CRYPTOCURRENCIES: Protocols for Consensus

Andrew Lewis-Pye, LSE

Two basic tools from cryptography

(1) Hash functions

Acts (essentially) like a random string generator.

This means you are unlikely ever to find two inputs which hash to the same value.

Two basic tools from cryptography

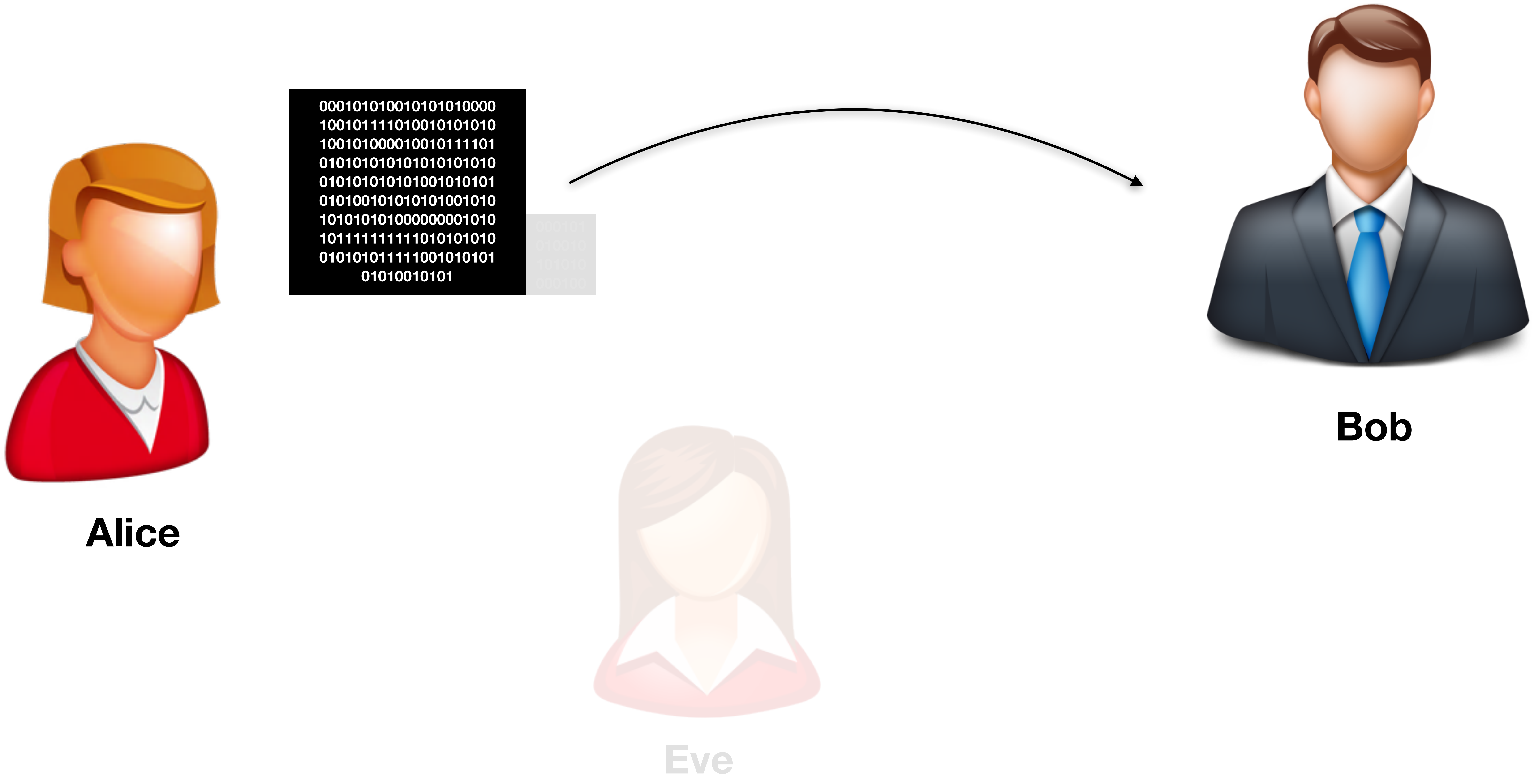
(1) Hash functions

Acts (essentially) like a random string generator.

This means you are unlikely ever to find two inputs which hash to the same value.

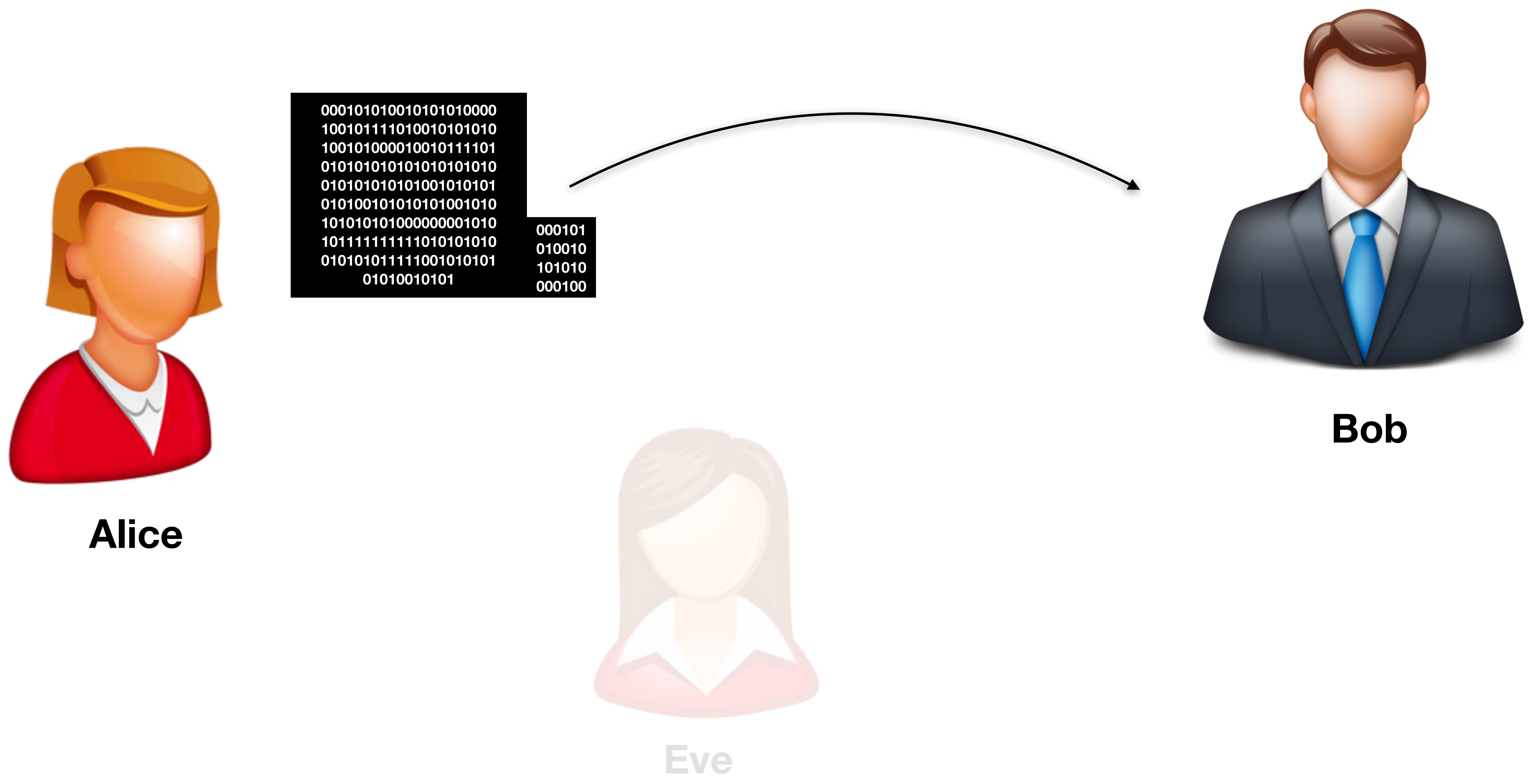
Two basic tools from cryptography

(2) Digital signatures



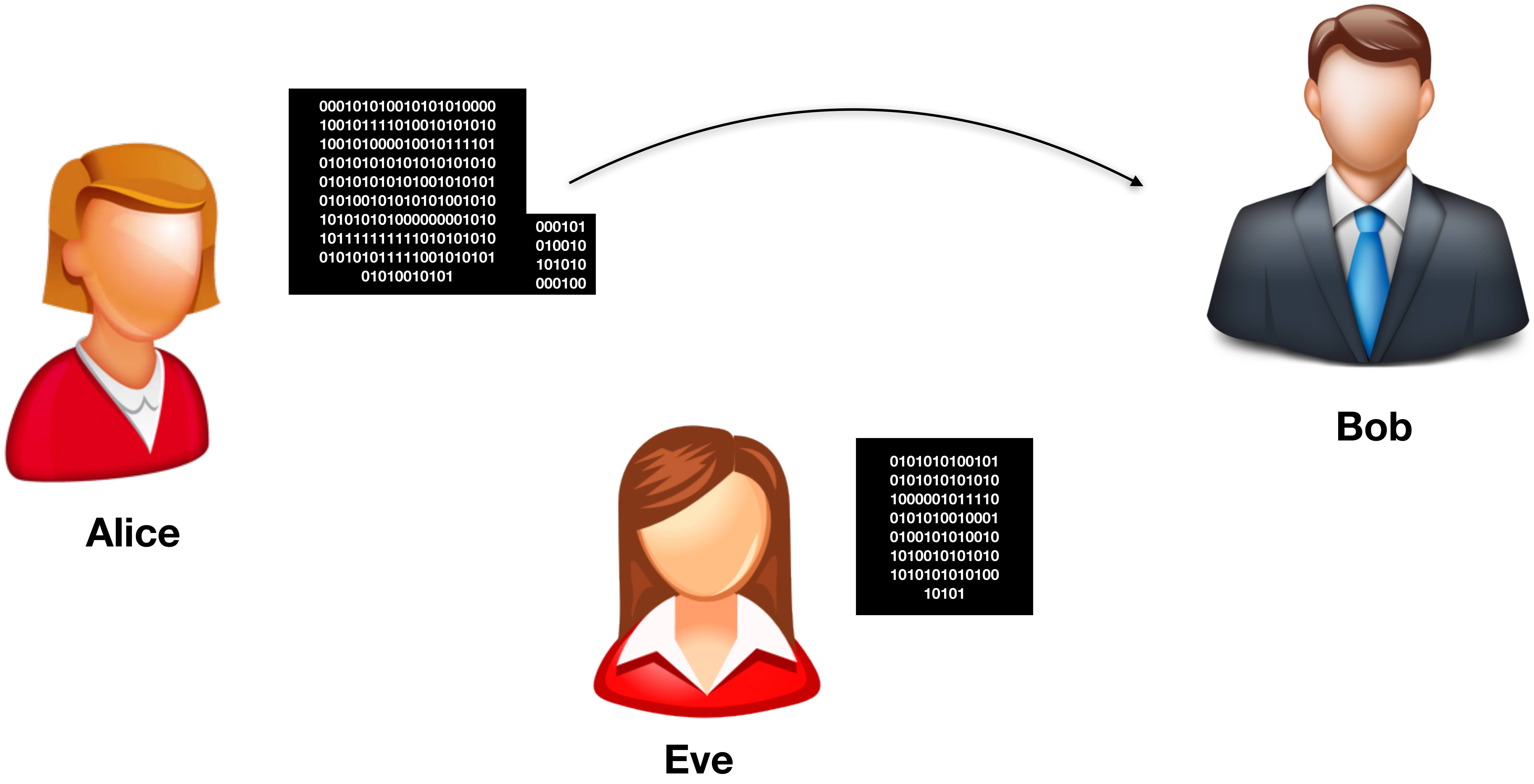
Two basic tools from cryptography

(2) Digital signatures



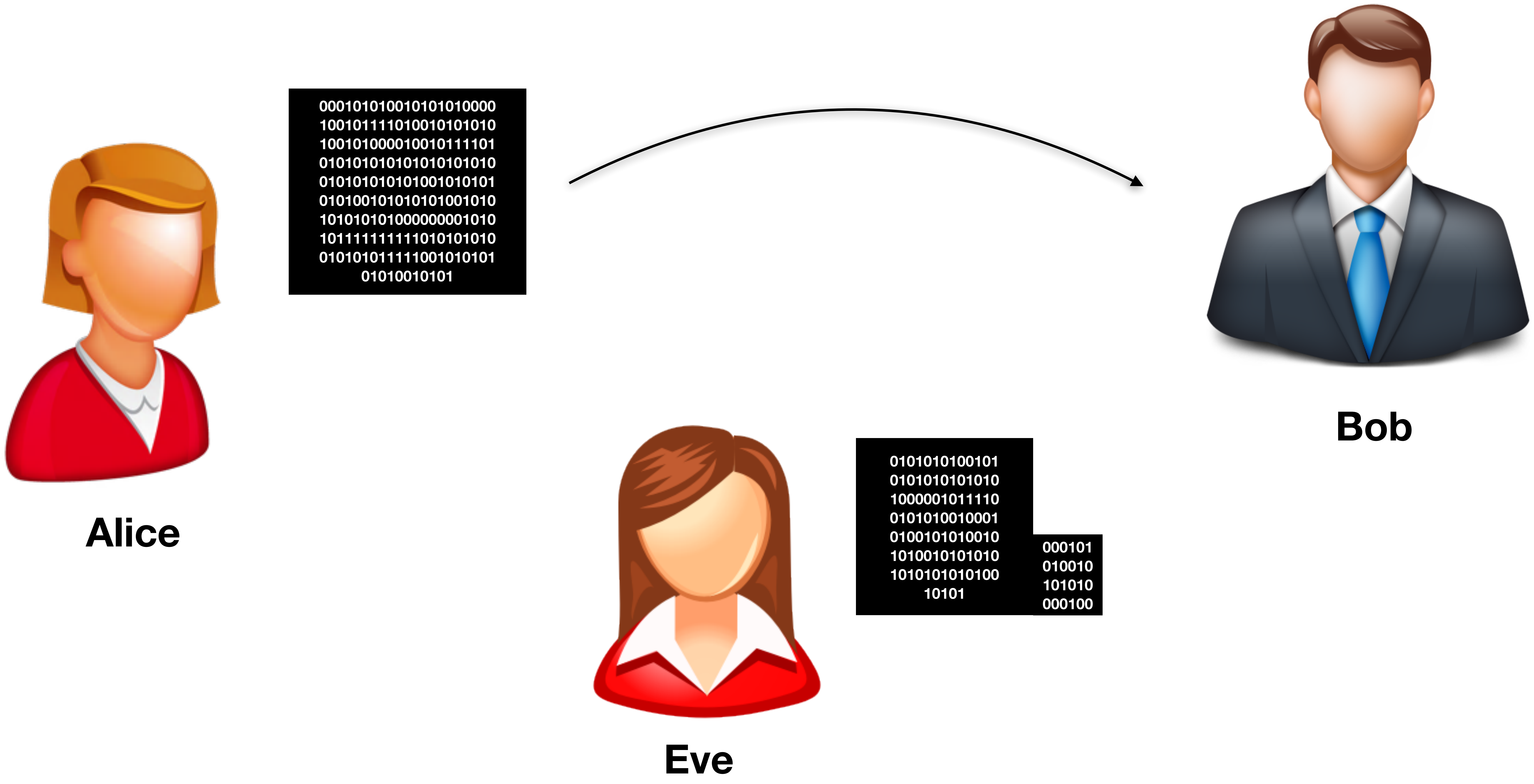
Two basic tools from cryptography

(2) Digital signatures



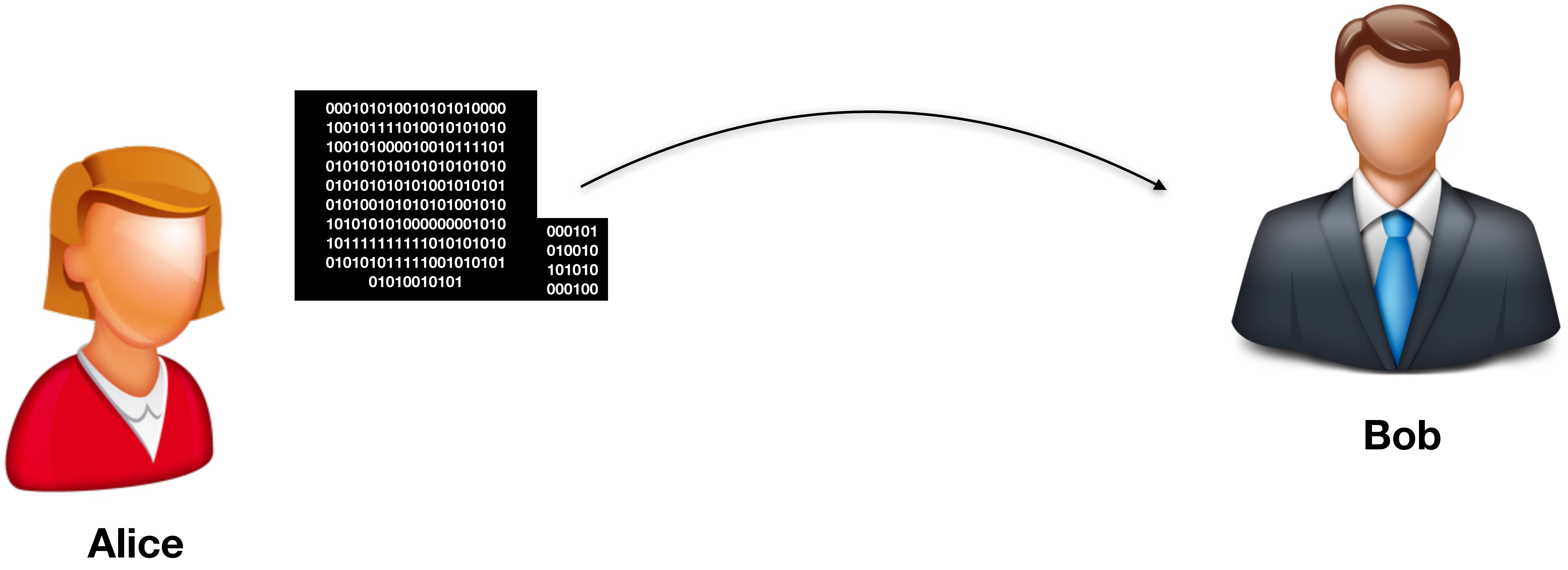
Two basic tools from cryptography

(2) Digital signatures



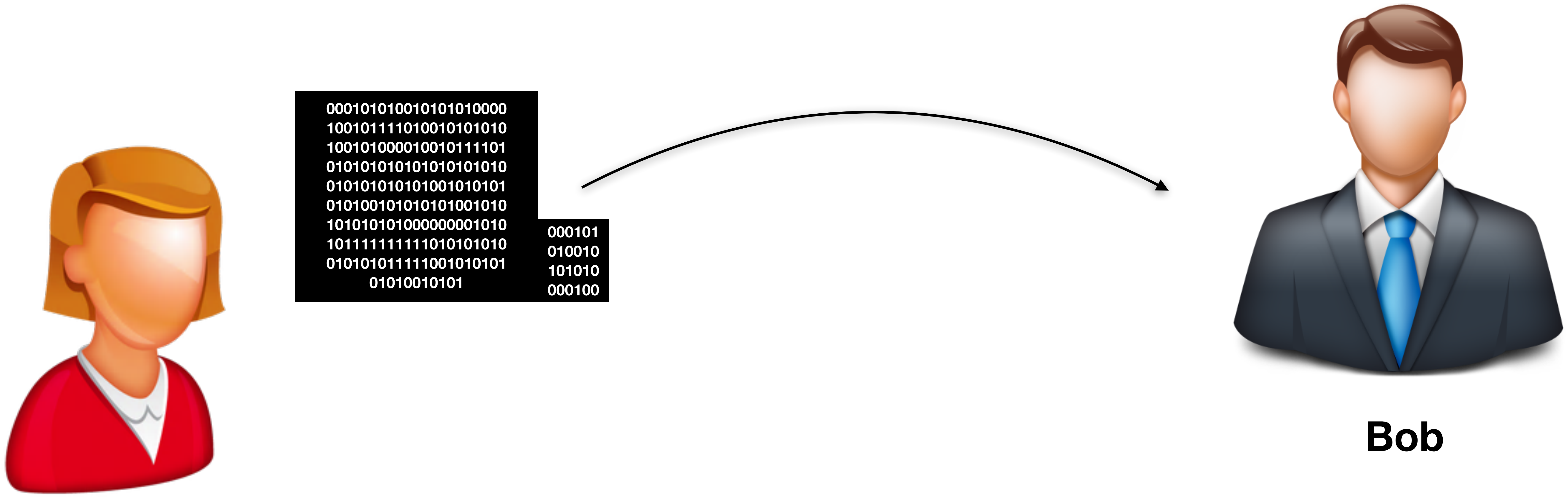
Two basic tools from cryptography

(2) Digital signatures



Two basic tools from cryptography

(2) Digital signatures



Alice

Bob

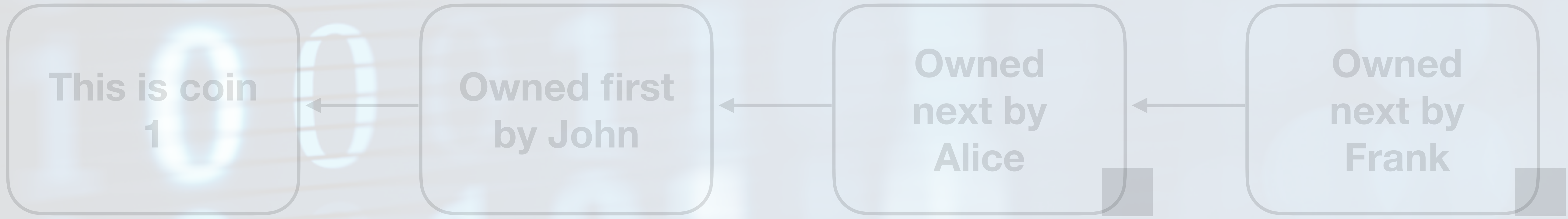
What is achieved:

When somebody sends a message, the receiver can be sure who it came from.

How to design a cryptocurrency?

The whole point of Bitcoin is that it should be decentralised. First of all, though, let’s consider how things might work with a central bank…

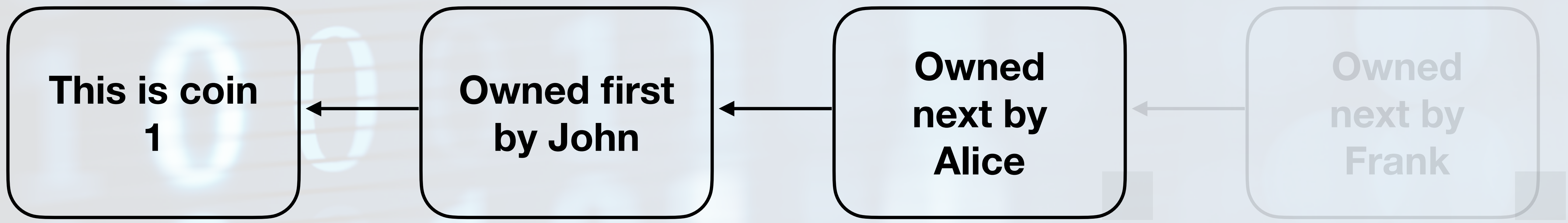
We could have a ledger for each coin…



How to design a cryptocurrency?

The whole point of Bitcoin is that it should be decentralised. First of all, though, let's consider how things might work with a central bank…

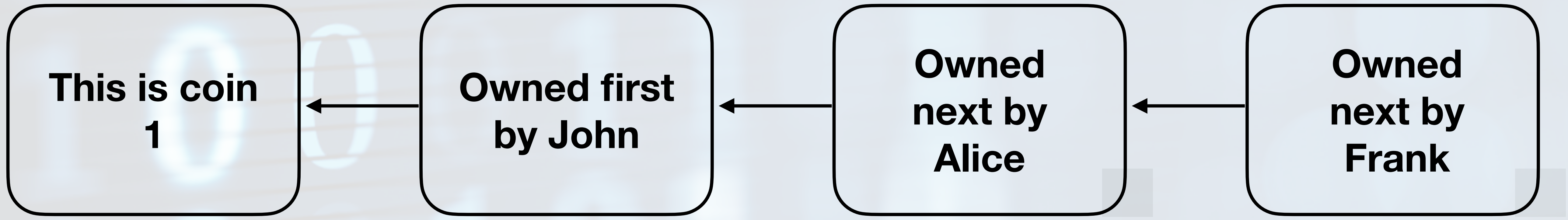
We could have a ledger for each coin…

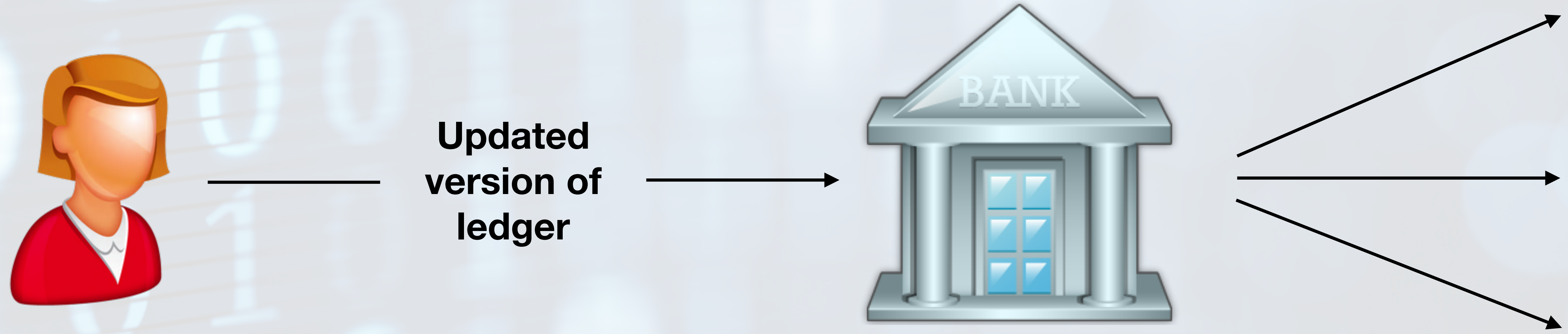


How to design a cryptocurrency?

The whole point of Bitcoin is that it should be decentralised. First of all, though, let's consider how things might work with a central bank…

We could have a ledger for each coin…

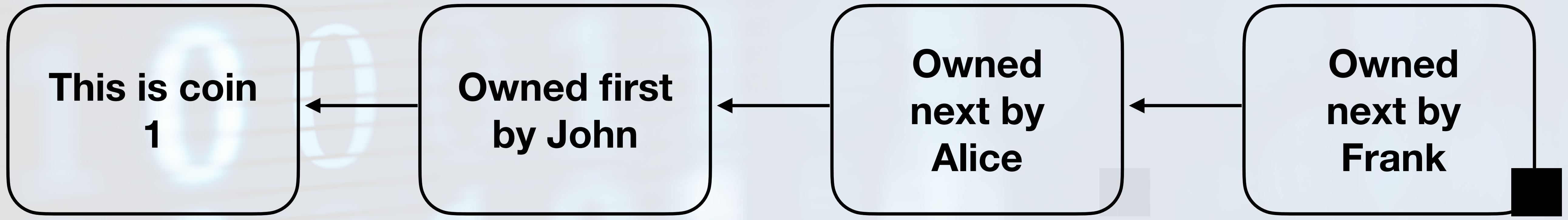


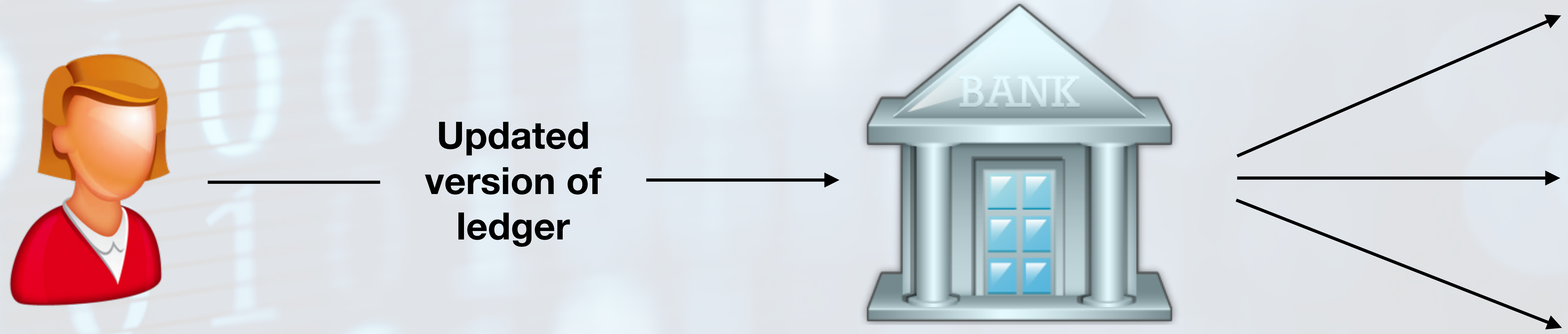


How to design a cryptocurrency?

The whole point of Bitcoin is that it should be decentralised. First of all, though, let's consider how things might work with a central bank…

We could have a ledger for each coin…

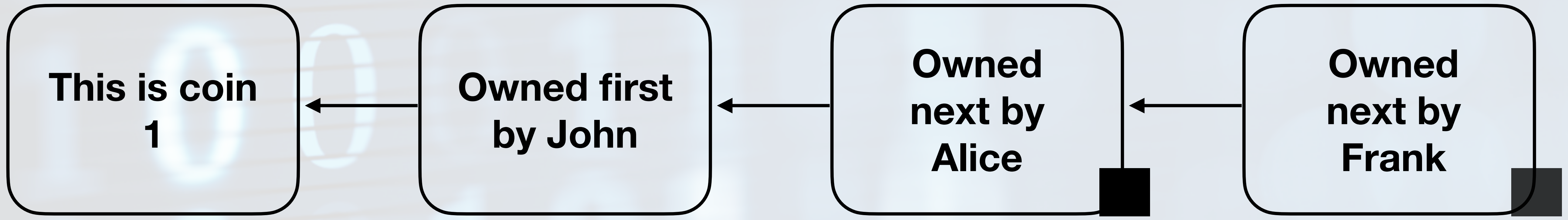


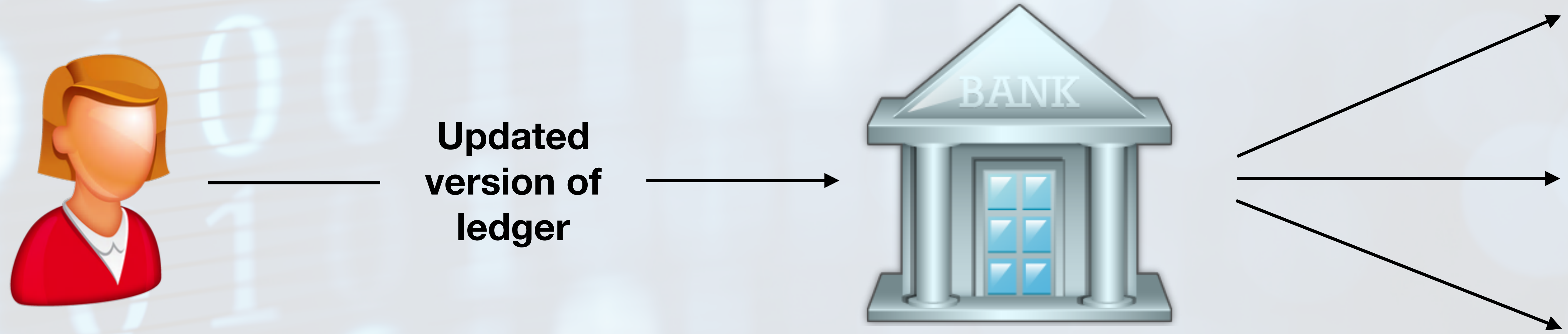


How to design a cryptocurrency?

The whole point of Bitcoin is that it should be decentralised. First of all, though, let's consider how things might work with a central bank…

We could have a ledger for each coin…





How to design a cryptocurrency?

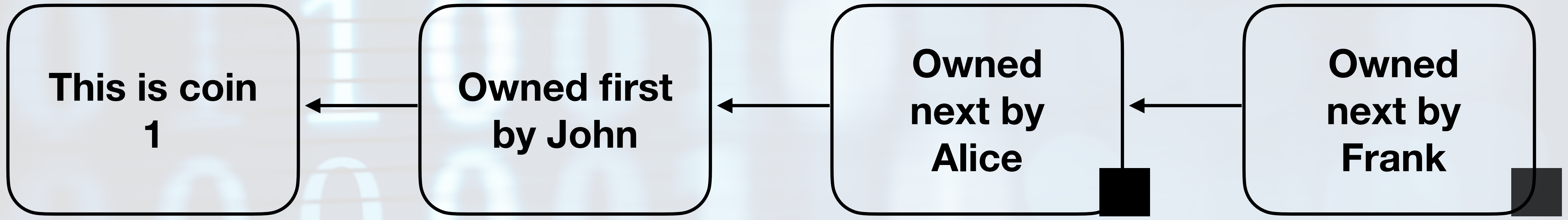
What this process (with the central bank) achieves:

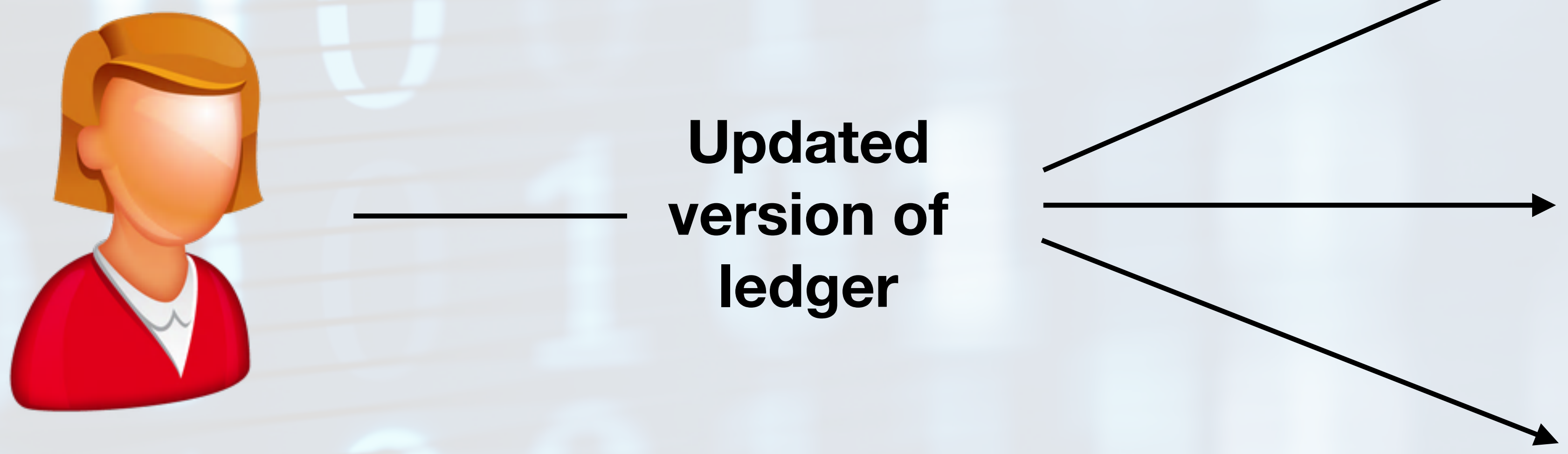
(1) Only Alice can spend her coin.

(2) She cannot spend it twice.

How to design a cryptocurrency?

Now what happens without the central bank?



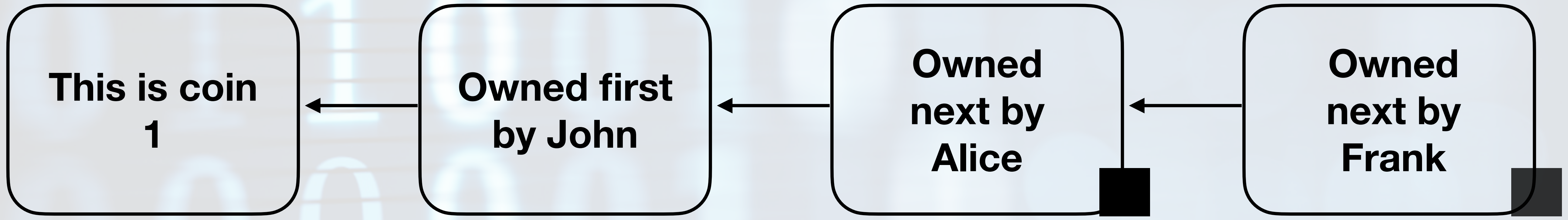


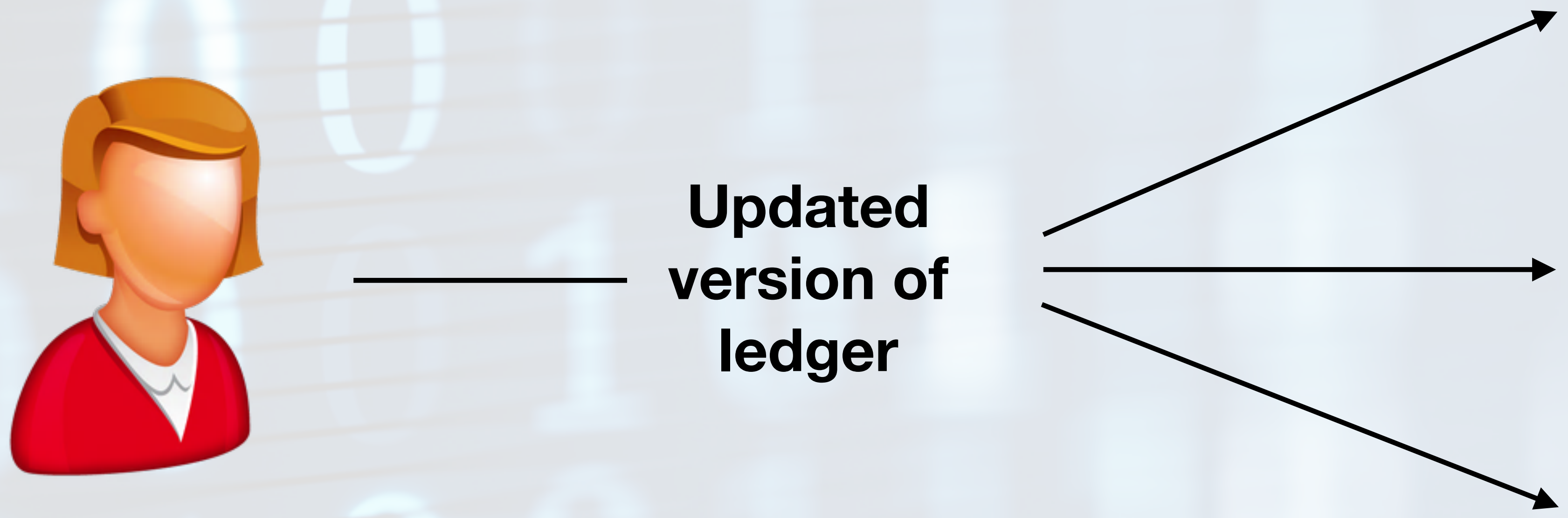
(1) Only Alice can spend her coin. ✓

(2) She cannot spend it twice. ✗

How to design a cryptocurrency?

Now what happens without the central bank?





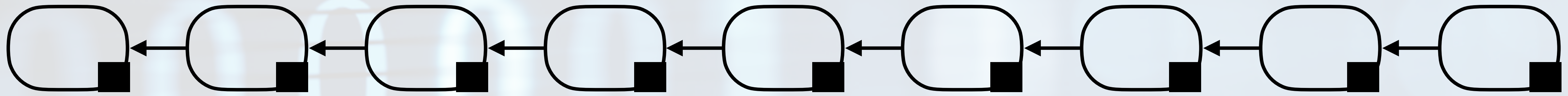
(1) Only Alice can spend her coin. ✓

(2) She cannot spend it twice. ✗

How to design a cryptocurrency?

…so how to avoid double spending?

1) Let's (have all users) keep a universal ledger of all coins.



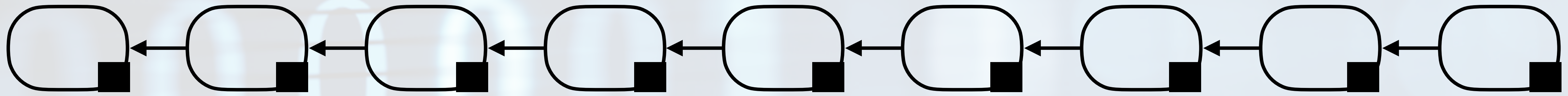
2) We could specify a Proof-Of-Work (a hard computational task) for each transaction, and only append transactions to the ledger once the corresponding POW has been completed.

…so now, when Alice wants to spend her coin, she sends the transaction out into the network of users who all start tying to provide the corresponding POW. Once somebody completes the POW the transaction is appended to the ledger.

How to design a cryptocurrency?

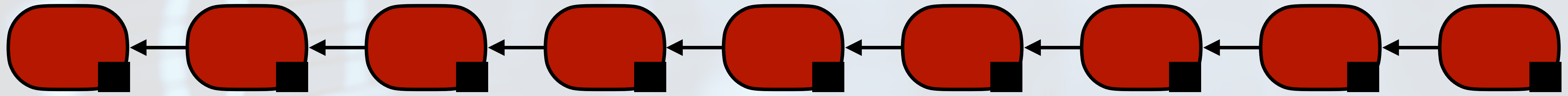
…so how to avoid double spending?

1) Let's (have all users) keep a universal ledger of all coins.



2) We could specify a Proof-Of-Work (result of a hard computational task) for each transaction, and only append transactions to the ledger once the corresponding POW has been completed.

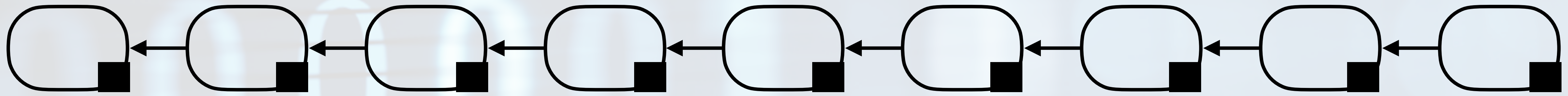
…so now, when Alice wants to spend her coin, she sends the transaction out into the network of users who all start trying to provide the corresponding POW. Once somebody completes the POW the transaction is appended to the ledger.



How to design a cryptocurrency?

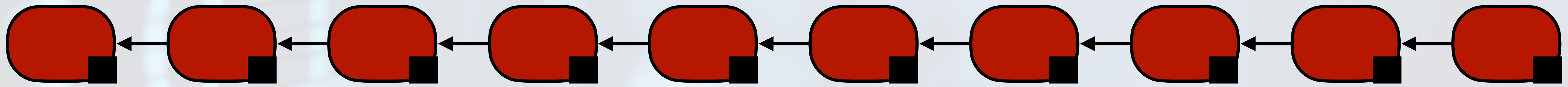
…so how to avoid double spending?

1) Let's (have all users) keep a universal ledger of all coins.



2) We could specify a Proof-Of-Work (result of a hard computational task) for each transaction, and only append transactions to the ledger once the corresponding POW has been completed.

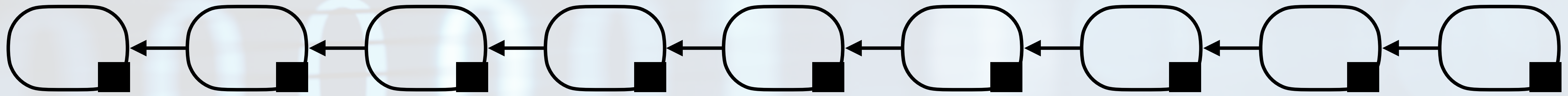
…so now, when Alice wants to spend her coin, she sends the transaction out into the network of users who all start trying to provide the corresponding POW. Once somebody completes the POW the transaction is appended to the ledger.



How to design a cryptocurrency?

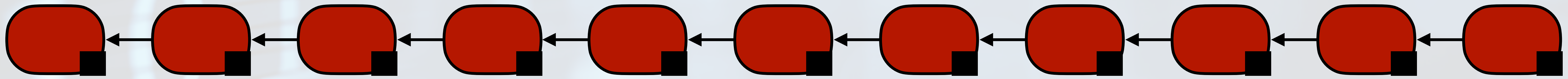
…so how to avoid double spending?

1) Let's (have all users) keep a universal ledger of all coins.



2) We could specify a Proof-Of-Work (result of a hard computational task) for each transaction, and only append transactions to the ledger once the corresponding POW has been completed.

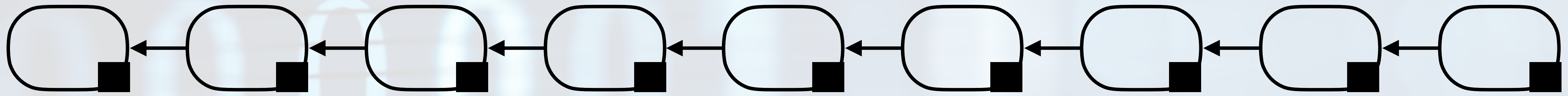
…so now, when Alice wants to spend her coin, she sends the transaction out into the network of users who all start trying to provide the corresponding POW. Once somebody completes the POW the transaction is appended to the ledger.



How to design a cryptocurrency?

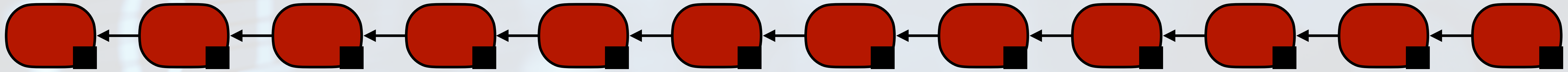
…so how to avoid double spending?

1) Let's (have all users) keep a universal ledger of all coins.



2) We could specify a Proof-Of-Work (result of a hard computational task) for each transaction, and only append transactions to the ledger once the corresponding POW has been completed.

…so now, when Alice wants to spend her coin, she sends the transaction out into the network of users who all start trying to provide the corresponding POW. Once somebody completes the POW the transaction is appended to the ledger.



How to design a cryptocurrency?

…so how to avoid double spending.. (ctd)..

3) We specify that the CORRECT version of the ledger is always the longest one.

4) We agree that a transaction is CONFIRMED once it is in the ledger and is followed by sufficiently many transactions.

How does this avoid double spending?

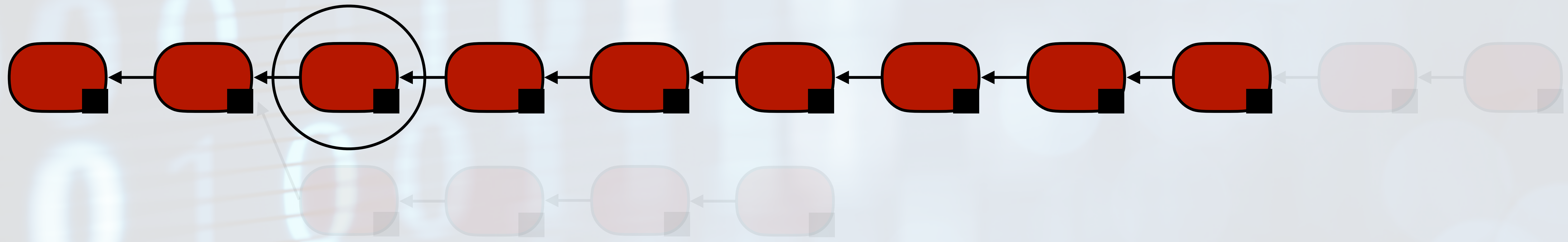
How to design a cryptocurrency?

…so how to avoid double spending.. (ctd)..

3) We specify that the CORRECT version of the ledger is always the longest one.

4) We agree that a transaction is CONFIRMED once it is in the ledger and is followed by sufficiently many transactions.

How does this avoid double spending?



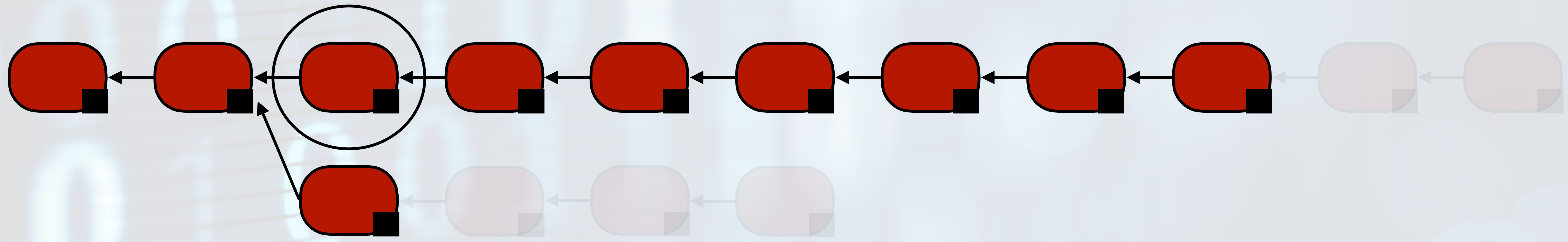
How to design a cryptocurrency?

…so how to avoid double spending.. (ctd)..

3) We specify that the CORRECT version of the ledger is always the longest one.

4) We agree that a transaction is CONFIRMED once it is in the ledger and is followed by sufficiently many transactions.

How does this avoid double spending?



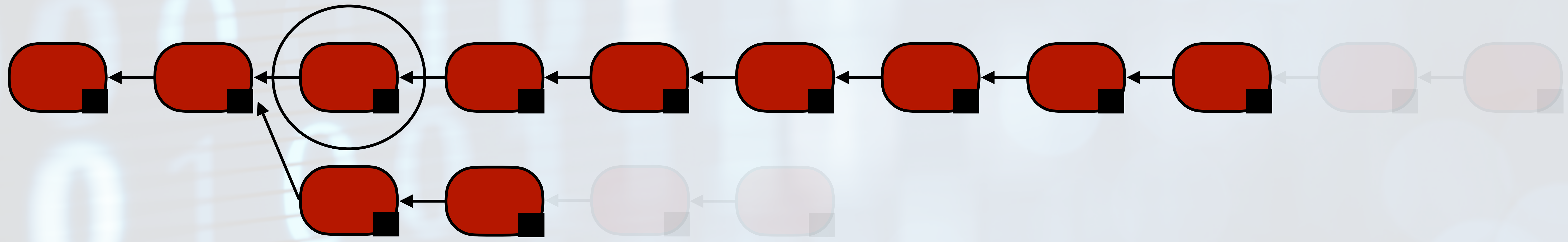
How to design a cryptocurrency?

…so how to avoid double spending.. (ctd)..

3) We specify that the CORRECT version of the ledger is always the longest one.

4) We agree that a transaction is CONFIRMED once it is in the ledger and is followed by sufficiently many transactions.

How does this avoid double spending?



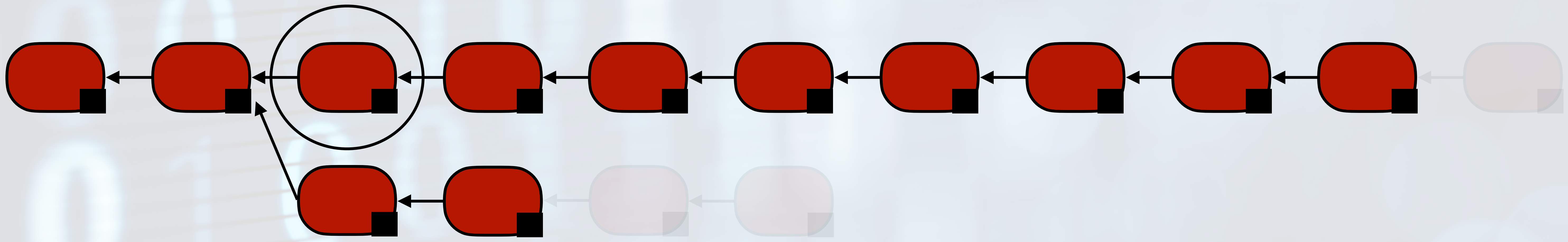
How to design a cryptocurrency?

…so how to avoid double spending.. (ctd)..

3) We specify that the CORRECT version of the ledger is always the longest one.

4) We agree that a transaction is CONFIRMED once it is in the ledger and is followed by sufficiently many transactions.

How does this avoid double spending?



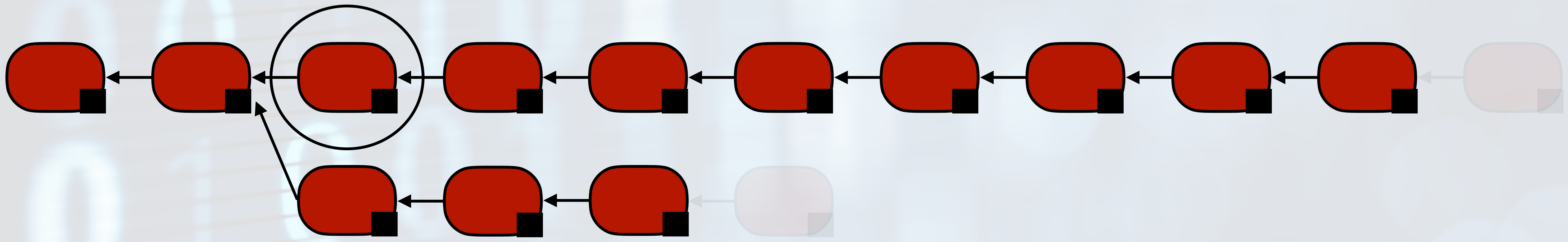
How to design a cryptocurrency?

…so how to avoid double spending.. (ctd)..

3) We specify that the CORRECT version of the ledger is always the longest one.

4) We agree that a transaction is CONFIRMED once it is in the ledger and is followed by sufficiently many transactions.

How does this avoid double spending?



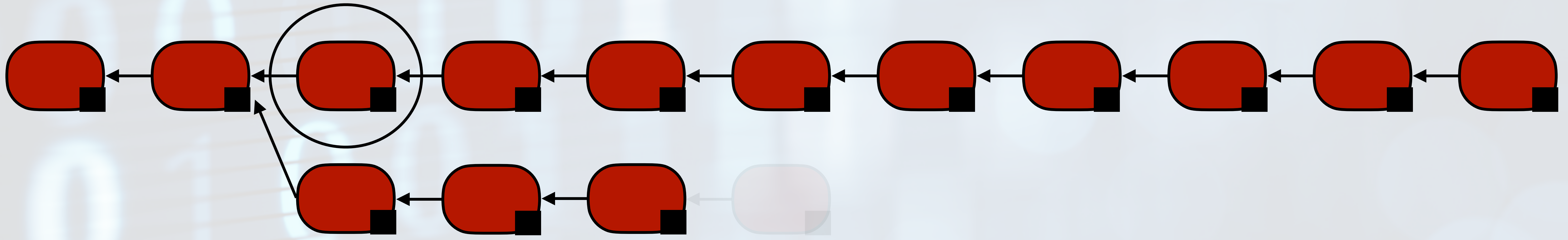
How to design a cryptocurrency?

...so how to avoid double spending.. (ctd)..

3) We specify that the CORRECT version of the ledger is always the longest one.

4) We agree that a transaction is CONFIRMED once it is in the ledger and is followed by sufficiently many transactions.

How does this avoid double spending?



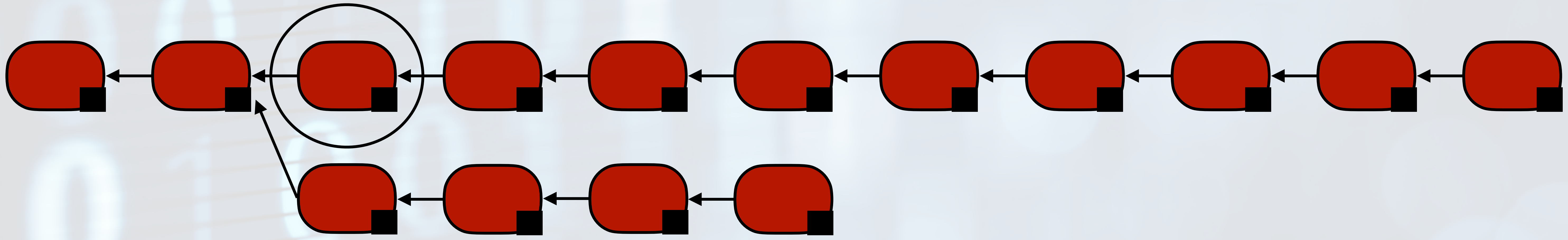
How to design a cryptocurrency?

...so how to avoid double spending.. (ctd)..

3) We specify that the CORRECT version of the ledger is always the longest one.

4) We agree that a transaction is CONFIRMED once it is in the ledger and is followed by sufficiently many transactions.

How does this avoid double spending?



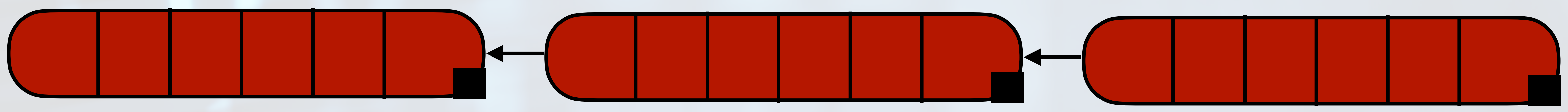
The adversary would need more computational power than the rest of the network combined!

How to design a cryptocurrency?

Some further details:

1) Let’s call the people looking for the necessary POWs miners. We better pay them for their effort.

2) If we actually append transactions individually this will cause timing problems. Much better to have the miners group transactions together into large blocks, and require a POW for each block.



How to design a cryptocurrency?

Some further details:

3) We can specify the POW for each block of transactions using an agreed on hash function.

Take the data which is the block:

1001............010101 | **10001** → **hash** → **11011....00000**

For any given k, by a NONCE for the block, we mean something we can append to the block, so that when it's fed into the hash function we get an output ending with k many 0s.

How to design a cryptocurrency?

Some further details:

3) We can specify the POW for each block of transactions using an agreed on hash function.

Take the data which is the block:

For any given k, by a NONCE for the block, we mean something we can append to the block, so that when the block and the nonce are fed into the hash function we get an output ending with at least k many 0s.

The POW required is a NONCE (for k which is chosen to make the task hard).

What are the limitations?

Many…including:

What are the limitations?

Many…including:

Massive energy consumption!

What are the limitations?

Many…including:

Massive energy consumption!

How secure is it really?

What are the limitations?

Many…including:

Massive energy consumption!

How secure is it really?

Transaction rates…

…and solutions?

Many…including:

Massive energy consumption!

How secure is it really?

→ Proof-of-stake is one approach

Transaction rates…

Scalability

The problem

The second bottleneck

(all or many nodes verify all transactions)

The first bottleneck

(network latency means blocks can't be produced too fast)

The solutions

Layer 2

(protocols which are implemented on top of the underlying cryptocurrency)

Layer 1

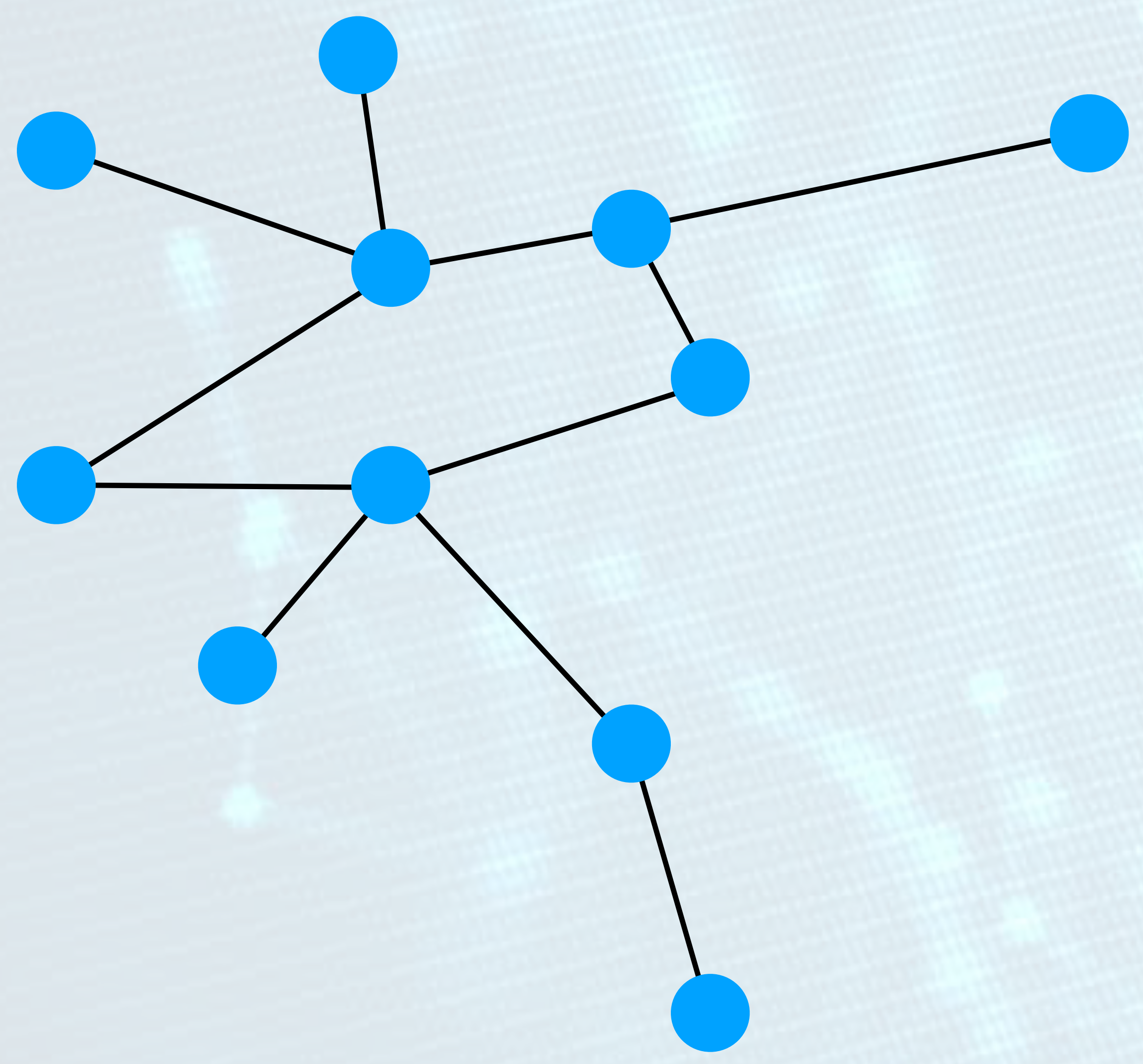
(solutions at the level of the protocol itself)

Layer 0

(underlying infrastructure used by the protocol)

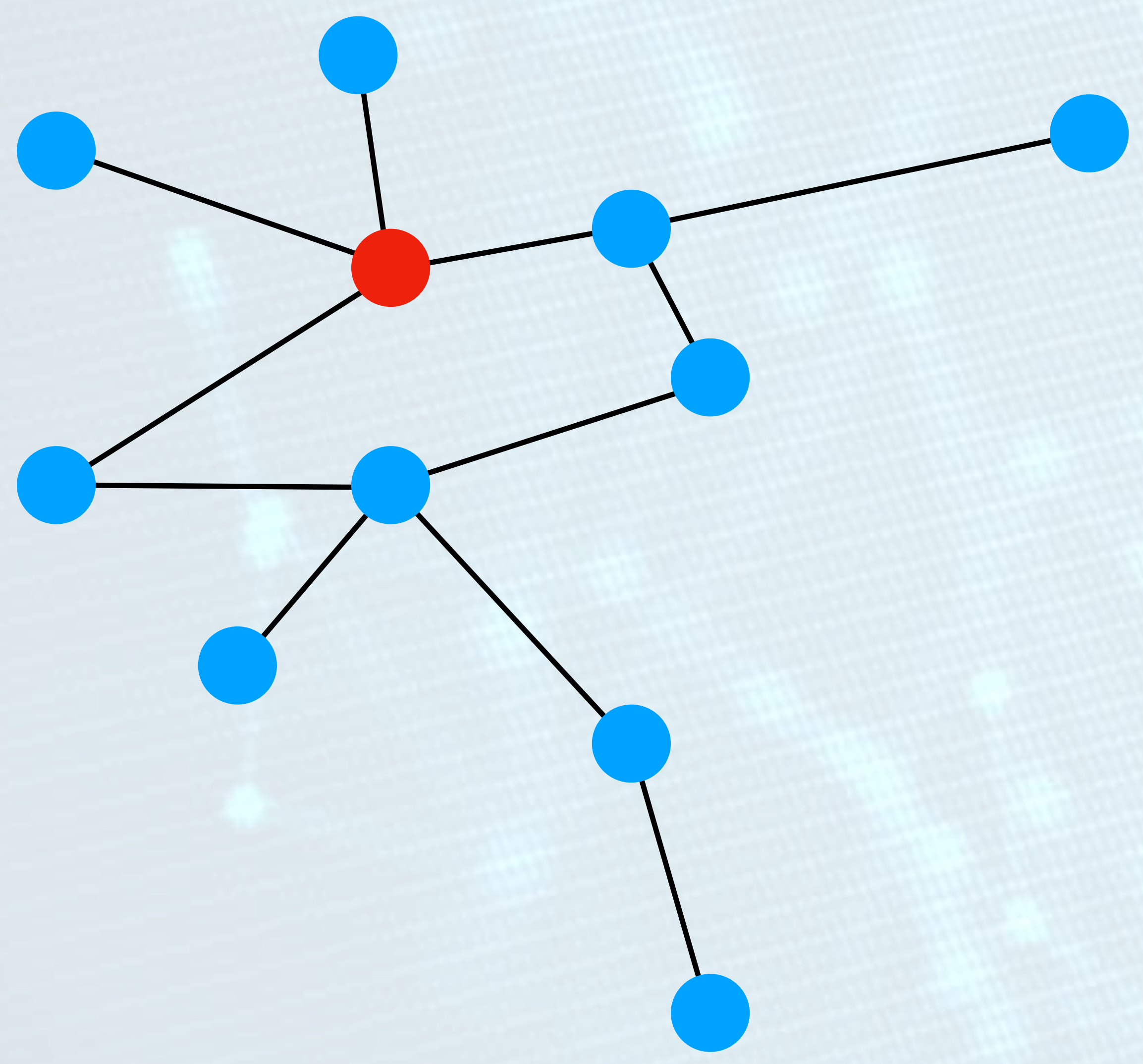
The first scaling bottleneck

The underlying communication network has latency, i.e. messages take time to travel.



The first scaling bottleneck

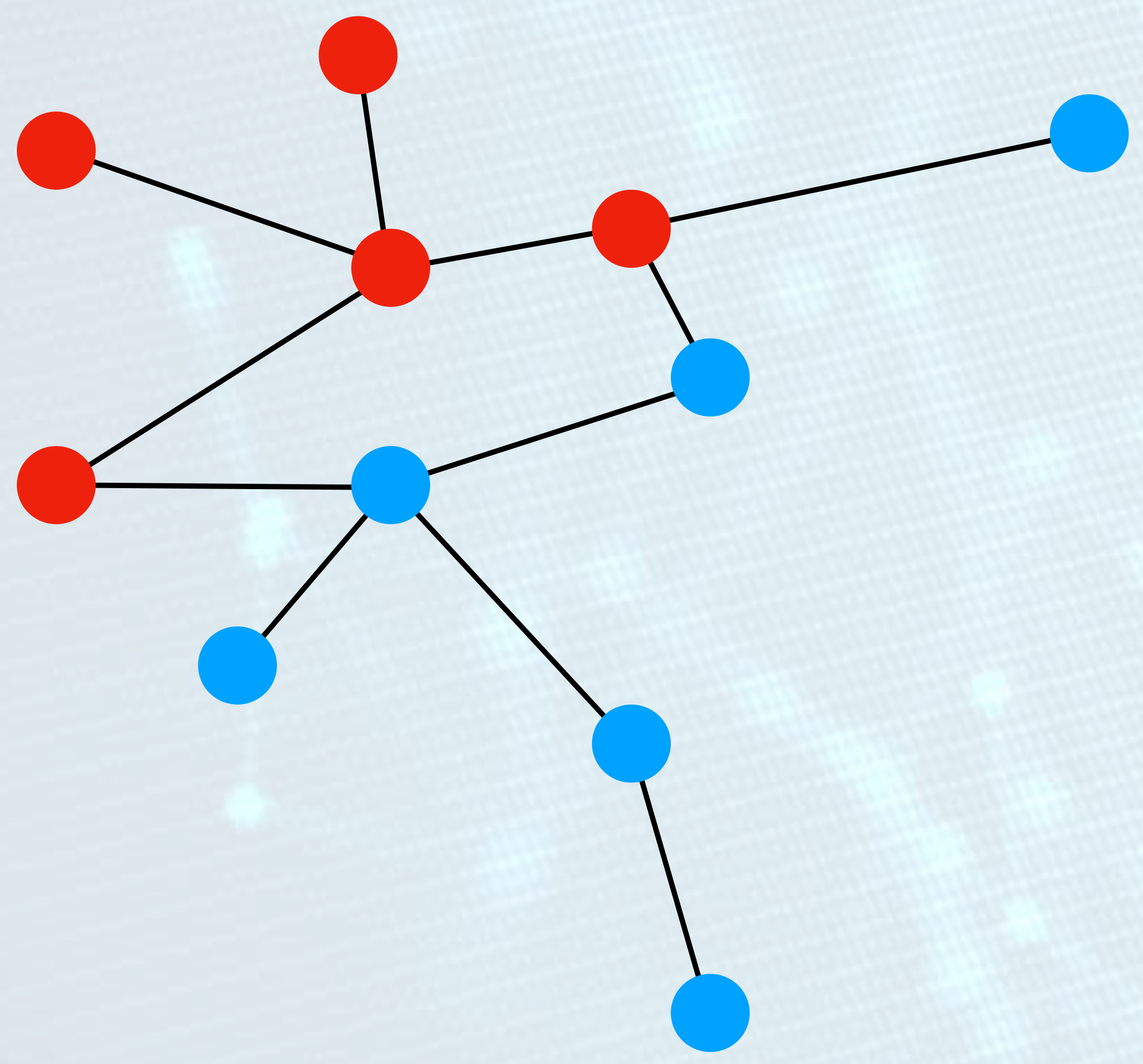
The underlying communication network has latency, i.e. messages take time to travel.



When a miner finds a block it takes time to propagate across the network.

The first scaling bottleneck

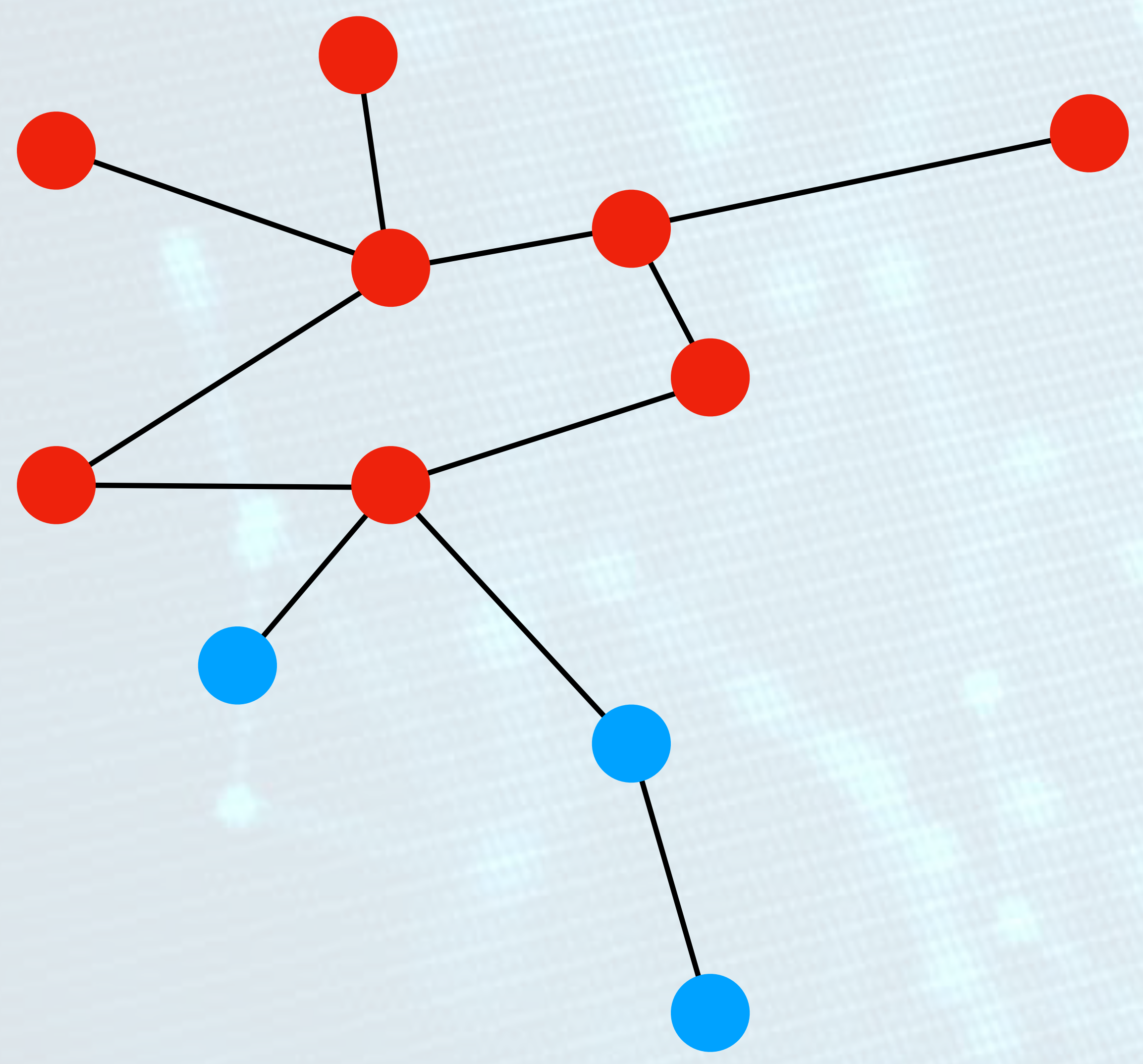
The underlying communication network has latency, i.e. messages take time to travel.



When a miner finds a block it takes time to propagate across the network.

The first scaling bottleneck

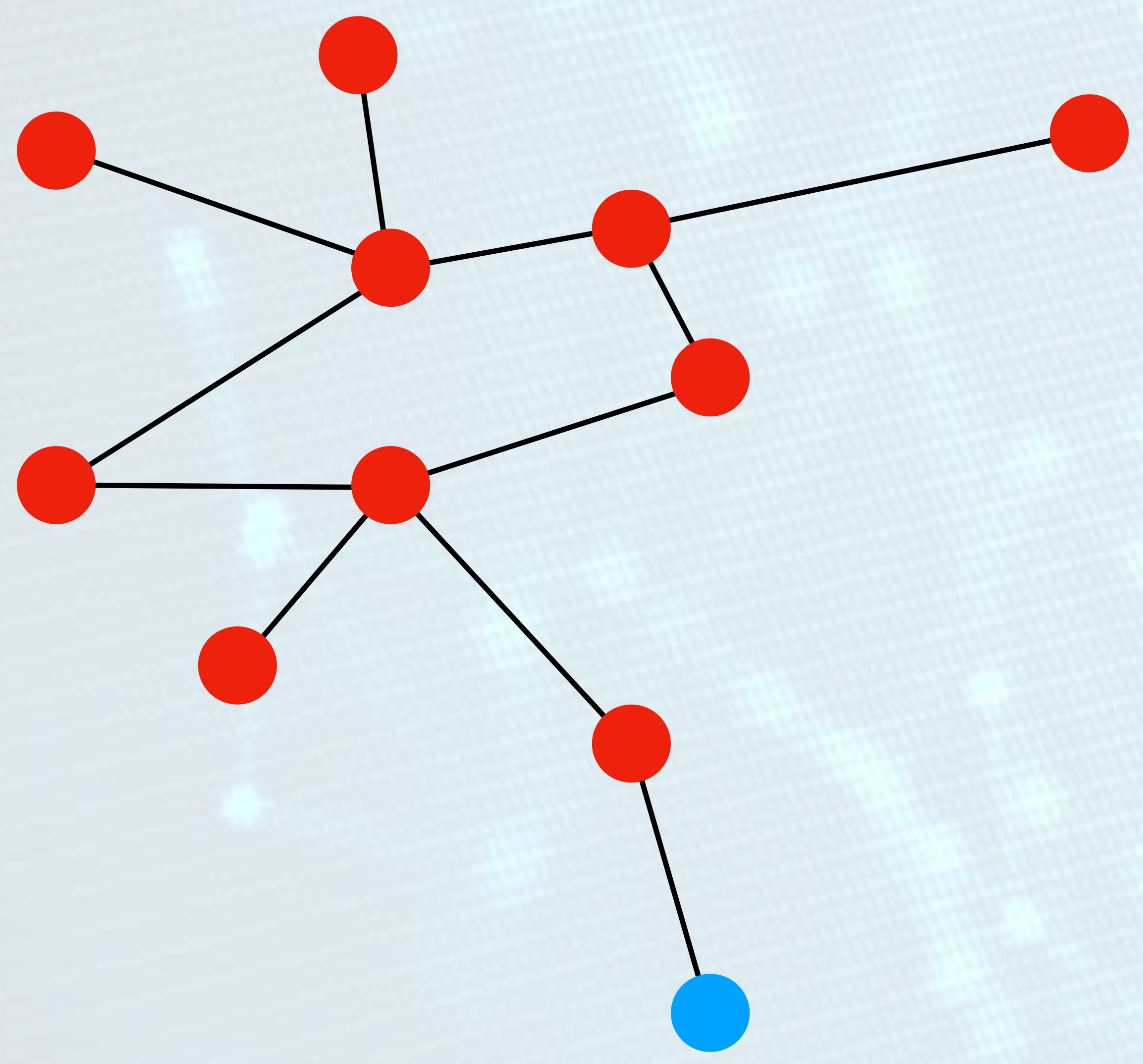
The underlying communication network has latency, i.e. messages take time to travel.



When a miner finds a block it takes time to propagate across the network.

The first scaling bottleneck

The underlying communication network has latency, i.e. messages take time to travel.



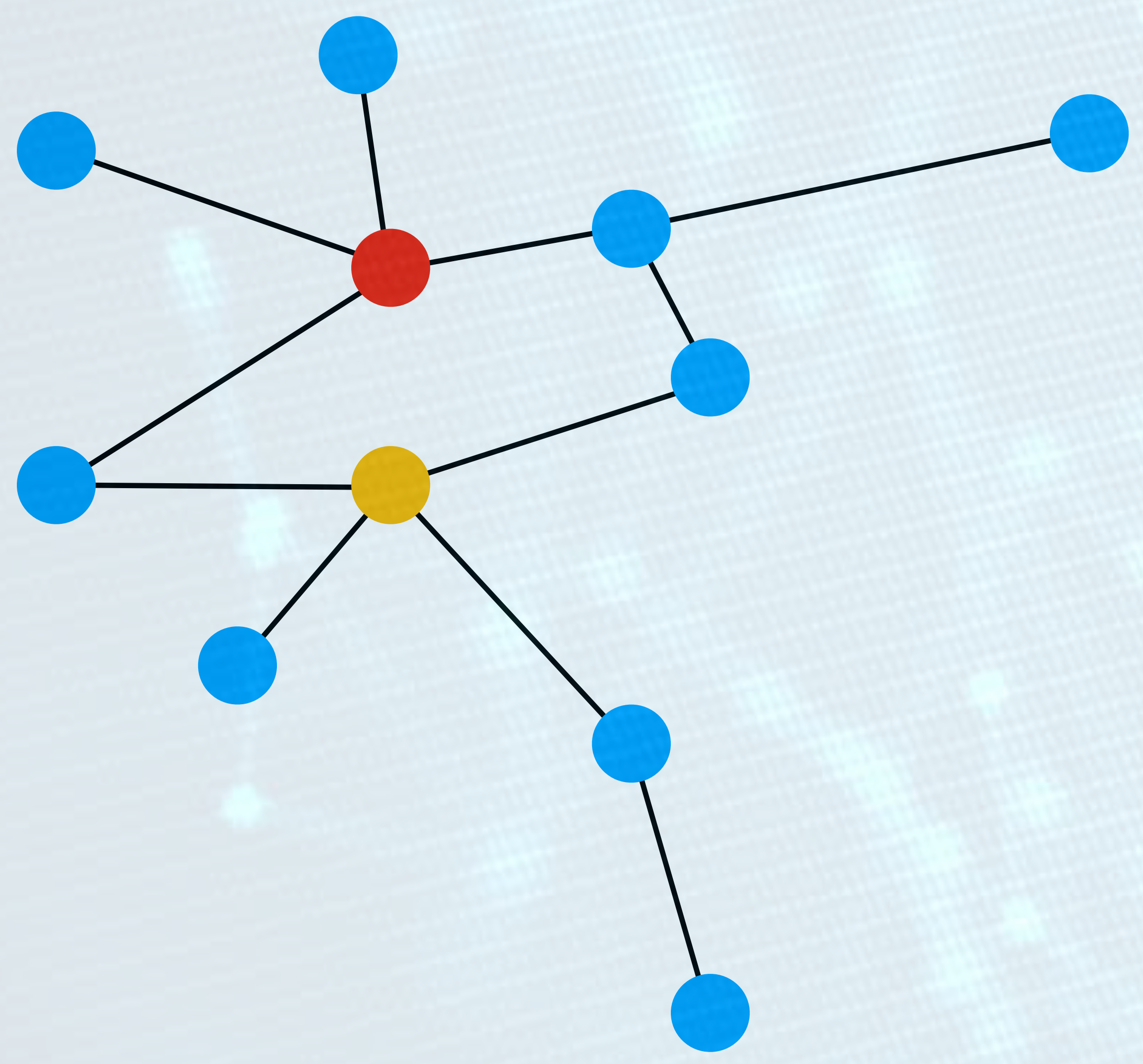
When a miner finds a block it takes time to propagate across the network.

The first scaling bottleneck

The underlying communication network has latency, i.e. messages take time to travel.

The first scaling bottleneck

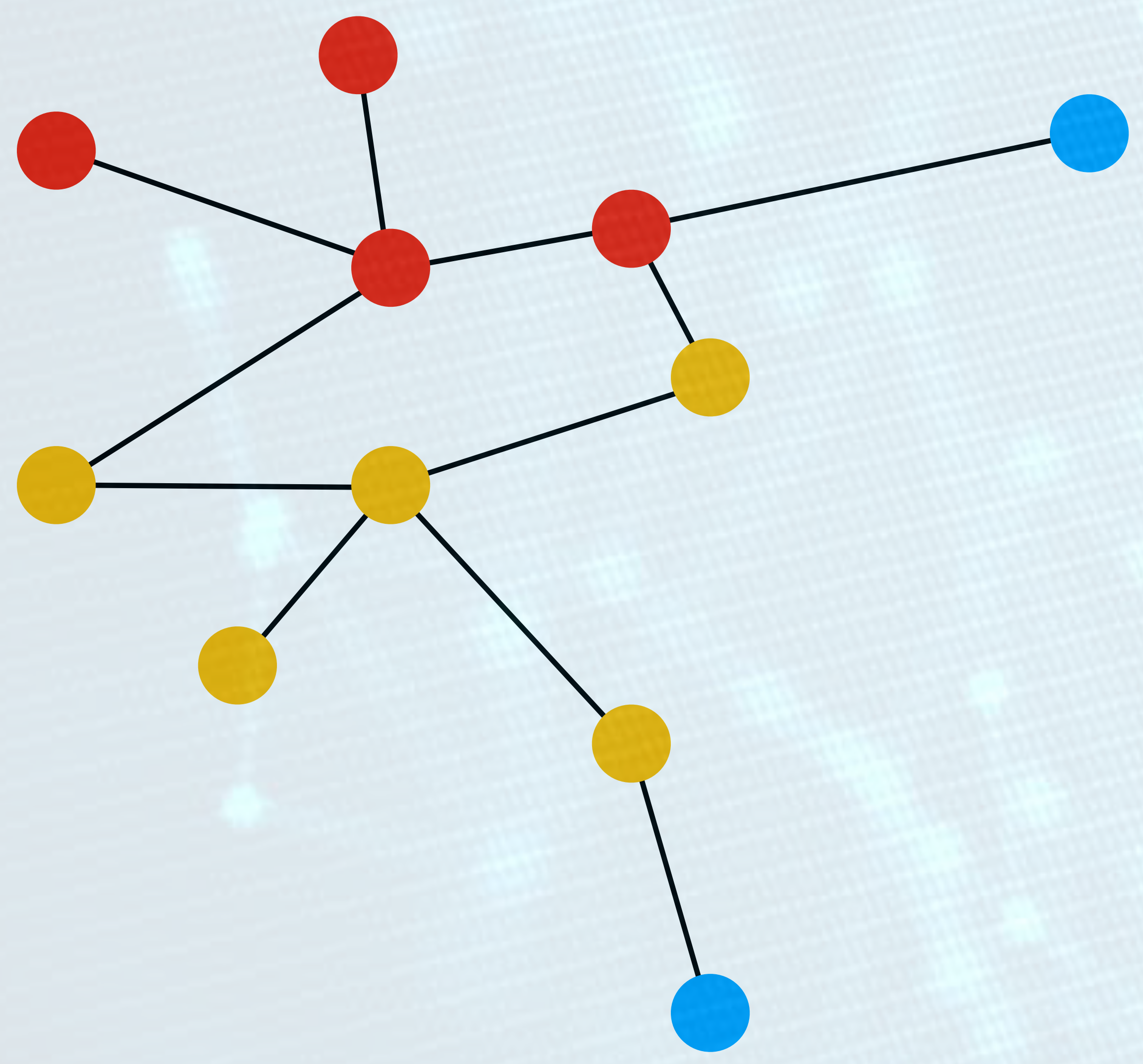
The underlying communication network has latency, i.e. messages take time to travel.



When two blocks are found almost simultaneously, this splits the network…causes a fork.

The first scaling bottleneck

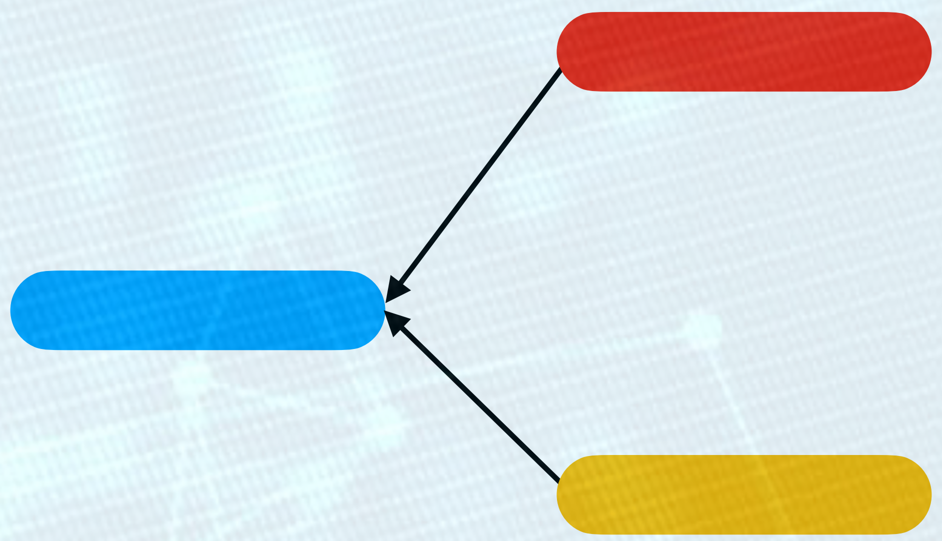
The underlying communication network has latency, i.e. messages take time to travel.



When two blocks are found almost simultaneously, this splits the network…causes a fork.

The first scaling bottleneck

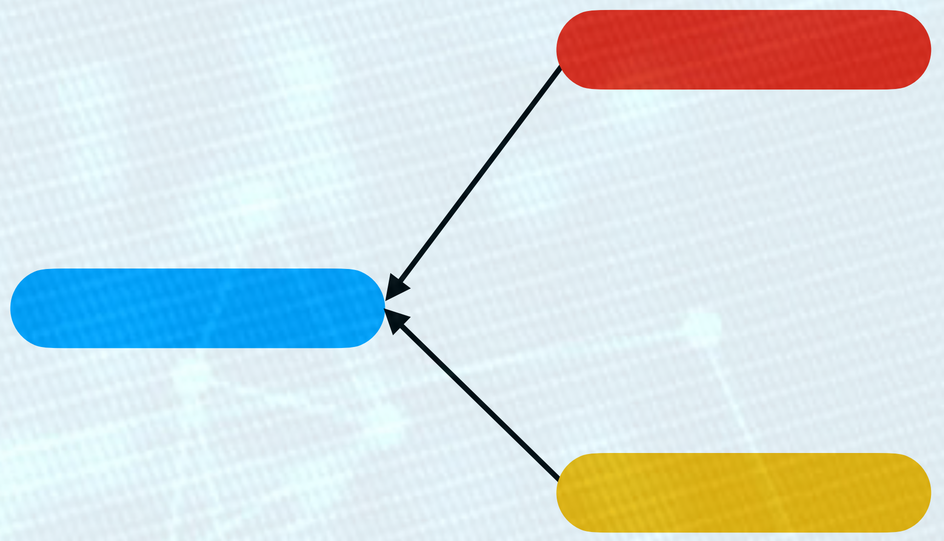
The underlying communication network has latency, i.e. messages take time to travel.



When two blocks are found almost simultaneously, this splits the network…causes a fork.

The first scaling bottleneck

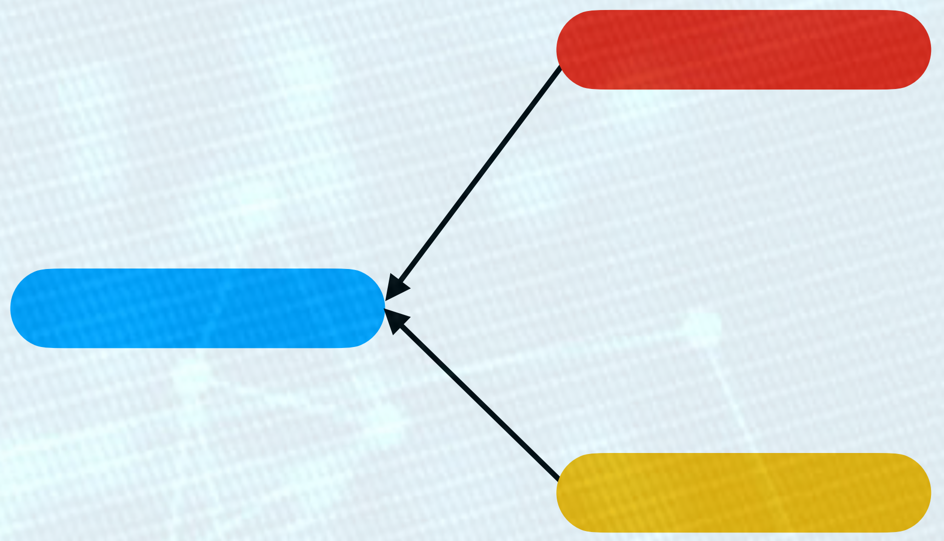
The underlying communication network has latency, i.e. messages take time to travel.



Now only half the network is working to find POWs above each side of the fork. This makes it twice as easy for our adversary.

The first scaling bottleneck

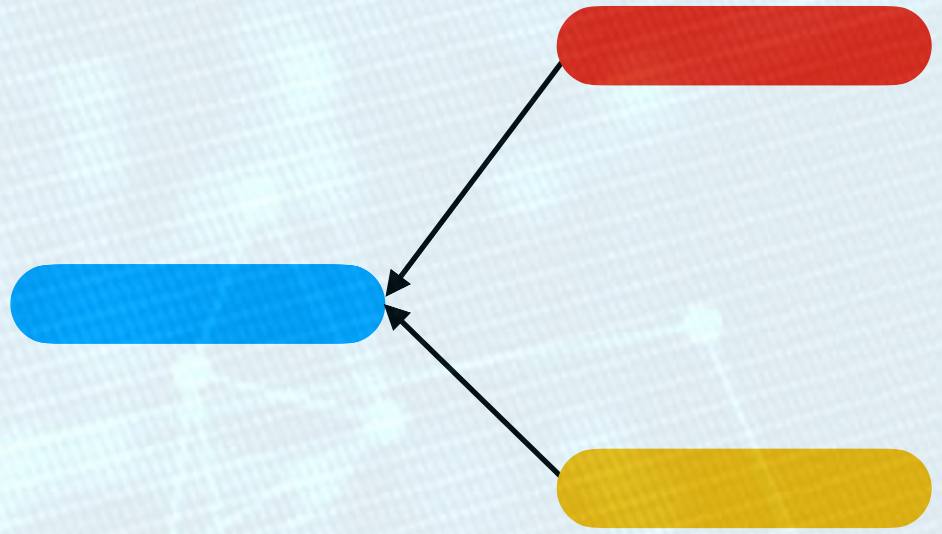
The underlying communication network has latency, i.e. messages take time to travel.



With Bitcoin this happens quite infrequently.. but..

The first scaling bottleneck

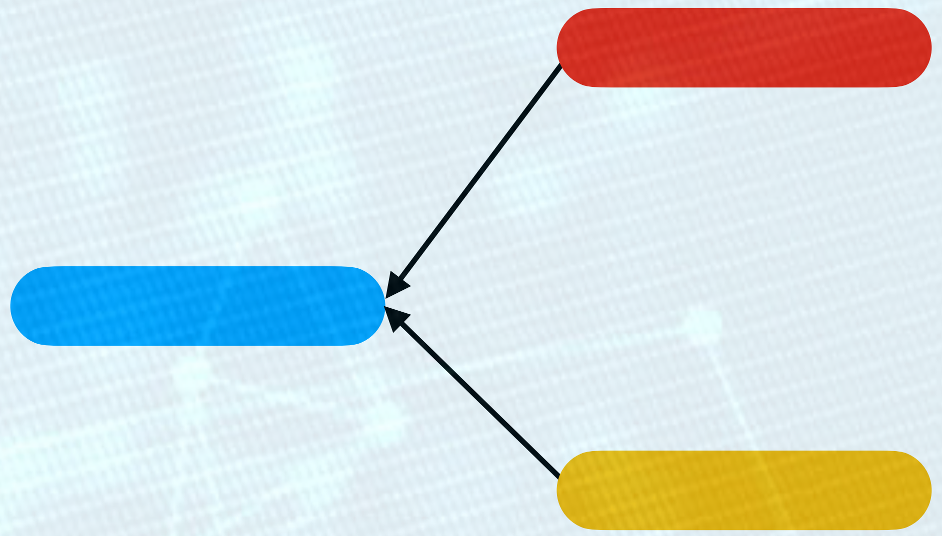
The underlying communication network has latency, i.e. messages take time to travel.



With Bitcoin this happens quite infrequently.. but..if we were to produce twice as often it would happen twice as much.

The first scaling bottleneck

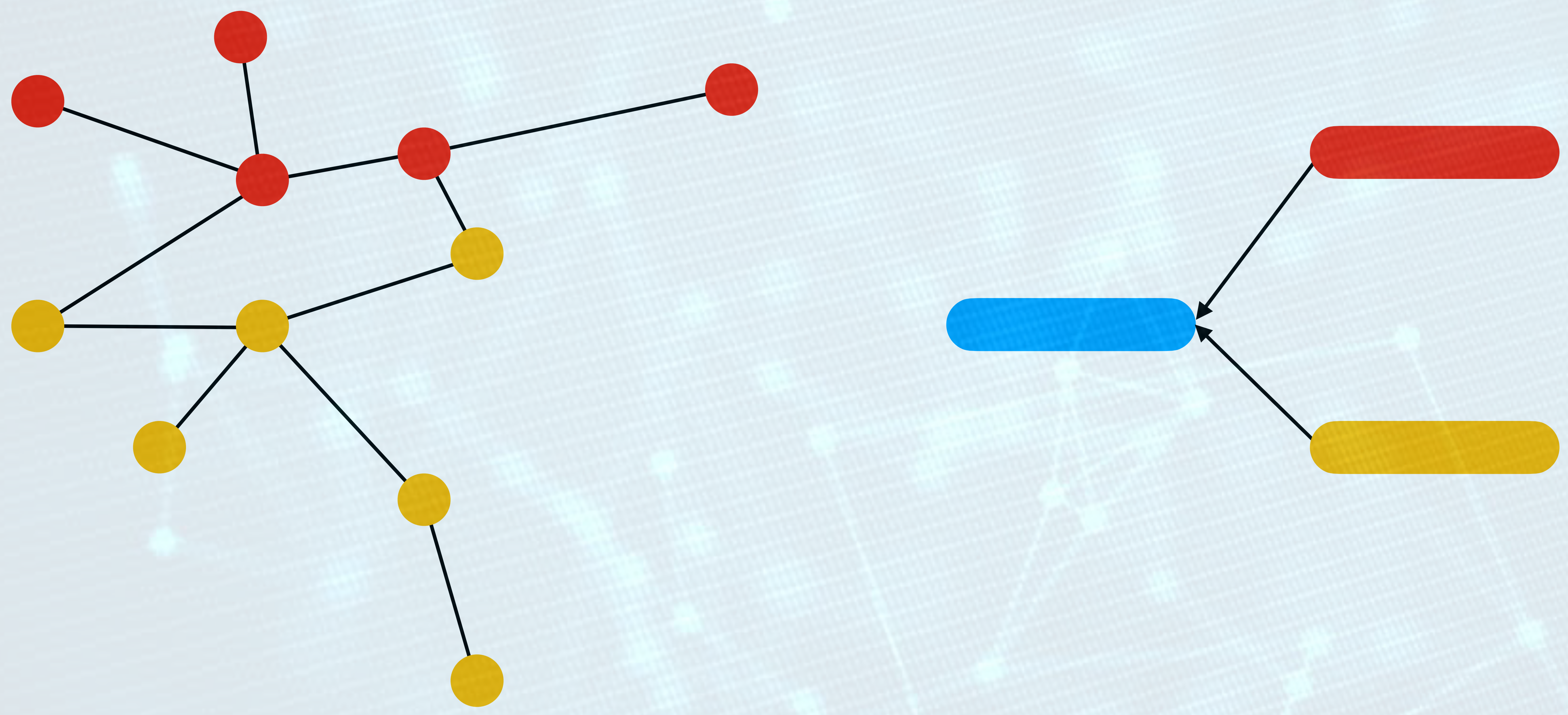
The underlying communication network has latency, i.e. messages take time to travel.



If we produce a block every 5 seconds, then we would have forks within forks within forks etc! Chaos would ensue..

The first scaling bottleneck

The underlying communication network has latency, i.e. messages take time to travel.



This is the first scaling bottleneck: network latency means blocks cannot be produced too quickly without sacrificing security.

The second scaling bottleneck

So long as all (or many) users have to verify all transactions, this severely limits the rate at which they can be processed.

In a decentralised Web 3.0, one couldn’t reasonably have many users verifying all actions of all users!

So scaling solutions dealing with this bottleneck aim to reduce the verification tasks of individual users without sacrificing (too much) security.

The second scaling bottleneck

So long as all (or many) users have to verify all transactions, this severely limits the rate at which they can be processed.

In a decentralised Web 3.0, one couldn’t reasonably have many users verifying all actions of all users!

So scaling solutions dealing with this bottleneck aim to reduce the verification tasks of individual users without sacrificing (too much) security.

Scalability

The problem

The second bottleneck

(all or many nodes verify all transactions)

The first bottleneck

(network latency means blocks can't be produced too fast)

The solutions

Layer 2

(protocols which are implemented on top of the underlying cryptocurrency)

Layer 1

(solutions at the level of the protocol itself)

Layer 0

(underlying infrastructure used by the protocol)

The future for cryptocurrencies?

The human element to all of this makes things especially hard to predict.

The future for cryptocurrencies?

The human element to all of this makes things especially hard to predict.

Can high transaction rates be achieved? Yes! But there are different routes…

Bitcoin remains king

Layer 2

Combination of layers 1 and 2

Stronger solution

The future for cryptocurrencies?

The human element to all of this makes things especially hard to predict.

Can high transaction rates be achieved? Yes! But there are different routes…

Bitcoin remains king

Layer 2

Combination of layers 1 and 2

Stronger solution

Not realistic or interesting to talk of cryptocurrencies replacing fiat currencies in the short term. If the appetite is there, then they will establish new functionalities and roles (e.g. in decentralised finance and web applications).

Thanks for listening