CSE 127 Computer Security

Alex Dent, Spring 2018

Lecture 15: Crypto-currency (just one – Bitcoin)

Lecture Objectives

- View currency as a the result and ability to perform transactions.
- Understand that ownership of currency can be tracked if all transactions are known (ledger principle).
- Understand how a ledger can be maintained by a group of mutually-distrusting peers in a dense network.
- Explain the different attacks that might be attempted against a peer-to-peer ledger and why they do not work in the long term.

What is money?

- Object that can be exchanged for goods or services.
  - Money is the power to execute a transaction
  - It is the ability to exchange money for other things that makes it useful.
- Traditionally linked to obtaining a commodity.
- Now almost every country uses “fiat money”.

The properties of money

Fiat Money
Fungible (interchangeable)
Hard to double-spend
Anonymous
Independent transactions
Vulnerable to inflation
Government-backed value

The properties of money

Fiat Money	Institutional Banking
Fungible (interchangeable)	Fungible (interchangeable)
Hard to double-spend	Mostly hard to double-spend
Anonymous	Not anonymous
Independent transactions	Bank-backed transactions
Vulnerable to inflation	Vulnerable to inflation and bank going broke
Government-backed value	Government-backed/bank-backed value

The properties of money

- Aims of e-cash:
  - Anonymous transactions
  - Not vulnerable to bank business-model (going broke)

- Challenges of e-cash:
  - How to establish value?
  - *Coins introduced slowly and represent cost of resources used to generate*

  - How to prevent double-spend?
  - *Distributed, public ledger with reconciliation process*

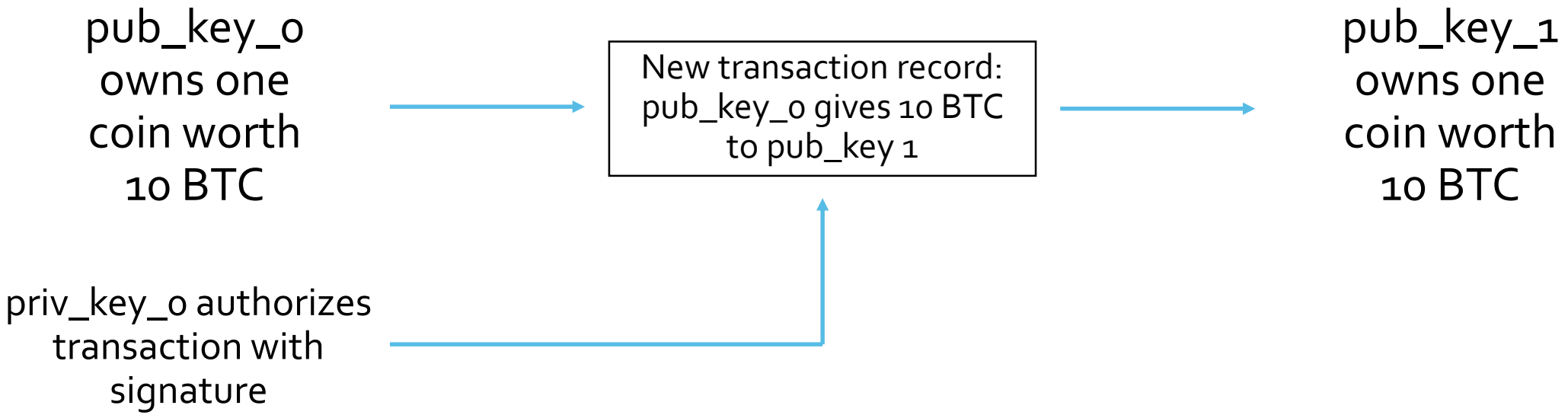
D. Chaum, Blind signatures for untraceable payments, 1983.
S. Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash Systems, 2008.

Transactions

- Money is the power to execute a transaction…
- … therefore the "value" of an electronic coin is the thing that allows you to transfer that coin to someone else.

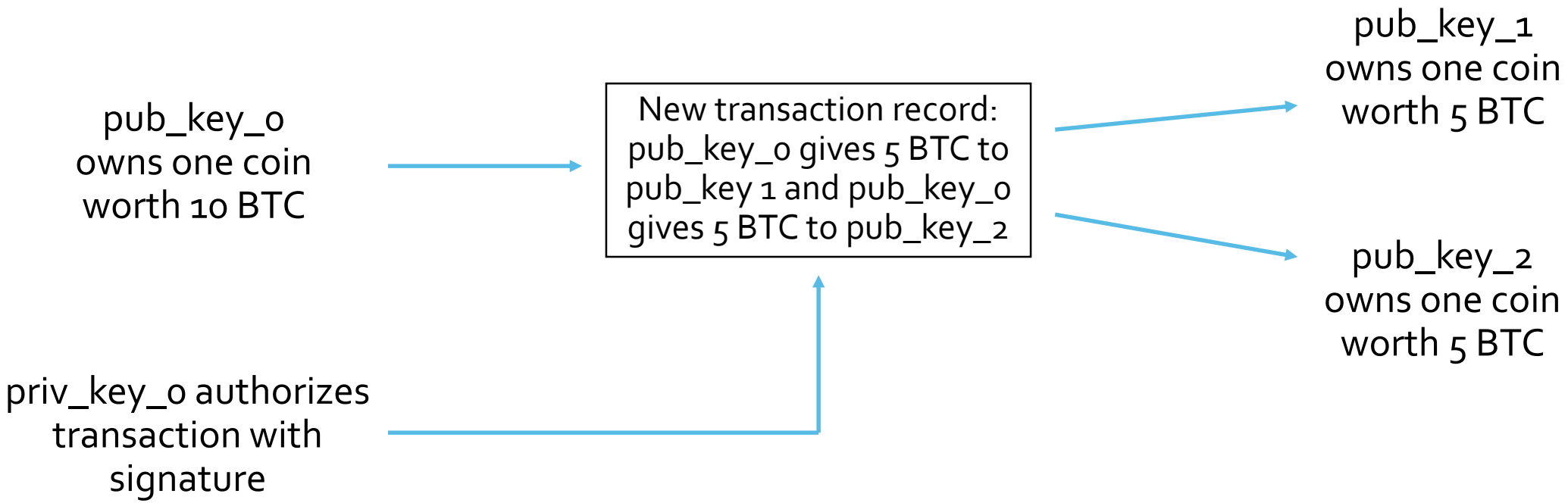
- Coin is going to be "owned" by public/private key pairs for a digital signature algorithm. A coin can be worth any amount of money.
- Public keys identify "ownership" of a coin.
- Private keys allow you to sign transactions which transfer ownership to a different public key.

Transactions



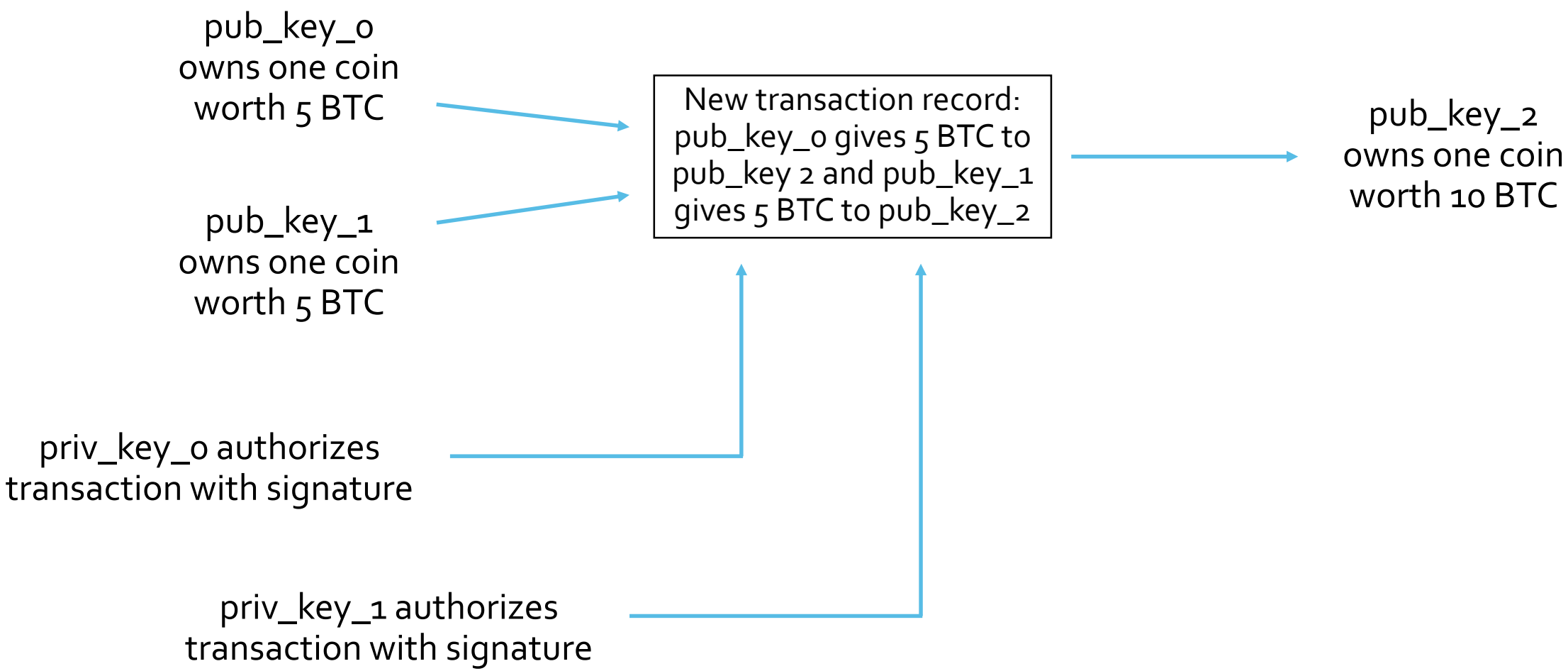
- Note that isn't like a bank account where unspent bitcoins are retained by pub_key_0. Transfer is whole amount of money.

Transactions



- Possible to spend part of a coin's value with a transaction that transfers some value back to a public key you own.

Transactions



Transactions

- Note the anonymity properties: public keys have no meaning.

- Nice in theory, but how do you demonstrate ownership of a coin?
- Refer back to the transaction record that gave you the coin.
- How do you demonstrate that transaction is valid?
- Need a list of all transactions back to the first one: a ledger.

Using a (perfect) ledger

Ledger
Transaction 1: pub_key_0 gives 0.5 BTC to …
Transaction 2: pub_key_1 gives 0.5 BTC to …
Transaction 3: pub_key_3 gives 0.25 BTC to …
Transaction 4: pub_key_4 gives 2 BTC to …
Transaction 5: pub_key_7 gives 0.25 BTC to …
Transaction 6: pub_key_2 gives 0.1 BTC to …
Transaction 7: pub_key_9 gives 0.1 BTC to …
Transaction 8: pub_key_11 gives 3 BTC to …
Transaction 9: pub_key_15 gives 0.25 BTC to …
Transaction 10: pub_key_12 gives 1.5 BTC to …
Transaction 11: pub_key_8 gives 1 BTC to …
Transaction 12: pub_key_19 gives 2.5 BTC to …
Transaction 13: pub_key_14 gives 0.25 BTC to …
Transaction 14: pub_key_18 gives 0.1 BTC to …
Transaction 15: pub_key_21 gives 1 BTC to …
Transaction 16: pub_key_20 gives 0.5 BTC to …
Transaction 17: pub_key_5 gives 0.75 BTC to …
…

- Ledger is a list of all of the transactions.
- Transaction isn't confirmed until it has been accepted by the ledger.
- A coin (public key) can show its worth by pointing to the transaction that gave it value.
- Ledger won't accept the transaction if the coin has already been spent.

Using a (perfect) ledger

Ledger
Transaction 1: pub_key_0 gives 0.5 BTC to …
Transaction 2: pub_key_1 gives 0.5 BTC to …
Transaction 3: pub_key_3 gives 0.25 BTC to …
Transaction 4: pub_key_4 gives 2 BTC to …
Transaction 5: pub_key_7 gives 0.25 BTC to …
Transaction 6: pub_key_2 gives 0.1 BTC to …
Transaction 7: pub_key_9 gives 0.1 BTC to …
Transaction 8: pub_key_11 gives 3 BTC to …
Transaction 9: pub_key_15 gives 0.25 BTC to …
Transaction 10: pub_key_12 gives 1.5 BTC to …
Transaction 11: pub_key_8 gives 1 BTC to …
Transaction 12: pub_key_19 gives 2.5 BTC to …
Transaction 13: pub_key_14 gives 0.25 BTC to …
Transaction 14: pub_key_18 gives 0.1 BTC to …
Transaction 15: pub_key_21 gives 1 BTC to …
Transaction 16: pub_key_20 gives 0.5 BTC to …
Transaction 17: pub_key_5 gives 0.75 BTC to …
…

- ✓ Anonymous
- ✓ Prevents double-spending (except by ledger)
- ✗ One central authority
- ✗ Impractical
- ✗ No details on how to create coins

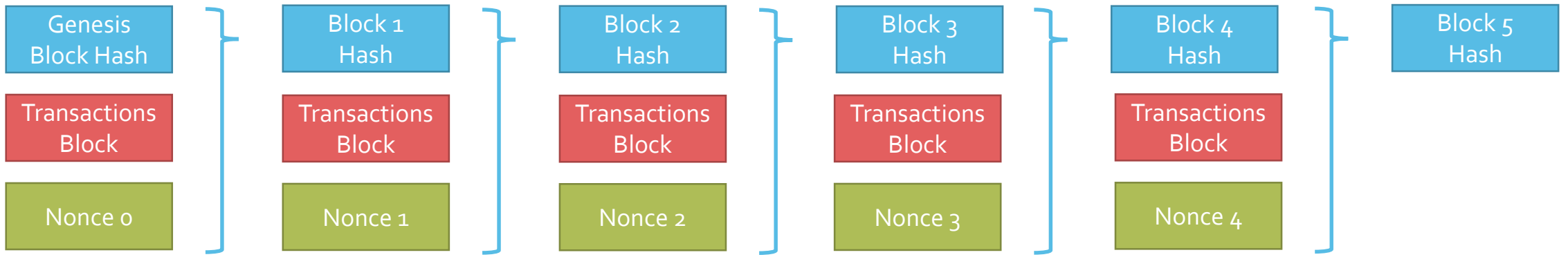
Using a peer-to-peer ledger

- Copies of the ledger stored with different servers (called nodes).
- New transactions are broadcast to all nodes:
  - Nodes check the authenticity of transaction compared to their ledger.
  - If valid, then add transaction to the ledger.
  - Honest nodes will have correct ledgers.
  - Dishonest nodes will have incorrect ledgers.
- How can a user tell which node is honest?
  - Dishonest nodes will accuse honest nodes of dishonesty.
  - Dishonest nodes will support other dishonest nodes.
  - Dishonest nodes may have behaved honestly up until the current transaction.
  - Everything becomes a childish squabble.

Using a peer-to-peer ledger

- Solution: Make cheating expensive!
- Adding an entry to the ledger costs resources (CPU time).
  - "Proof of Work"
  - Amount of work is the same for adding good and bad transaction to ledger.
- Once a valid solution is found, then the new ledger can be distributed and checked by all other nodes (updating their ledgers).
- If more then half of the CPUs are honest, then real ledger will grow faster than any fake ledger as they have more resources.
- The real ledger is the longest ledger.

Using a peer-to-peer ledger



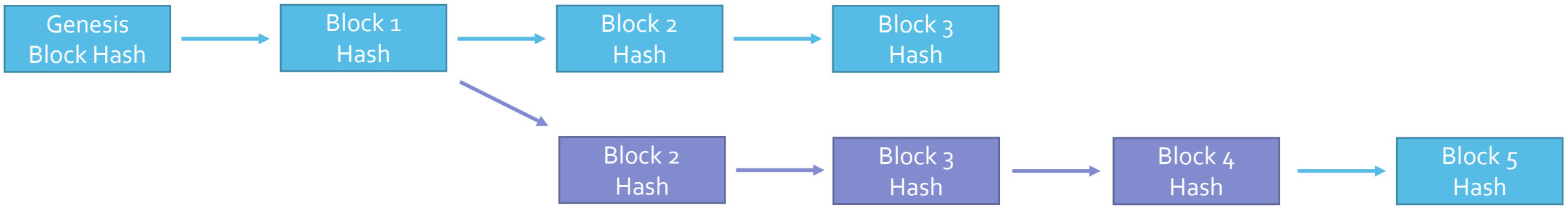
- Transactions are group into blocks.
- Blocks are placed into a block (hash) chain.
- Blocks only accepted into chain if hash has correct form -- usually starting with some number zeroes. This is the proof of work.
- Transaction is valid if it exists on the block chain.

Using a peer-to-peer ledger

1. New transactions are broadcast to all nodes.
2. Each node collects new transactions into a block.
3. Each node works on finding proof-of-work for its block.
4. When node finds proof-of-work, it broadcasts block and nonce.
5. Nodes accept block if all transactions are valid, not spent and new block hash has the correct form.
6. Nodes express their acceptance by working on creating the next block in the chain, using the new block hash.

Using peer-to-peer ledger

- What if two nodes compute proof-of-work for different blocks simultaneously?
- The longest chain is always the correct one.
- Ties go to the first one each node received.



Worked example: User 0 wants to send User 1 a coin worth 5 BTC.

User 0:
{pub_key_0,
transaction_id_17}

Signs script: {pub_key_0, transaction_id_17} -> pub_key_1

Broadcasts signed script to all nodes

Checks transaction is valid according to longest chain in ledger

Collates transactions into block and starts computing the hash value (proof of work)

Receives new block from other node.

Checks that new block is valid

Checks to see if current transaction is in block that it just received.

Collates unsigned transactions into new block and starts computing the hash value

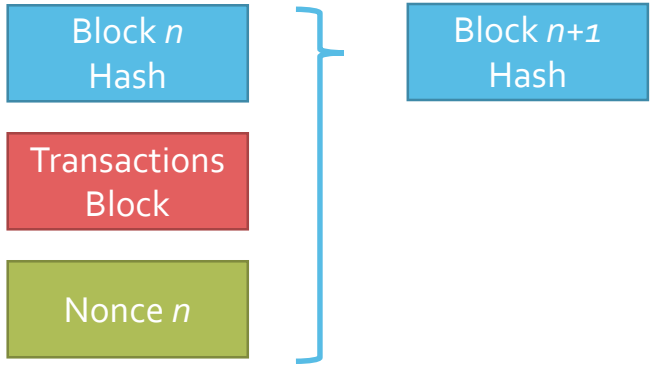
Broadcasts new block to other nodes and marks old transaction as "spent"

Returns new coin:
{pub_key_1,
transaction_id_86}

User 1:
{pub_key_1}

Checks transaction is in block chain in a few nodes

Using a peer-to-peer ledger



- ✓ Anonymous
- ✓ Prevents double-spending (except by majority)
- ✓ Distributed authority
- ✗ Impractical
- ✗ No details on how to create coins

What if… bad node broadcasts a false block with an invalid hash?

- Users and nodes will reject the block chain as invalid.

What if… a bad node broadcasts a false block with an invalid transaction?

- Users probably would accept the block chain as valid.
- Nodes will reject the block as invalid.
- The bad block will never be accepted into a good node's chain.
- If there is a majority of good nodes/CPU power, then eventually good nodes' chain will be longer than bad nodes' chain and block will be rejected.
- So transactions that might initially be accepted as valid will become invalid as the longest chain switches over to good nodes' chain.
- Moral: Wait to make sure transaction was committed.

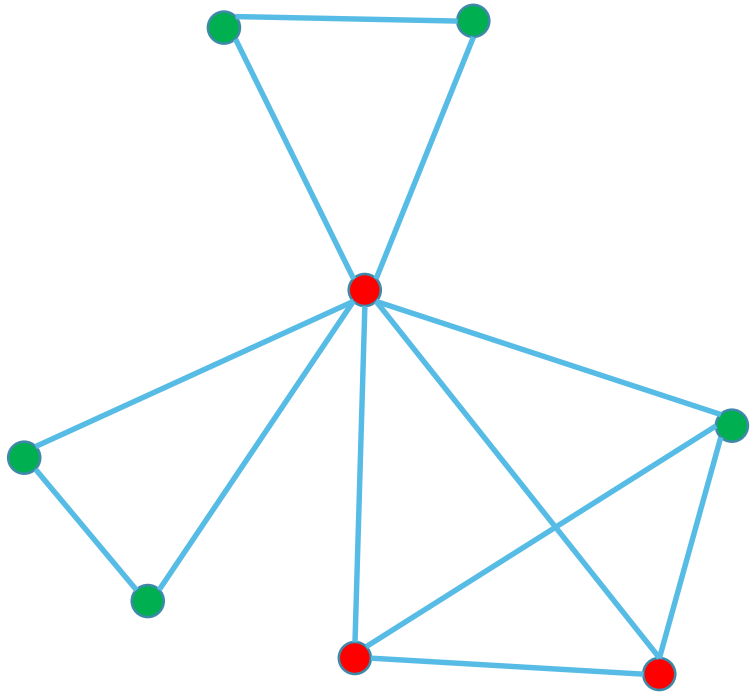
What if… no transactions are submitted?

- All nodes have to compute the next block anyway.
- Recall that the longest chain is the right chain.
- If there is a majority of good nodes/CPU power, then the good nodes chain will always be the longest …
- … but this requires that computation never stops.

What if… bad spender submits different transaction to many different nodes?

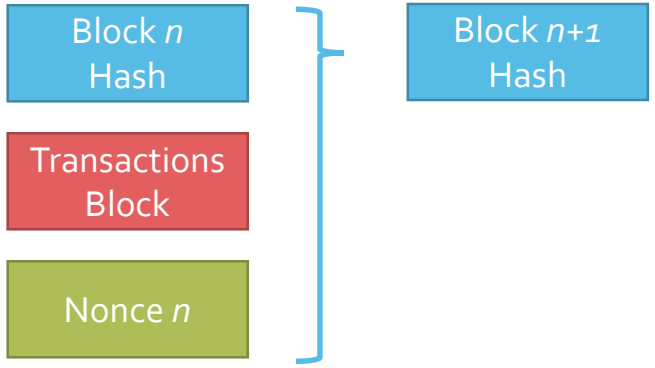
- This is an example of double-spend.
- Both nodes will accept the initial transaction and start to try to find proof-of-work for the block containing that transaction.
- One node's chain will eventually win and that transaction will be valid.
- Moral: Wait to make sure transaction was committed.

What if… we don’t really have a broadcast channel?



- Broadcast channel is never practical.
- Some network topologies can lead to security issues.
- If the central node refuses to propagate correctly formatted blocks, then longest chain will be computed by largest sub-clique, which is under attacker control.
- Solution is to require large number of connections to randomly chosen nodes.

What if…

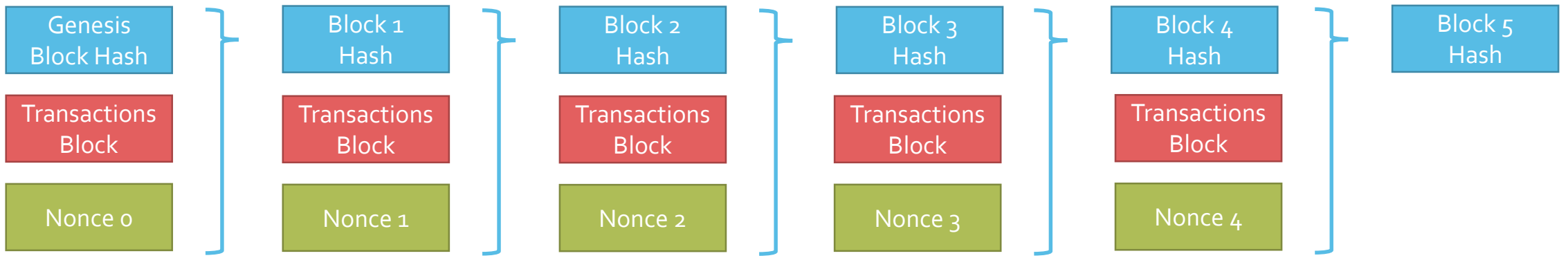


- ✓ Anonymous
- ✓ Prevents double-spending (except by majority)
- ✓ Distributed authority
- ✗ Impractical
- ✗ No details on how to create coins

Improving practicality

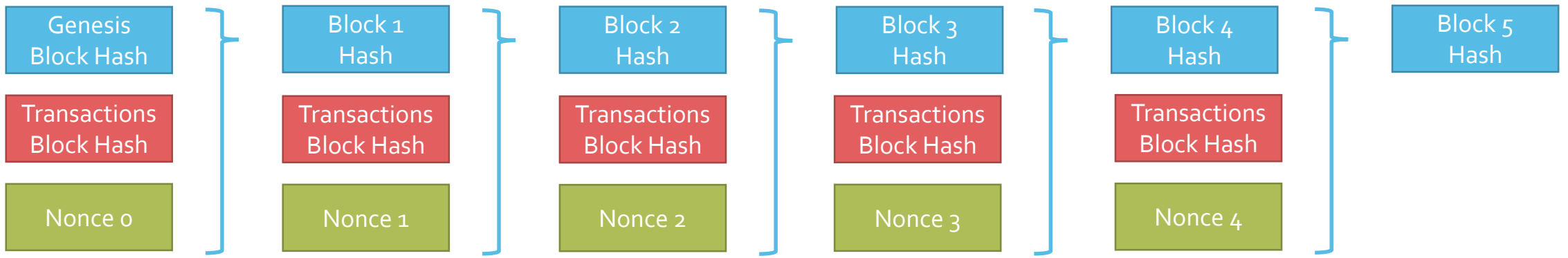
- Storage space requirement continually increasing as no transaction can ever be removed from block chain.
- Verifying a single transaction requires hash all transactions in the block chain (which will be a large number)
- Verifying a single transaction requires hashing all transactions in that block (which may be large number).
- Running a node costs money and has no reward.
- The number of coins in circulation is constant.

Improving practicality



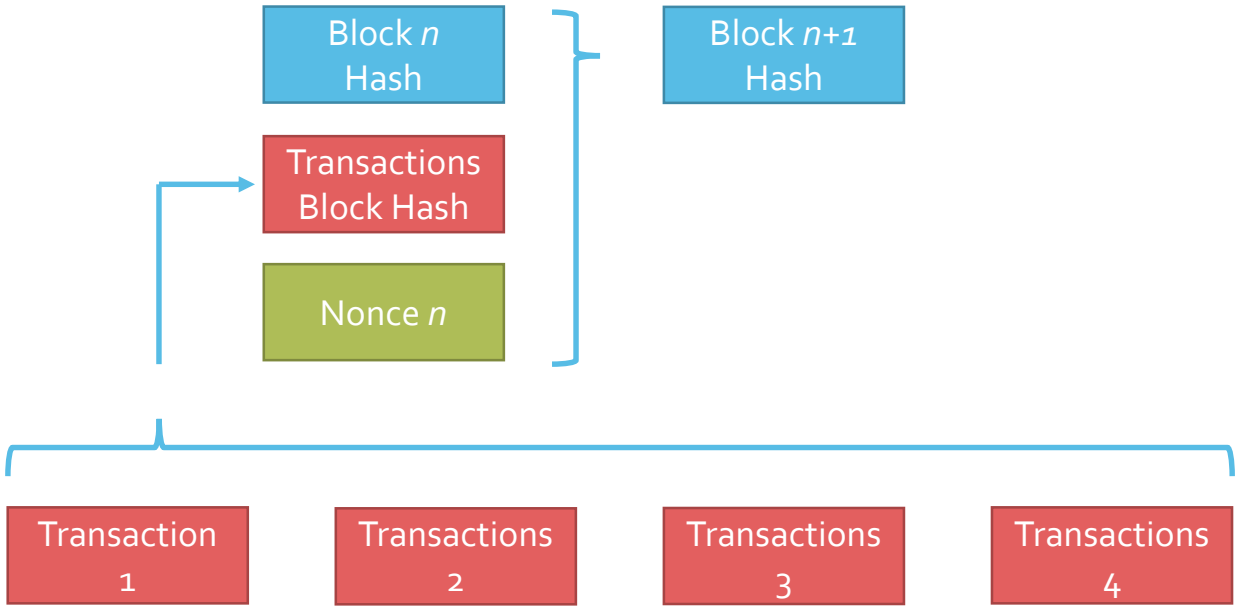
- Verifying the correctness of the block chain requires the user to download all the transactions.
- No transactions can ever be removed from the block chain even though recording old transactions is not required for scheme.

Improving practicality



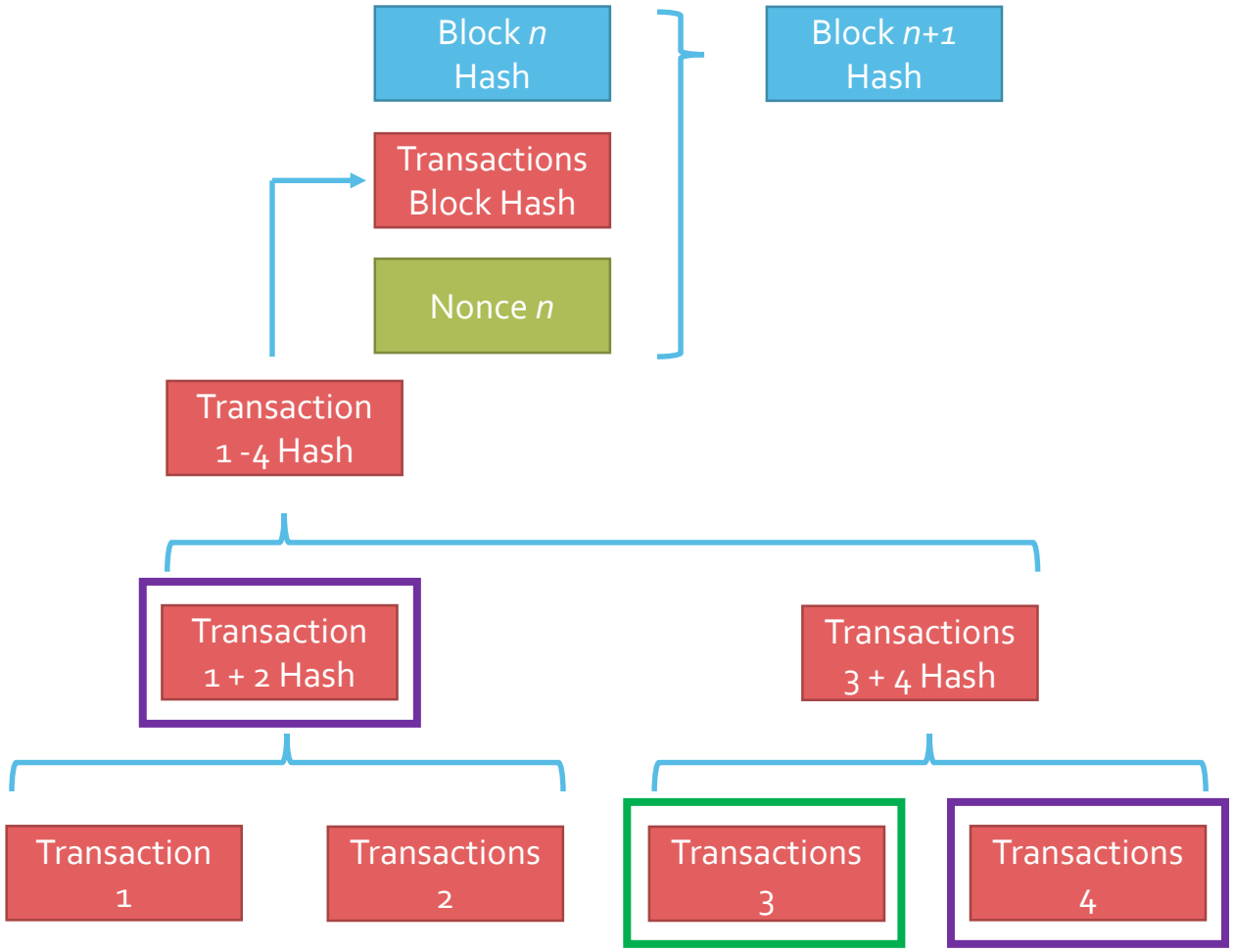
- Use hash of transaction block in block chain.
- User require all transactions block hashes to check block chain.
- Nodes can delete transaction block after all transactions in that block have been spent (just keep the transaction block hash).

Improving practicality



- Block transaction hash still requires user to process all transactions in a block.
- Block transaction hash still requires nodes to store all transactions in a block.
- Use a Merkle tree instead of "straight" hashing.

Improving practicality



- Merkle tree hashes inputs in pairs in a tree structure.
- Verifying one transaction requires you to download less data and hash fewer bytes.
- Node can also delete parts of the tree when all children represent spent transactions.

Improving practicality

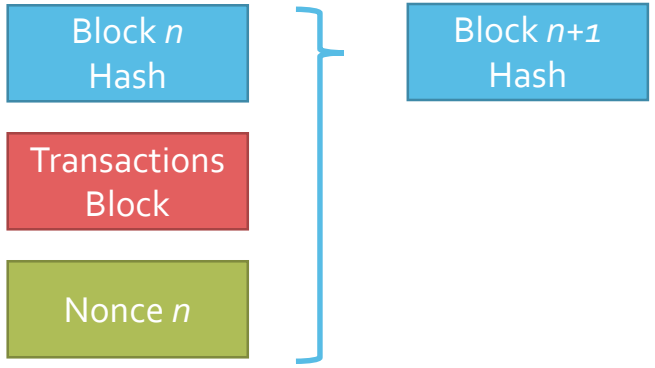
- Nodes are always-on, always-computing servers.
- Not reasonable to expect altruism in a $130T industry.
- Any money not accounted for in a transaction is transaction fee that is kept by the node that found proof-of-work in block chain.
  - Nodes can reject transactions that do not offer sufficiently high fees.
  - Block chain records now require "space" to give public key of node that produced block / found proof-of-work.
- Nodes can also use processing power to create new coins.
  - Not a corruption of block chain as bitcoin allows new coins to be created.
  - Provides an incentive to be honest even if transaction fees/volume is low.

Improving practicality

- Coins need to be initially created.
- It's a good idea to allow coins to be added to the pool slowly.
- No central authority that produces coins.

- Solution:
  - Node can create new coins whenever they create a new block.
  - Value of new coin reduces over time (number of coins created).
  - Genesis block only created coins.

Improving practicality



- ✓ Anonymous
- ✓ Prevents double-spending (except by majority)
- ✓ Distributed authority
- ✓ Practical
- ✓ No details on how to create coins

Review

- Ownership of coins is provided by tracing transactions through ledger.
- Ledger is distributed between many peers (nodes).
- Ledger is captured in block chain which cannot be altered after it is produced; only extended by adding new blocks.
- Adding an entry to the block chain costs CPU power, but node gets transaction fees and some newly created coins.
- Longest block chain is the valid block chain. Longest chain likely to be correct if majority of CPUs are honest.

Next Lecture…

Who knows?