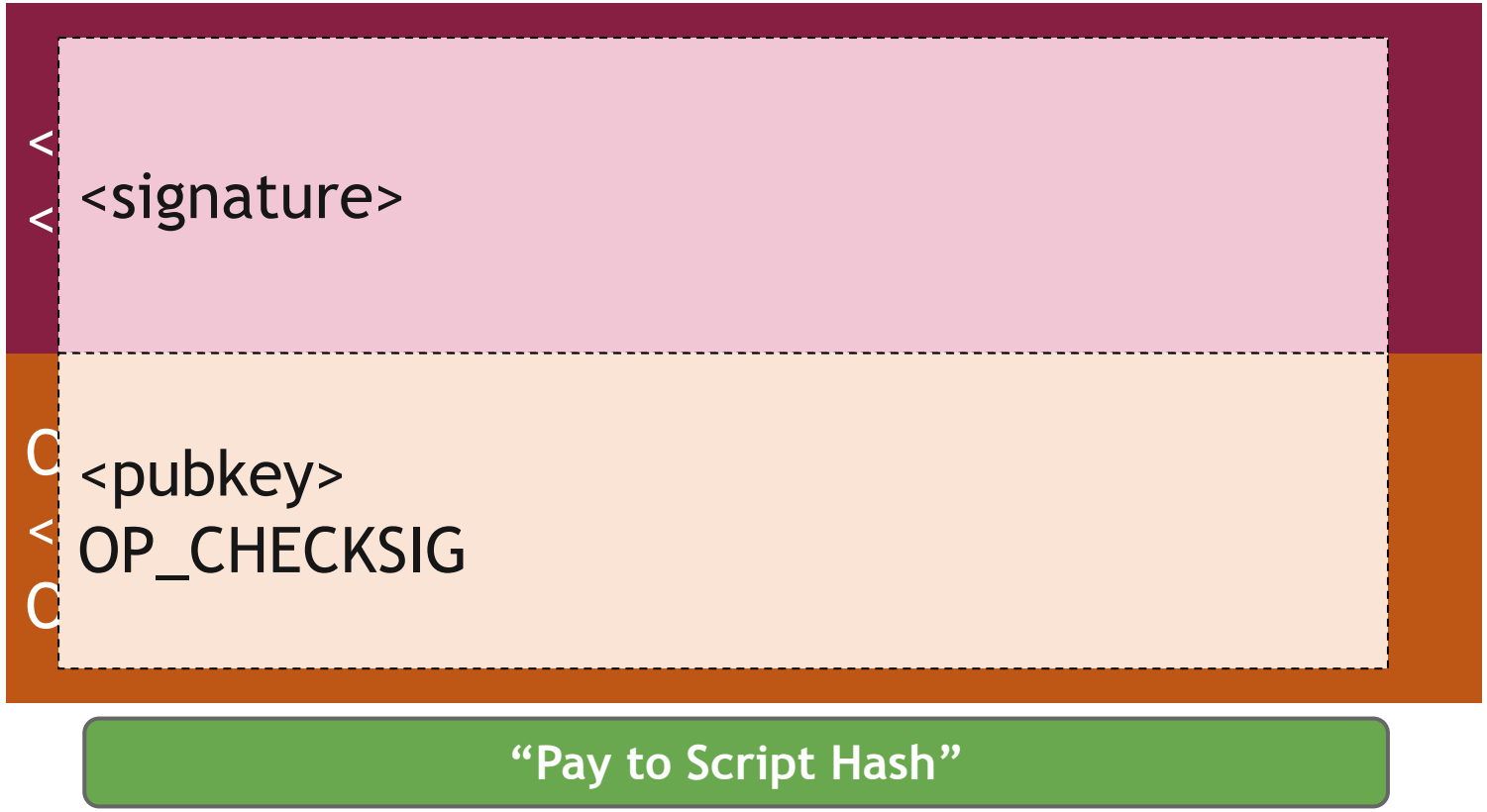
I'm ready to pay for my purchases!

Cool! Well we're using MULTISIG now, so include a script requiring 2 of our 3 account managers to approve. Don't get any of those details wrong. Thanks for shopping at Big Box!



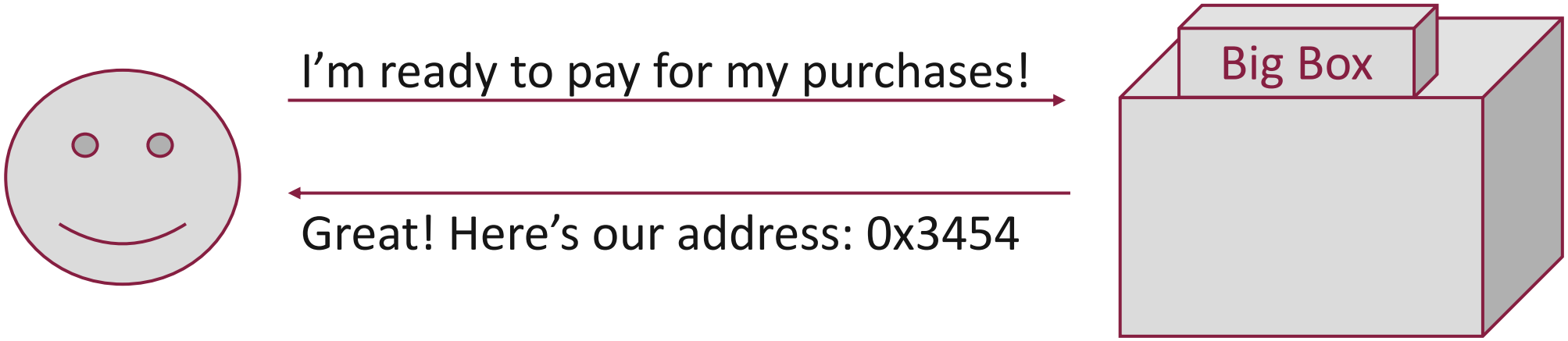
Pay-to-Script-Hash

- Idea: use the hash of script



Pay-to-Script-Hash

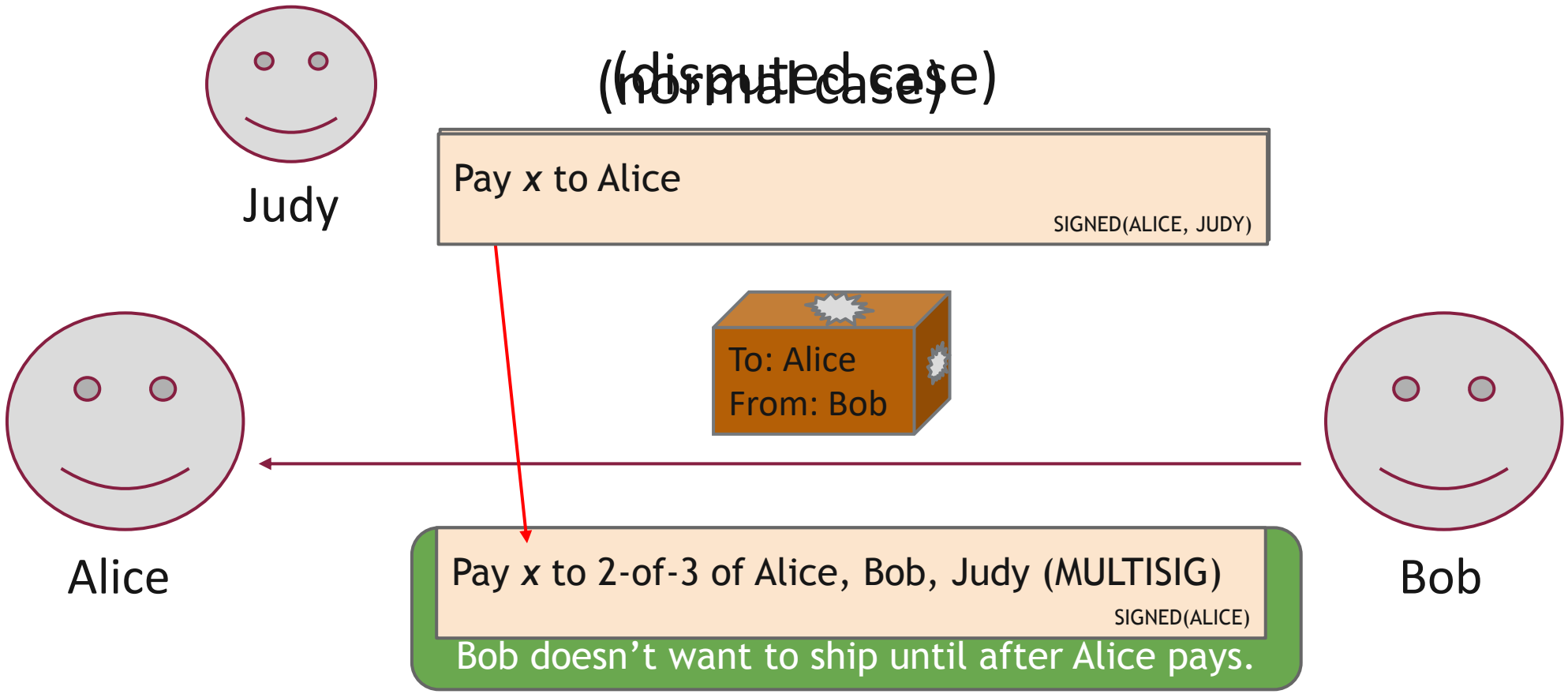
Idea: use the hash of script



Bitcoin Scripts Applications

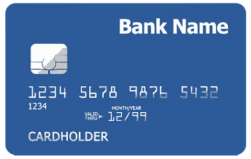
Escrow Transactions

Third party to approve/dispute transactions



Bitcoin Scripts Applications

Green Addresses



It's me, Alice! Could you make out a green payment to Bob?

Bank

004 days since last double spend!

Faraday cage

Alice

Pay x to Bob, y to Bank

No double spend

SIGNED(BANK)

Bob

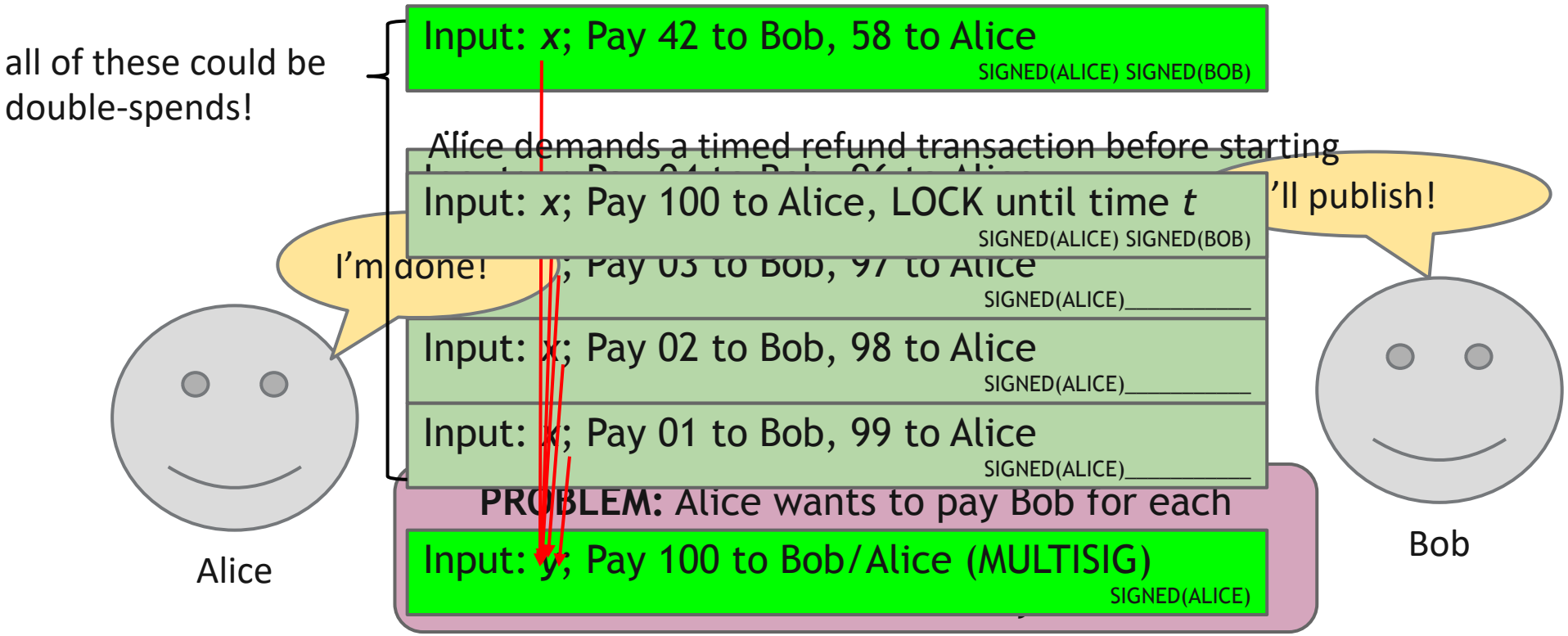
PROBLEM: Alice wants to pay Bob.
Bob can't wait 6 verifications to guard against double-spends, or is offline completely.

Bitcoin Scripts Applications

Micropayments

What if Bob never signs the last transaction?

- Alice coins sit in escrow forever



Bitcoin Scripts Applications

lock_time

- Lock transactions until some time in the future
- If Bob does not sign the last transaction, Alice can get refund


```
{
    "hash":"5a42590...b8b6b",
      "ver":1,
      "vin_sz":2,
      "vout_sz":1,
      "lock_time":315415,
      "size":404,
...
}
```

More Advanced Bitcoin Scripts

Multiplayer lotteries

Hash pre-image challenges

Coin-swapping protocols

“Smart contracts”

(to be discussed in next lectures)

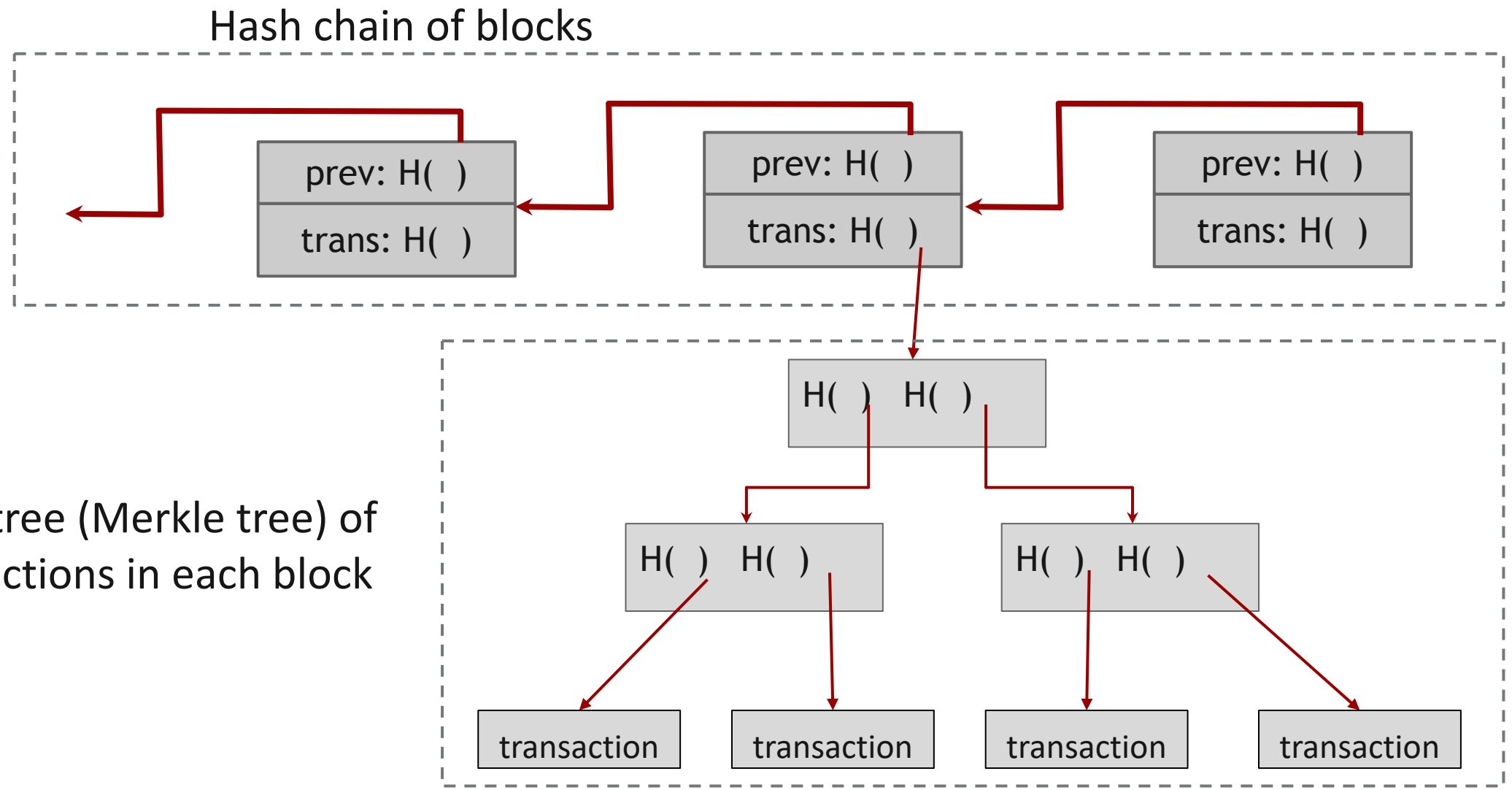
Bitcoin Block

Contain multiple transactions

Why bundle transactions together?

- Single unit of work for miners
- Limit length of hash-chain of blocks
  - Faster to verify history

Bitcoin Block Structure



Hash tree (Merkle tree) of transactions in each block

Bitcoin Block Structure in Script

block header

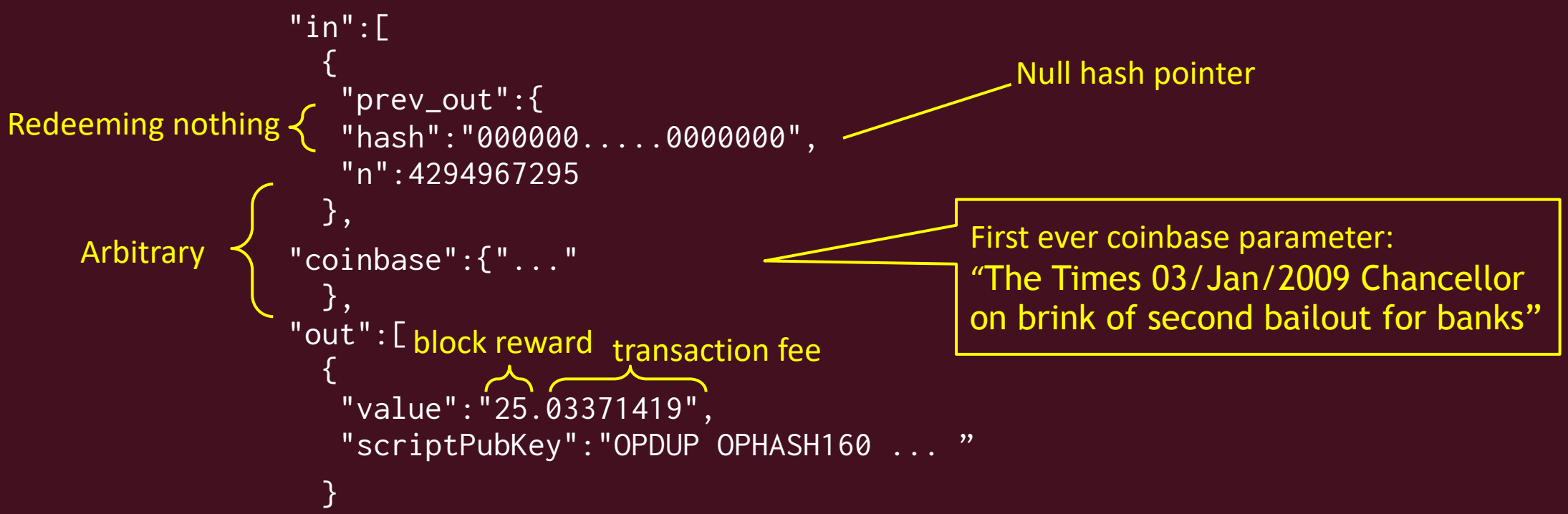
transaction data

```
{
  "hash":"00000000000000001aad2...",
  "ver":2,
  "prev_block":"00000000000000003043...",
  "time":1391279636,
  "bits":419558700,
  "nonce":459459841,
  "mrkl_root":"89776...",
  "n_tx":354,
  "size":181520,
  "tx":[
    ...
  ],
  "mrkl_tree":[
    "6bd5eb25...",
    ...
    "89776cdb..."
  ]
}
```

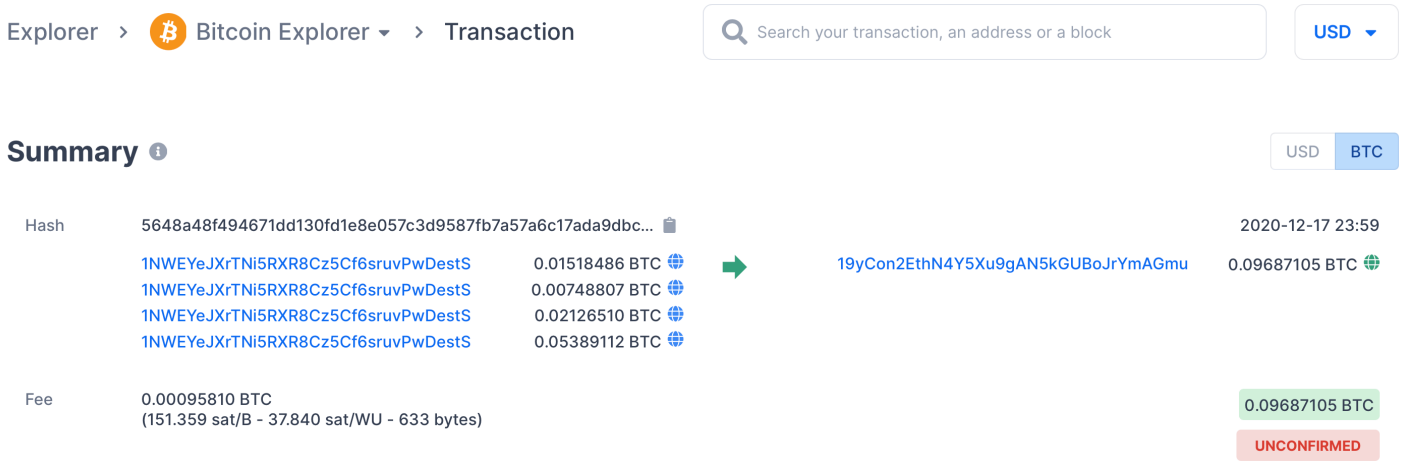
Coinbase Transaction

Bitcoin block has a special transaction called “coinbase” transaction

- To create new coin for mining/incentivizing
- Does not redeem previous output (null hash pointer)



Bitcoin Block



Details

Hash	5648a48f494671dd130fd1e8e057c3d9587fb7a57a6c17ada9dbc4c4a240df9a
Status	Unconfirmed
Received Time	2020-12-17 23:59
Size	633 bytes
Weight	2,532
Included in Block	Mempool
Confirmations	0
Total Input	0.09782915 BTC
Total Output	0.09687105 BTC
Fees	0.00095810 BTC

Explore yourself!

Blockchain.com

Google…

Consensus Mechanism

Bitcoin Transaction Validation

Once new transactions are broadcast in the network

Some nodes collect them to form a block
A random node is selected to propose the next block to the chain

How to select the node in network?

Other nodes accept the block only if all transactions in it are valid

Unspent / no double spent

Valid signatures

Nodes express their acceptance of the block by including its hash in the next block they create

Bitcoin Block Validation

Once the block is formed,

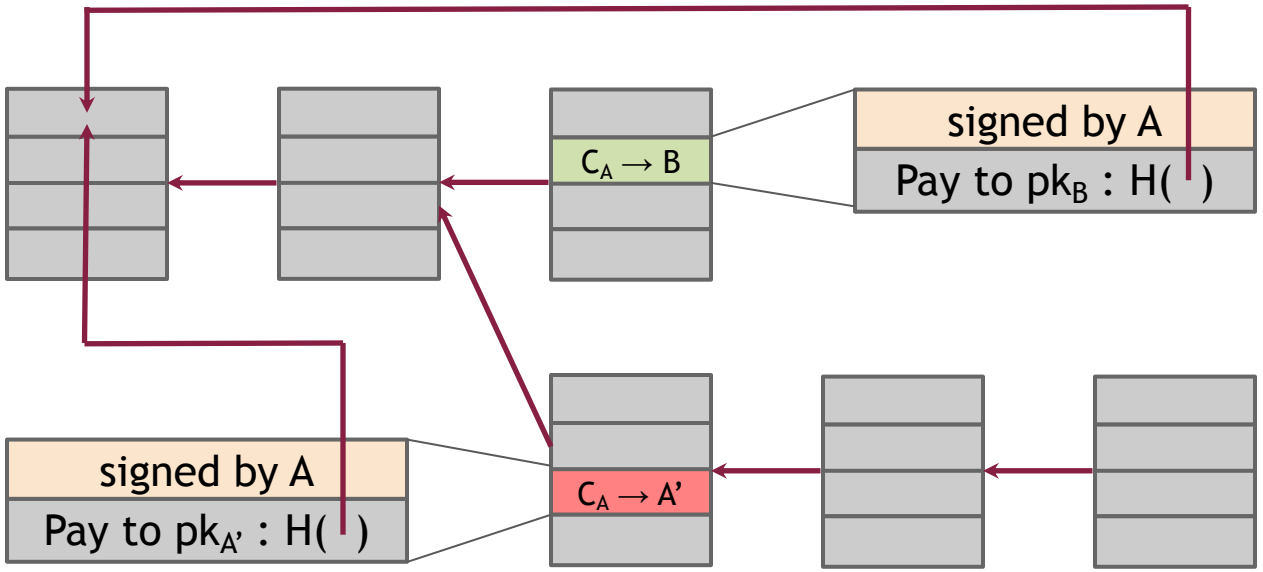
- It must be verified before included in the chain
- A block contains multiple transactions
- All transactions in the block must be verified

Malicious Node

What if a malicious node is selected to propose new block

Adversary may try to

- Steal bitcoin from other nodes?
  - Need to forge signature (break cryptographic primitives)
- Execute Denial-of-Service (DoS) attacks?
  - Ignore transactions from specific address
- Attempt double-spending attacks?



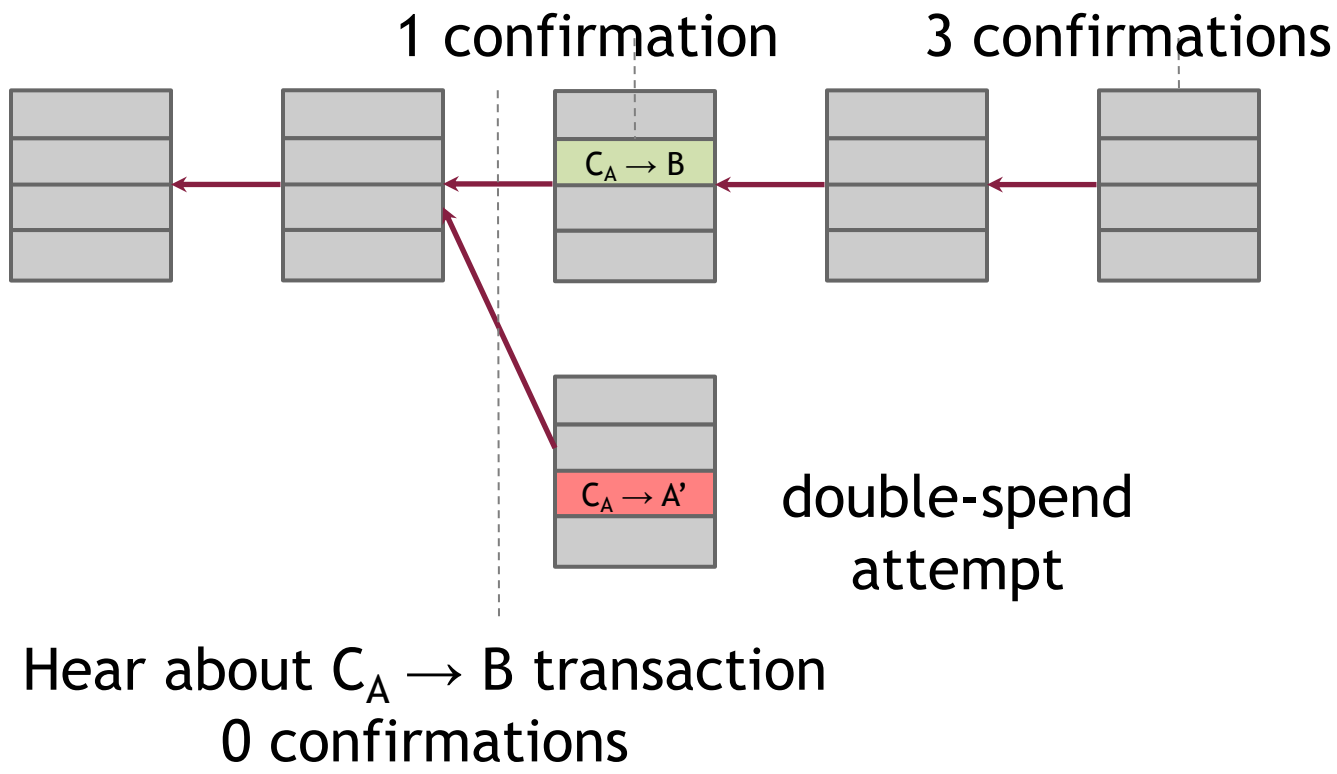
Double-spending attack

Blockchain Consensus

Prevent double-spending attack

How many confirmation? 0? 1? 2?

The more confirmations transaction gets,
the higher probability end up in longest term chain



Double-spend probability
decreases exponentially with
# of confirmations

Most common heuristic:
6 confirmations

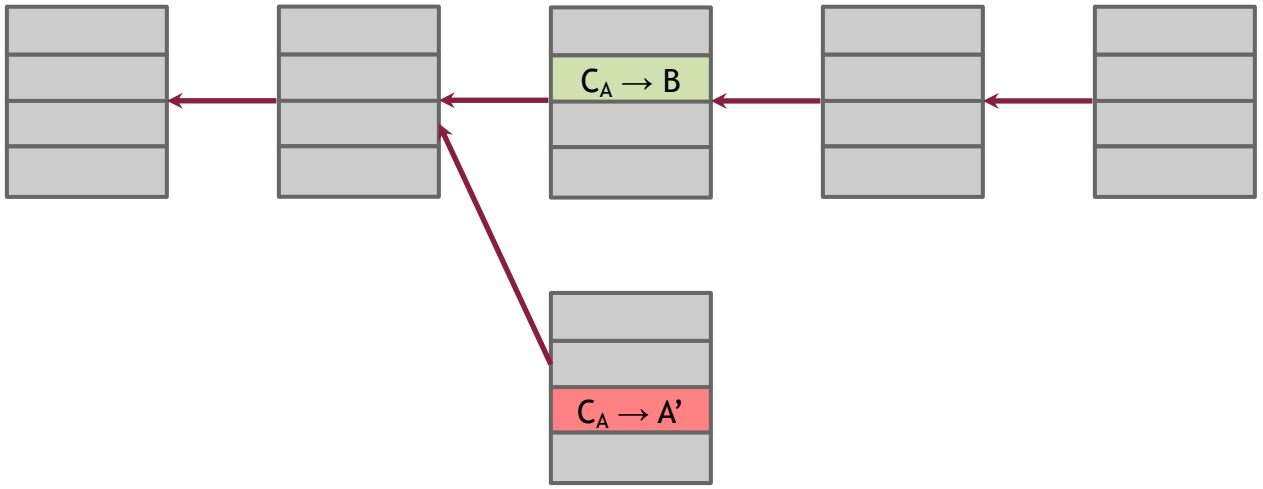
Blockchain Consensus

Protection against invalid transactions (malicious nodes) is cryptographic, but enforced by consensus

Majority is honest

Protection against double-spending is purely by consensus

Never 100% sure a transaction is in consensus branch. Guarantee is only probabilistic

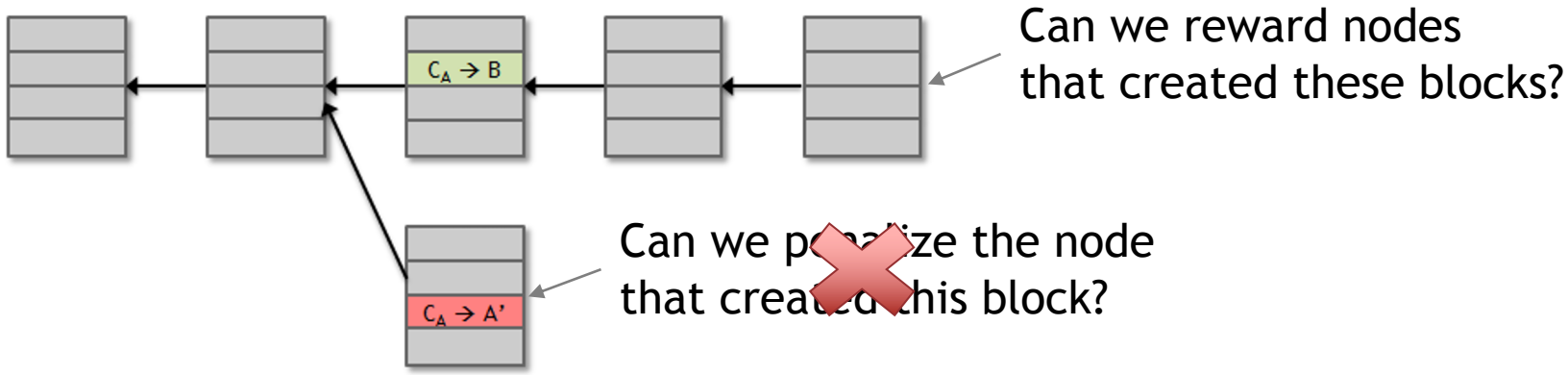


Incentive Mechanism

Assumption of honesty is problematic

What is the benefit of behaving honestly?

Solution: Give incentives to nodes for behaving honestly



Currency has real value to harden robustness of (traditional) consensus protocol

Incentive Mechanism

Incentive method 1: Block reward

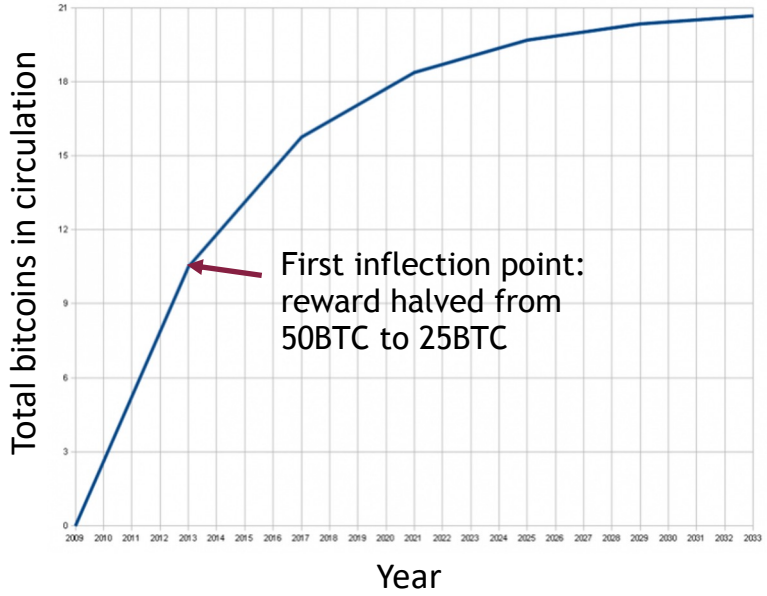
Node that creates new block gets to

- Include special coin-creation transaction in the block
- Choose recipient address of this transaction
- Collect the reward if its block ends up on long-term consensus branch

Value is fixed: currently 12.5 BTC, halves every 4 years

- Finite source of coin supply

Runs out in 2040. No new coins unless rules change



Incentive Mechanism

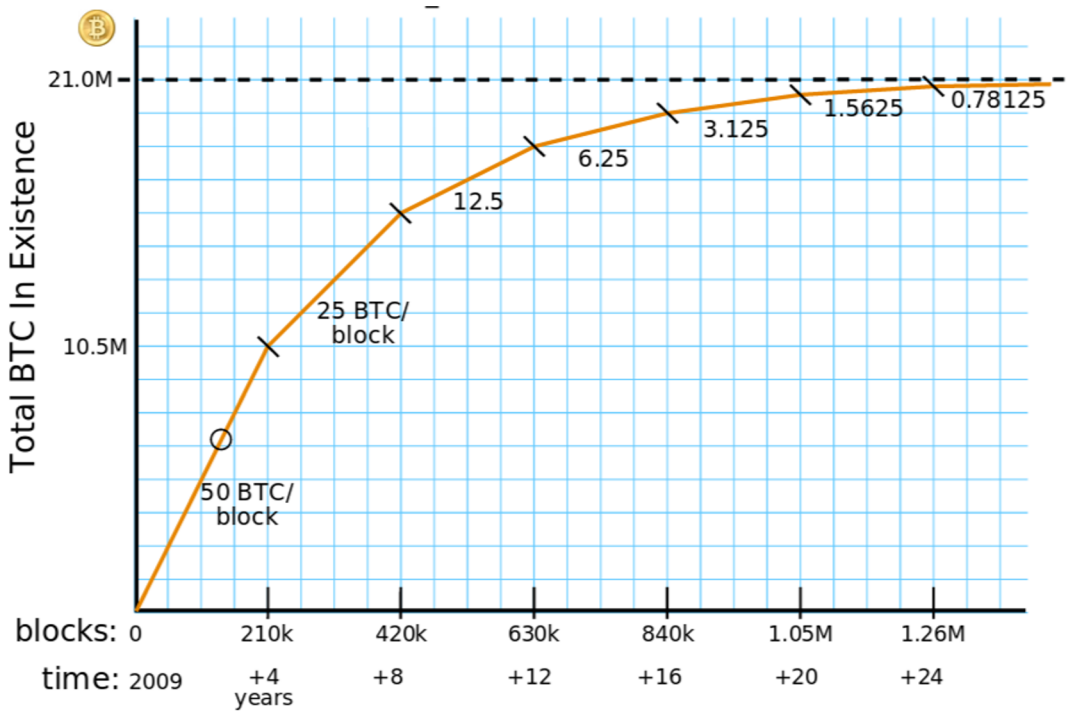
Incentive #2: Transaction fees

Creator of transaction can choose to make output value less than input value

Remainder is a transaction fee and goes to block creator

Purely voluntary (like a tip)

Will be more important when new coins run out in 2040



Incentive Mechanism

Incentive given to encourage honest behavior

Yet, there are still remaining problems

- How to select a random node?
- How to avoid a free-for-all due to rewards?
  - Everyone wants to capture reward
- How to prevent Sybil attacks?
  - What if adversary control 51% # of verifying nodes?

Bitcoin Components

Mining

Mining Mechanisms

To approximate selecting a random node:

Select nodes in proportion to a resource that no one can monopolize hopefully)

Proof-of-Work: In proportion to computing power

- Nodes compete with each other to propose the new block
- Make it hard to create new identities (prevent Sybil attacks)

Proof-of-Stake: in proportion to ownership

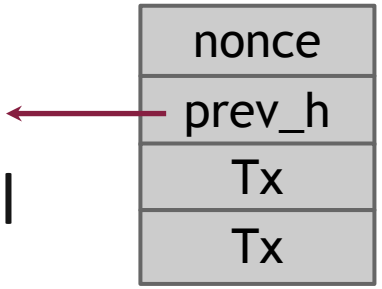
- Not used in bitcoin
- Will be discussed later

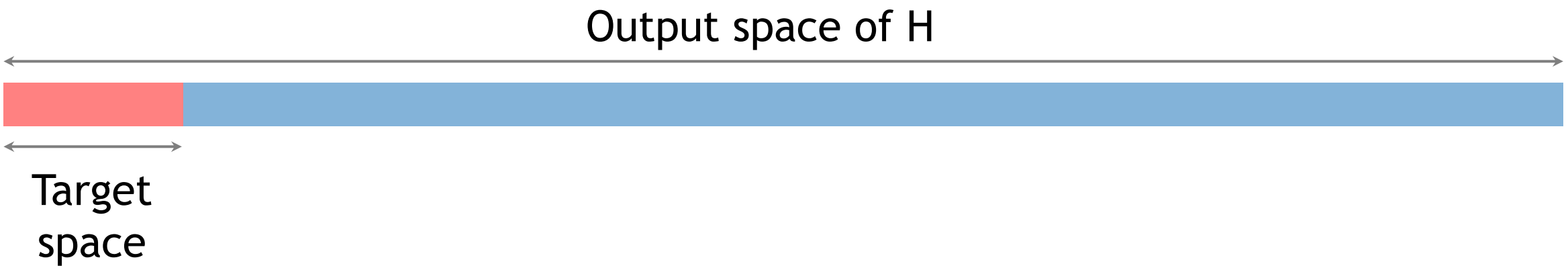
Proof of Work

Goal: Solve a **hash puzzle**

To create a new block, find a **nonce** such that

H(**nonce** || prev_hash || tx || ... || tx) is very small





If H is secure:
only way to succeed is to try enough nonces until we get lucky

Mining Mechanism: Proof-of-Work

Three desirable properties of PoW:

- Difficult to compute
  - ~10^{20} hashes/block as of Aug 2014:
  - Only some nodes bother to compete — miners
- Parameterizable cost (Why?)
  - Automatically re-calculate the target every two weeks
  - Goal: keep average time between blocks ~ 10 minutes.
    - Prob (Alice wins next block) = fraction of global hash power she controls
- Trivial to verify
  - Fully decentralized: no need CA to do this job!
  - Nonce published as part of block so other miners verify that

$$H(\text{nonce} \,\|\, \text{prev\_hash} \,\|\, \text{tx} \,\|\, \ldots \,\|\, \text{tx}) < \text{target}$$

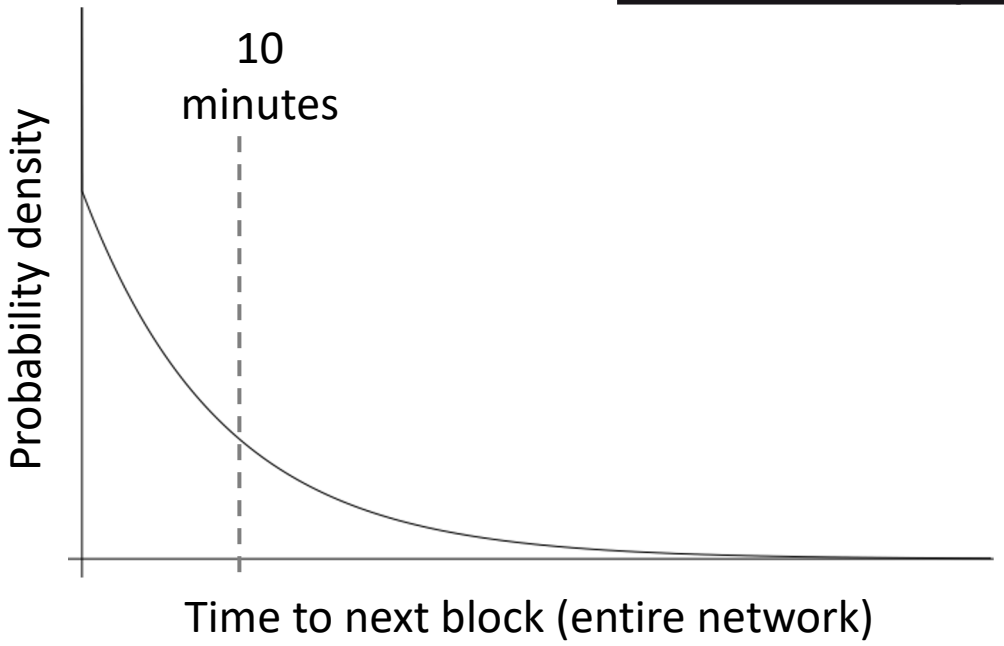
Mining Mechanism: Proof-of-Work

Attacks infeasible if honest majority

>50% miners weighted by hash power follow the protocol

Solving hash puzzles is probabilistic

Bernoulli trials: Probability density function of the time to find the next block by any node in the network is reduced exponentially



For individual miner:

$$\text{mean time to find block} = \frac{10 \text{ minutes}}{\text{fraction of hash power}}$$

Mining and Incentive Mechanism

Mining and incentive mechanisms significantly limits the impact of malicious nodes in Bitcoin network

In summary, what can a “51 percent” attacker do?

- Steal coins from existing address? ✗
- Suppress some transactions?
  - From the block chain ✓
  - From the P2P network ✗
- Change the block reward? ✗
- Destroy confidence in Bitcoin? ✓✓

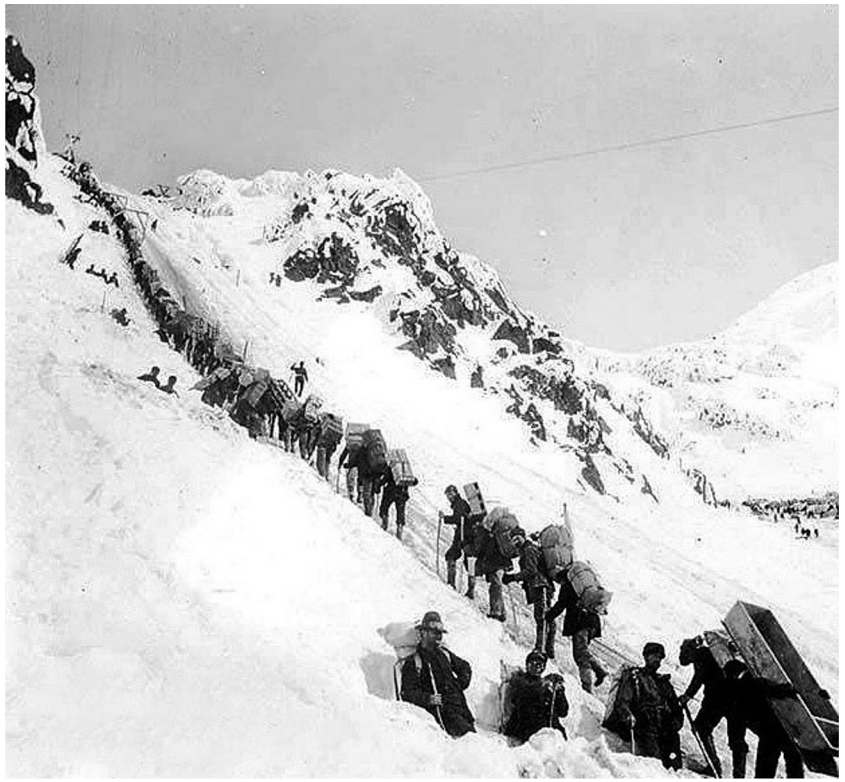
Bitcoin Mining

Bitcoin needs miners to operate

What do miners do?

- Store and broadcast blockchain
- Listen and verify transactions
- Form blocks and add to the chain
- Vote on consensus

Super Pit gold mine in Western Australia (Wikipedia)

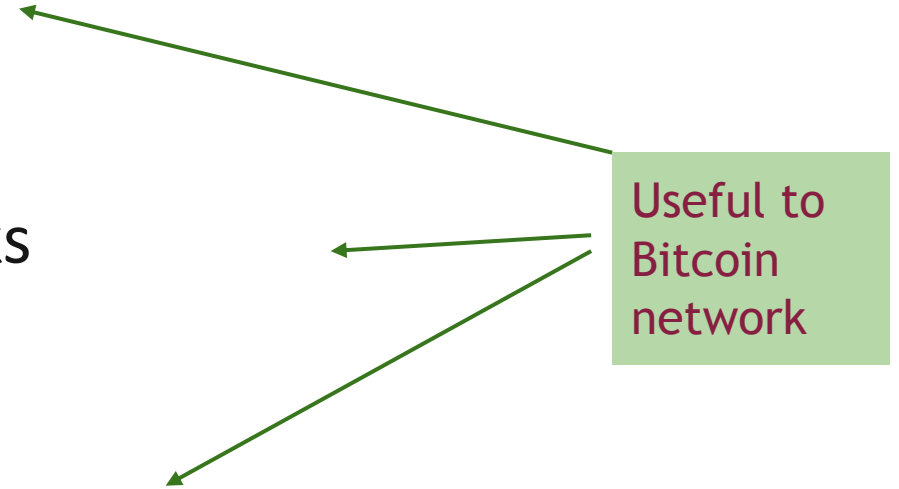


Gold miners ascending the Chilkoot pass
Klondike gold rush of 1898

Bitcoin Mining

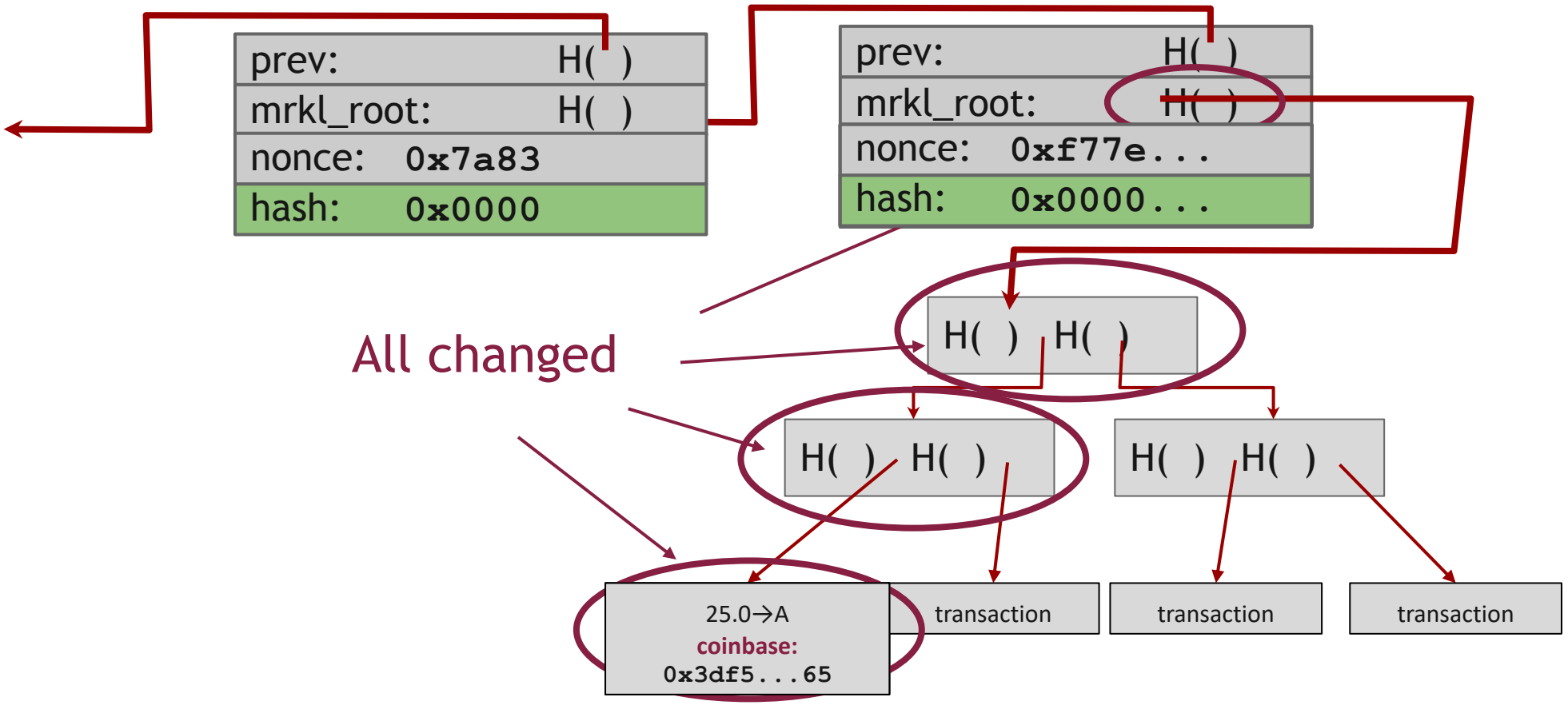
Steps to become a miner:

- Join the network, listen for new transactions
  - Validate all proposed transactions
- Listen for new blocks, maintain chain of blocks
  - When a new block is proposed, validate it
- Assemble a new valid block
- Find the nonce to make the block valid
  - Called “mining”
- Hope everybody accepts the block
- Make profit!



Bitcoin Mining

Can two miners solve the same puzzle?



Mining Difficulty Target

256 bit hash output

0000000000000003AAEA200000000000000000000000000000000000000000

64+ leading zeroes required

Current difficulty = $2^{66.2}$

=84,758,978,290,086,040,000

Mining Difficulty

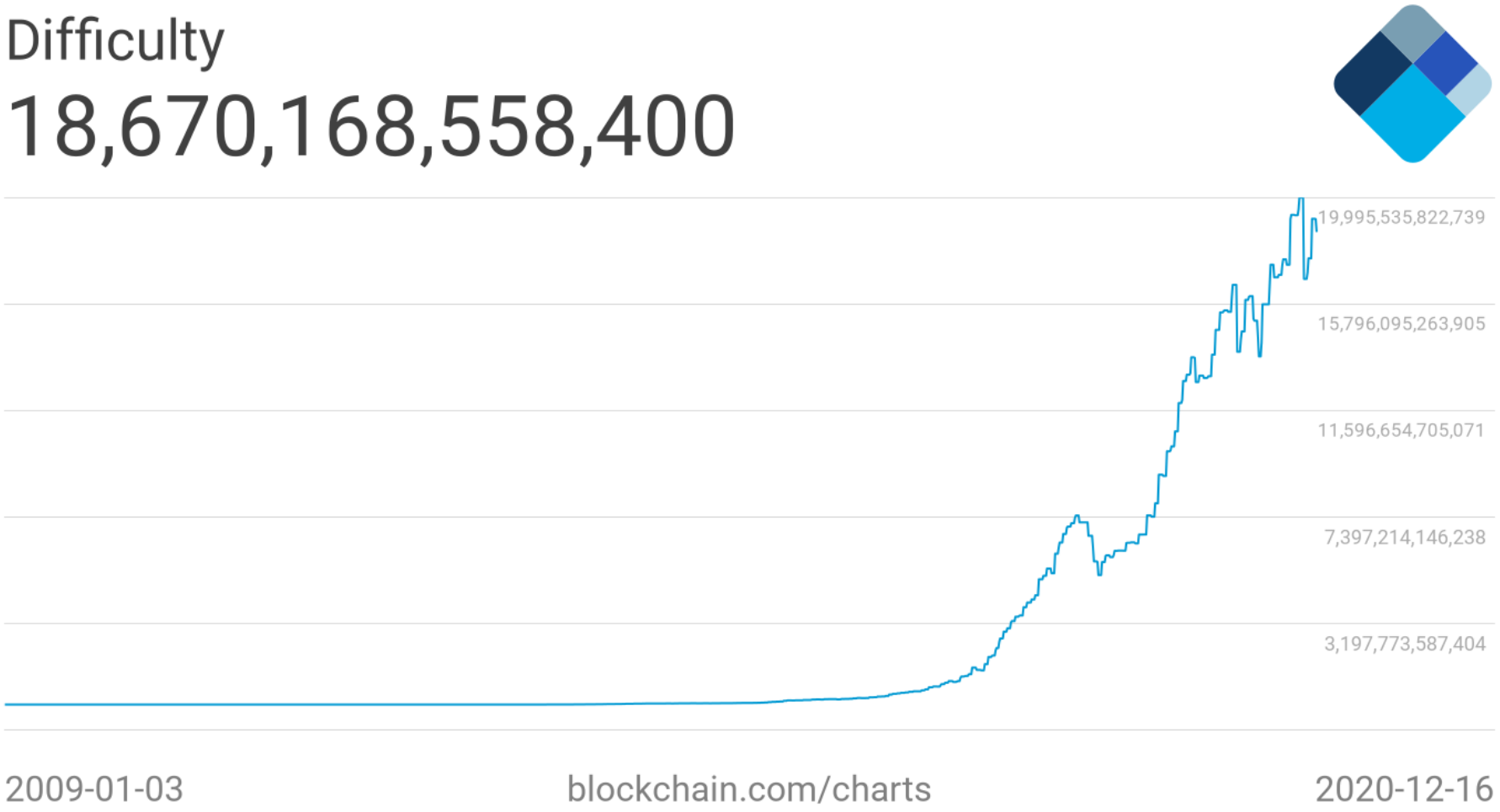
(Recap) PoW allows mining difficulty to be adjusted dynamically based on how long 2016 blocks are found

Every two weeks, compute

```
next_difficulty = previous_difficulty *
    (2 weeks)/(time to mine last 2016 blocks)
```

Expected number of blocks in 2 weeks at 10 minutes/block

Mining Difficulty Over Time



SHA-256 Hash

General purpose crypto hash function

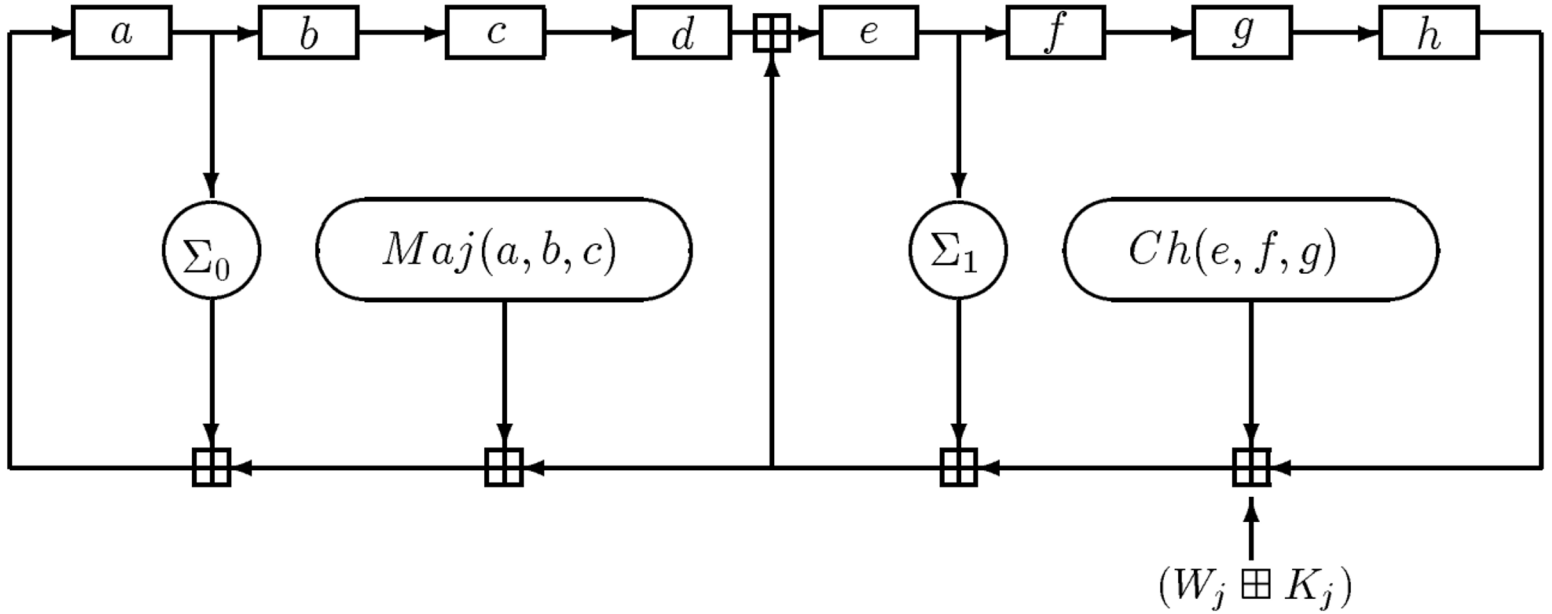
Part of SHA-2 family: SHA-224,SHA-384,SHA-512

Remains unbroken cryptographically

Weaknesses found though!

SHA-3 (replacement) under standardization

256-bit state



x80 iterations

round constants

Figure 1: j^{th} internal step of the SHA-256 compression function C

Mining Economics

If mining reward (block reward + Tx fees)	>	hardware + electricity cost	→	Profit

Complications

- Fixed vs. variable costs
- Reward depends on global hash rate

Mining Hardware

CPU

```
while (1){
    HDR[kNoncePos]++;
    IF (SHA256(SHA256(HDR)) < (65535 << 208)/ DIFFICULTY)
        return;
}
```

two hashes

High-end PC throughput ≈ 2^{24}

139,461 years

GPU

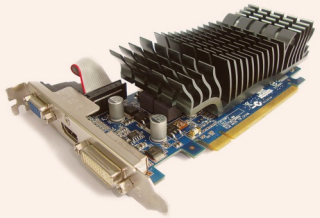
Parallel ALUs, overclockable

Poor cooling, high power consumption

High-end throughput ≈ 2^{27}

173 years w/100 cards

ATI better than NVIDIA at mining. Why?

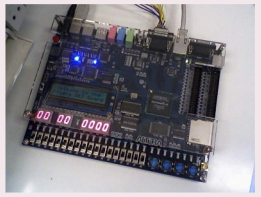


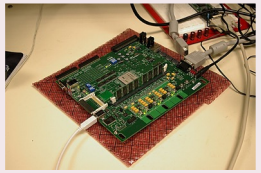
Source: LeonardH, cryptocurrenciestalk.com

Mining Hardware

FPGA (Field Programmable Gate Area)

- High customization, optimization
- Better cooling
- Expensive, high power consumption
- High-end throughput ≈ 2^{30}
  - 25 years w/ 100 boards





Bitcoin ASICs

- Special purpose
- Longevity
- TerraMiner IV ($6,000)
  - 14 months to find a block

Mining Future

Only ASIC and professional mining profitable to Bitcoin

- Expensive
- Somewhat violate original vision of Bitcoin?

Can smaller miners stay in the game?

Would we be better off without ASICs?

Bitcoin Limitations

Many hardcoded constraints with tight economic implication

- Block size
- # signatures per block
- Currency divisibility
- # coins
- Block rewards
- Fixed algorithms
- …

Scalability

Protocol Upgrade

Bitcoin Limitations

Very low throughput: 7 transactions per sec

- 10 mins (on average) to create a block
- Block size: 1 MB, TX size: 250 bytes => 4000 TXs per block
- VISA: 2K – 10K TX/ sec, Paypal 50 - 100 TX /sec

20,000 signature operations per block

21M total bitcoins maximum

- 50, 25, 12.5 …bitcoin mining reward
- 100M satoshis per coin

} Impact economic balance of power
Too much to change now

Bitcoin Limitations

Cryptographic limits

- Hard-coded crypto primitives (ECDSA/P256, SHA1)
- Crypto primitives may be broken by 2040

Hard to upgrade (outdated) protocols

- Impossible to ensure all nodes upgrade

Hard fork

Soft fork

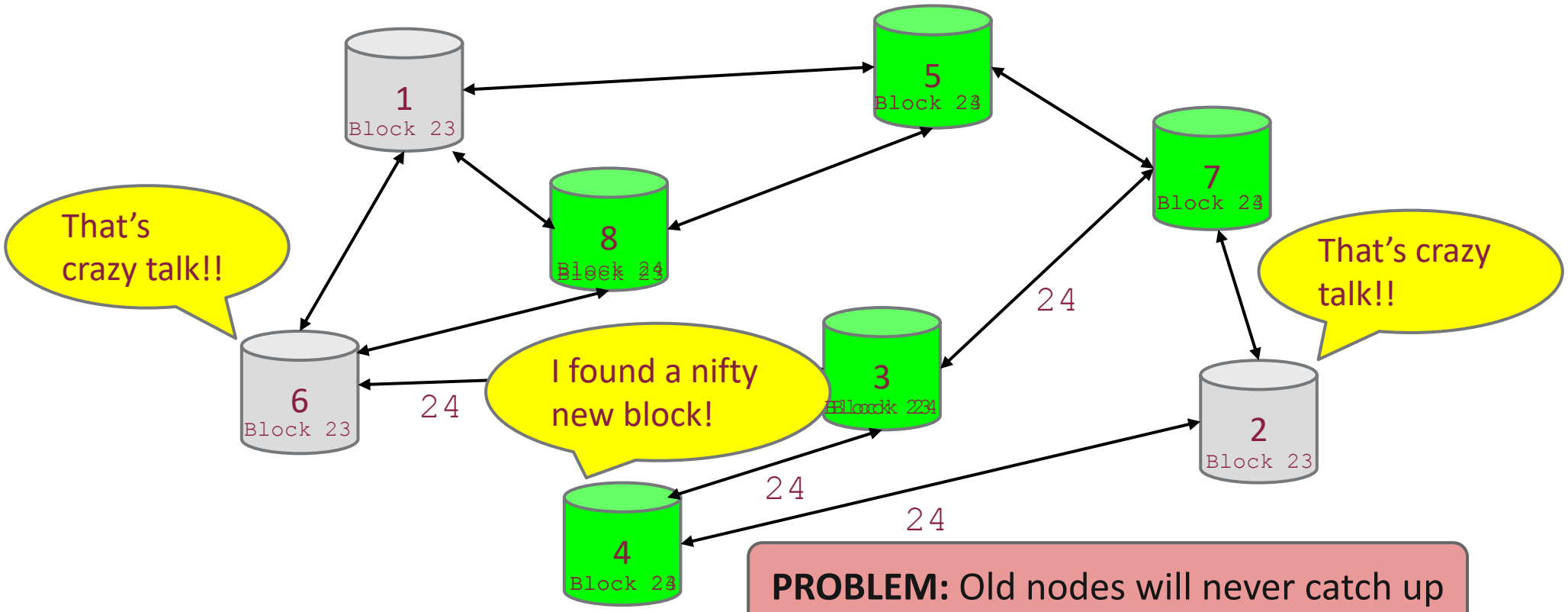
“Hard-forking” Changes to Bitcoin

Validate blocks that were previously considered invalidate

Two branches never join again

Permanent split may occur

Old nodes will have to upgrade to stay on the same chain eventually



“Soft-forking” Changes to Bitcoin

Enforce stricter validation rules

New rules reject blocks that were previously accepted by old rule

Require majority of nodes to enforce new rules

Old nodes will approve

```
<signature>
<<pubkey> OP_CHECKSIG>
```

```
OP_HASH160
<hash of redemption script>
OP_EQUAL
```

Old nodes will just approve the hash, not run the embedded script

RISK: Old nodes might mine now-invalid blocks

Using Bitcoin

Bitcoin is available on many platforms now

Explore more yourself! ☺