

Building Blockchains With Ethereum

Overview

A blockchain is an immutable, verifiable and distributed ledger of transactions

Ethereum is an open blockchain-based platform which supports Smart Contracts

AWS Blockchain Templates allow us to quickly spin up an Ethereum network on AWS

Smart Contracts are digital contracts which are stored on and interact with the Ethereum blockchain

Ethereum transactions incur a fee in proportion to their computational complexity



Ethereum Basics

Ethereum

Open-source, public blockchain-based platform that supports smart contracts

Smart Contract

Mechanism that allows common contractual clauses to be specified, verified or enforced even in the absence of trust between contracting parties and in the absence of a third party

Ether

Cryptocurrency whose blockchain is generated by Ethereum and is used to compensate nodes on Ethereum for their participation

Ethereum Virtual Machine (EVM)

Execution environment that all Ethereum nodes run

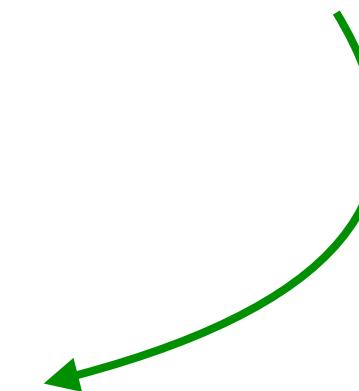
Blockchain

A blockchain is a growing list of records (called blocks) which are linked cryptographically

Blockchain

A blockchain is a growing list of records (called blocks) which are linked cryptographically

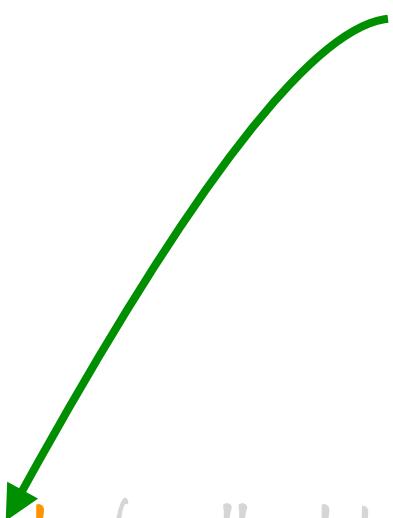
Each record is
called a block



Blockchain

A blockchain is a growing **list of records** (called blocks) which are linked cryptographically

Ledger == list of records



Blockchain

A blockchain is a **growing list** of records (called blocks) which are linked cryptographically

This ledger is open (not closed)

Blockchain

A blockchain is a **growing list** of records (called blocks) which are linked cryptographically

This ledger is open to every node on peer-to-peer network

Blockchain

A blockchain is a **growing list** of records (called blocks) which are linked cryptographically

Any node on that network can add records

Blockchain

A blockchain is a **growing list** of records (called blocks) which are linked cryptographically

Ledger is distributed (not centralized)

Blockchain

A blockchain is a **growing list** of records (called blocks) which are linked cryptographically

The list is not stored in its entirety on any one node of that peer-to-peer network

Blockchain

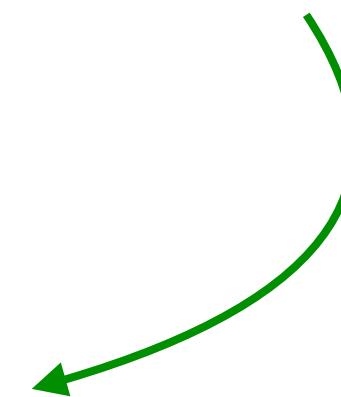
A blockchain is a growing list of records (called blocks) which are linked cryptographically

“Open, distributed ledger”

Blockchain

A blockchain is a growing list of records (called blocks) which are linked cryptographically

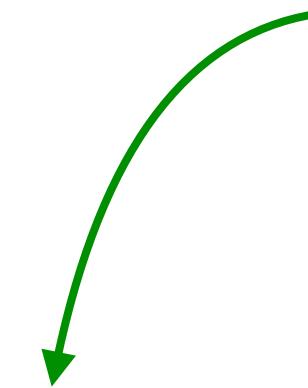
Each record is called
a block



Blockchain

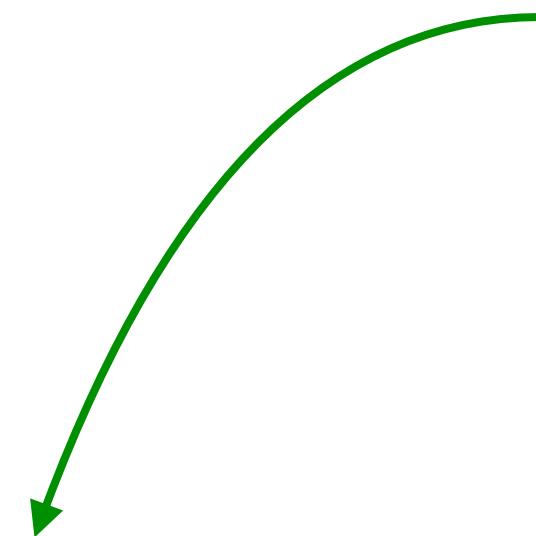
A blockchain is a growing list of records (called blocks) which are linked cryptographically

Each record contains
transactions



Blockchain

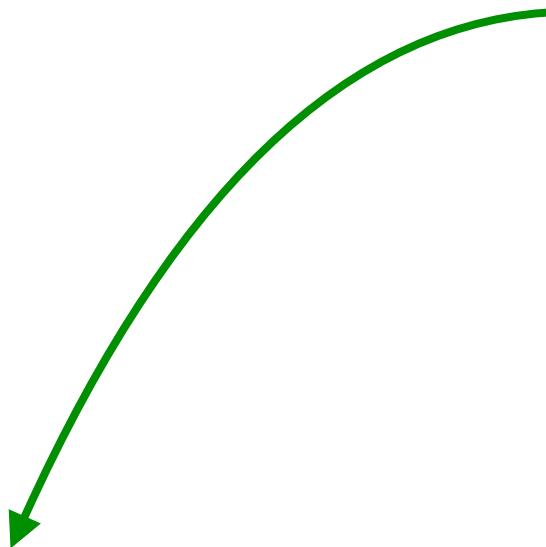
A blockchain is a **growing list** of records (called blocks) which are linked cryptographically



The list is not stored in its entirety on any one node of that peer-to-peer network

Blockchain

A blockchain is a **growing list** of records (called blocks) which are linked cryptographically



However state of list can always be recovered from the network as a whole

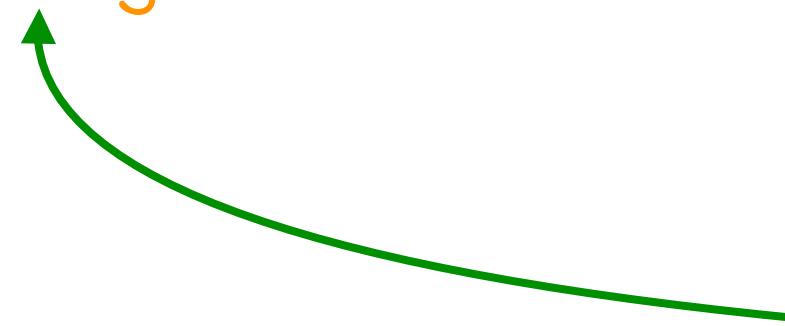
Blockchain

A blockchain is a **growing list** of records (called blocks) which are linked cryptographically

Transactions are permanent

Blockchain

A blockchain is a growing list of records (called blocks) which are **linked cryptographically**



Each block “knows about” all previous blocks

Blockchain

A blockchain is a growing list of records (called blocks) which are **linked cryptographically**



Can not unilaterally change
transaction after-the-fact

Blockchain

A blockchain is a growing list of records (called blocks) which are linked cryptographically



Transactions are verifiable

Verifiable and Permanent Transactions

Cryptographic Link



Each block contains

- cryptographic hash of previous block
- timestamp
- transaction data

Cryptographic Link



Transaction data in that block can not be altered after-the-fact

- except by altering all subsequent blocks
- which is complicated and needs consensus

Cryptographic Link



Each transaction is

- verifiable
- permanent

Blockchain

A blockchain is a growing list of records (called blocks) which are linked cryptographically

Blockchain

A blockchain is a growing list of records (called blocks) which are linked cryptographically

Blockchain

An open, distributed ledger that can record transactions in a verifiable and permanent manner

Blockchain

An open, distributed ledger that can record transactions in a verifiable and permanent manner

Blockchain

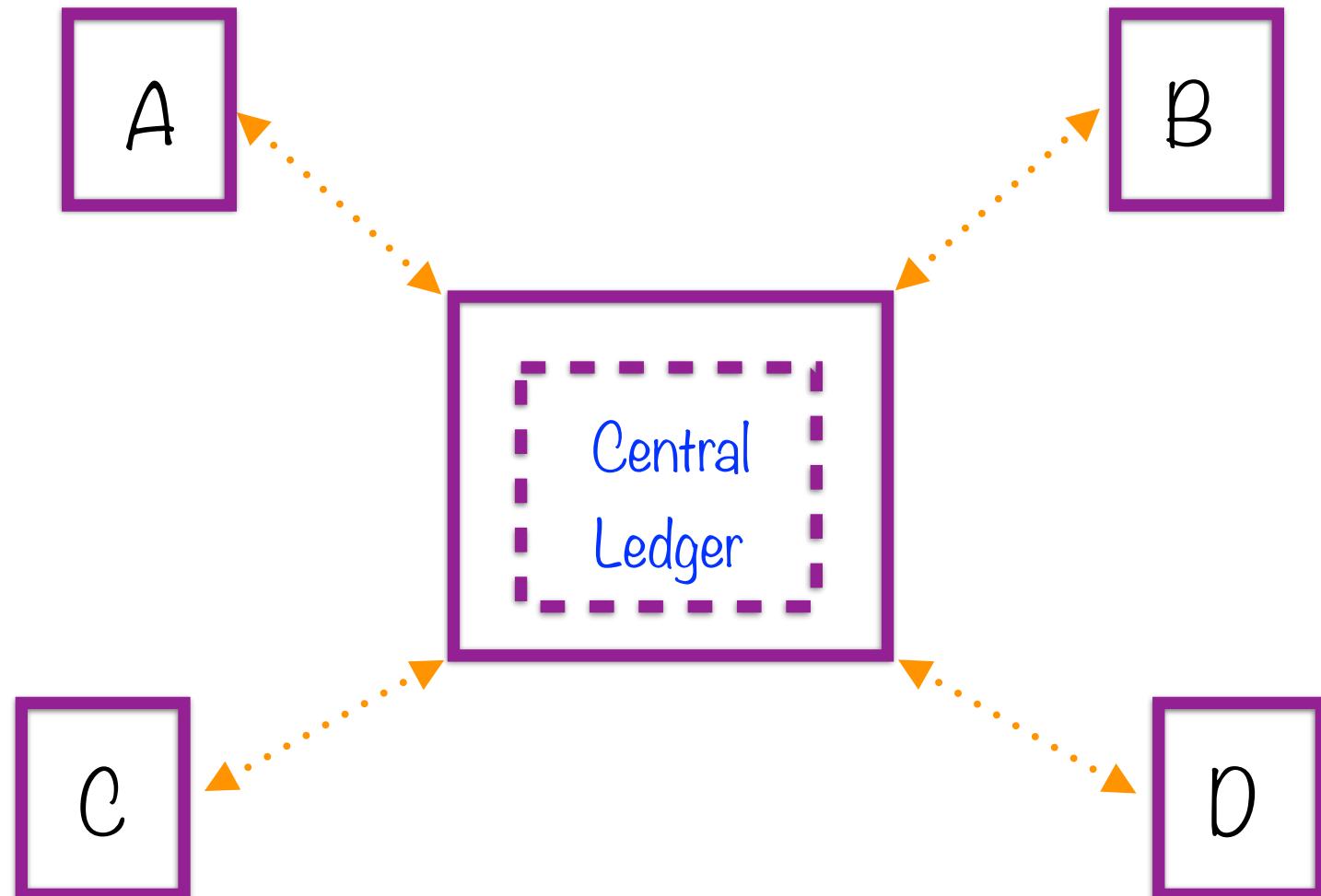
An open, distributed ledger that can record transactions in a verifiable and permanent manner

Blockchain

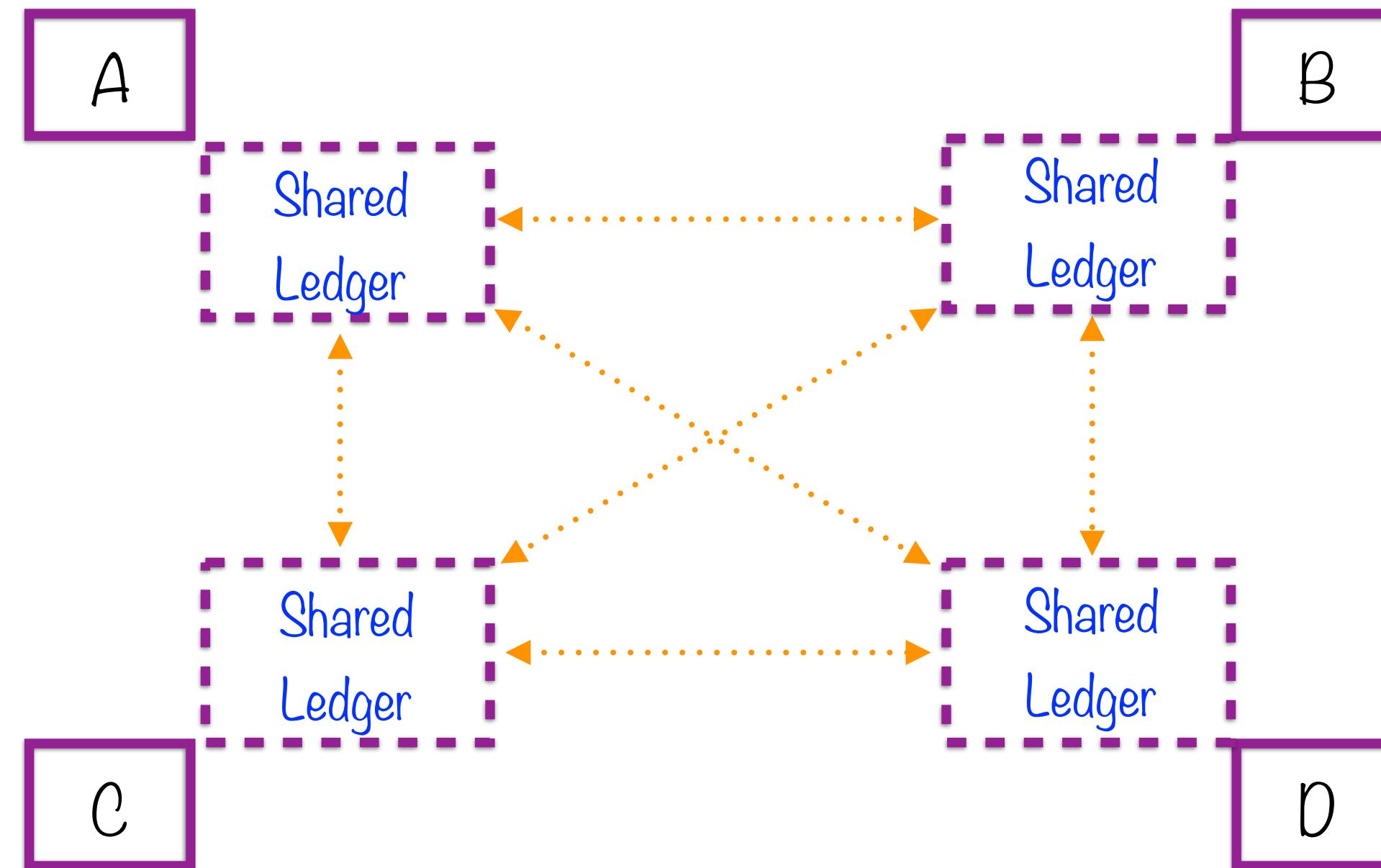
An open, distributed ledger that can record transactions in a verifiable and permanent manner

Accounts and Nodes

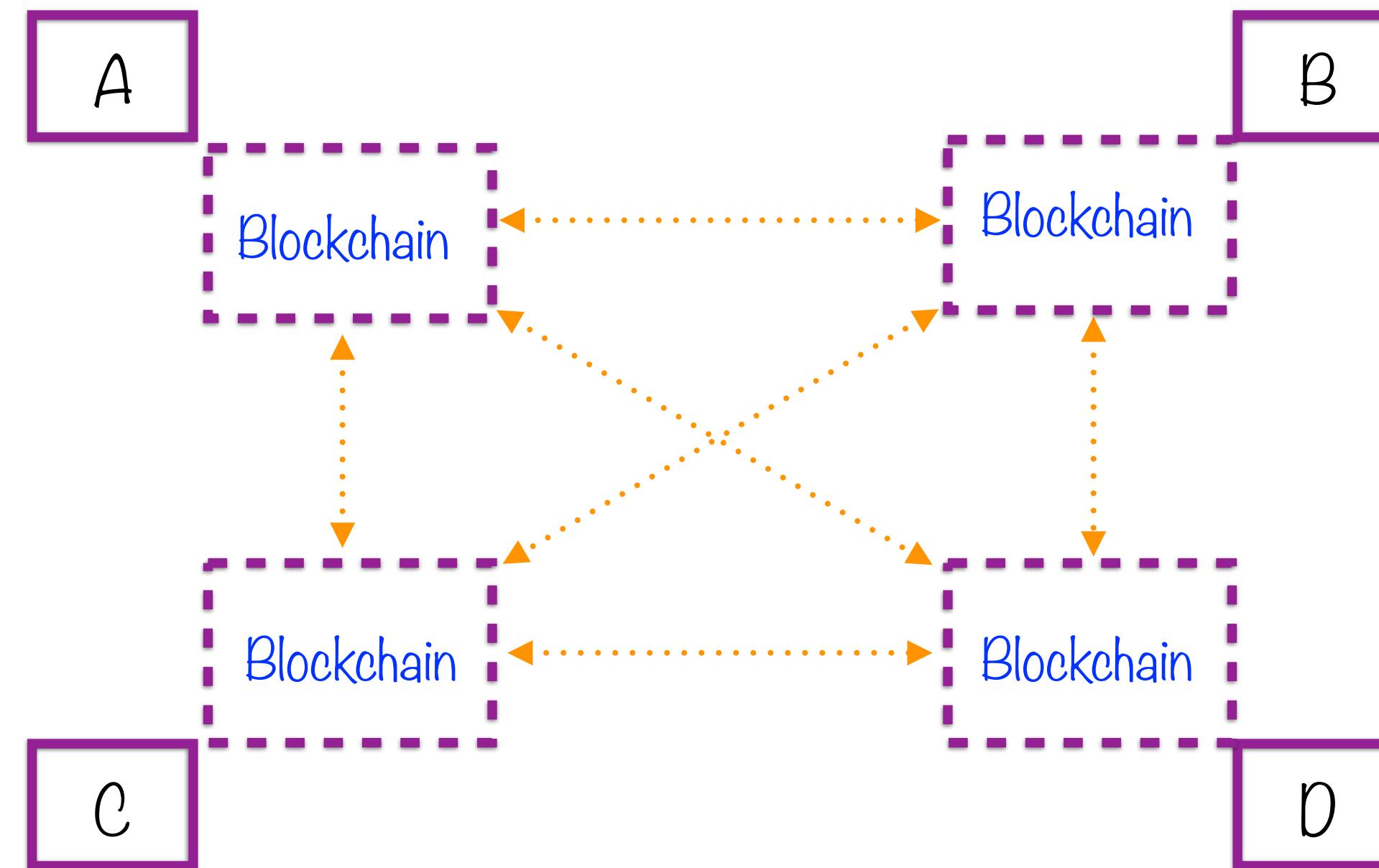
Centralized Ledger



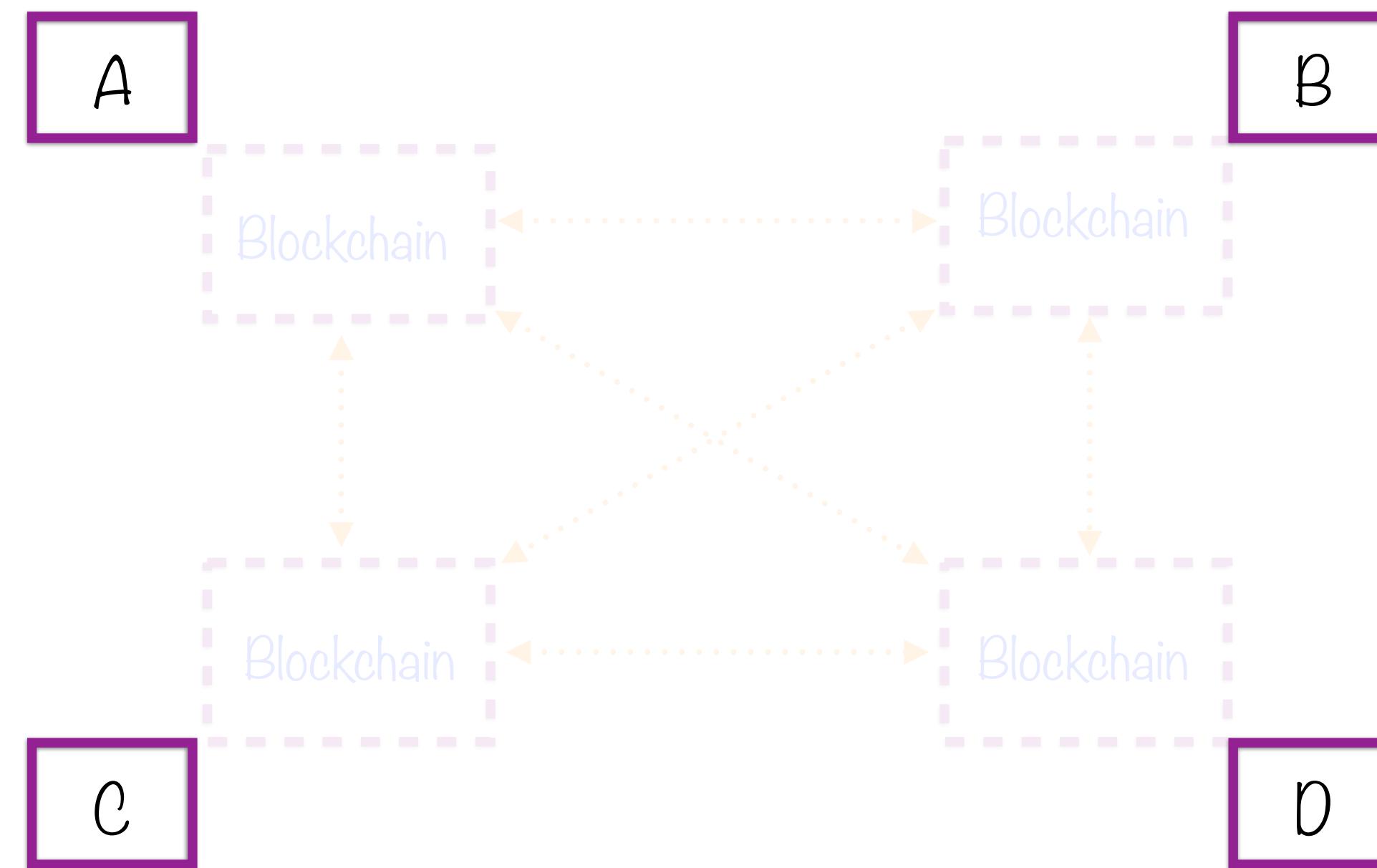
Distributed Ledger



Blockchain Network



Nodes in a Blockchain Network



Nodes



Each node runs the Ethereum Virtual Machine (EVM)

EVM is runtime environment responsible for

- Executing contract code
- Calculating transaction complexity (gas consumption)
- Verifying transactions

Nodes



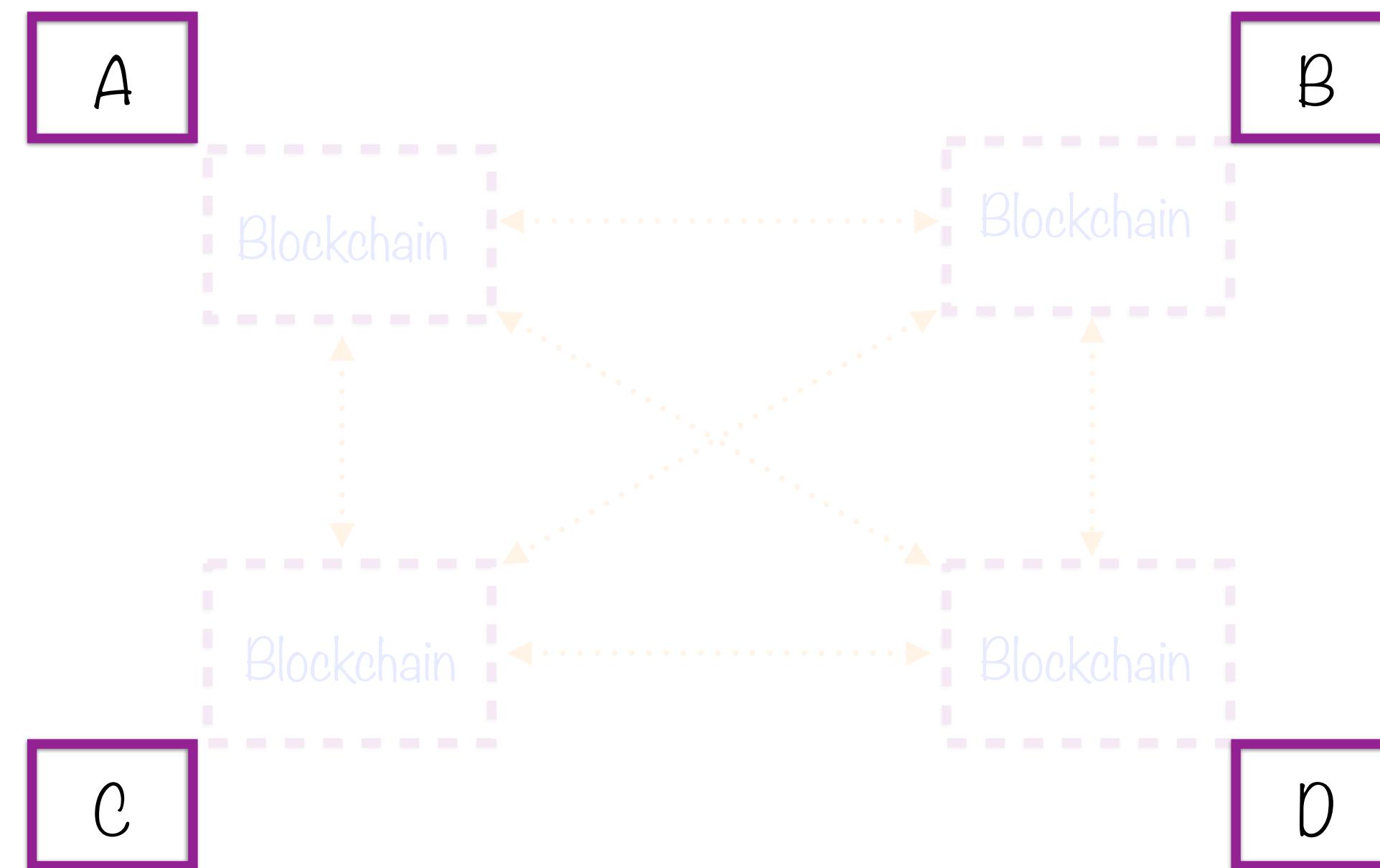
Each node has a key pair

- Private key: used to sign (encrypt)
- Public key: used to verify (decrypt)

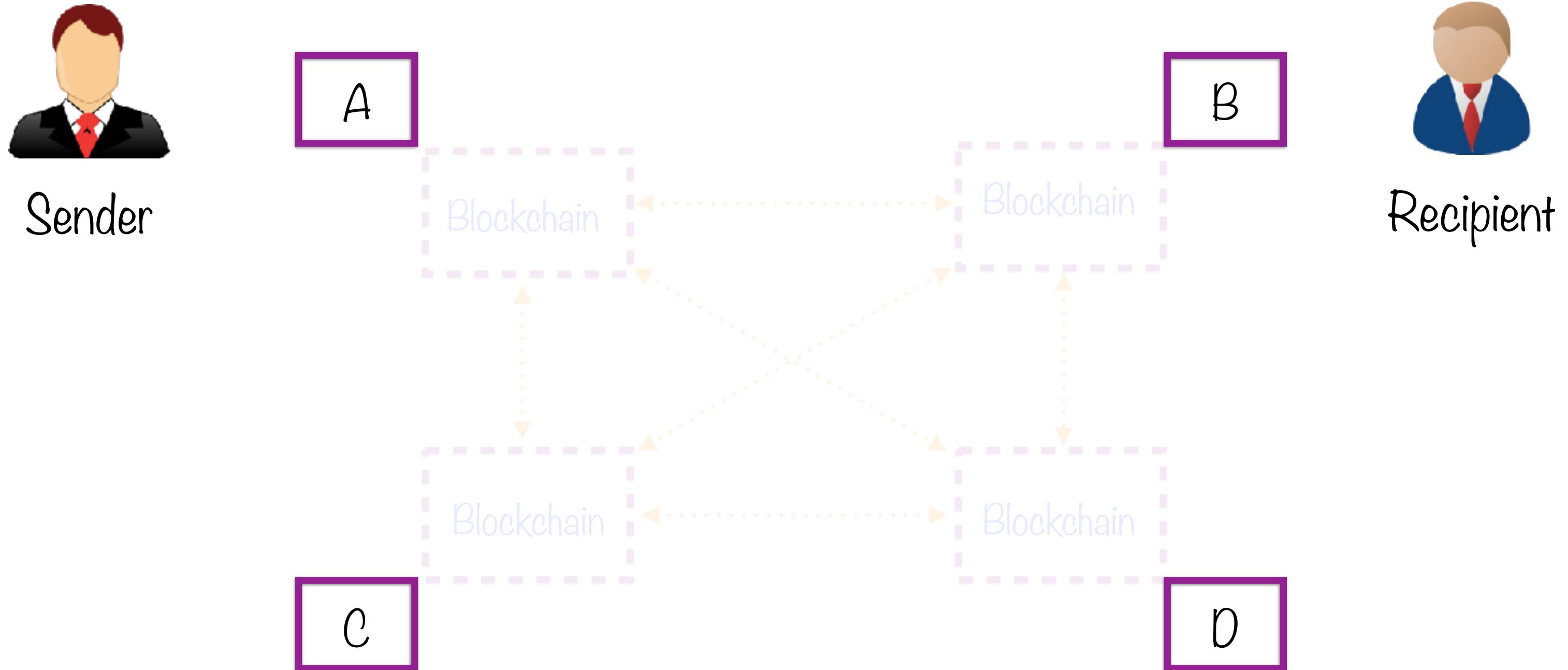
Private key of each node is private

Public key of each node is public

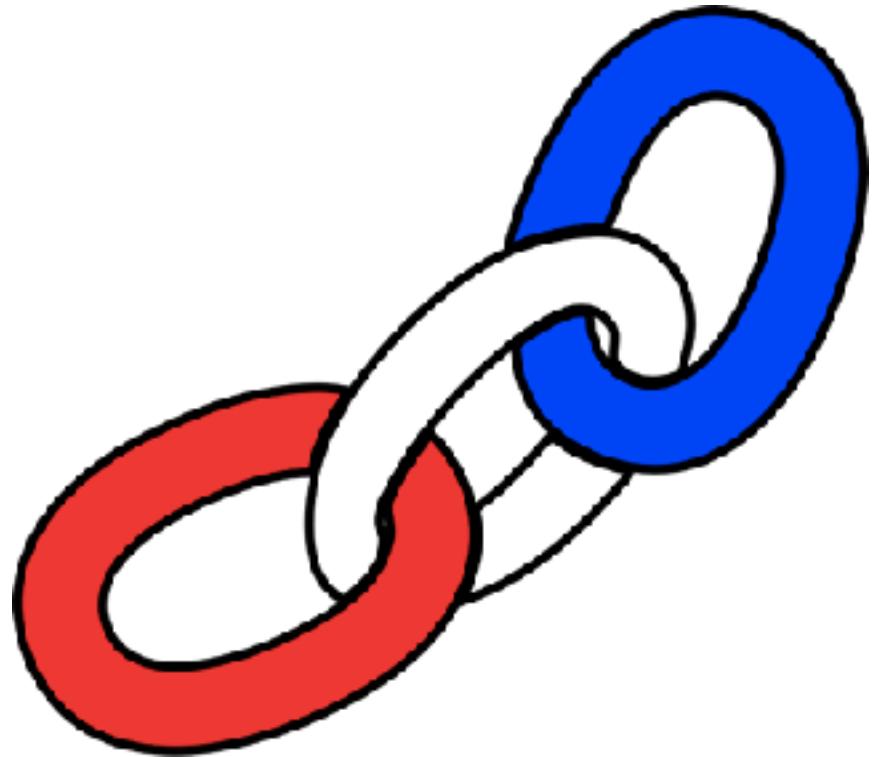
Nodes in a Blockchain Network



Accounts in a Blockchain Network



Accounts



Transactions are initiated from accounts

Accounts are of two types

- Externally owned accounts (EOAs)
- Contract accounts

EOAs



Possess an Ether balance

Can fire transactions

Possess a key pair (private and public keys)

No associated code

Contract Accounts



Possess an Ether balance

Can fire transactions

Associated code which is triggered by

- transactions
- messages sent by other contracts

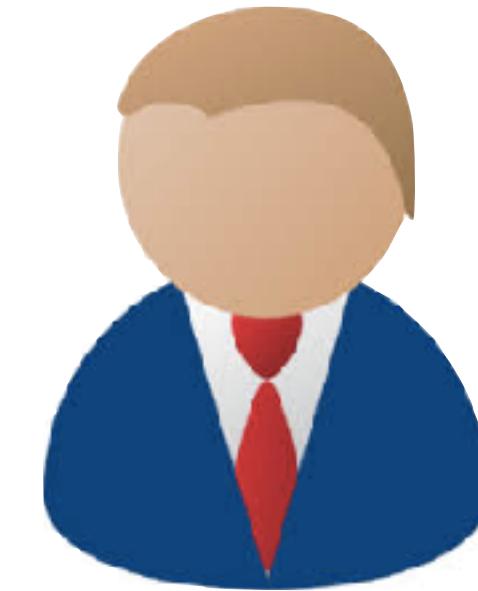
Perform operations when executed

Transactions

Transactions



A
Sender



B
Recipient



Messages sent by EOAs to each other to transfer value

Transactions



Transactions are interactions between accounts

Messages sent from EOA to any other account
on the network

Transactions

Contents of transaction

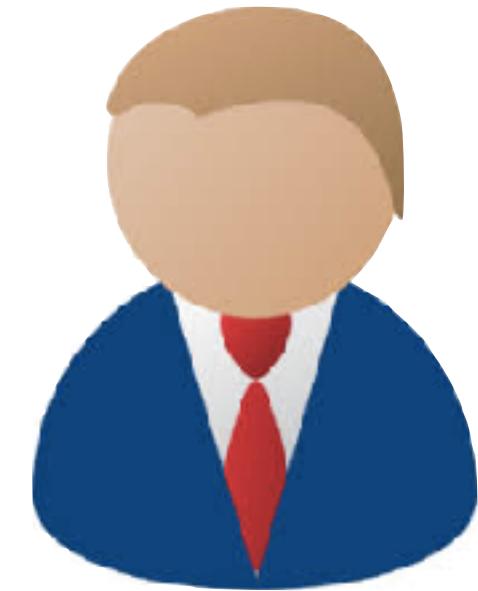


- recipient
- signature of sender
- value of assets to send to recipient
- gas price: incentive for transaction to be processed (in units of ether)
- gas limit: hard cap on that incentive

Transactions



A
Sender

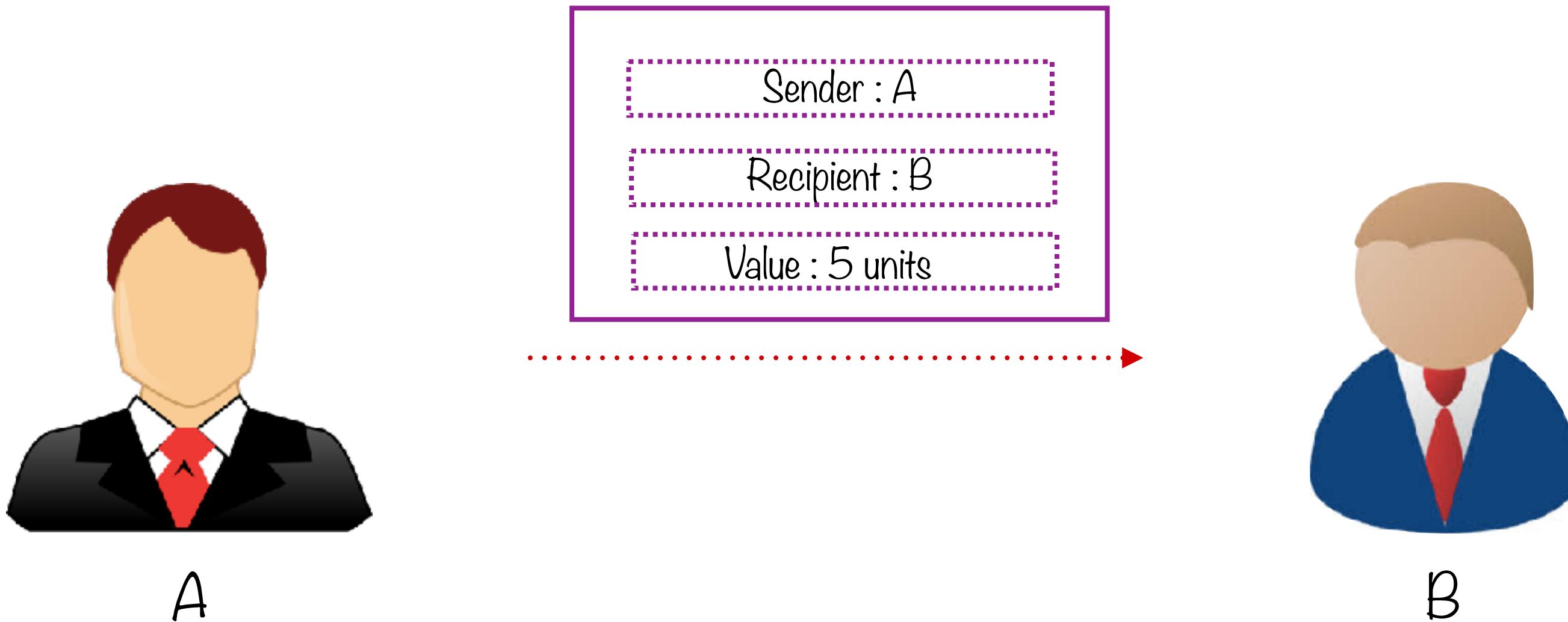


B
Recipient



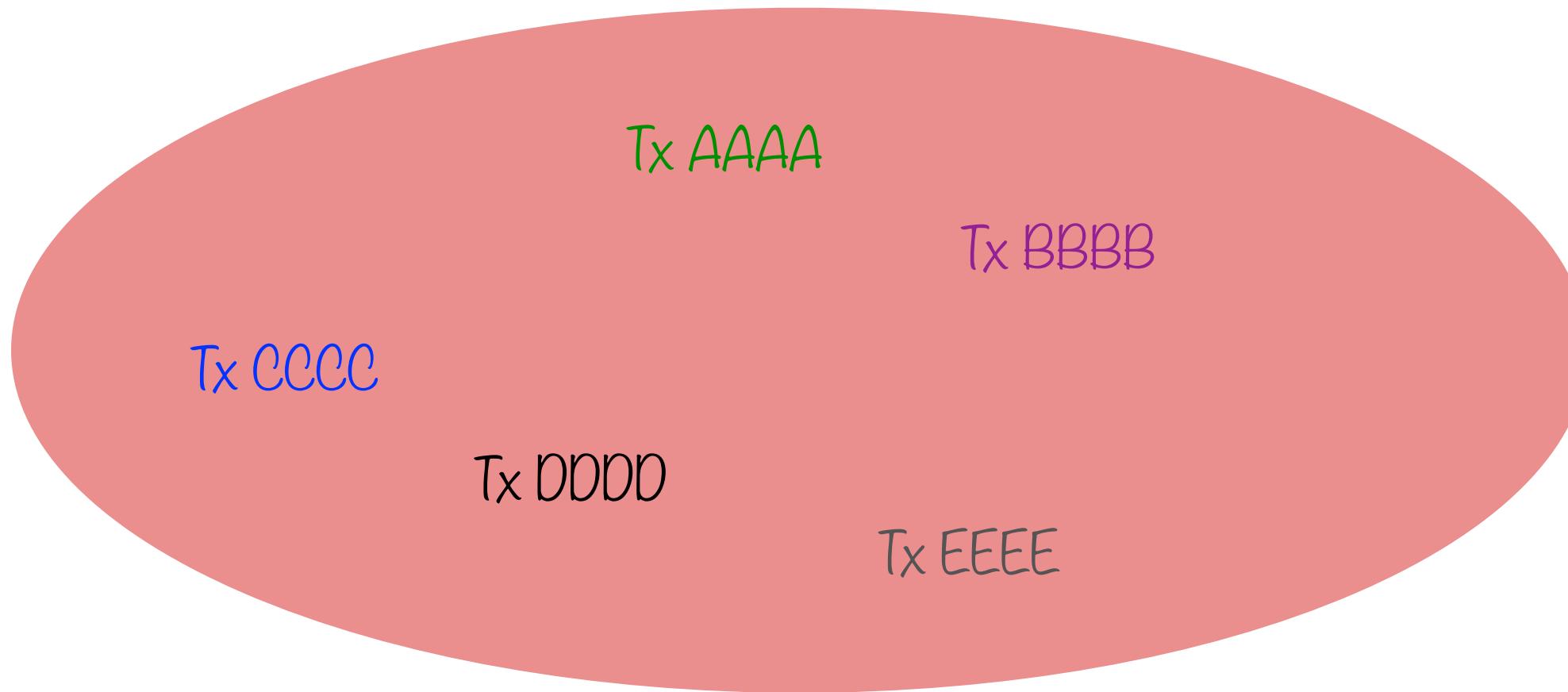
These transactions need to be verified and reflected in the state of all nodes in the network

Transaction Verification



Initiated transaction requests are broadcast to the entire blockchain network

Unverified Transaction Pool

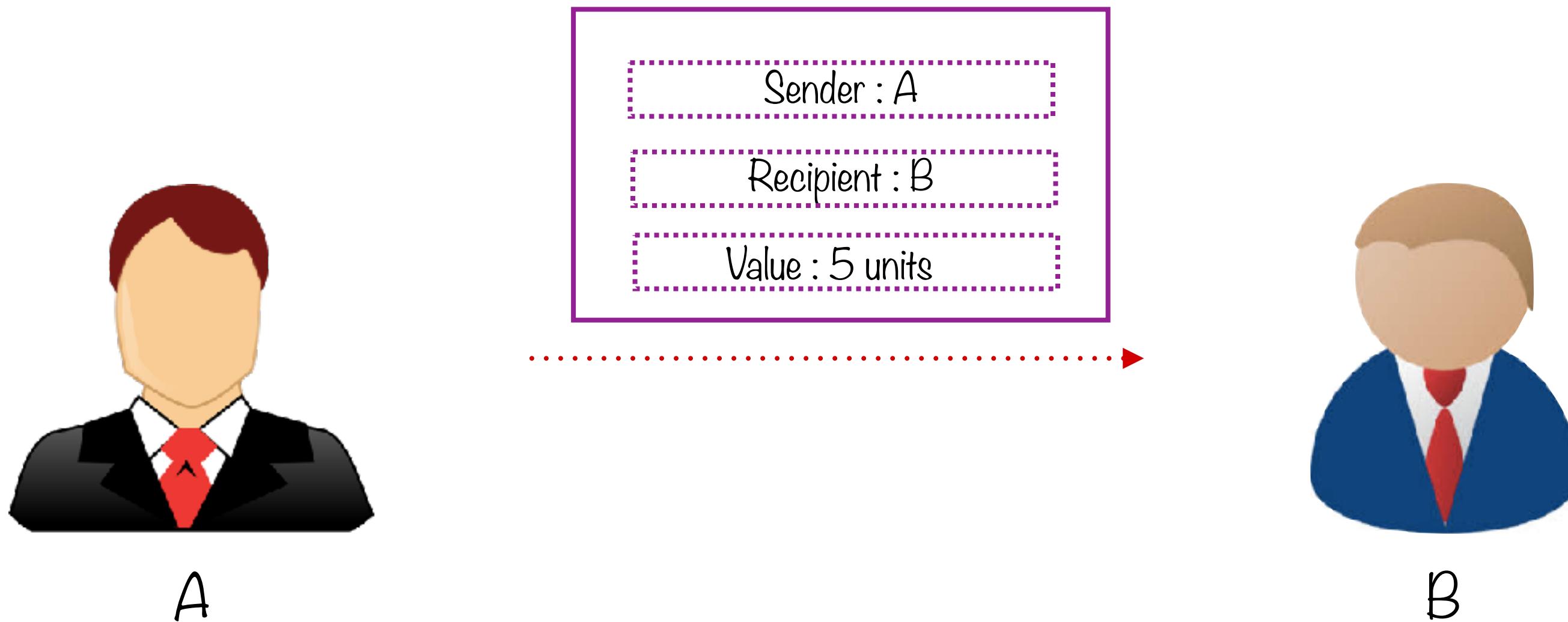


The initiated transactions are entered into a pool of unverified transactions

The process of verifying transactions is called mining and the participants are called miners

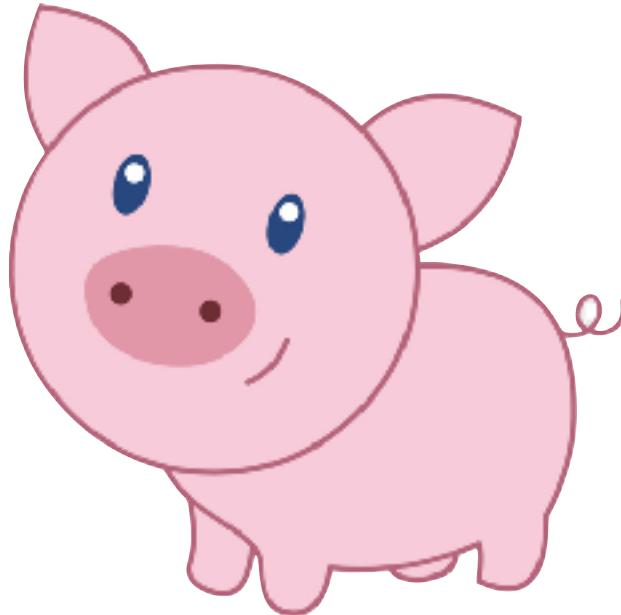
Miners and Gas

Transaction Verification



In order to incentivize miners to accept a transaction, the initiator must pay a fee

Gas

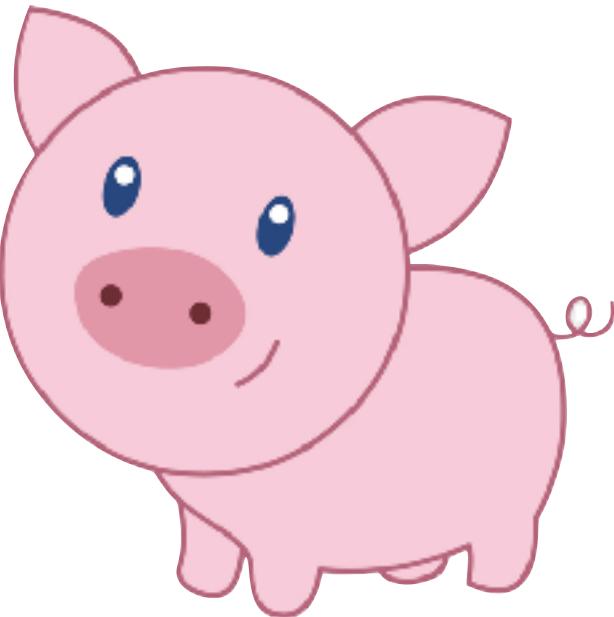


Computation work need to verify a transaction is called **Gas**

Each transaction contains the **Gas Price** that the initiator is willing to pay

Each transaction also contains a hard absolute **Gas Limit**

Gas



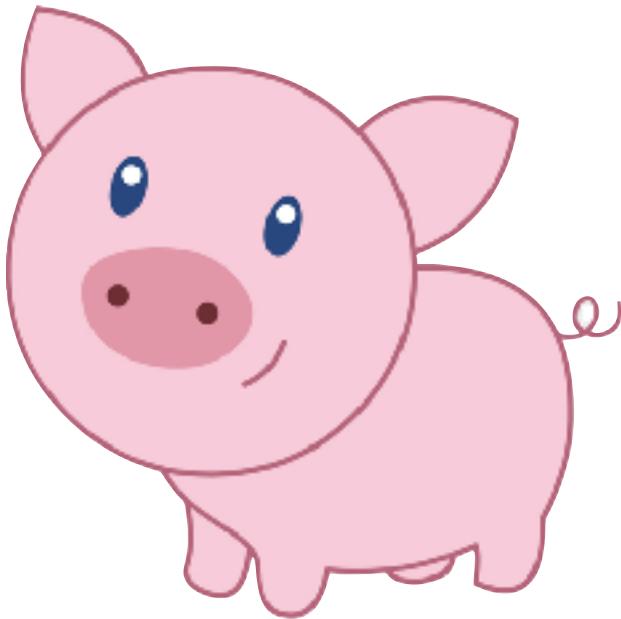
Every node must be running the Ethereum Virtual Machine (EVM)

EVM is a common runtime environment

Runs during verification protocol

EVM measures gas consumption

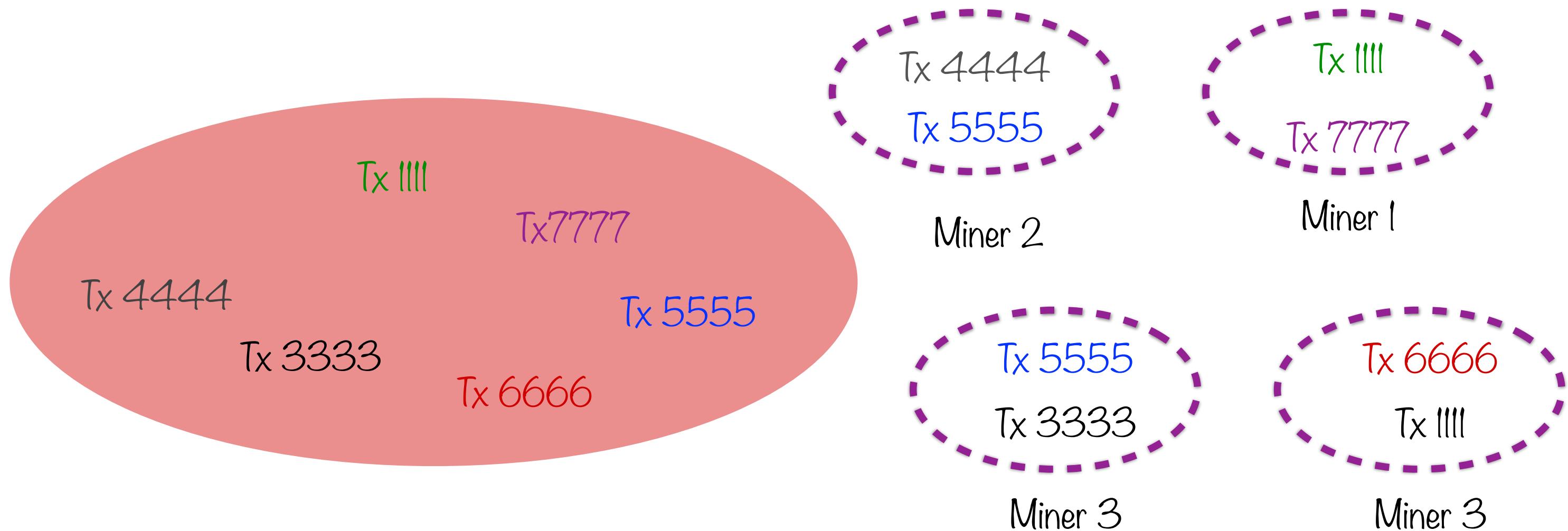
Miner's Prize



TotalCost = gasUsed \times gasPrice

The miner that verifies a transaction collects this
prize

Unverified Transaction Pool



Miners add the transactions to their blocks to collect the gas fee

Transaction Verification



- Verifying sender and transaction integrity
- Confirm that sender owns assets to send

Transaction Verification



Ethereum is a distributed system

Nodes need to converge (agree) on

- consistency of transactions
- order of transactions

Consensus Algorithms



Need to achieve consensus in the absence of trust

Trustless consensus algorithms

- proof-of-work
- proof-of-stake

Cryptographic Hashing in Blockchain

Blockchain

A blockchain is a growing list of records (called blocks) which are linked cryptographically

Blockchain

A blockchain is a growing list of records (called blocks) which are linked cryptographically

Blockchain

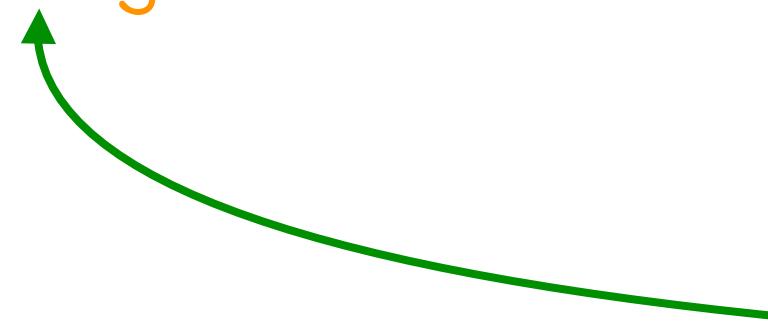
An open, distributed ledger that can record transactions in a verifiable and permanent manner

Blockchain

An open, distributed ledger that can record transactions in a verifiable and permanent manner

Blockchain

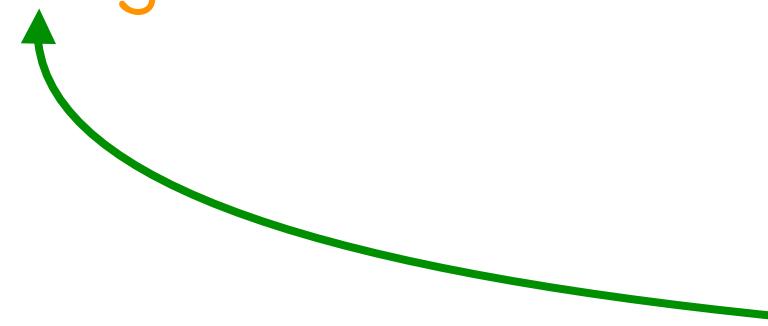
A blockchain is a growing list of records (called blocks) which are **linked cryptographically**



Each block “knows about” all previous blocks

Blockchain

A blockchain is a growing list of records (called blocks) which are **linked cryptographically**



Can not unilaterally change
transaction after-the-fact

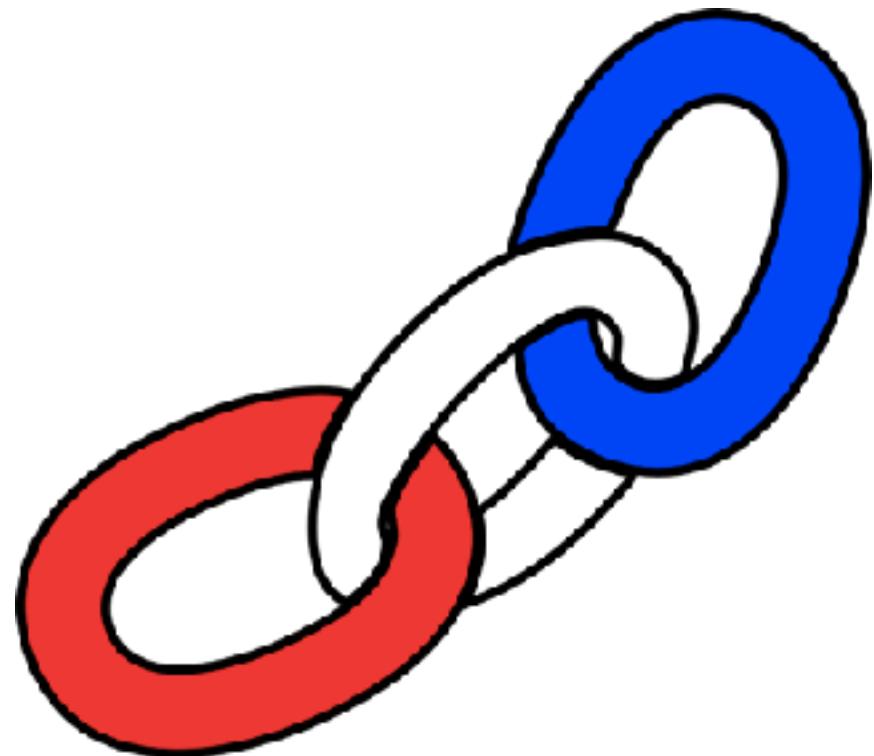
Blockchain

A blockchain is a growing list of records (called blocks) which are linked cryptographically



Transactions are verifiable

Cryptographic Link



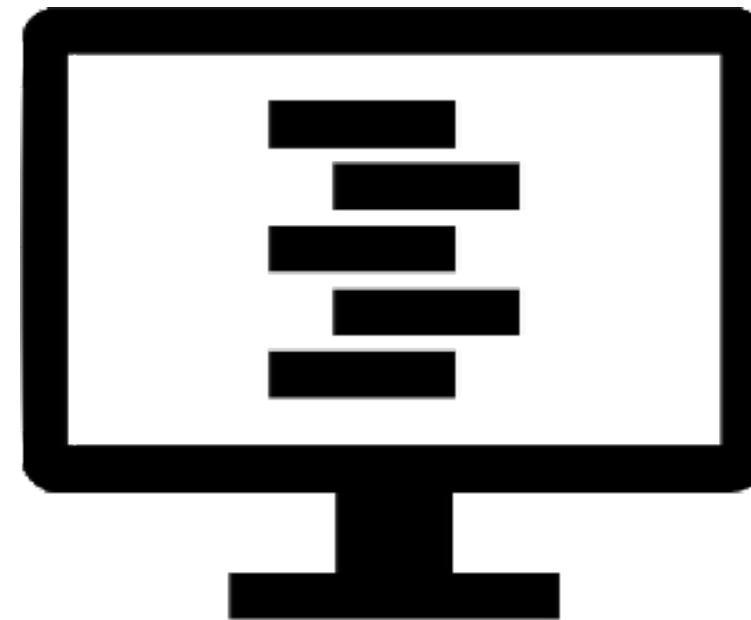
Each block contains

- cryptographic hash of previous block
- timestamp
- transaction data

Hash Algorithm



Input message



Hash Function



Hash value

Hash Algorithm



Hash Value



Also known as

- message digest
- digest
- digital fingerprint
- checksum

Hash Function



Ideal hash function properties

- Easy to compute hash value given message
- Low probability of hash collisions
 - hash collision: two different messages have same hash value

Cryptographic Hash Function



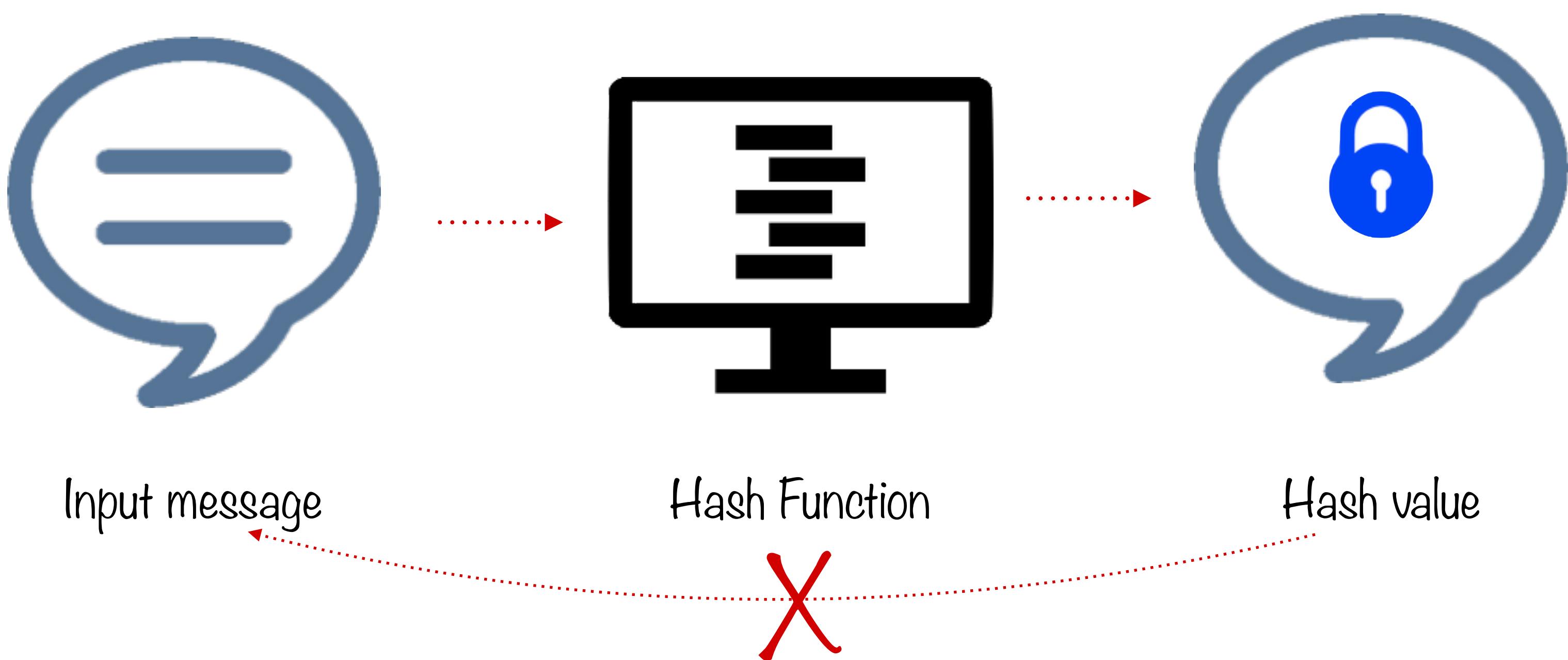
One additional property

- Easy to compute hash value given message
- Low probability of hash collisions
- Impossible for malicious entity to recreate message given only the hash value

Cryptographic Hash

Hash function in which it is impossible to recreate message given only the hash value

Cryptographic Hash Algorithm



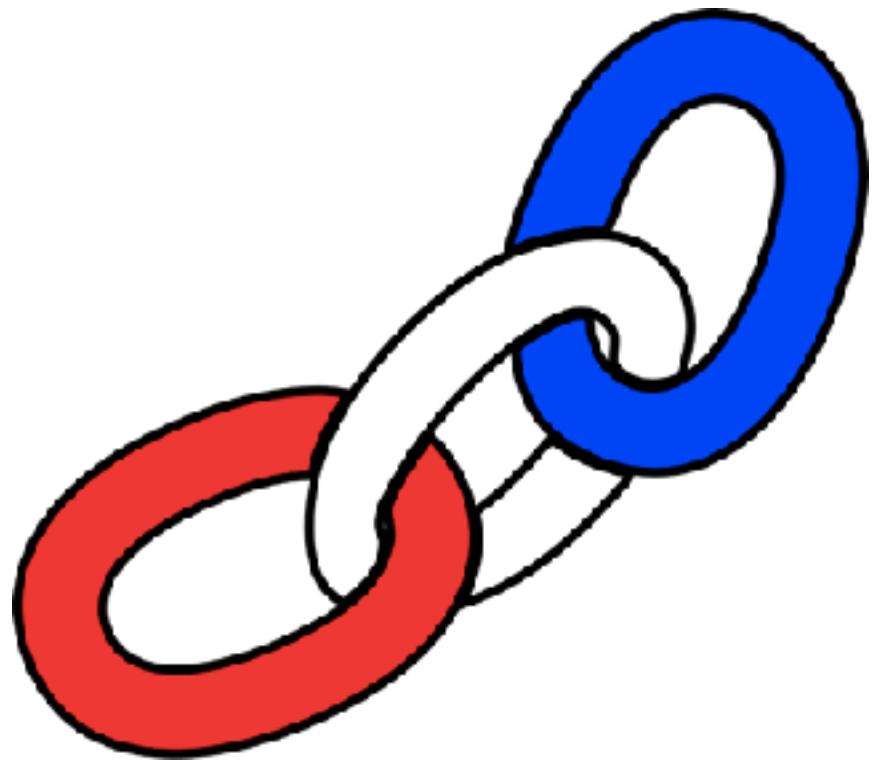
Cryptographic Hash



Each block contains

- cryptographic hash of previous block
- timestamp
- transaction data

SHA-256



SHA-2: Secure Hash Algorithm

Designed by US NSA

SHA-256 belongs to this family

256 = number of bits in digest

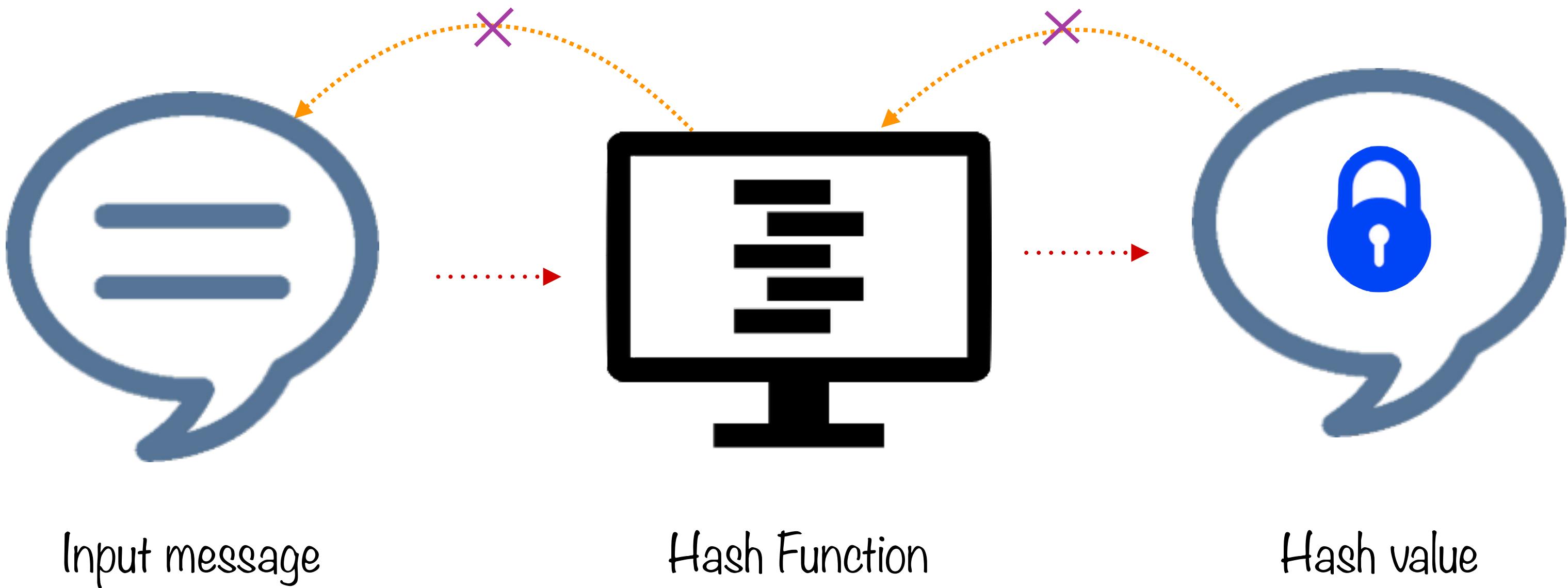
SHA256 Hash Generator

Rose



E10D850954A8E6113899EDA0B14AA4706CAAA04A73A804A693360816F6360C8D

Secure non reversible function



SHA256 Hash Generator

Rose



E10D850954A8E6113899EDA0B14AA4706CAAA04A73A804A693360816F636008D

SHA256 Hash Generator

E10D850954A8E6113899EDA0B14AA4706CAAA04A73A804A693360816F6360C8D



I2A32E1422274ABE08DA2FD6F91C72F9562236CD1219AA74BEFD932987EC4C22

SHA256 Hash Generator

Rose



E10D850954A8E6113899EDA0B14AA4706CAAA04A73A804A693360816F636008D

SHA256 Hash Generator

Rose.

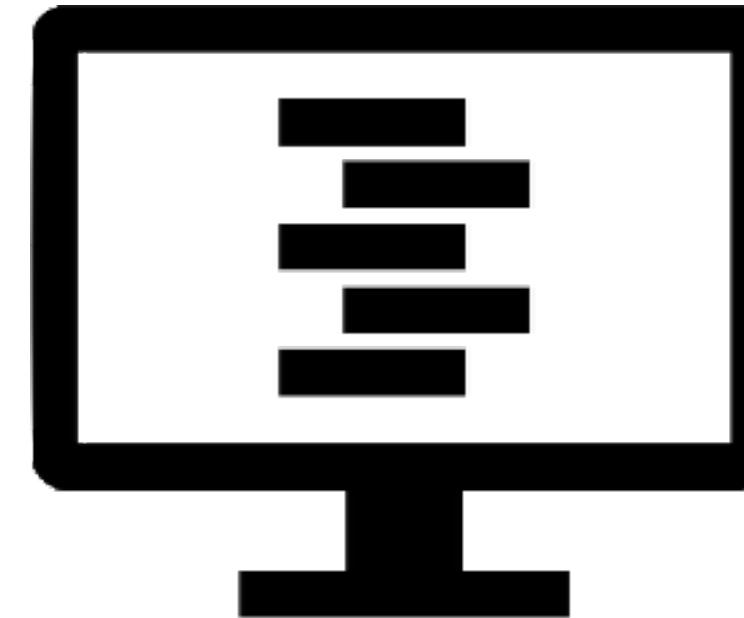


077B18A8C344516182D8B939D70457CFF0239FE1ED77707EDBB9ACF6AD4C50BB

SHA256 Hash Generator



Input message



Hash Function

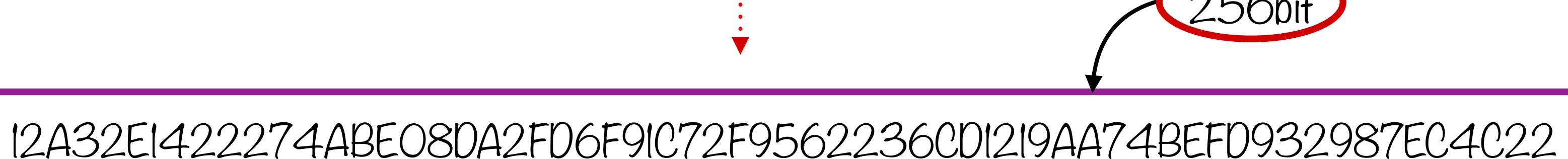


Hash value

Size of hash value is always 256 bits (independent of file size)

SHA256 Hash Generator

If we give a big paragraph or even a book as an input the hashed file will always be in 256bit



Chains of Blocks

Blockchain

A blockchain is a growing list of records (called blocks) which are linked cryptographically

Blockchain

A blockchain is a **growing list** of records (called blocks) which are linked cryptographically

Ledger is distributed (not centralized)

Blockchain

A blockchain is a **growing list** of records (called blocks) which are linked cryptographically

The list is not stored in its entirety on any one node of that peer-to-peer network

Blockchain

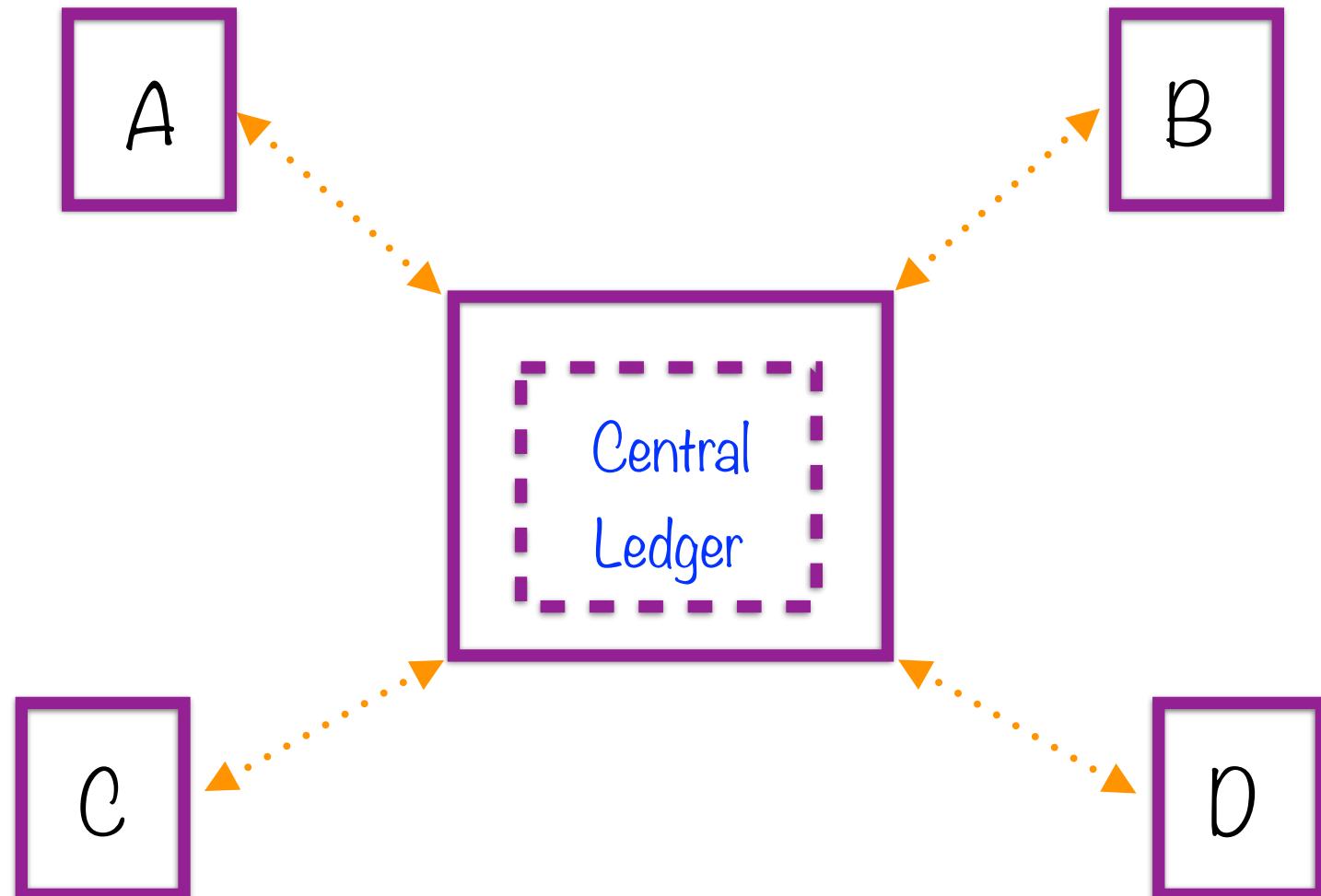
A blockchain is a growing list of records (called blocks) which are linked cryptographically

“Open, distributed ledger”

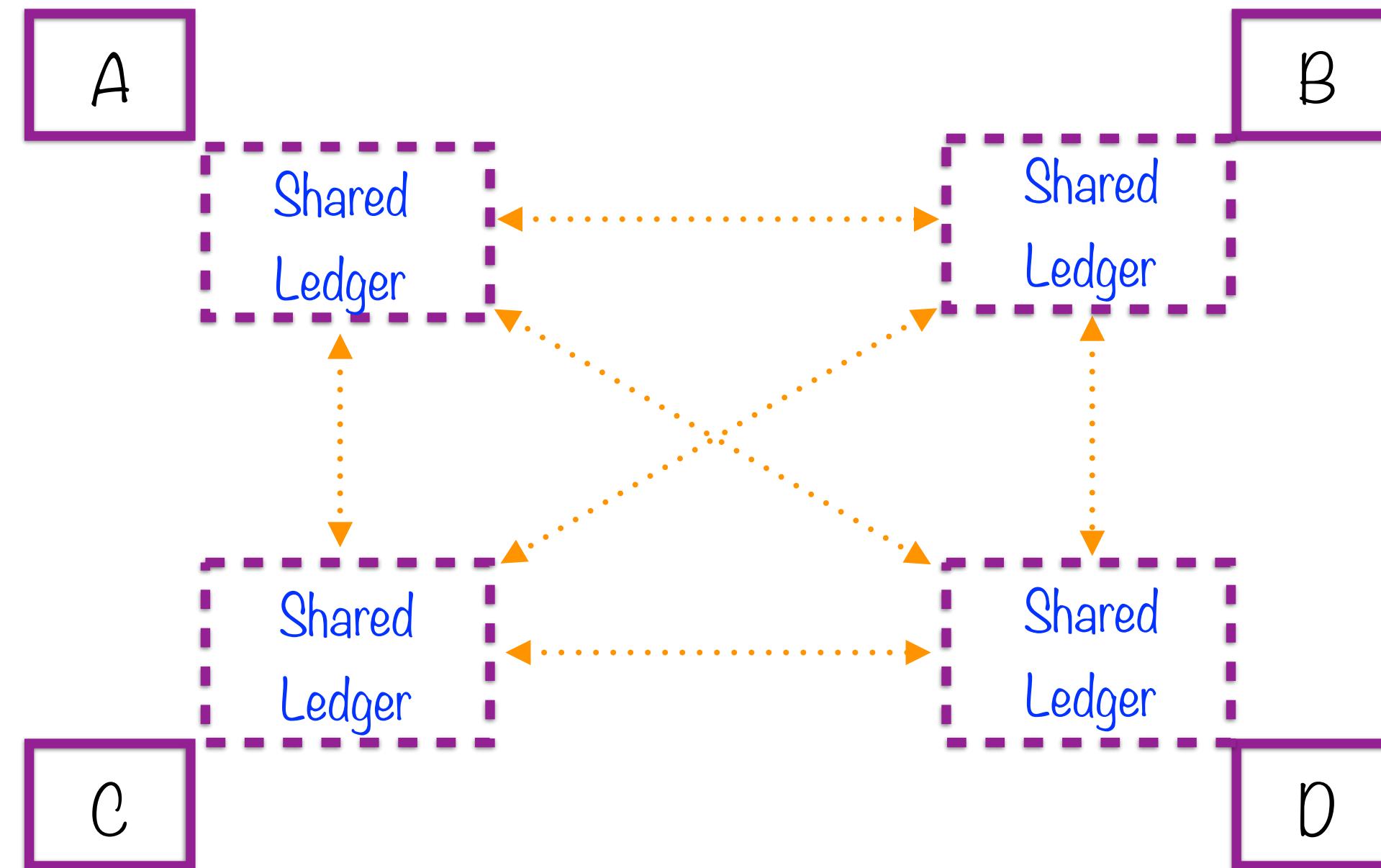
Ledger

List of records, each corresponding to a transaction

Centralized Ledger



Distributed Ledger



Distributed Ledger

A's Ledger		B's Ledger		C's Ledger	
	Transaction amount		Transaction amount		Transaction amount
1st → A → B	10.5	A → B	10.5	A → B	10.5
B → C	8.5	B → C	8.5	B → C	8.5
10 ⁶ th → X → Y	50	B → Y	35	C → Z	45

Everyone in the network shares the same ledger

Distributed Ledger

A's Ledger		B's Ledger		C's Ledger	
	Transaction amount		Transaction amount		Transaction amount
1st → A → B	10.5	A → B	10.5	A → B	10.5
B → C	8.5	B → C	8.5	B → C	8.5
10 ⁶ th → X → Y	50	B → Y	35	C → Z	45

Need to check: Consistency and ordering

Distributed Ledger

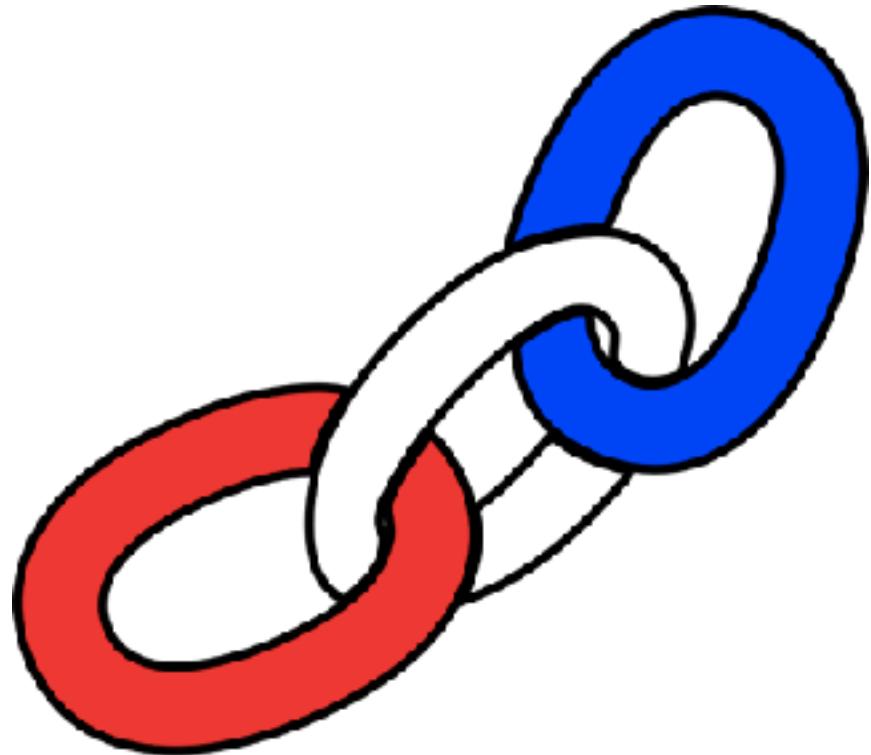


No single point-of-failure

However, need to ensure

- consistency
- ordering

Distributed Ledger



Use hashing to ensure consistency

Each node stores hash of entire ledger

Distributed Ledger



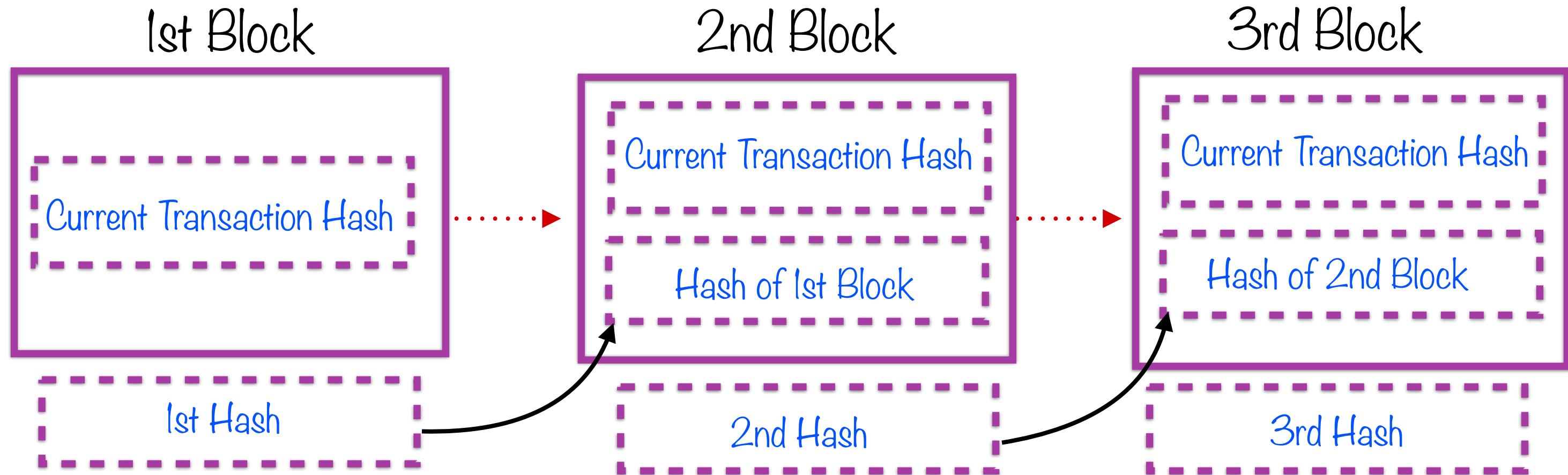
Include hash of each record while computing hash
of next record

Divide ledger into **blocks** to simplify

Ledger is now a **chain of blocks**

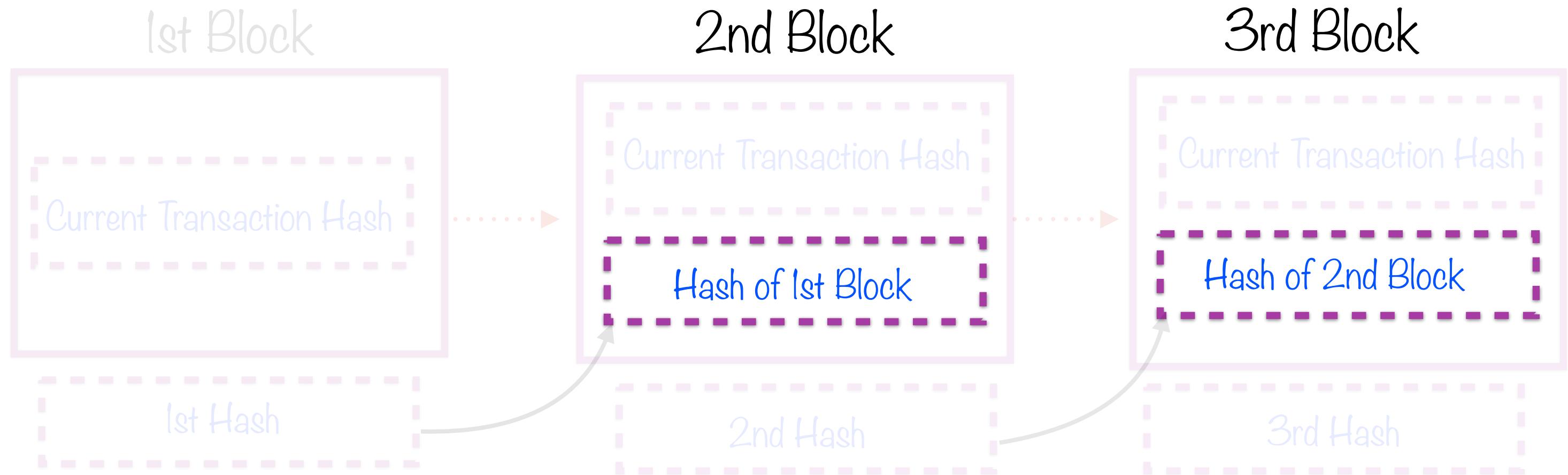
Hence, the name “blockchain”

Chain of Blocks



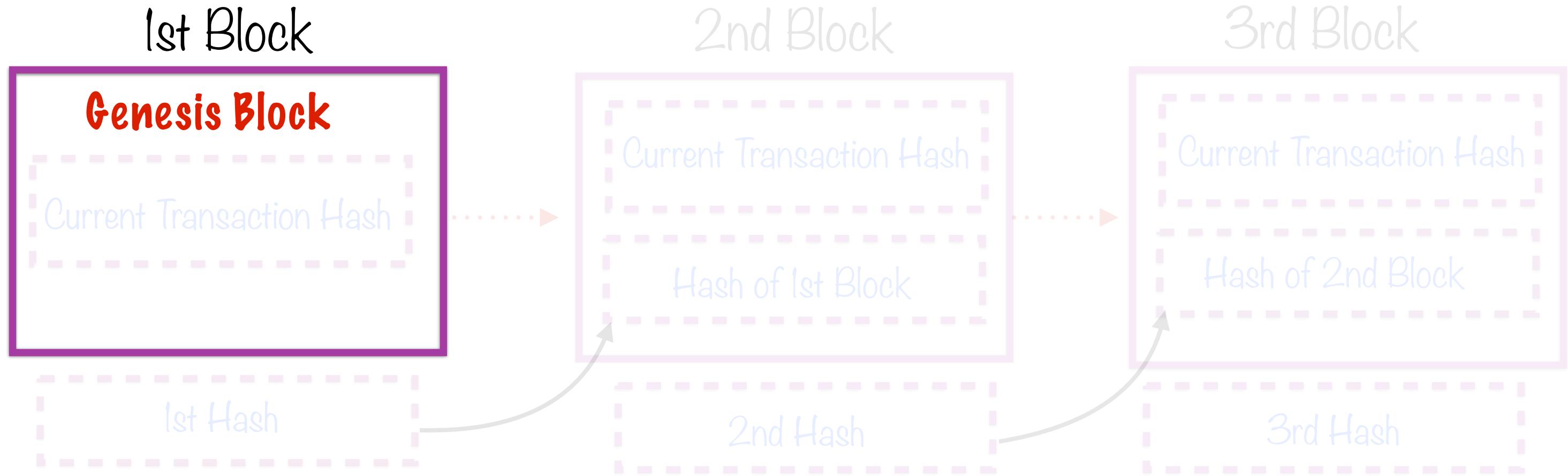
Each block contains hash of the preceding one in chain

Chain of Blocks



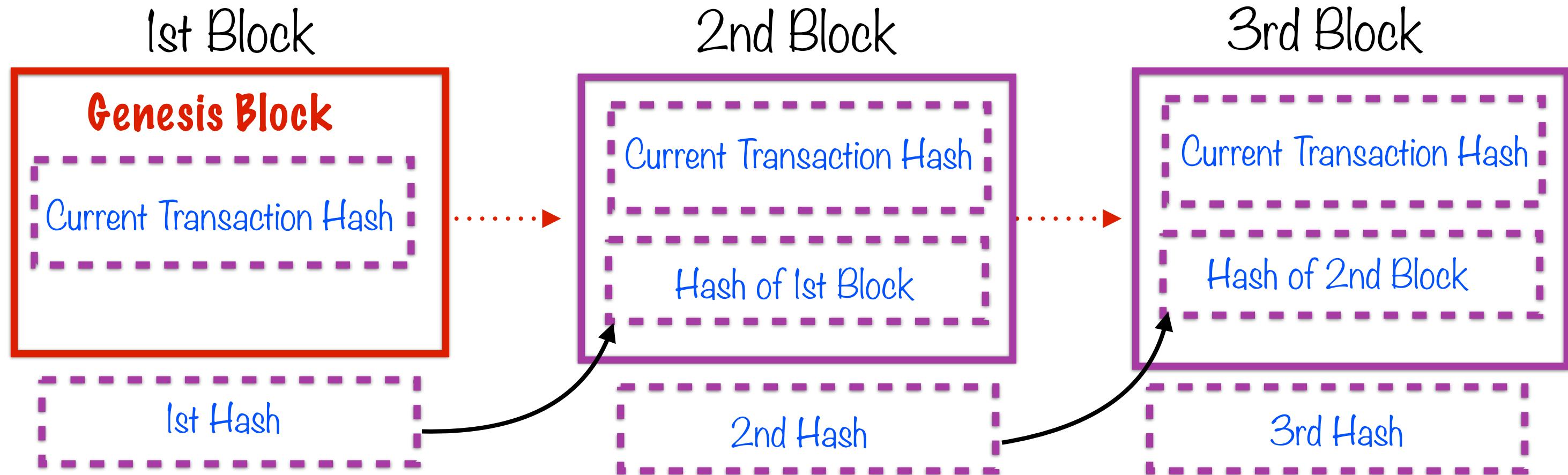
Each block contains hash of the preceding one in chain

Chain of Blocks



This is not true for the first block - called the **Genesis Block**

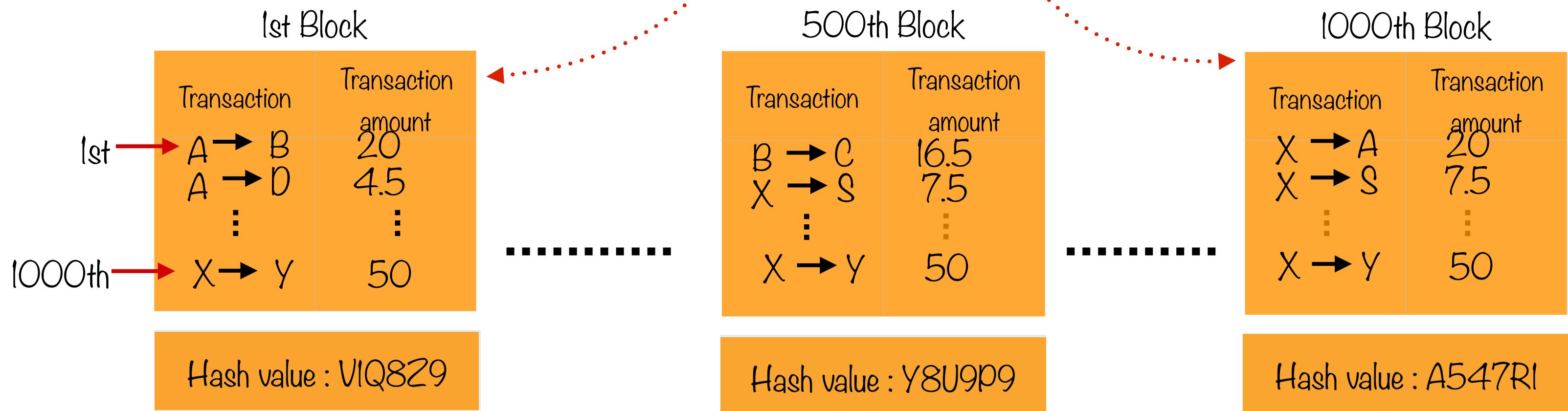
Chain of Blocks



Genesis block does not contain previous hash value

Merkle Trees

B's Ledger



Transactions are divided into 1000 blocks each containing
1000 transactions for each party

Identifying Discrepancies

	A's Ledger	B's Ledger	C's Ledger
1st	→ A1V2M2	A1V2M2	A1V2M2
	E4R6Y8	E4R6Y8	E4R6Y8
	⋮	⋮	⋮
500th	→ K3L5M7	Y8U9P9	K3L5M7
	⋮	⋮	⋮
1000th	→ A547RI	A547RI	A547RI

By comparing each block's hash discrepancies can be found quickly and efficiently

Distributed Ledger

A's Ledger		B's Ledger		C's Ledger	
Transaction	Transaction amount	Transaction	Transaction amount	Transaction	Transaction amount
1st → A → B	10.5	A → B	10.5	A → B	10.5
B → C	8.5	B → C	8.5	B → C	8.5
10 ⁶ th → X → Y	50	B → Y	35	C → Z	45
Hash value : F1F2F3		Hash value : F1F2F3		Hash value : F1F2F3	

Each node stores hash of entire ledger

Distributed Ledger

A's Ledger		B's Ledger		C's Ledger	
Transaction	Transaction amount	Transaction	Transaction amount	Transaction	Transaction amount
1st → A → B	10.5	A → B	10.5	A → B	10.5
B → C	8.5	B → C	8.5	B → C	8.5
10 ⁶ th → X → Y	50	B → Y	35	C → Z	45
Hash value : F1F2F3		Hash value : F1F2F3		Hash value : F1F2F3	

Each node stores hash of entire ledger

Distributed Ledger

A's Ledger		B's Ledger		C's Ledger	
Transaction	Transaction amount	Transaction	Transaction amount	Transaction	Transaction amount
1st → A → B	10.5	A → B	10.5	A → B	10.5
B → C	8.5	B → C	8.5	B → C	8.5
10 ⁶ th → X → Y	50	B → Y	35	C → Z	45
Hash value : F1F2F3		Hash value : F1F2F3		Hash value : F1F2F3	

Each node stores hash of entire ledger

Distributed Ledger

A's Ledger		B's Ledger		C's Ledger	
Transaction	Transaction amount	Transaction	Transaction amount	Transaction	Transaction amount
1st → A → B	10.5	A → B	10.5	A → B	10.5
B → C	8.5	B → C	16.5	B → C	8.5
10 ⁶ th → X → Y	50	B → Y	35	C → Z	45
Hash value : F1F2F3		Hash value : 3AE81C		Hash value : F1F2F3	

Hash mismatch => transaction details mismatch

Distributed Ledger

A's Ledger		B's Ledger		C's Ledger	
Transaction	Transaction amount	Transaction	Transaction amount	Transaction	Transaction amount
1st → A → B	10.5	A → B	10.5	A → B	10.5
B → C	8.5	B → C	16.5	B → C	8.5
10 ⁶ th → X → Y	50	B → Y	35	C → Z	45
Hash value : F1F2F3		Hash value : 3AE81C		Hash value : F1F2F3	

Only now will need to scan entire ledger to identify discrepancy

Tree vs. List

Problem: Search ledger to find transaction that caused hash mismatch

If ledger is represented as a list, this problem is $O(N)$

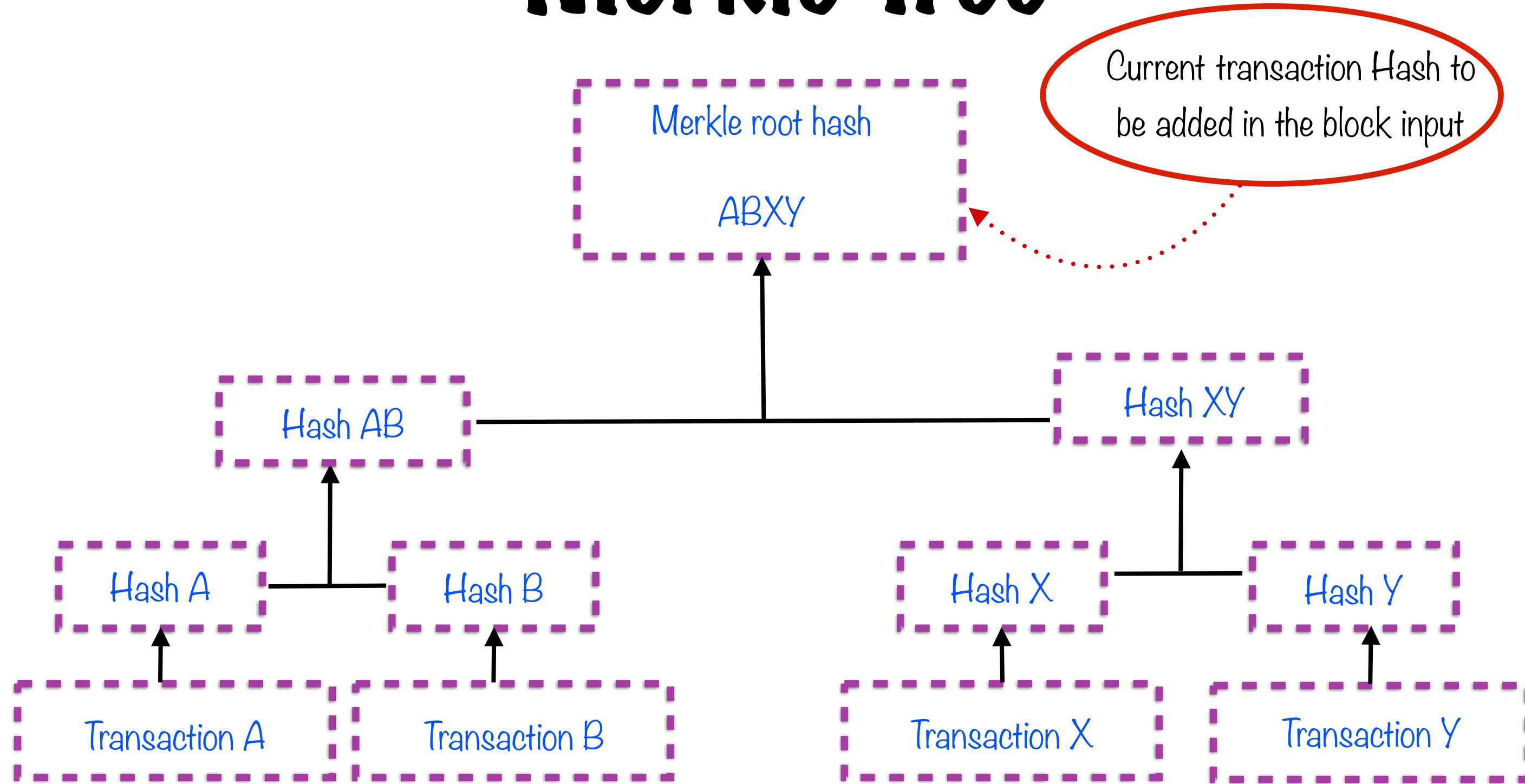
If ledger is represented as a tree, this improves to $O(\log N)$

Scanning ledger for discrepancy is possible
in $O(\log N)$ using a Merkle Tree

Merkle Tree

A tree in which every leaf node is labelled with the hash of a data block and every non-leaf node is labelled with the cryptographic hash of the labels of its child nodes

Merkle Tree



Order Matters

Merkle Tree encapsulates entire state of ledger so far

Each node computes this tree while verifying transactions

The order of transactions in the tree matters in calculating Merkle root hash

Need for Consensus

Nodes must agree on the order of transactions

Nodes do not trust each other

Need for “trustless consensus” algorithm

- proof-of-work
- proof-of-stake

Verifying Sender Identity

Transaction Verification

Initiated transaction goes to unconfirmed transaction pool from where nodes take transactions and verify them before adding them to their blocks

Transaction Verification



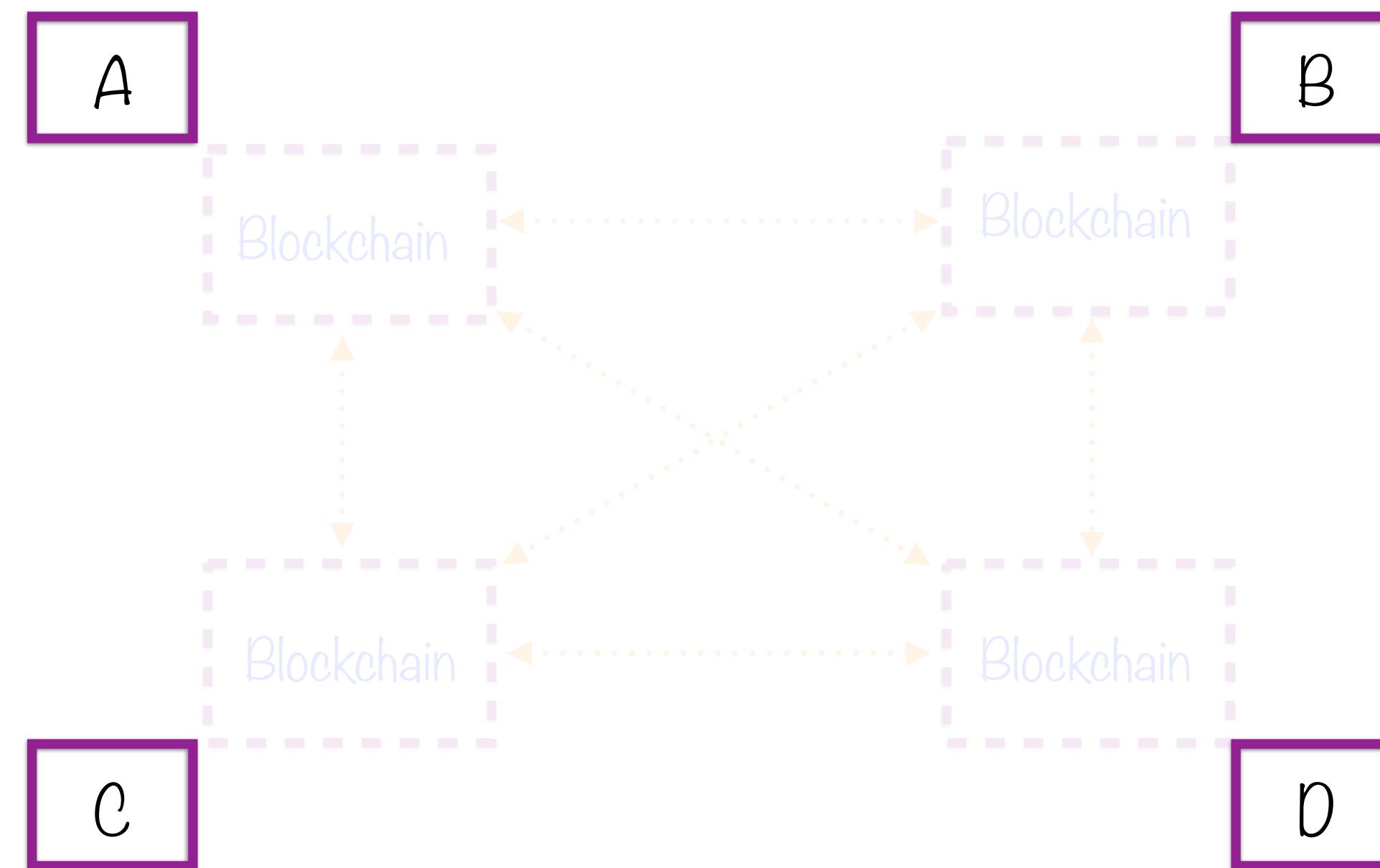
- Verifying sender and transaction integrity
- Confirm that sender owns assets to send

Transaction Verification

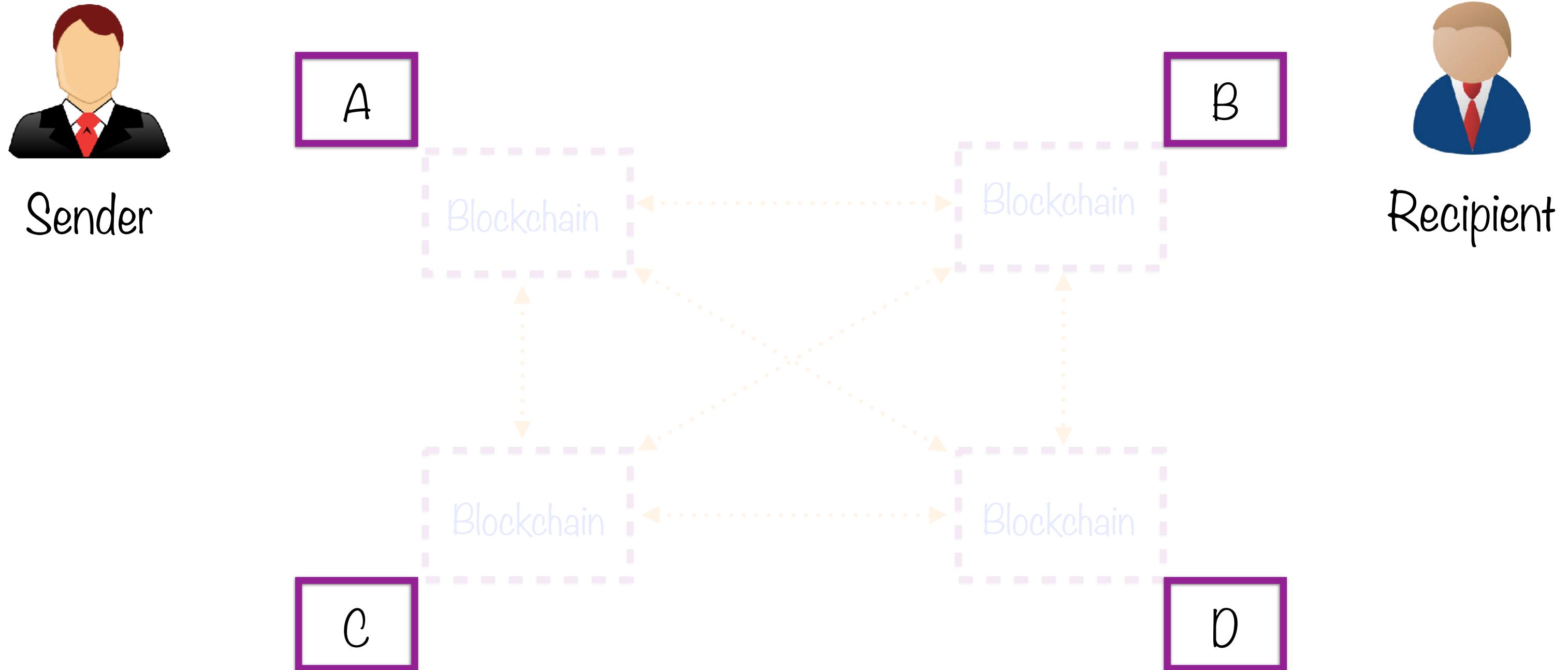


- Verifying sender and transaction integrity
- Confirm that sender owns assets to send
- Check transaction nonce

Nodes in a Blockchain Network



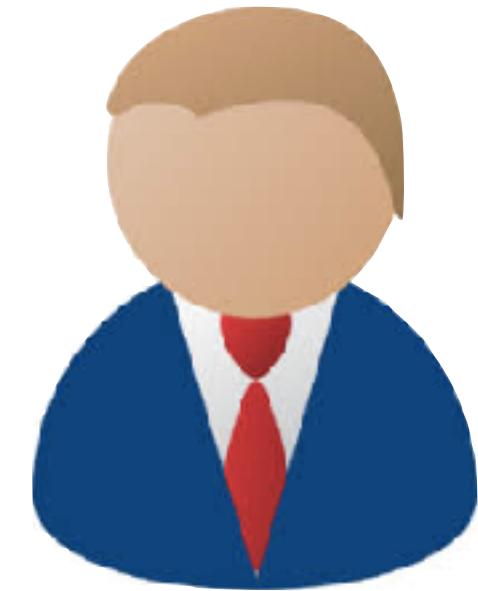
Accounts in a Blockchain Network



Transaction Verification



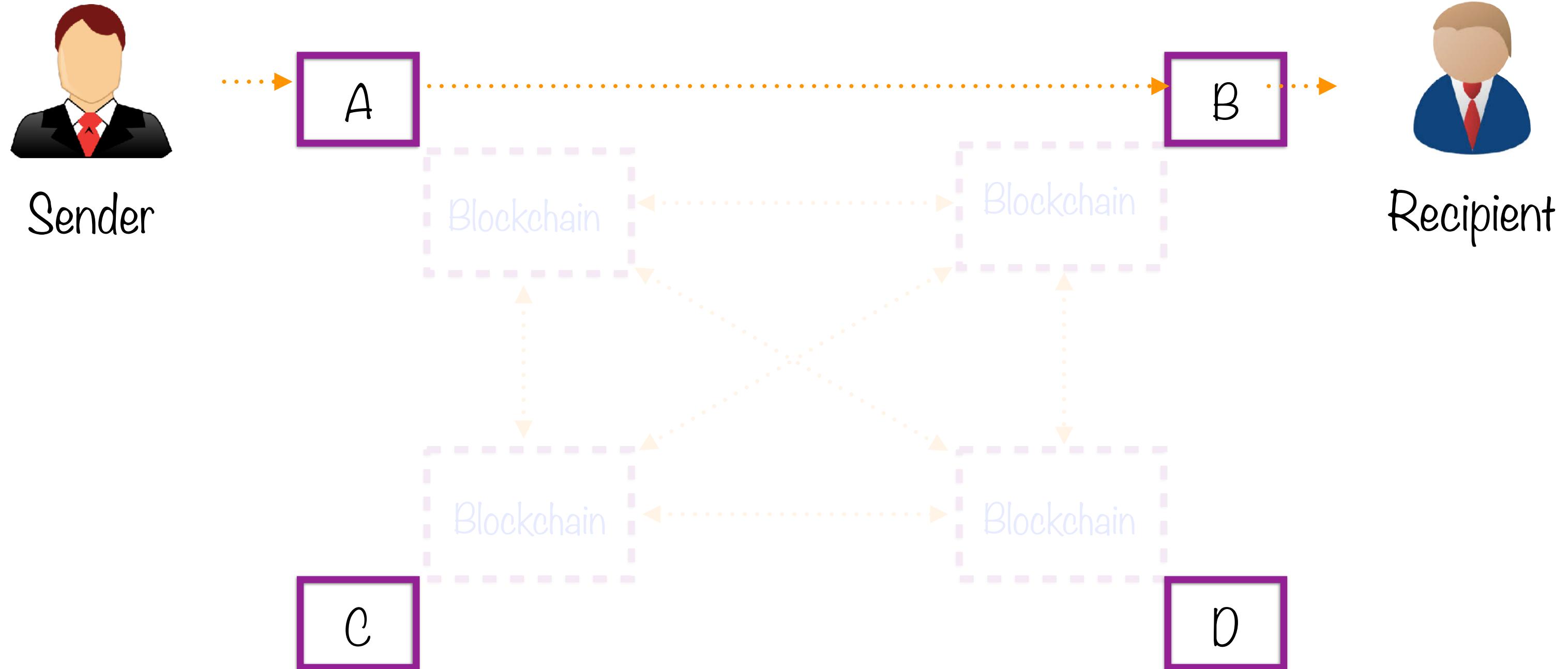
A
Sender



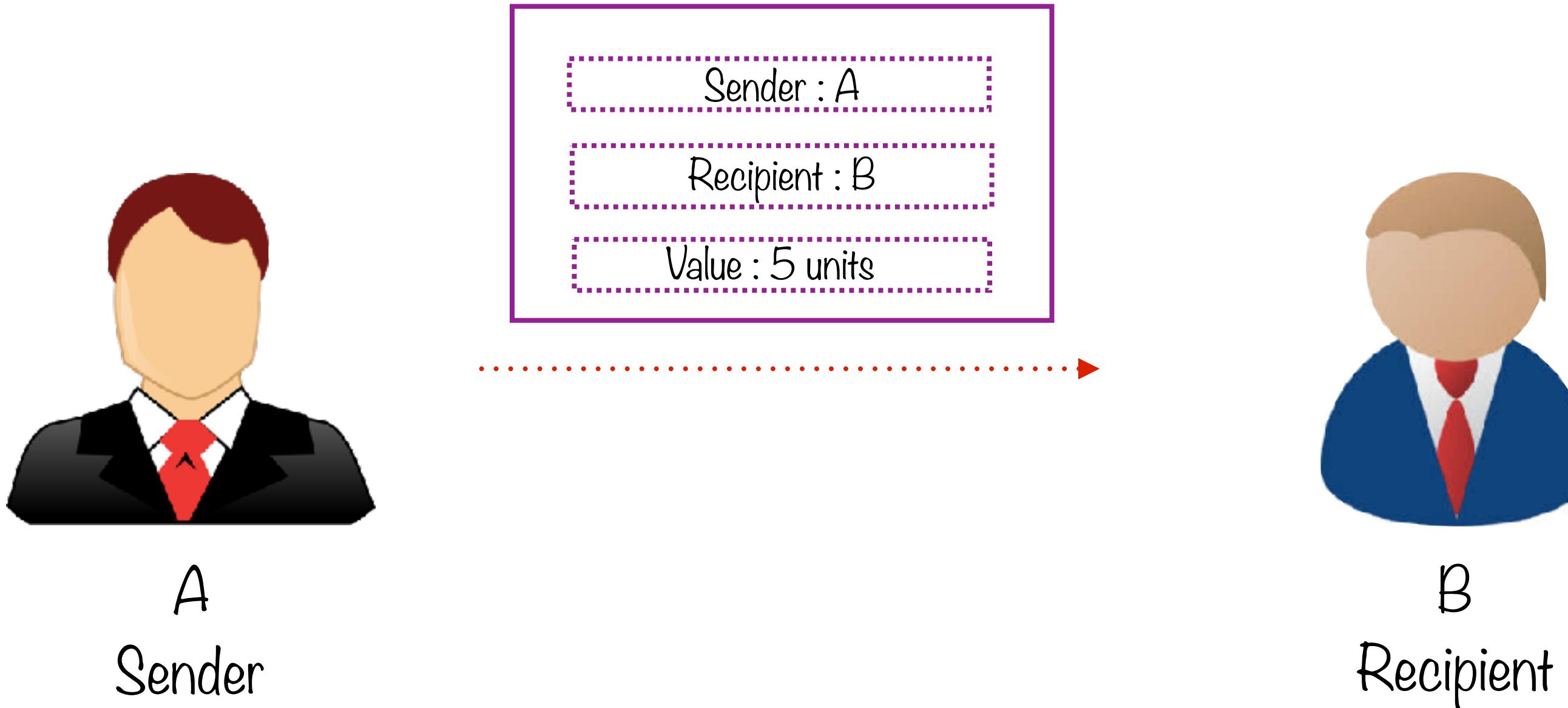
B
Recipient



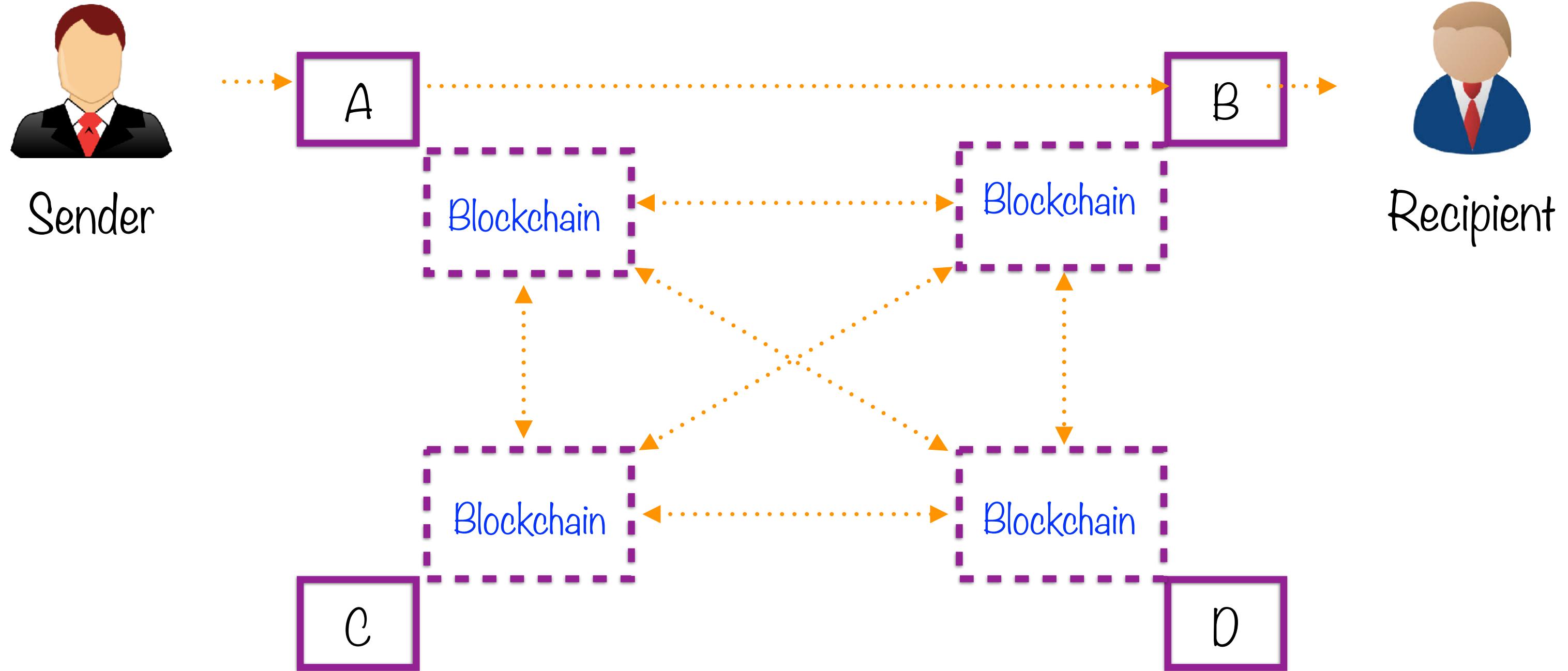
Accounts in a Blockchain Network



Transaction Verification

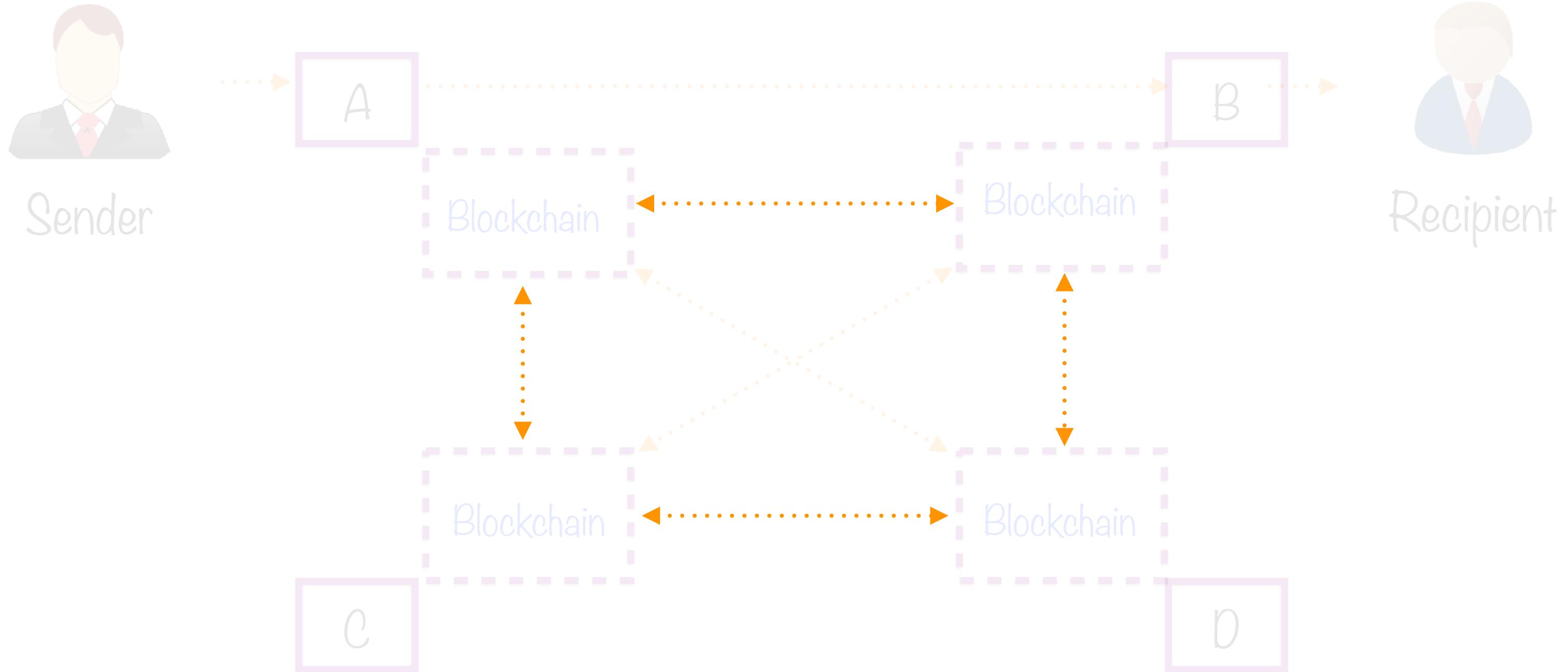


Accounts in a Blockchain Network



Every transaction is broadcast across the network

Accounts in a Blockchain Network



Every transaction is broadcast across the network

Every transaction is broadcast across entire
the network

Consensus



Many transactions may be broadcast
simultaneously

Many miners may pick these up in differing
orders

How is the order determined?

- Via a consensus algorithm

Accounts



Each account possesses a key pair

- Private key (analogy: ATM PIN)
- Public key (analogy: Card Number, CVV)

Accounts



Sender sends across two messages

- digitally signed transaction
- raw (unsigned) transaction

Digital Signed Message



Authentication

Non-repudiation

Integrity

Digital Signed Message

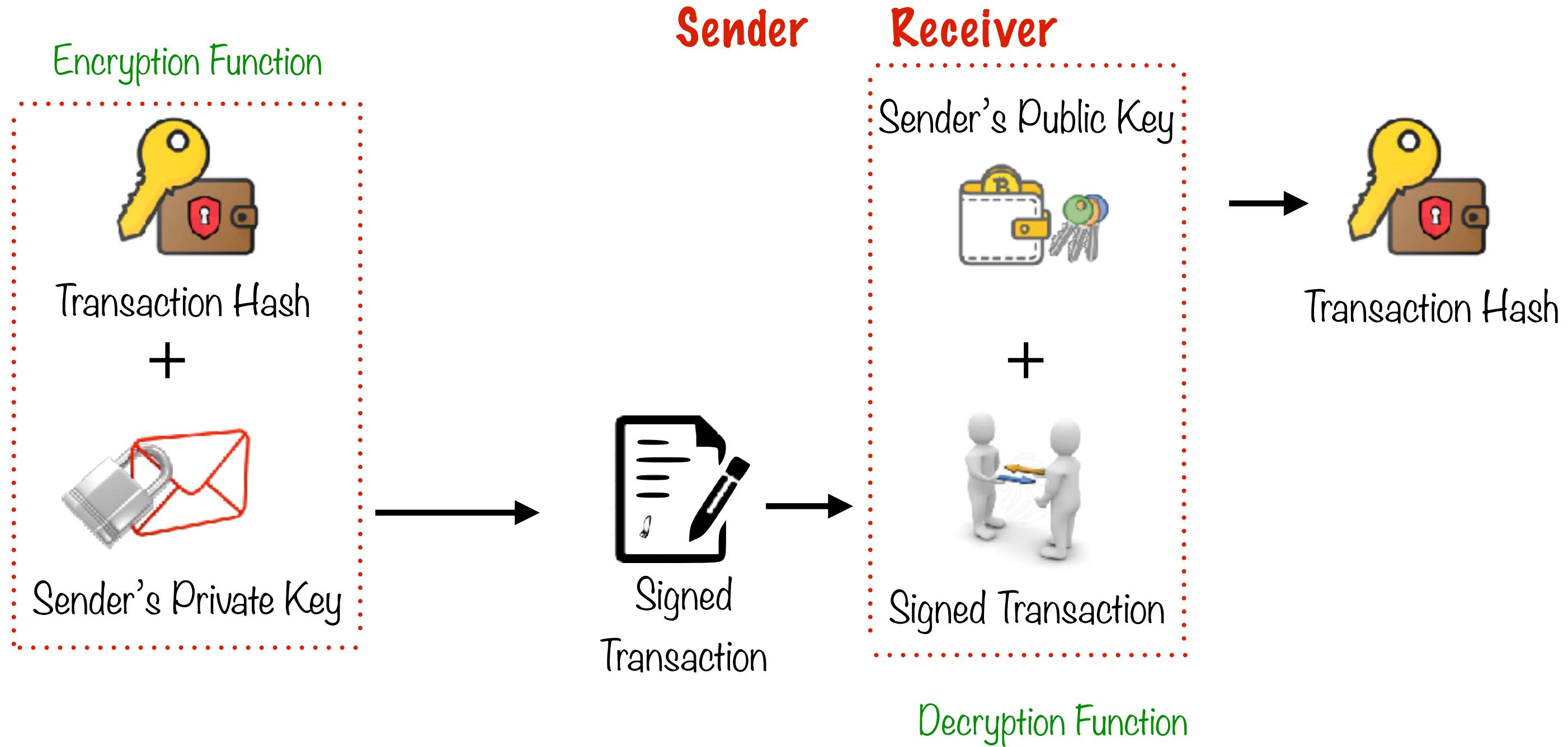


Authentication: Confirm identity of sender

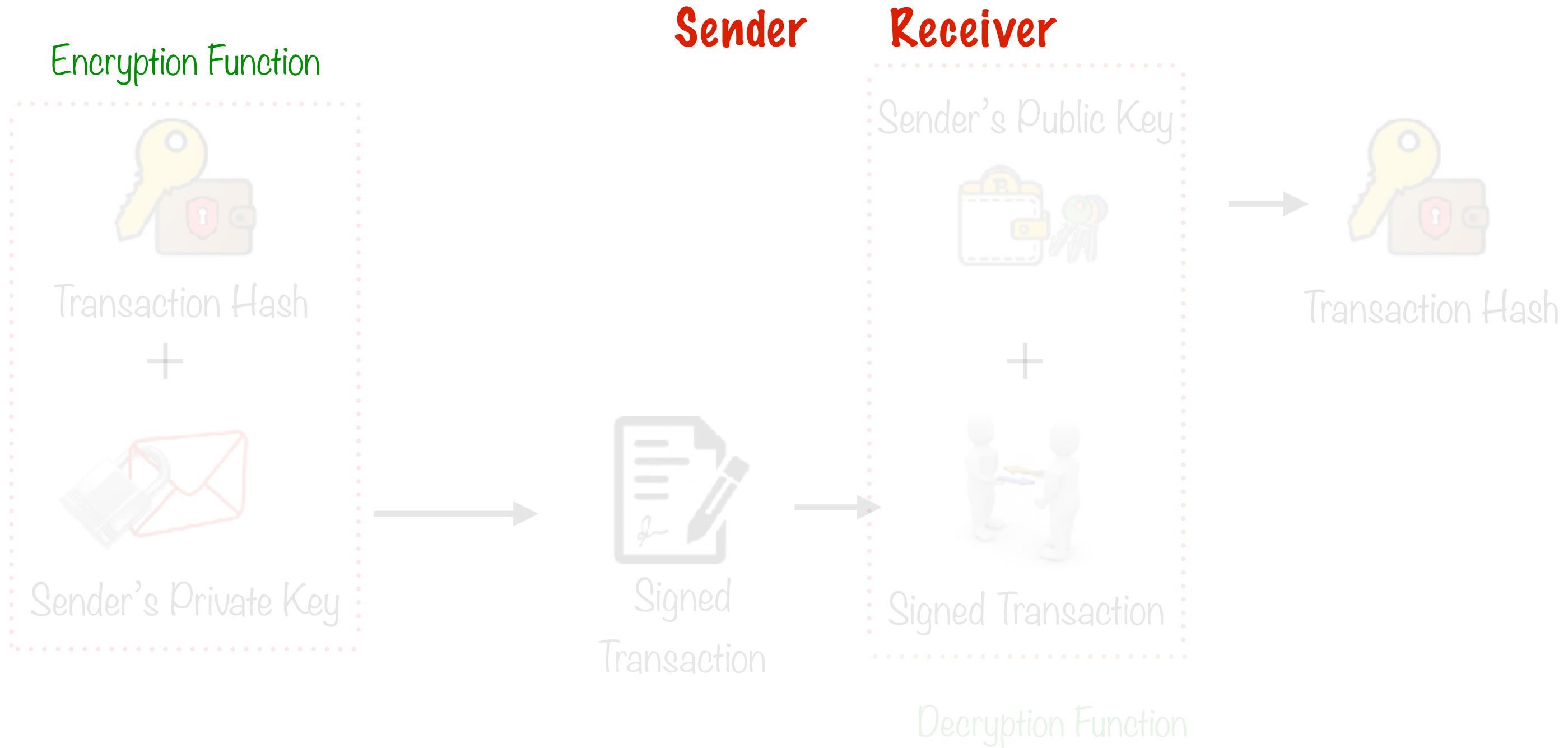
Non-repudiation: Sender can not deny having sent the message

Integrity: Message was not altered during transmission

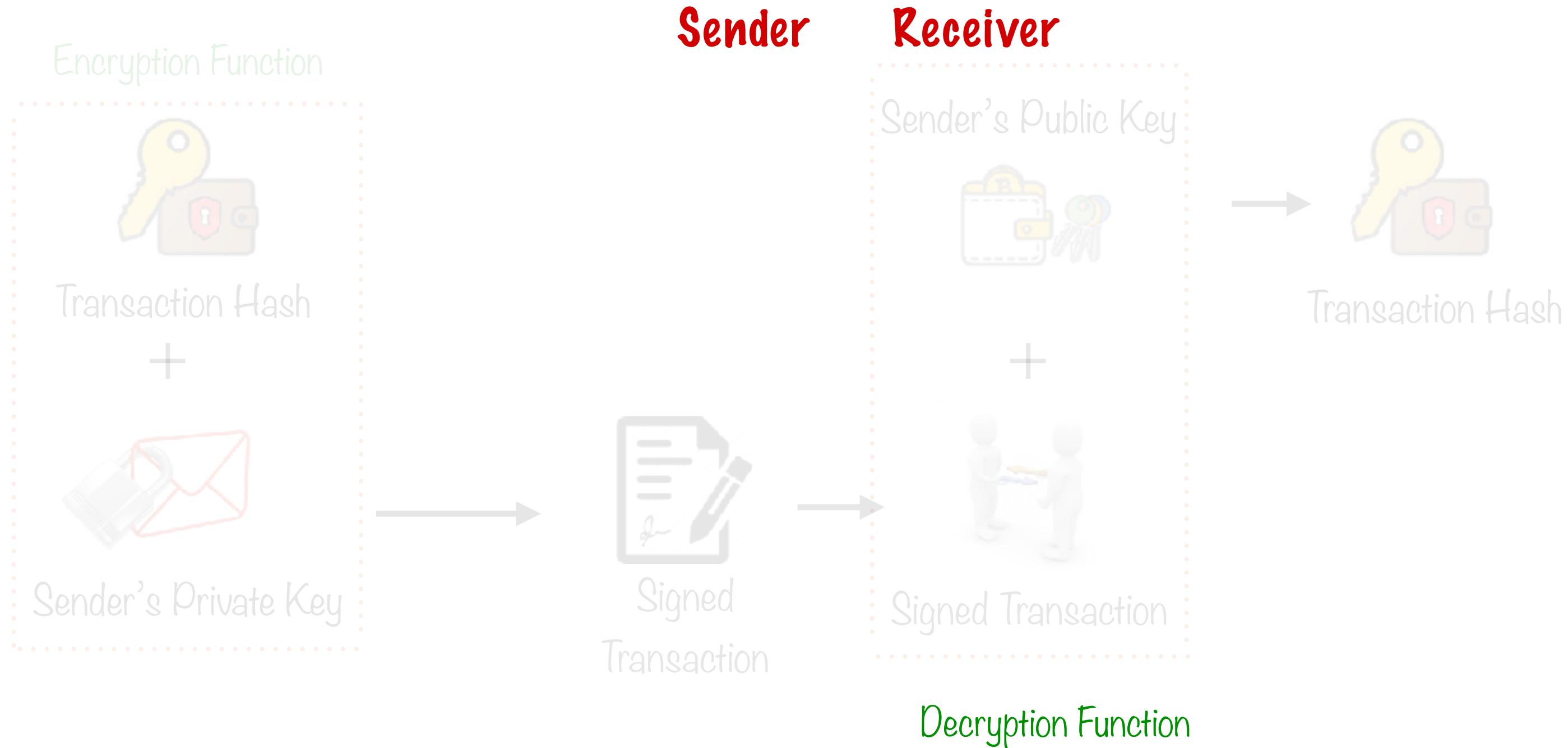
Signed Transaction Sent Across



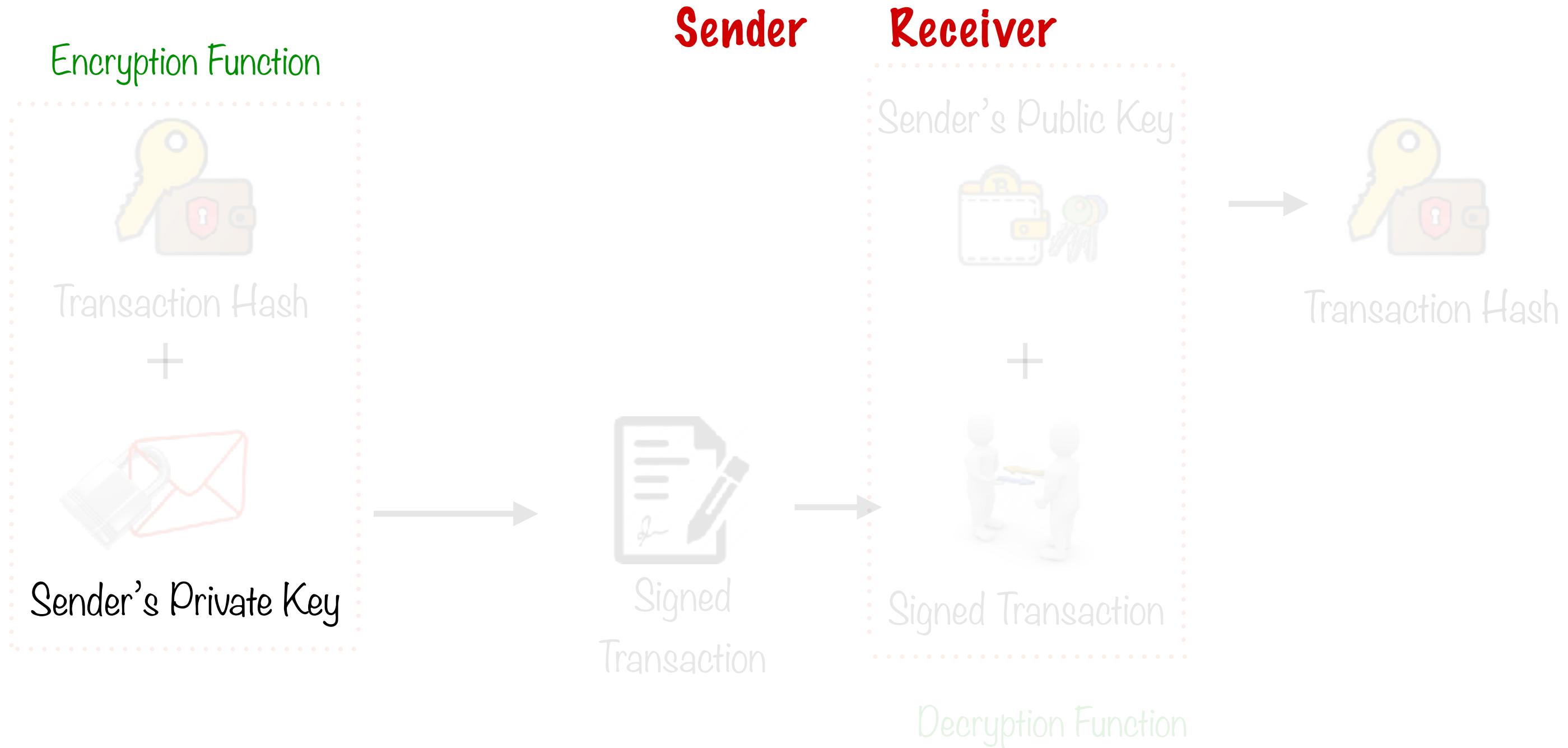
Signed Transaction Sent Across



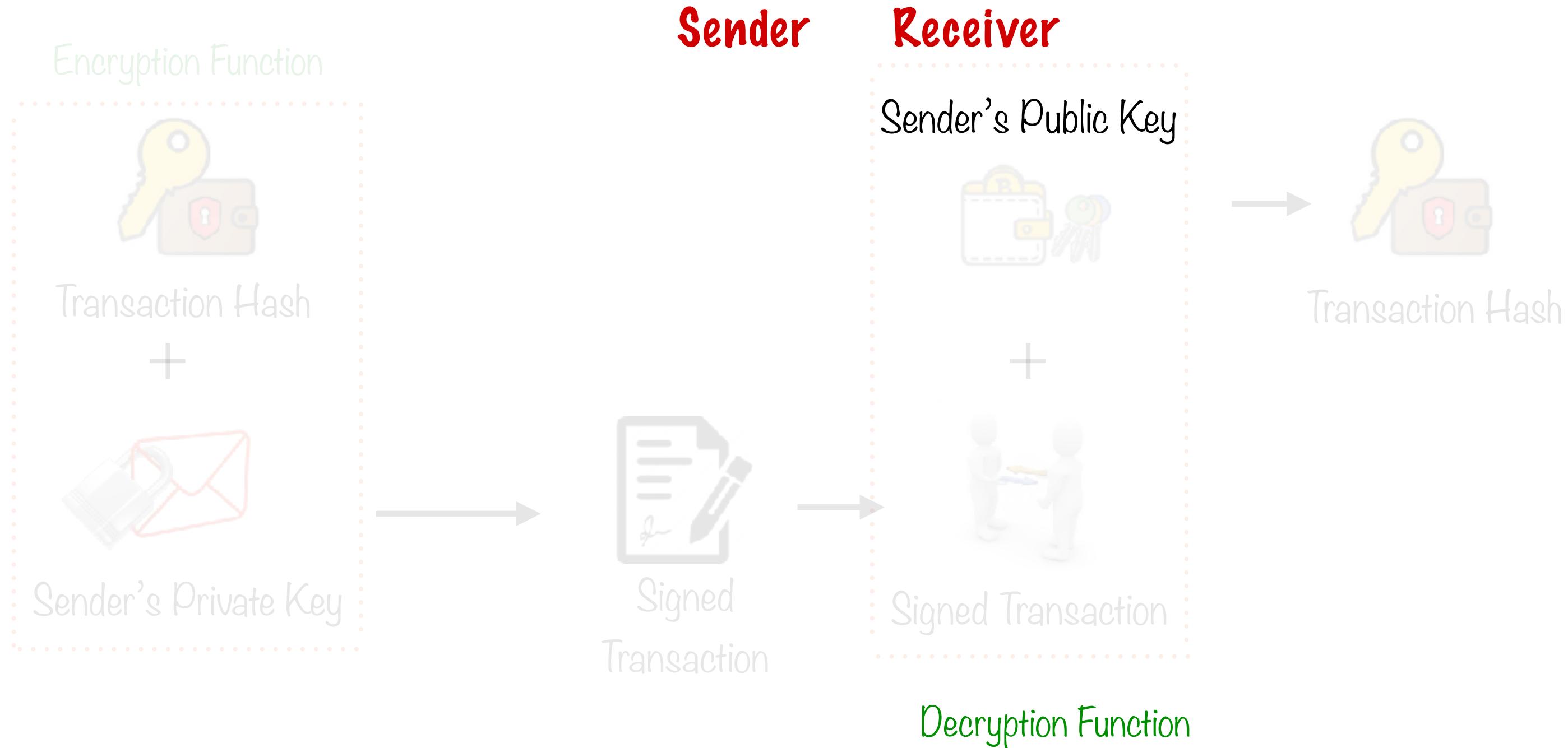
Signed Transaction Sent Across



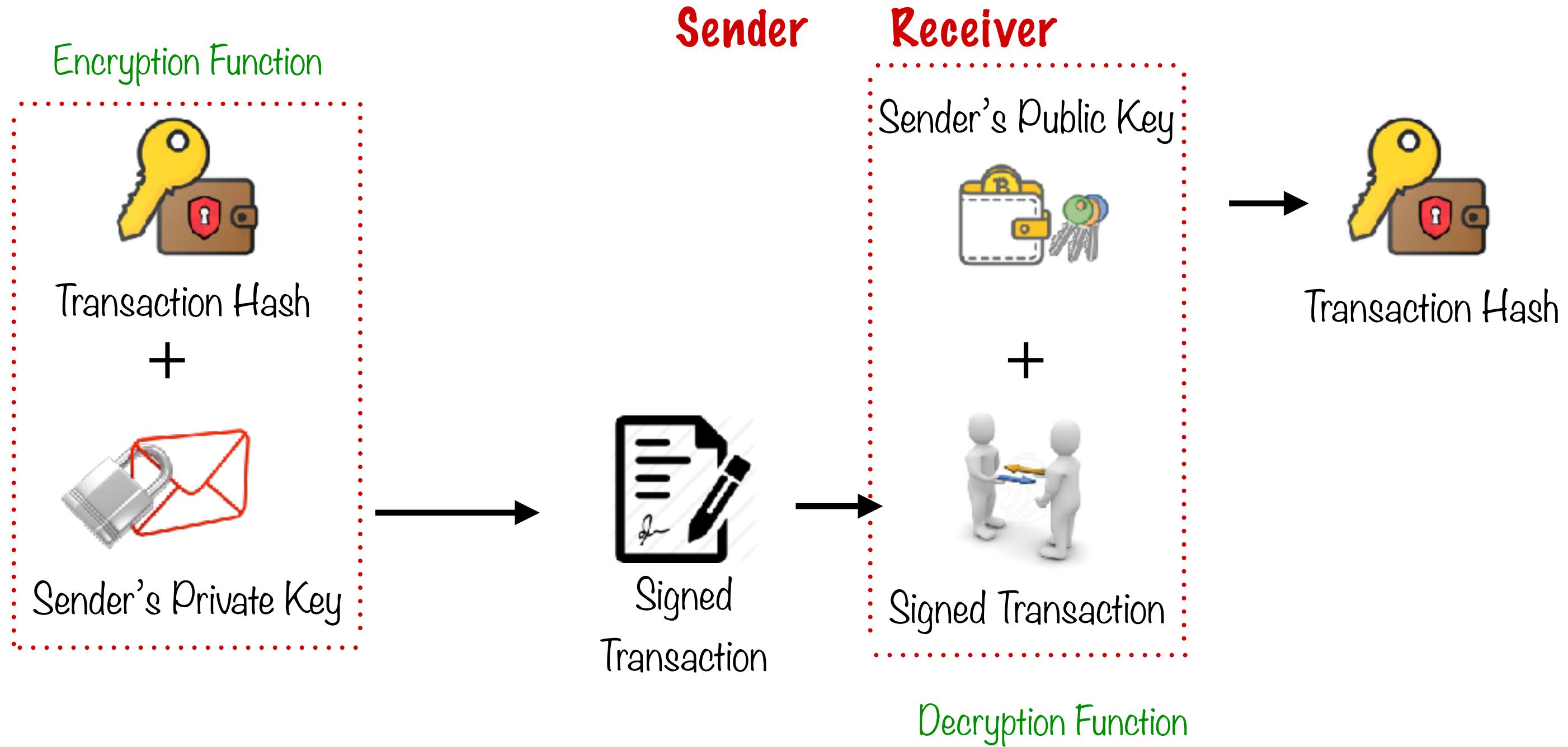
Signed Transaction Sent Across



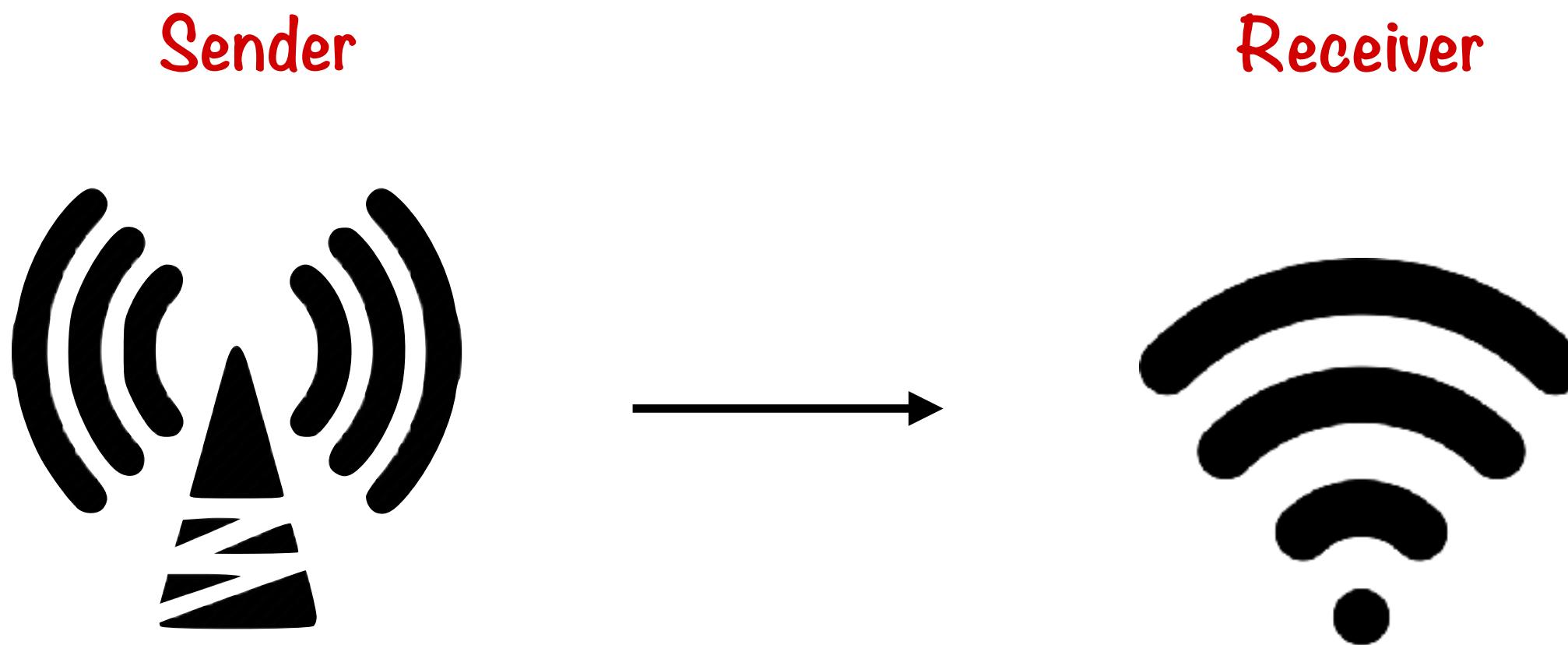
Signed Transaction Sent Across



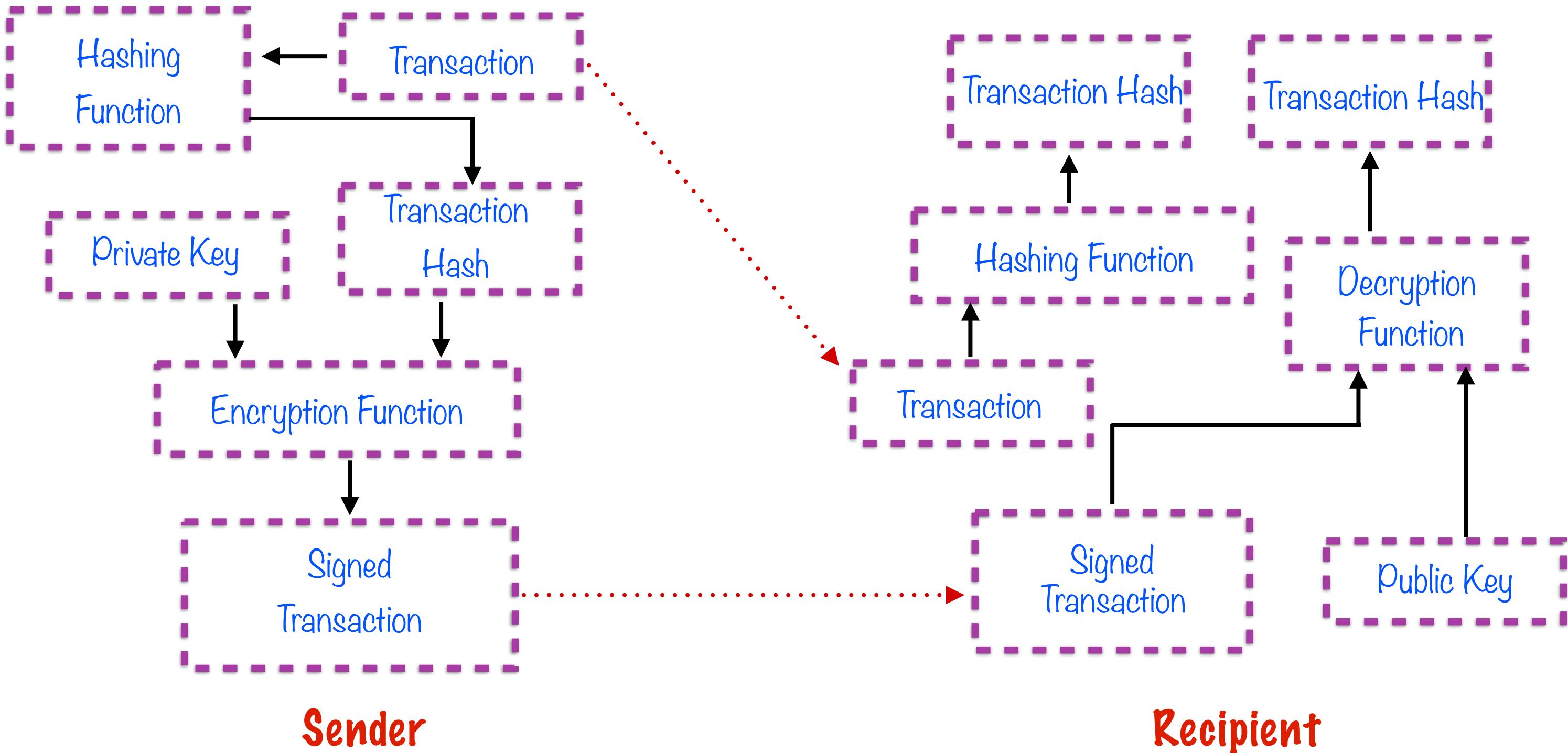
Signed Transaction Sent Across



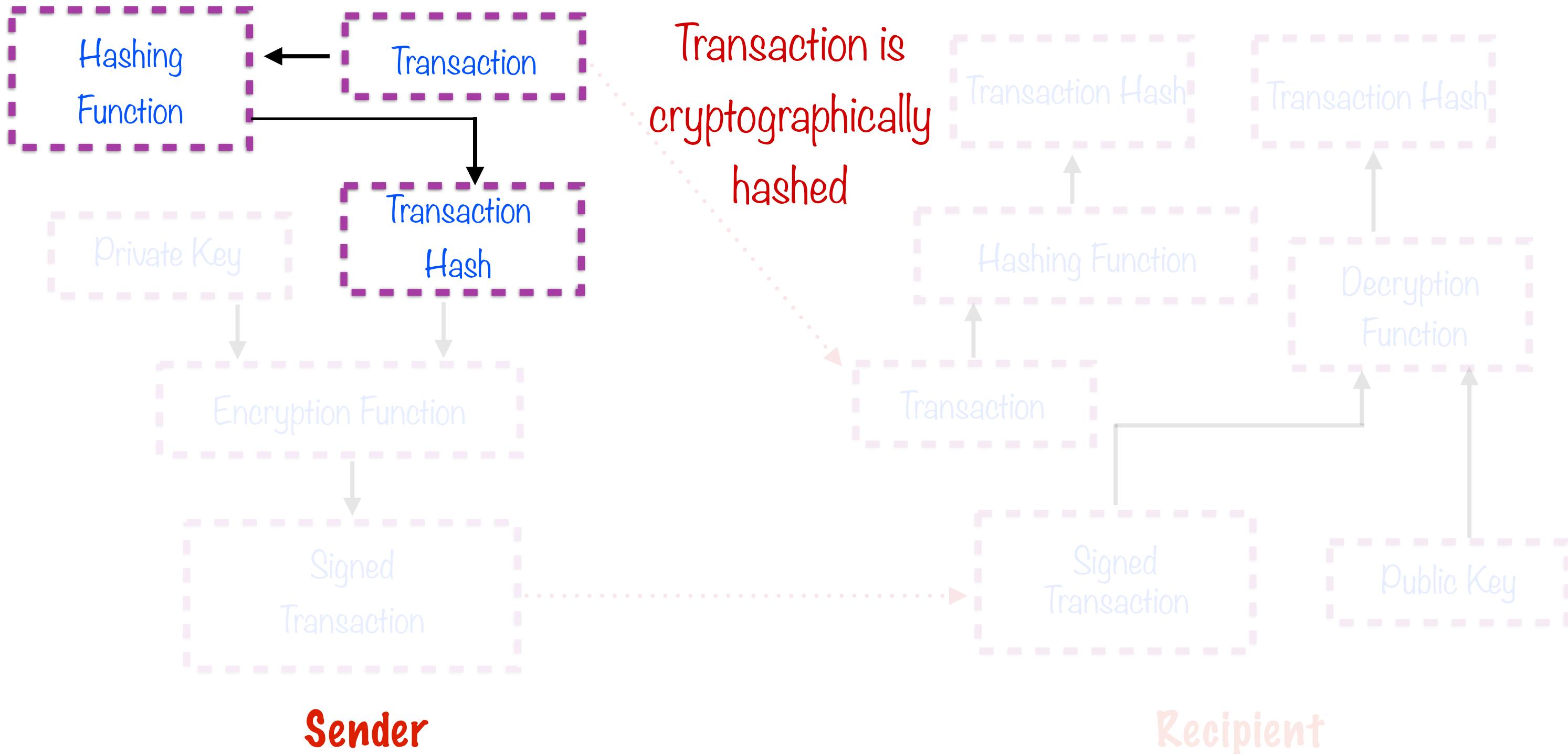
Unsigned Transaction Sent Across Also



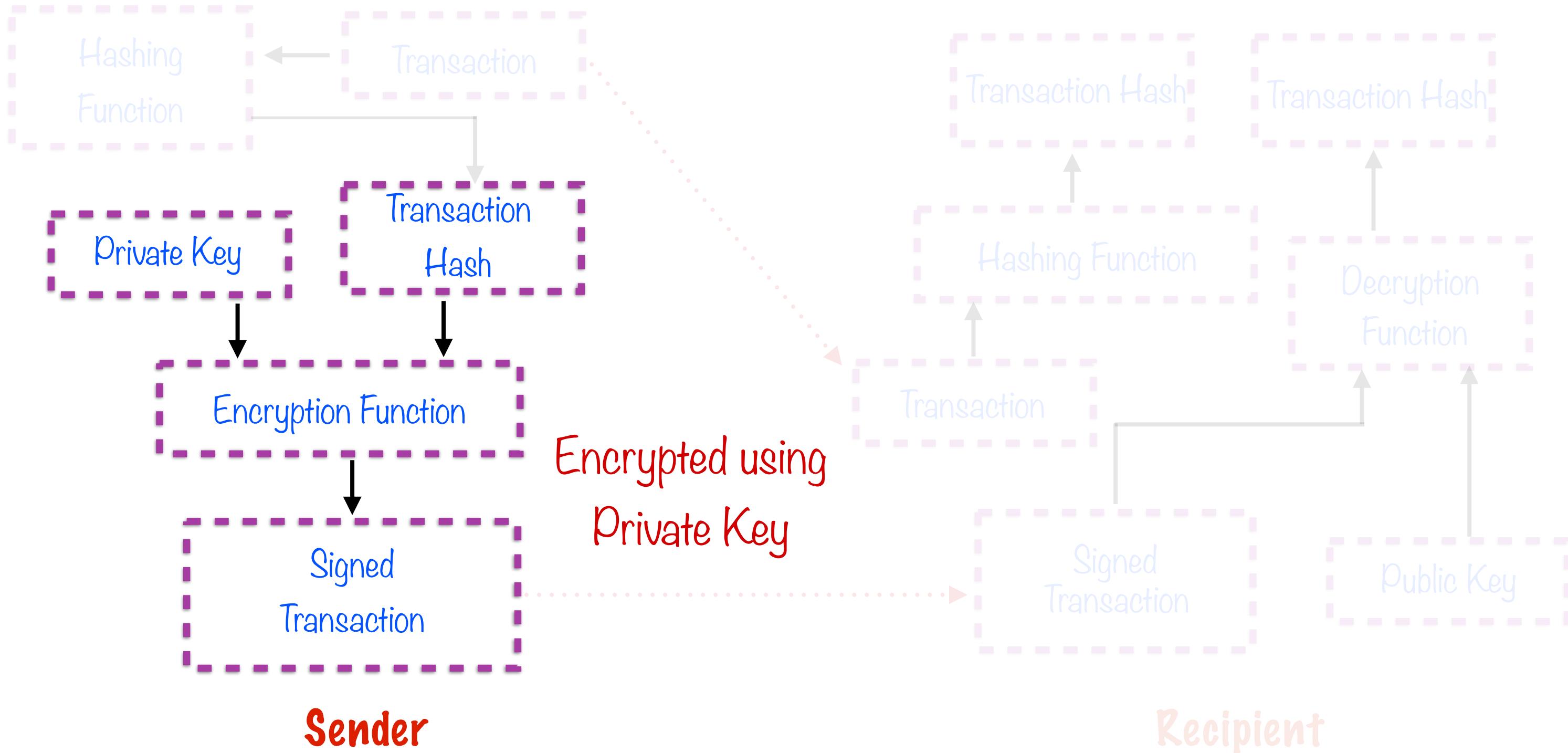
Transaction Verification



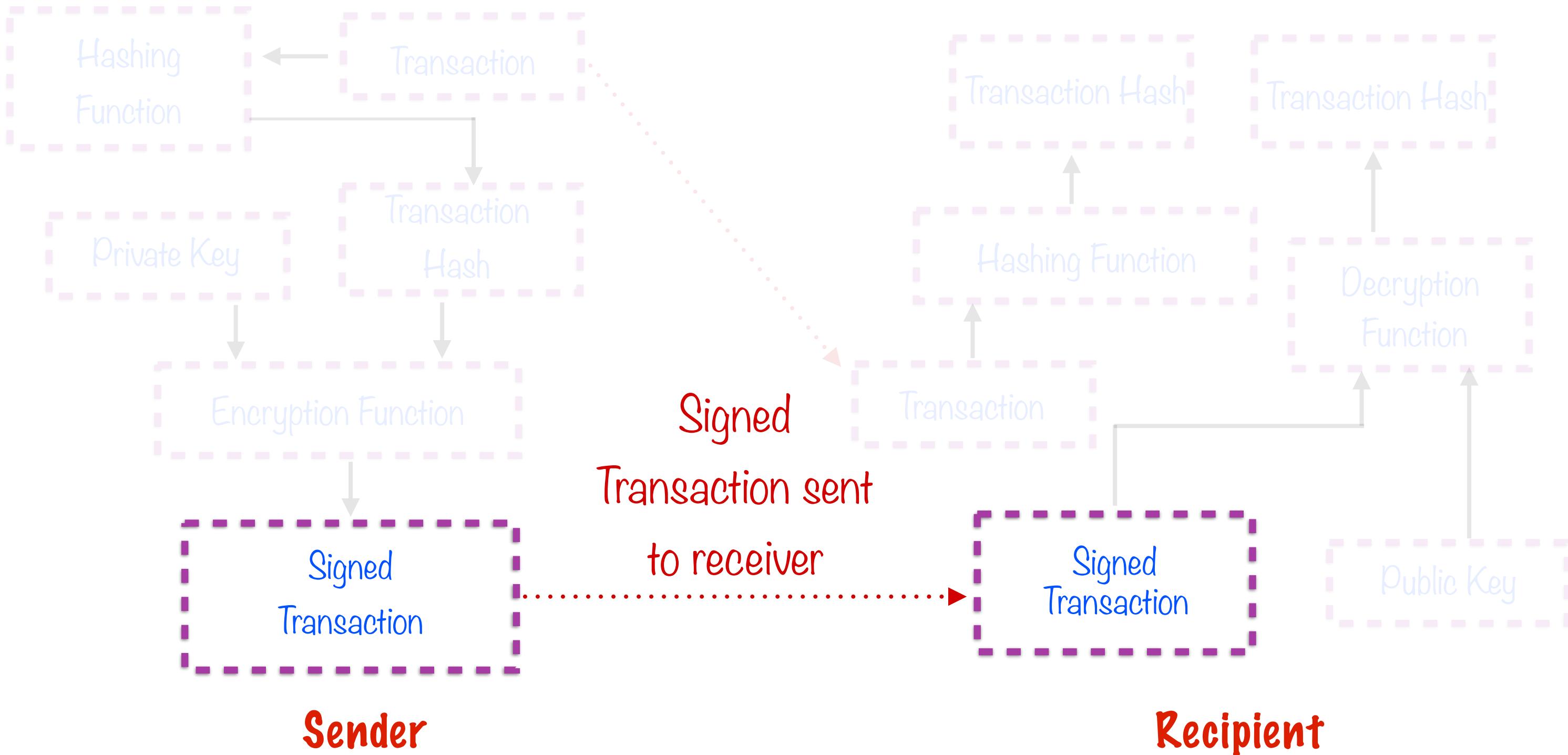
Transaction Verification



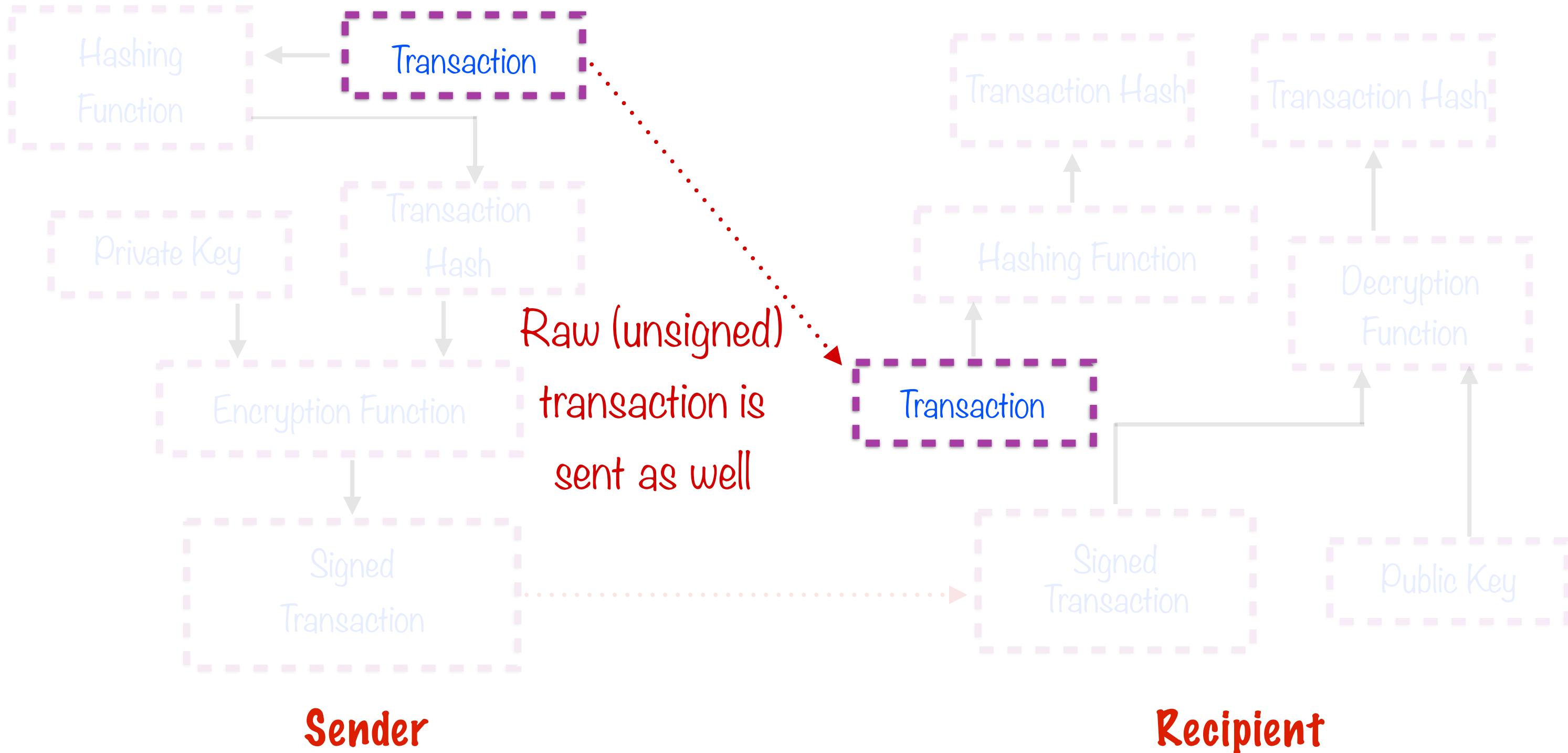
Transaction Verification



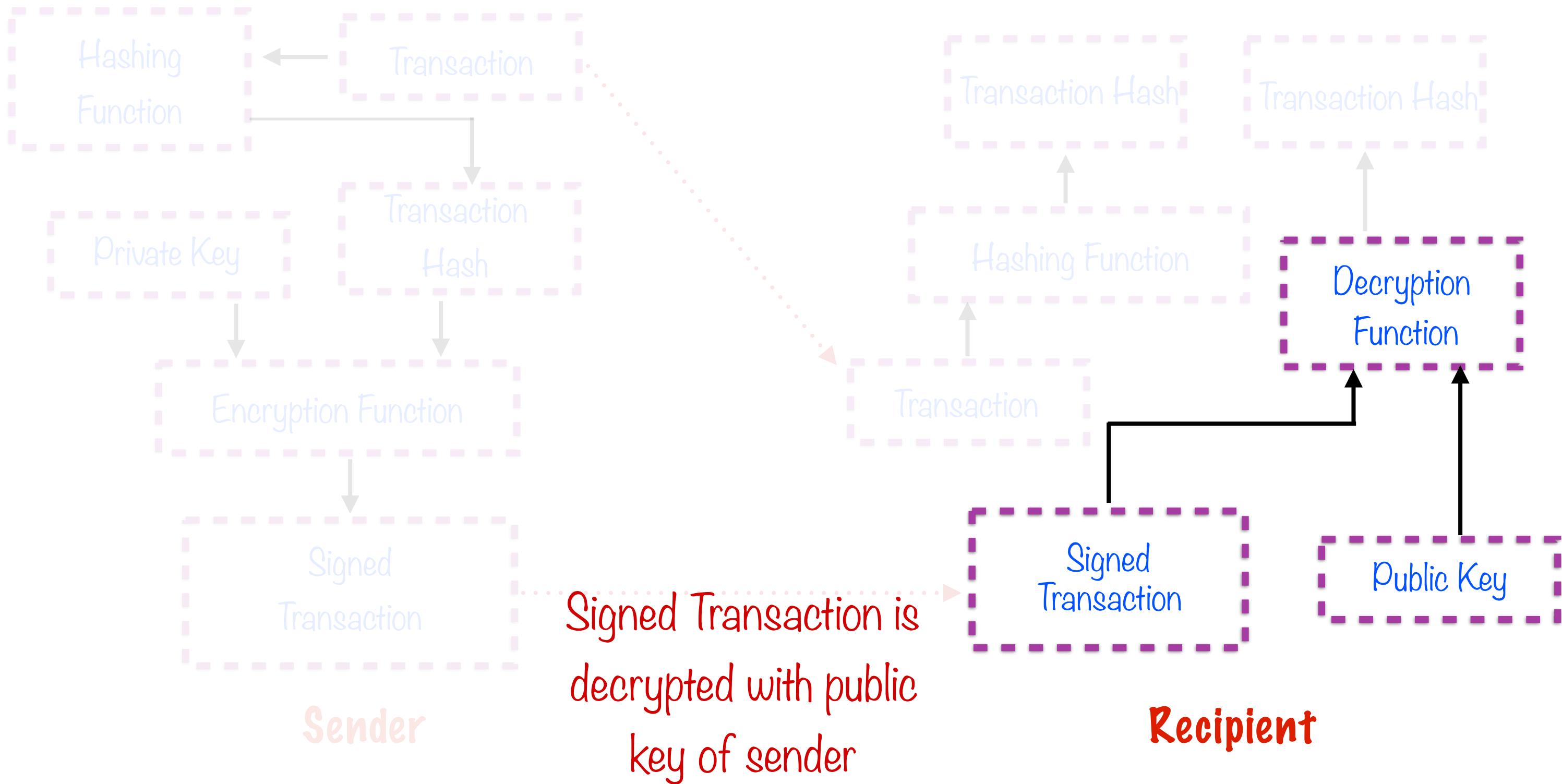
Transaction Verification



Transaction Verification

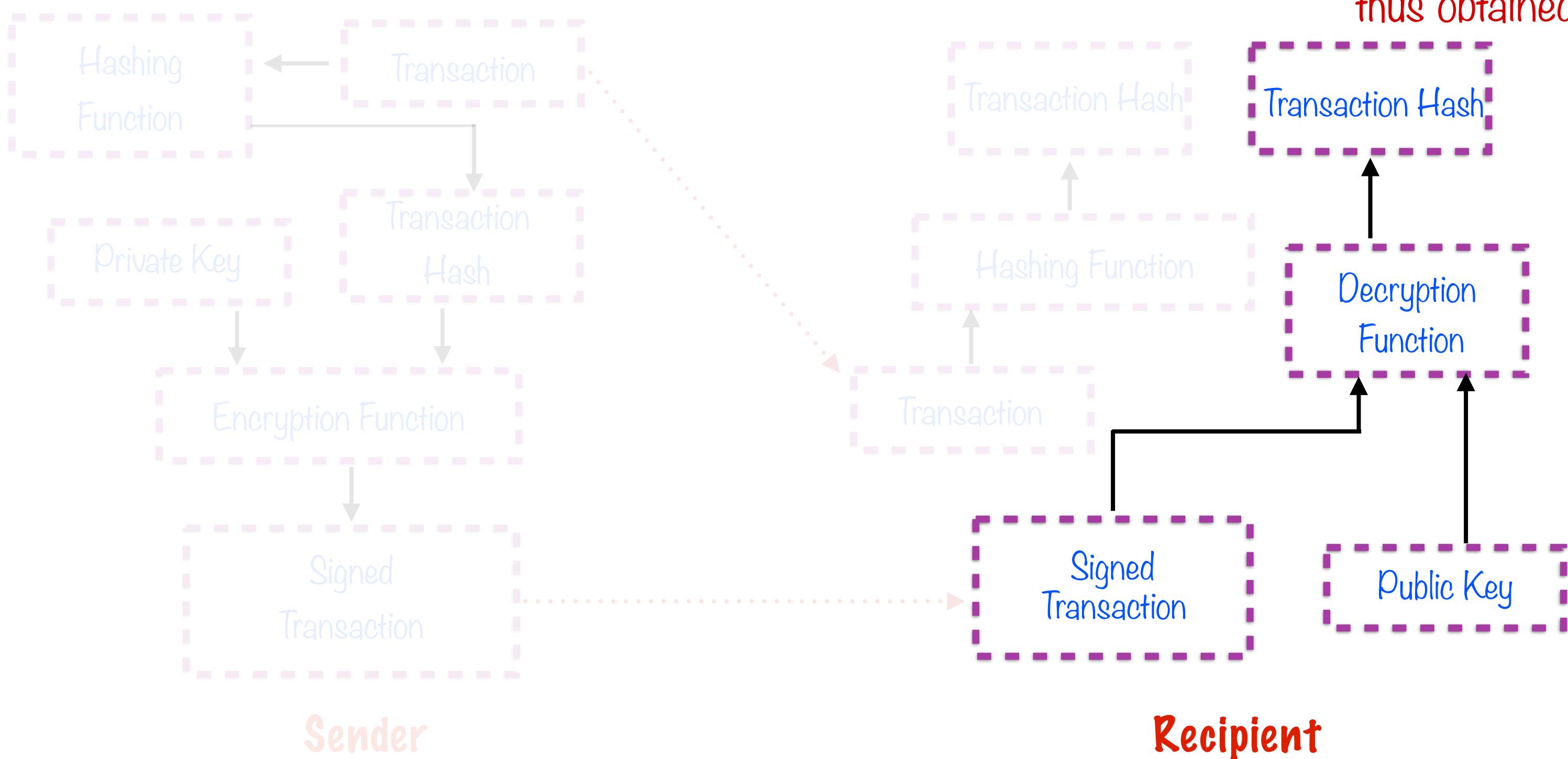


Transaction Verification

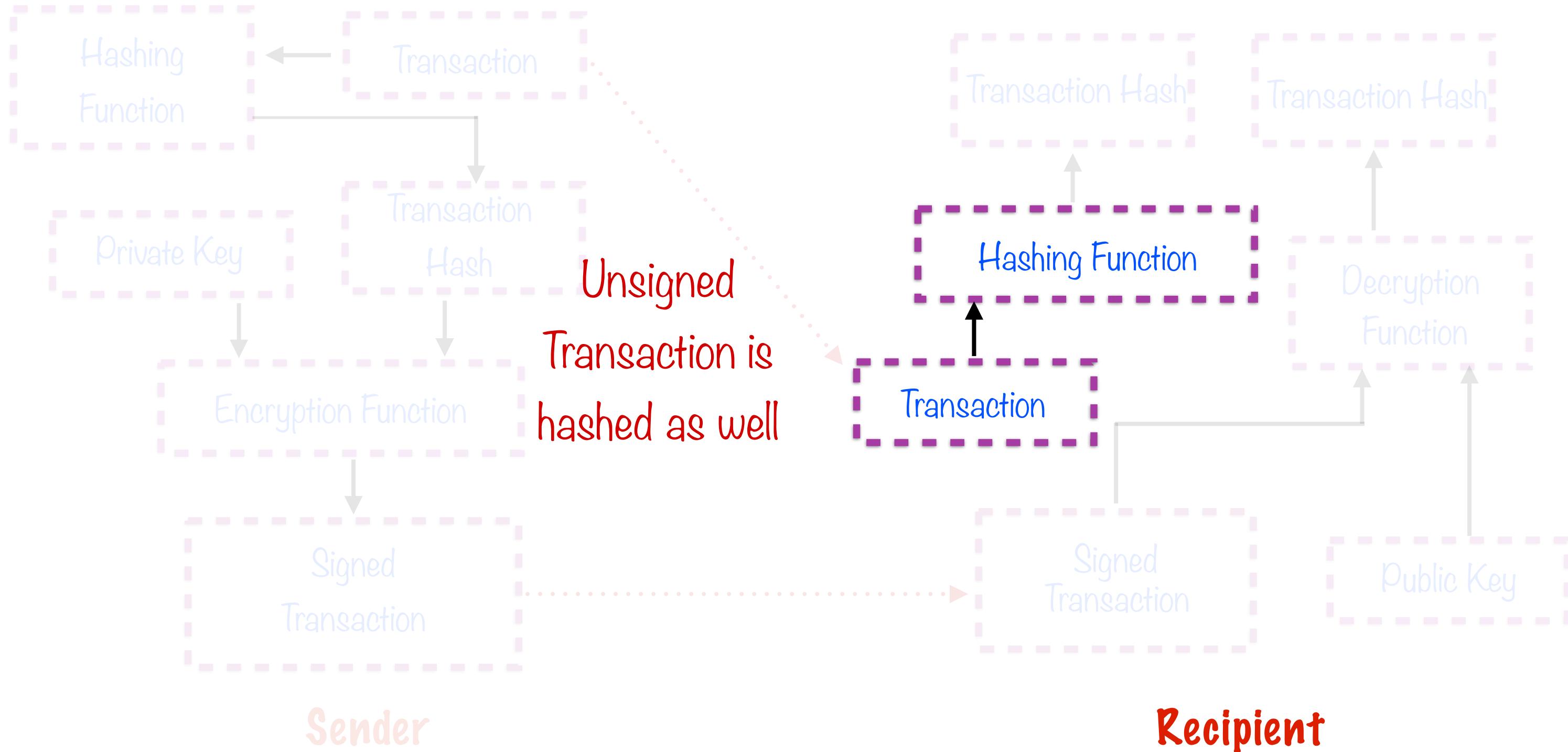


Transaction Verification

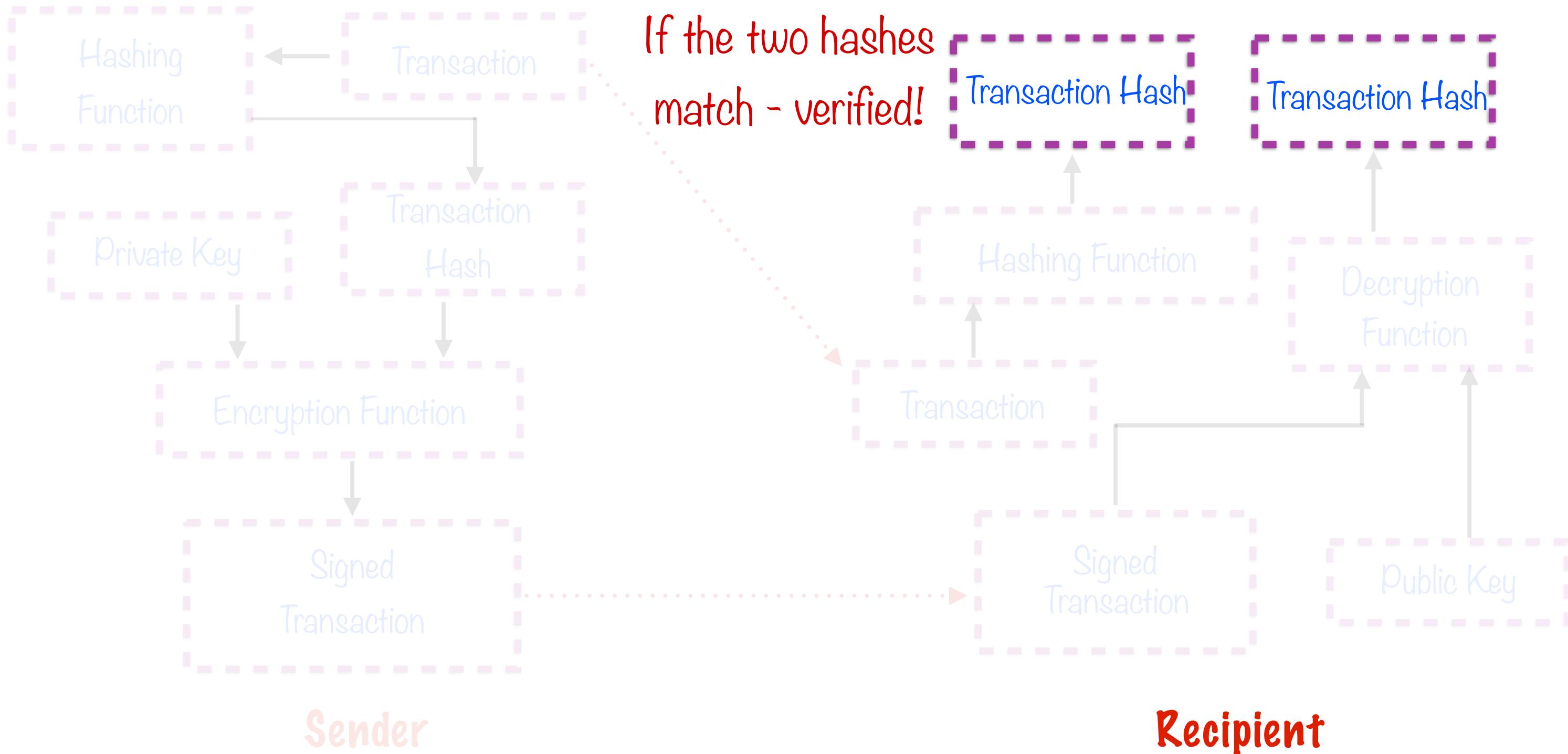
Transaction hash is thus obtained



Transaction Verification



Transaction Verification



Public and Private Blockchain Networks

Public Network

Truly open membership - anyone can join

Lots of nodes

Slower to converge on consensus*

Higher probability of fraud

Private Network

Restricted entry - selection criteria

Fewer nodes

Faster convergence on consensus*

Lower probability of fraud

*Will be explained in just a bit

Verifying Sender's Balance

Transaction Verification



- Verifying sender and transaction integrity
- Confirm that sender owns assets to send
- Check transaction **nonce**

Transaction Verification



Balance must be sufficient to cover

- Actual assets being transferred
- Gas limit: Max total prize that may need to paid out for this particular transaction

Transaction Verification



- Verifying sender and transaction integrity
- Confirm that sender owns assets to send
- Check transaction nonce

Transaction Verification



- Verifying sender and transaction integrity
- Confirm that sender owns assets to send
- Check transaction nonce

Sender's Balance



Need mechanism to ensure sender owns the asset being sent

State database maintained by each node

Transaction Verification



- Verifying sender and transaction integrity
- Confirm that sender owns assets to send
- Check transaction nonce

Transaction Nonce

Transaction Verification



- Verifying sender and transaction integrity
- Confirm that sender owns assets to send
- Check transaction nonce

Nonce (Number used Once)

A one-off - something meant for use exactly once

Transaction Nonce

A unique integer ID, incremented by 1, that identifies each transaction initiated from a particular account

Transaction Nonce



Effectively a transaction counter

Maintained by each node

Incremented by 1 for each transaction

Transaction Nonce



Nodes must follow rules while generating nonce

- Nonces must be generated in order
- Nonces generated in unit increments

Transaction Nonce



Uses of nonce

- Ordering
- De-duplication (prevent double-counting)

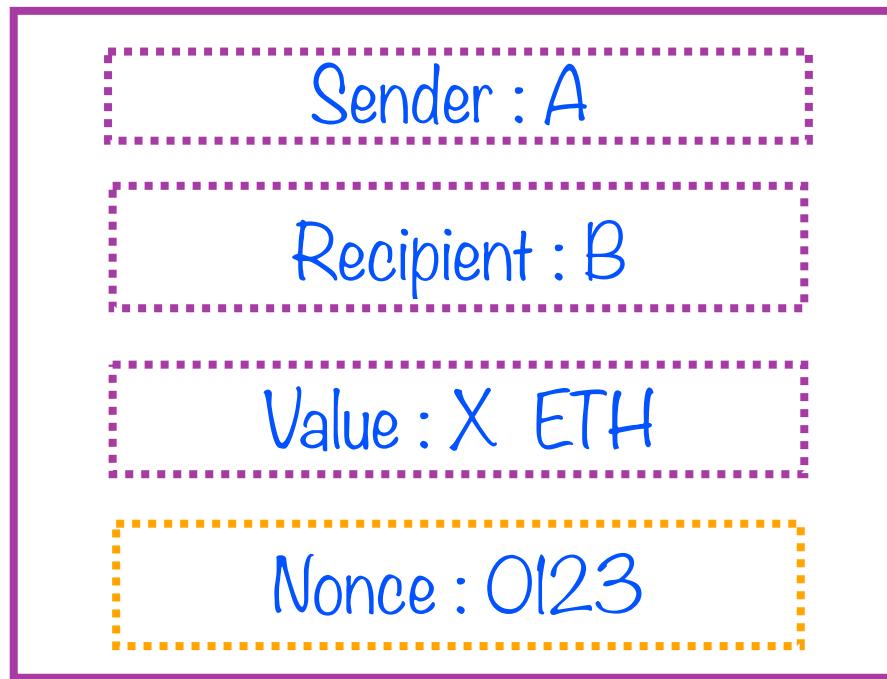
Transaction Nonce



Nonce : Blockchain transaction

Cheque number : Bank transaction

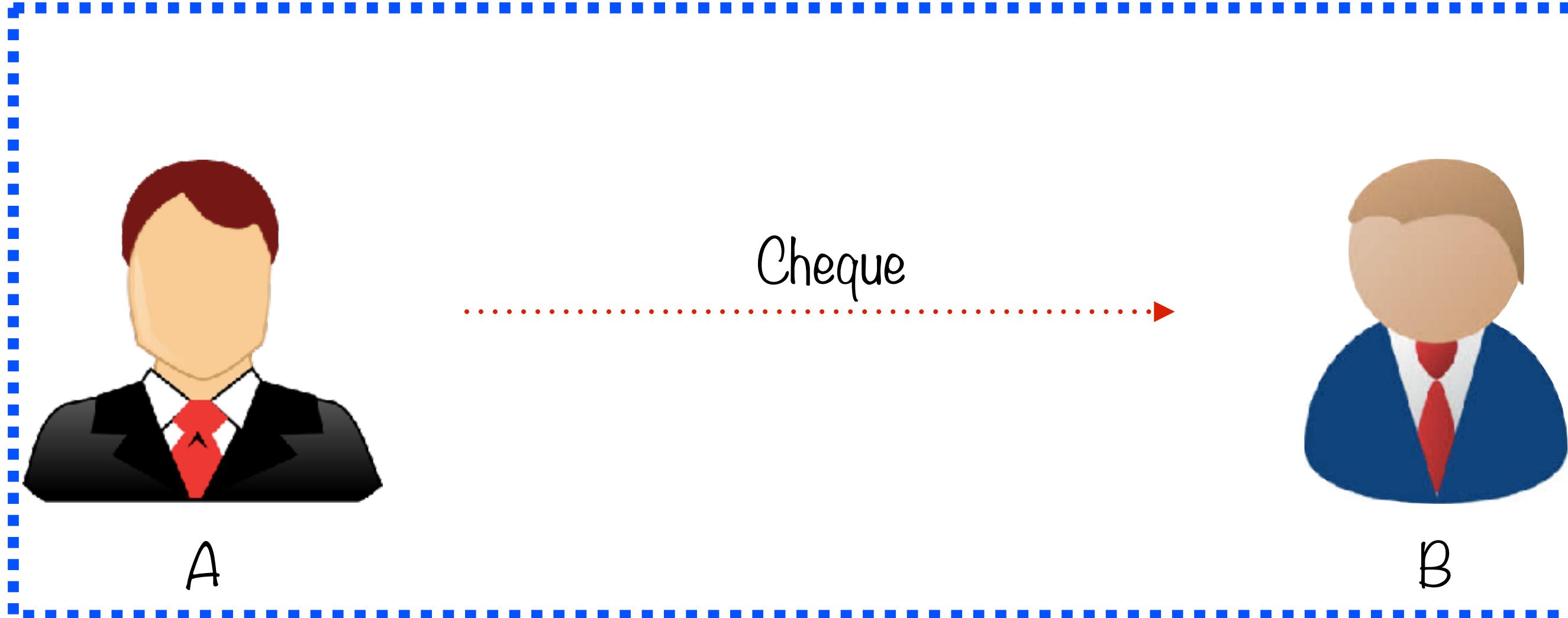
Cheque Number asNonce



Nonce : Blockchain transaction

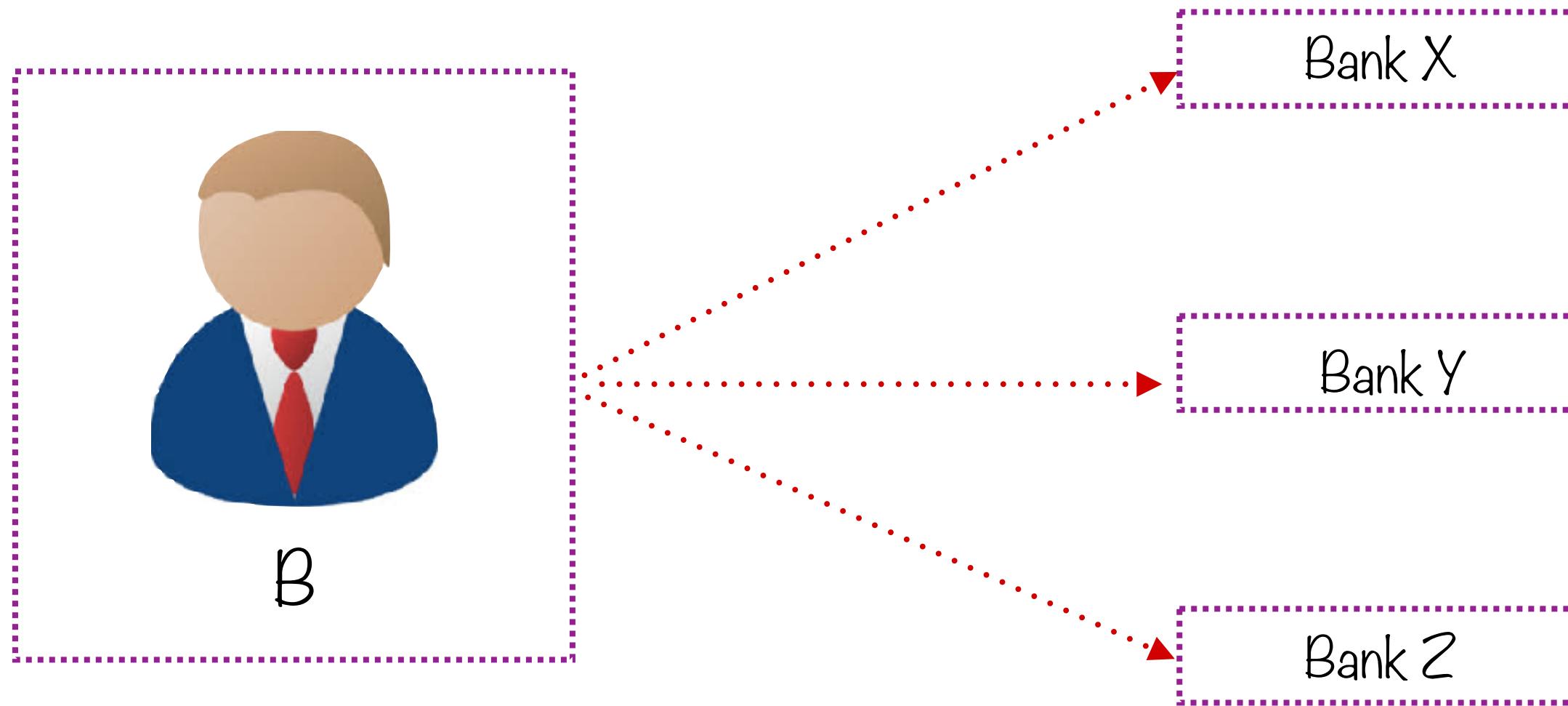
Cheque number : Bank transaction

Cheque Number asNonce



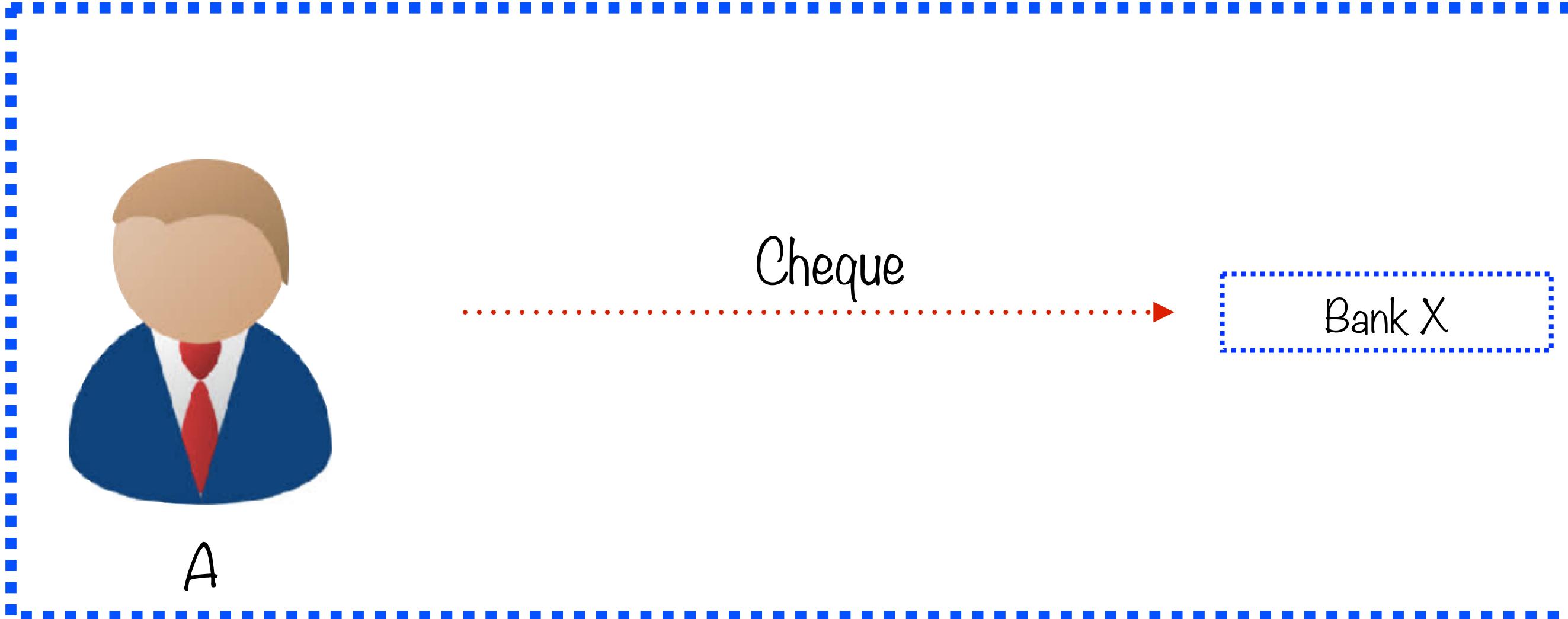
A has given a cheque to B issued by Bank Of R

Cheque Number asNonce



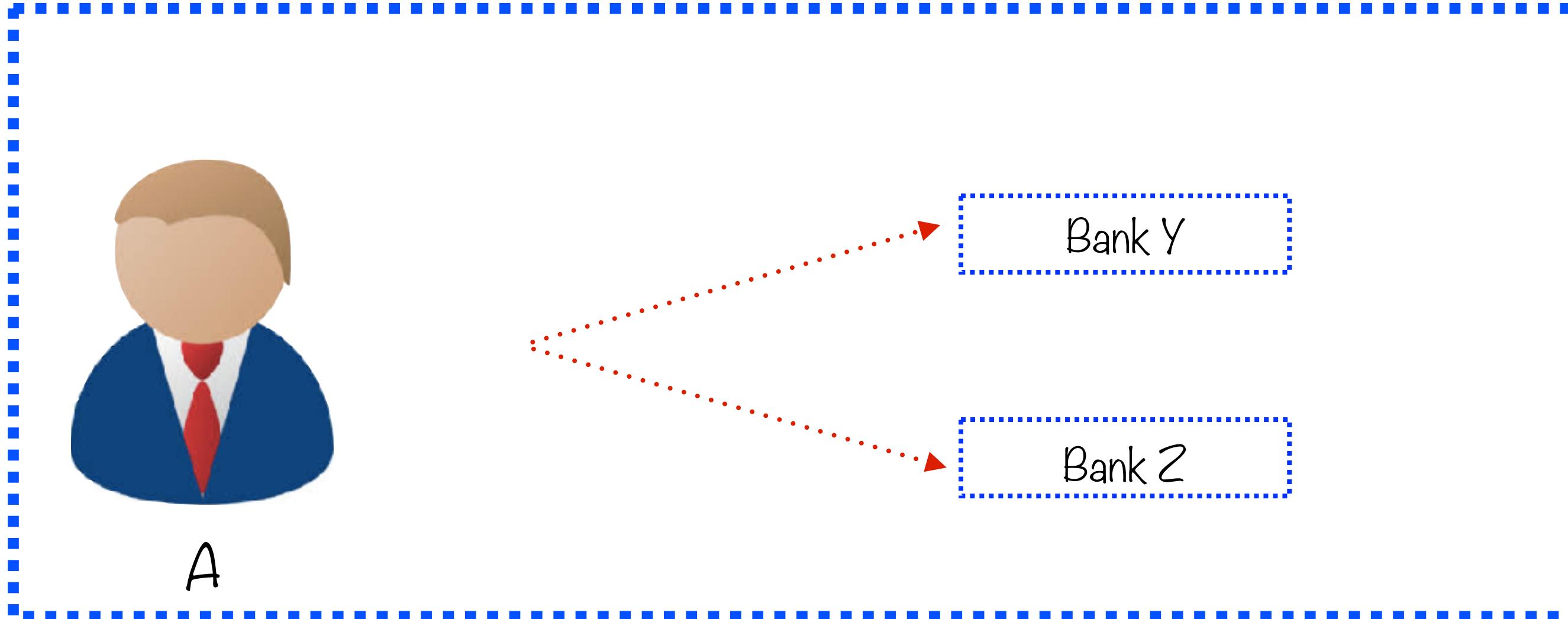
B maintains accounts with multiple banks

Cheque Number as Nonce



B deposits cheque in bank X

Cheque Number as Nonce



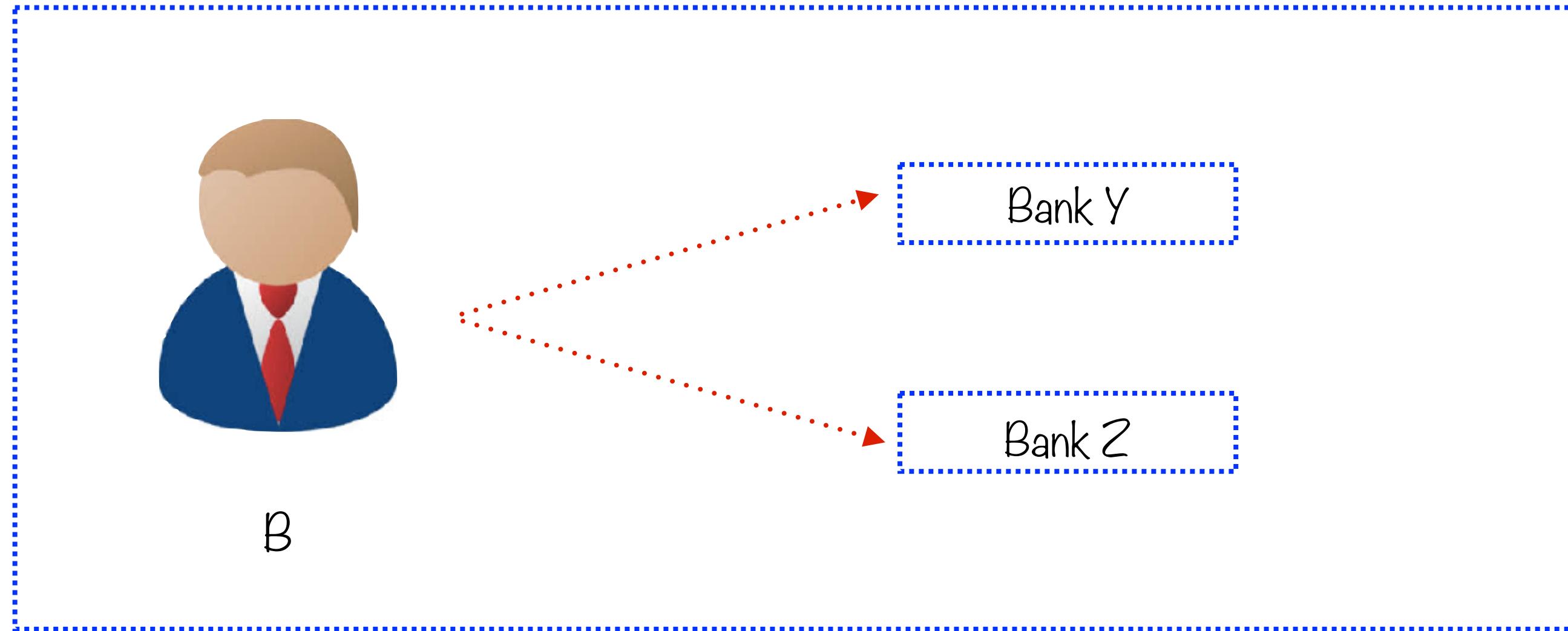
B can not now deposit the same cheque again with Bank 2

Cheque Number asNonce



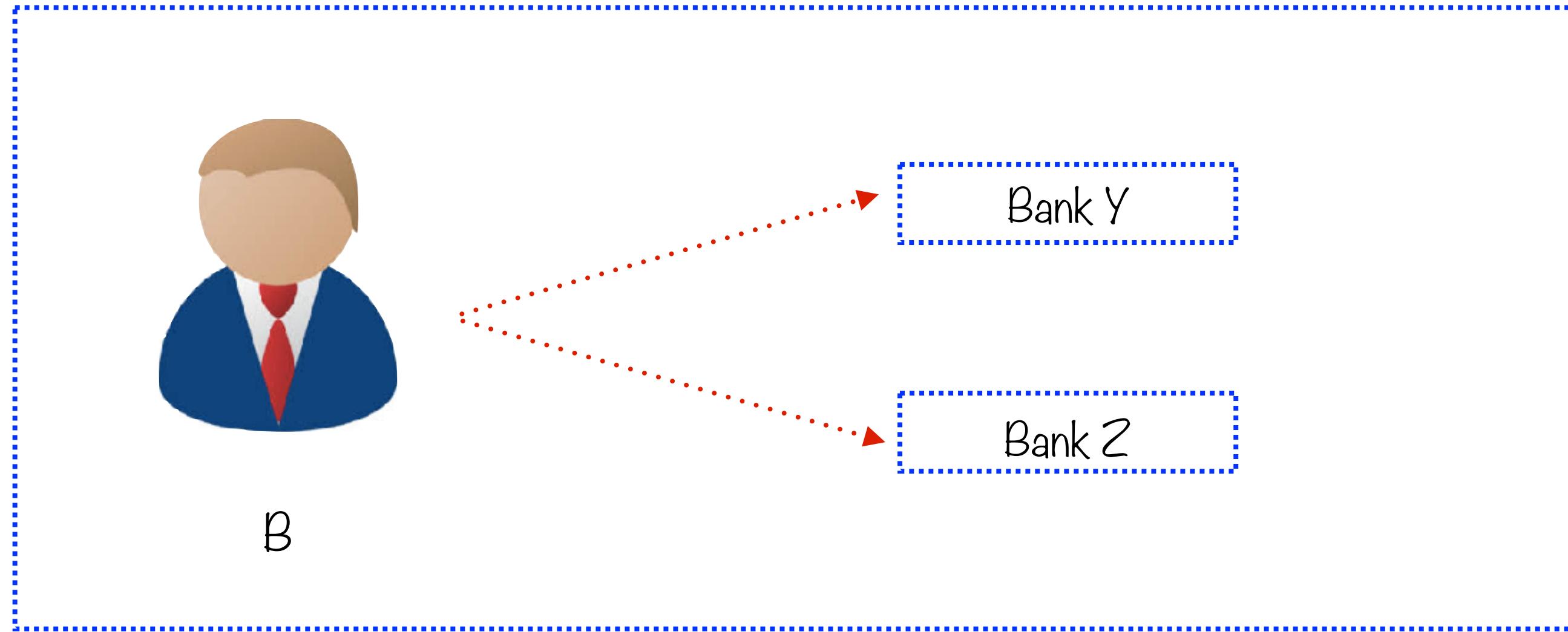
That's because every cheque contains a serial number which is unique and related to issuer's account

Cheque Number asNonce



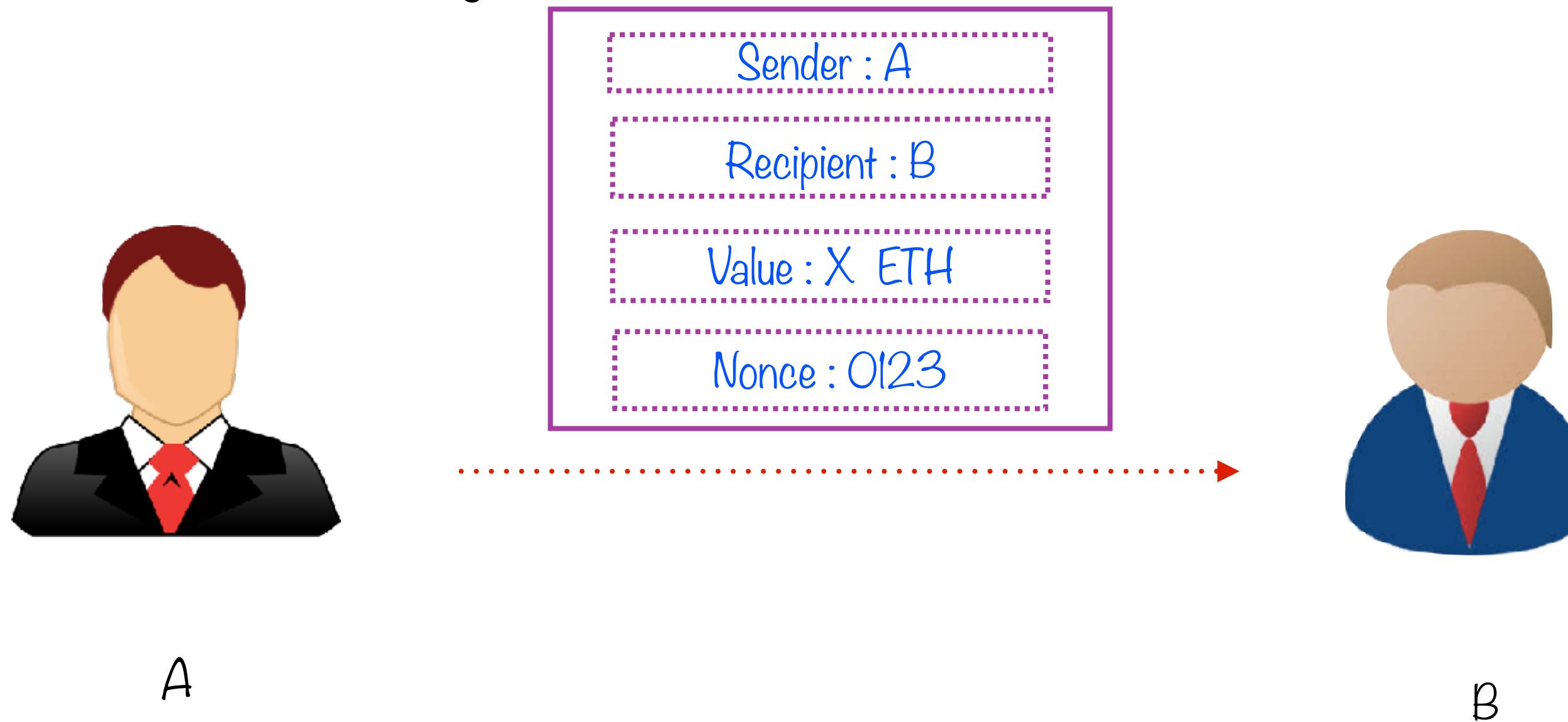
Bank Y and Bank Z will each verify the serial number with the issuing bank of the cheque - preventing double-deposit

Cheque Number asNonce



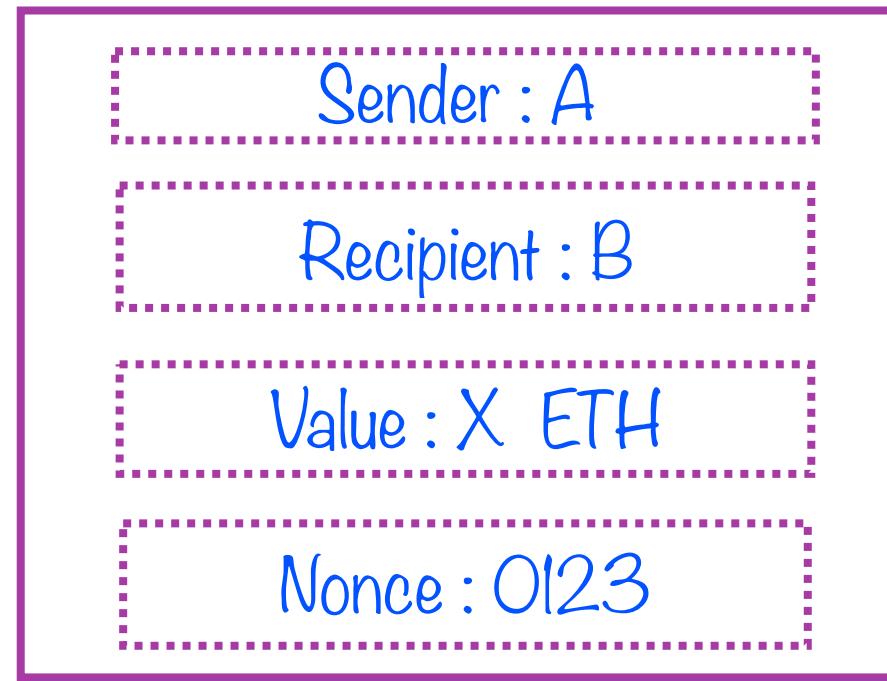
Cheque numbers prevent double-deposit of the same cheque

Cheque Number asNonce



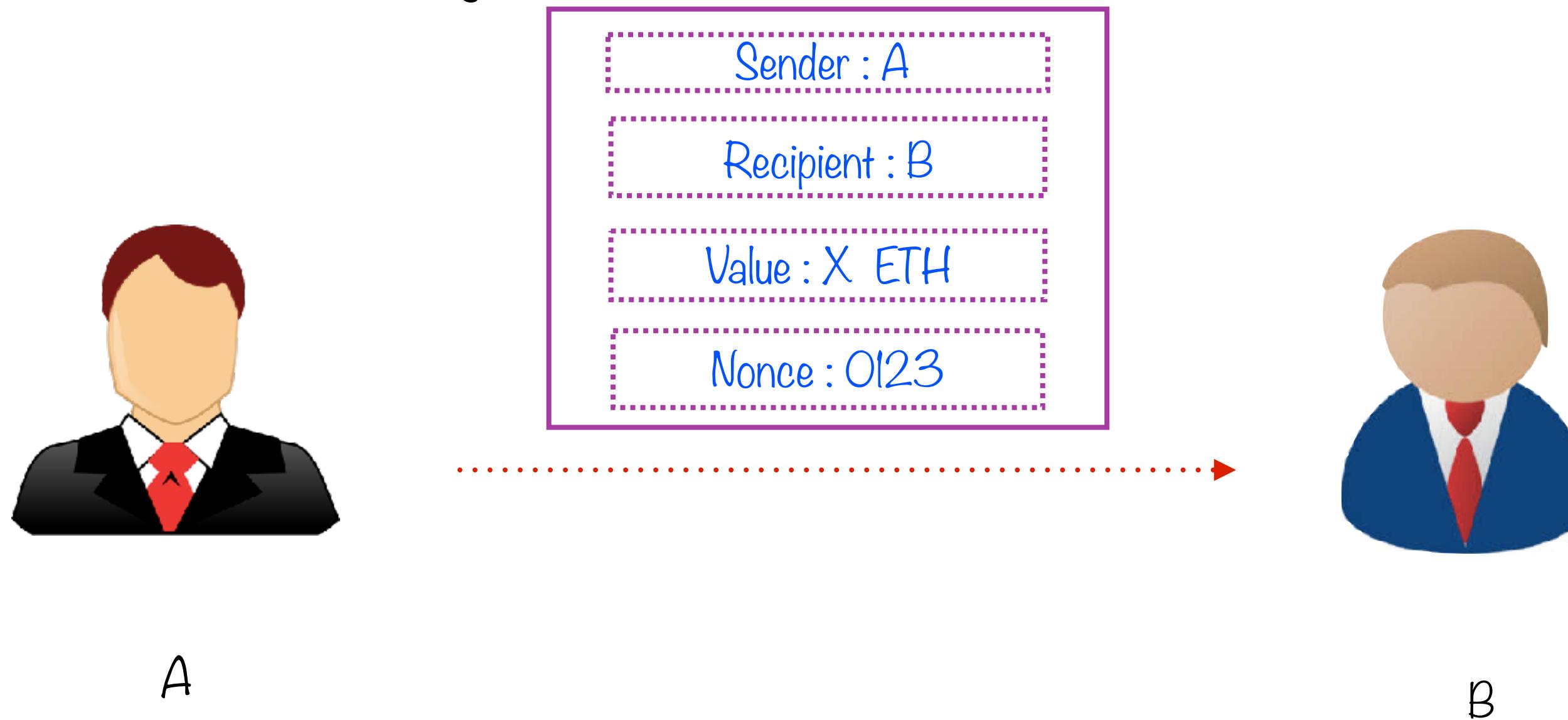
Nonce plays the same role in an Ethereum network

Cheque Number asNonce



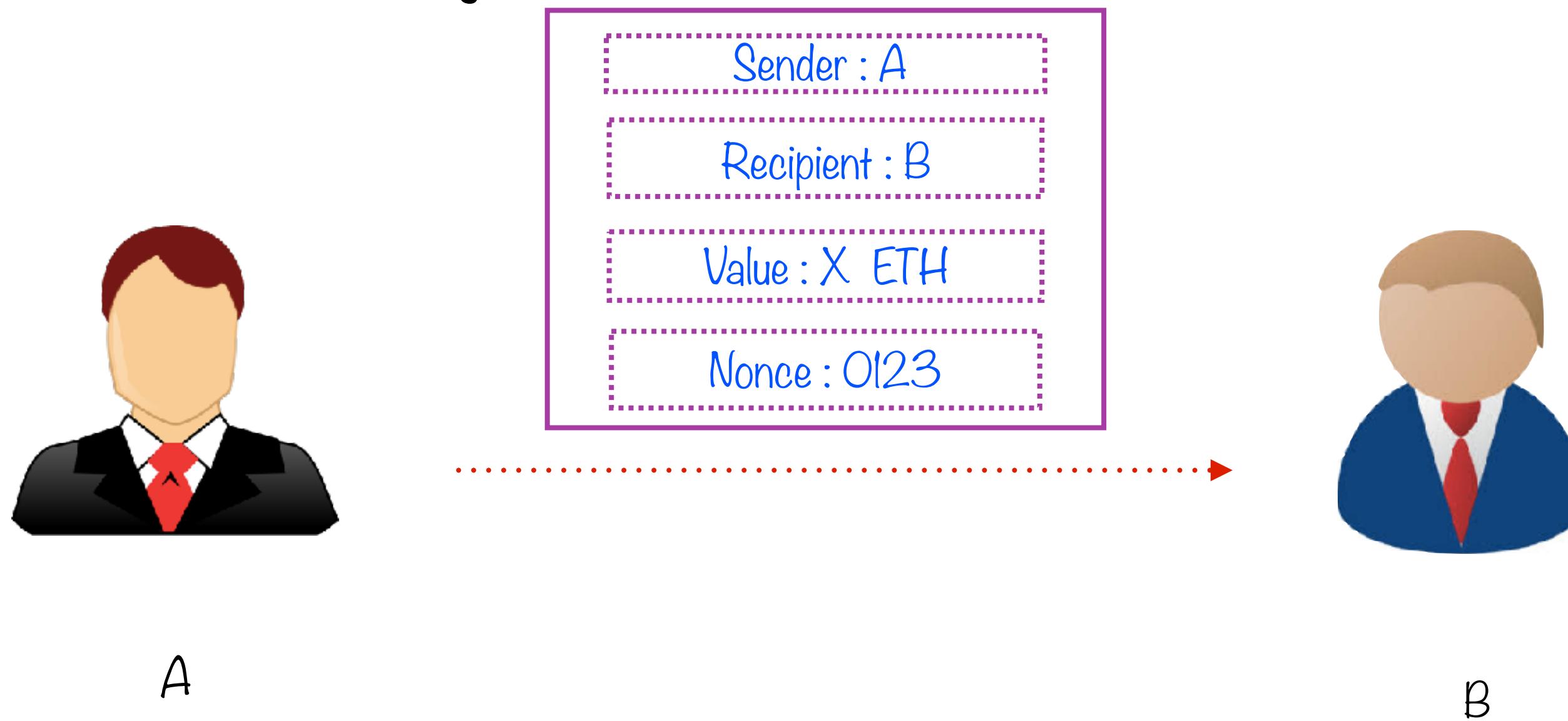
A sends B a sum of X units of ETH

Cheque Number asNonce



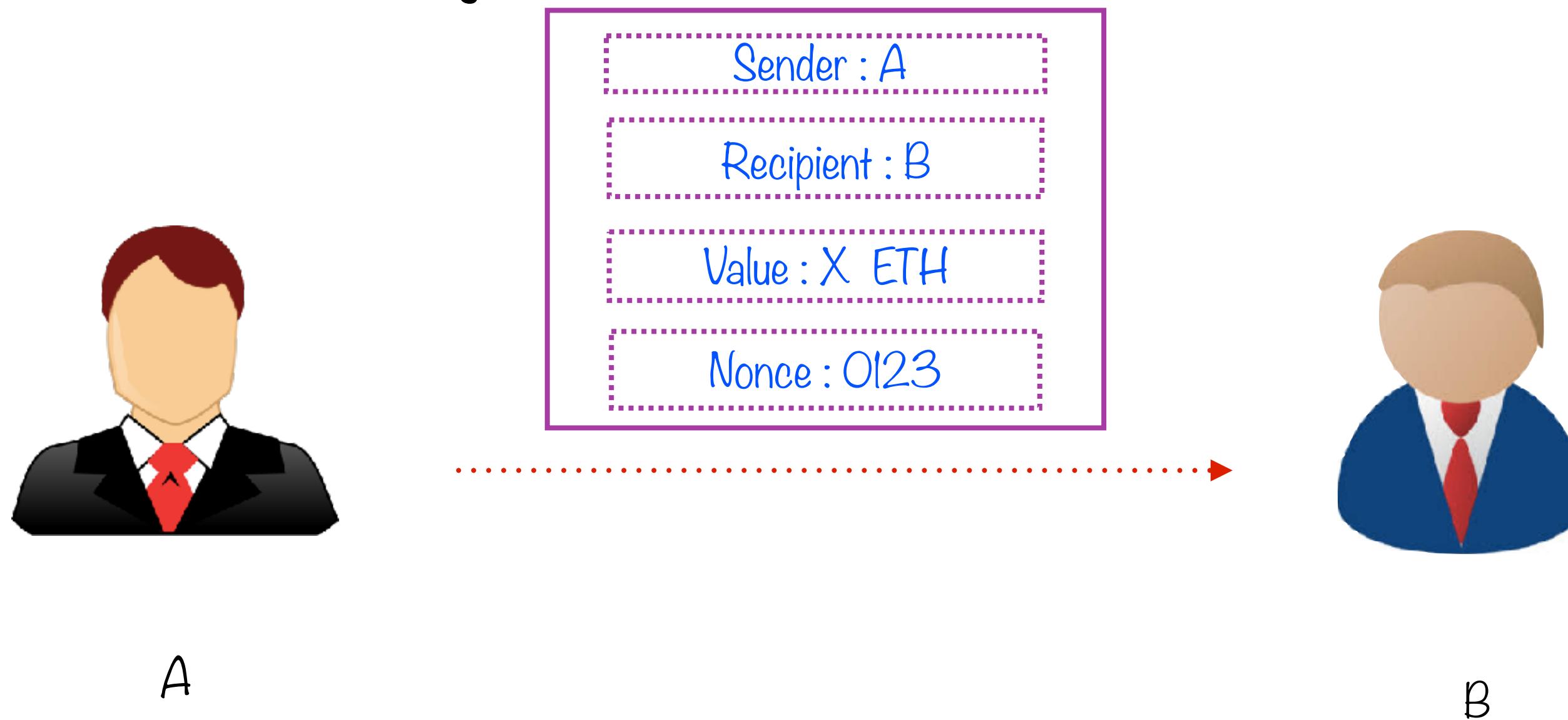
A sends his signed Transaction with the nonce **0123** to B

Cheque Number asNonce



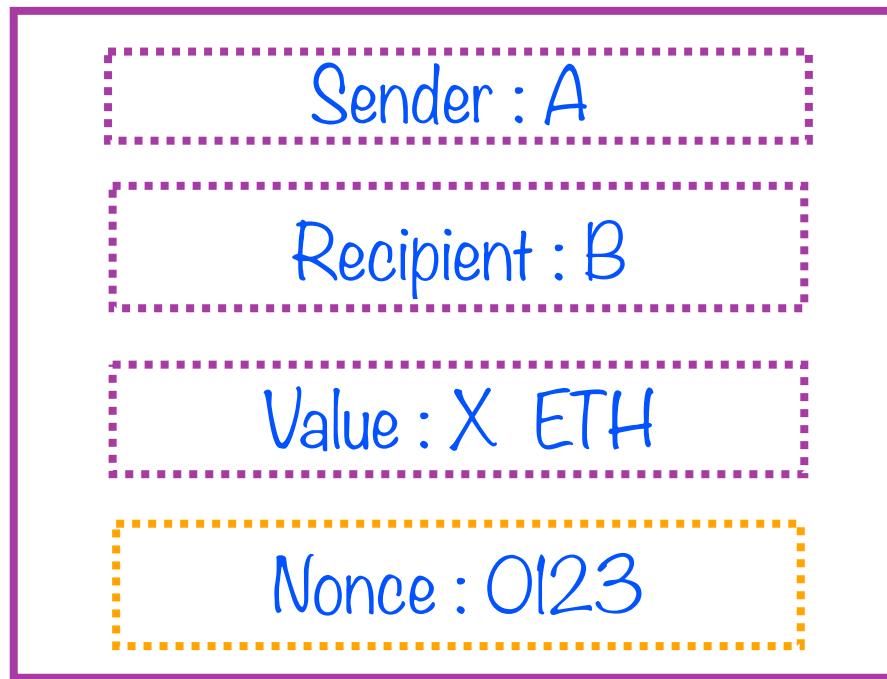
This is broadcast to the entire network so that no double-deposit occurs

Cheque Number asNonce



B gets X ETH after the miners in the network verify the the transaction

Cheque Number asNonce

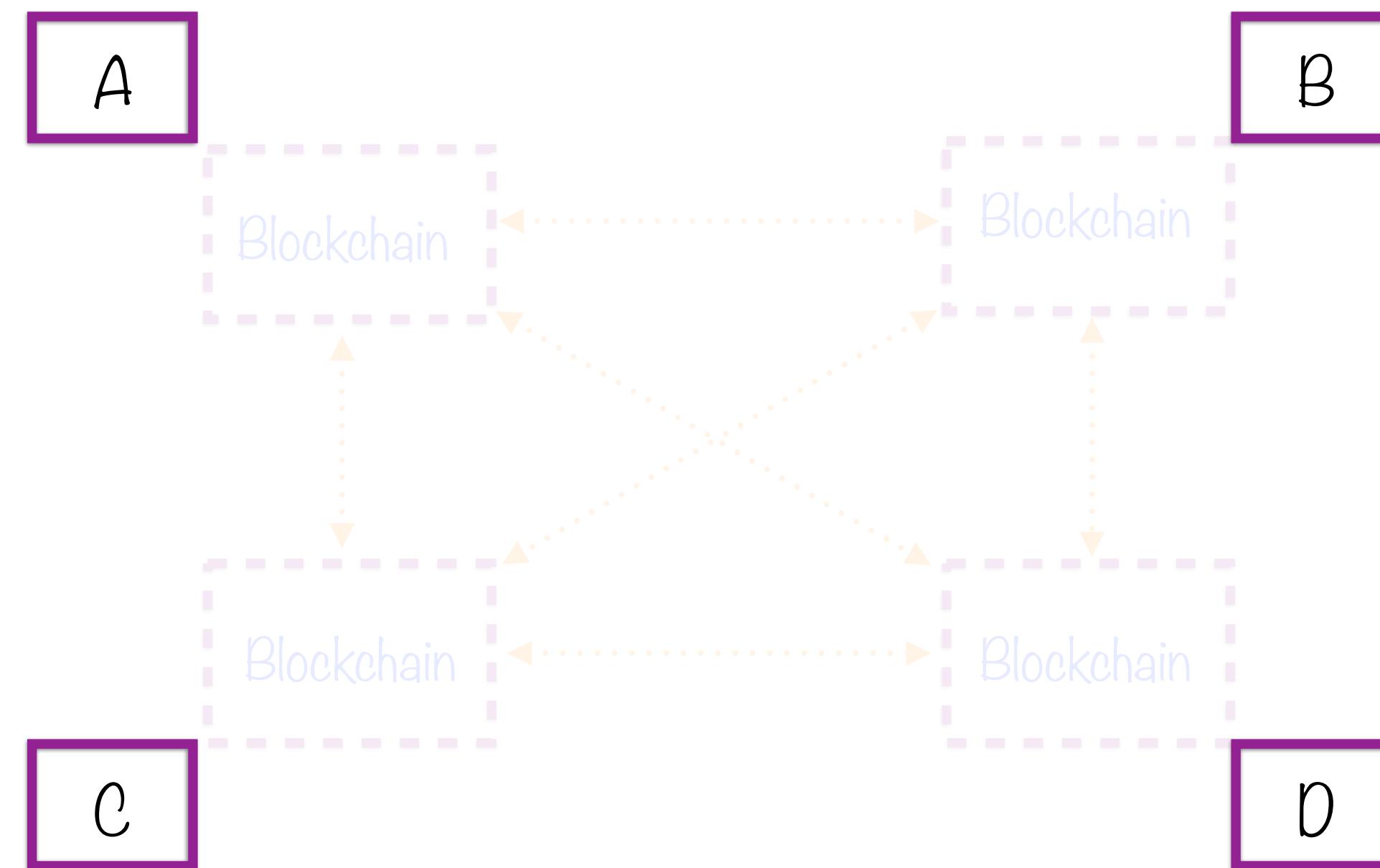


Nonce : Blockchain transaction

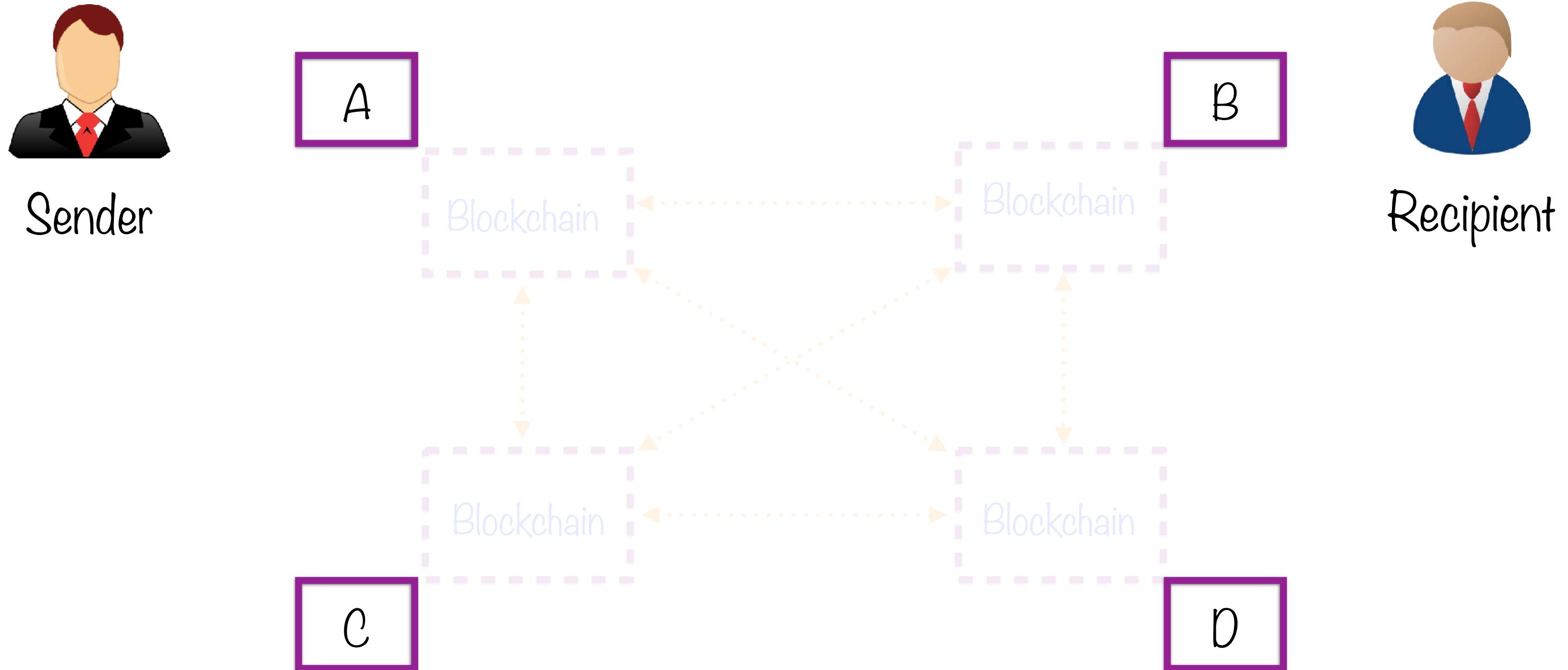
Cheque number : Bank transaction

Transaction Ordering and Consensus

Nodes in a Blockchain Network



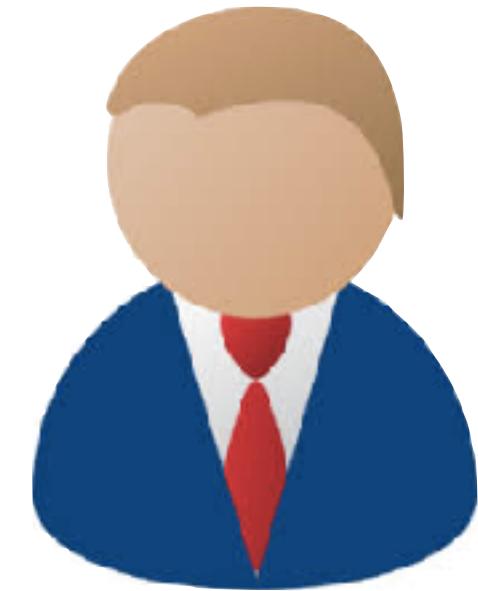
Accounts in a Blockchain Network



Transaction Verification



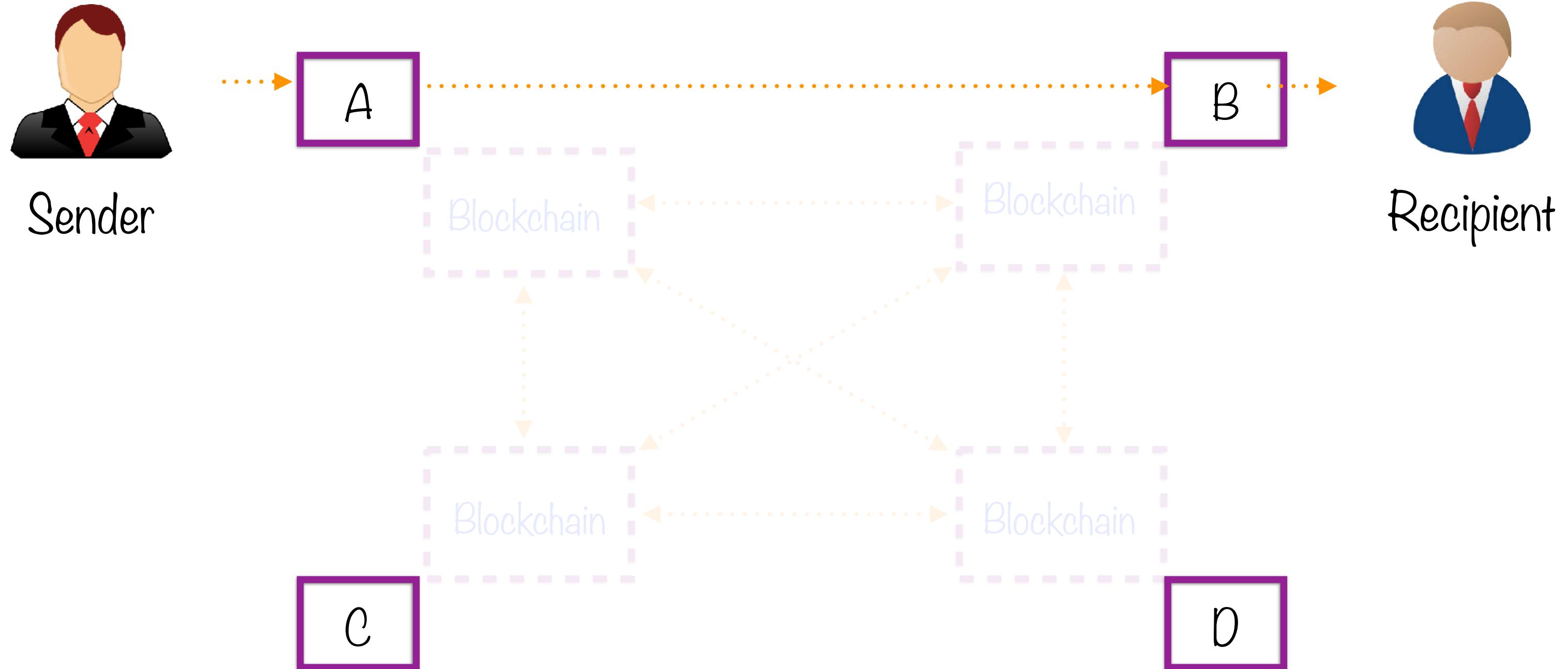
A
Sender



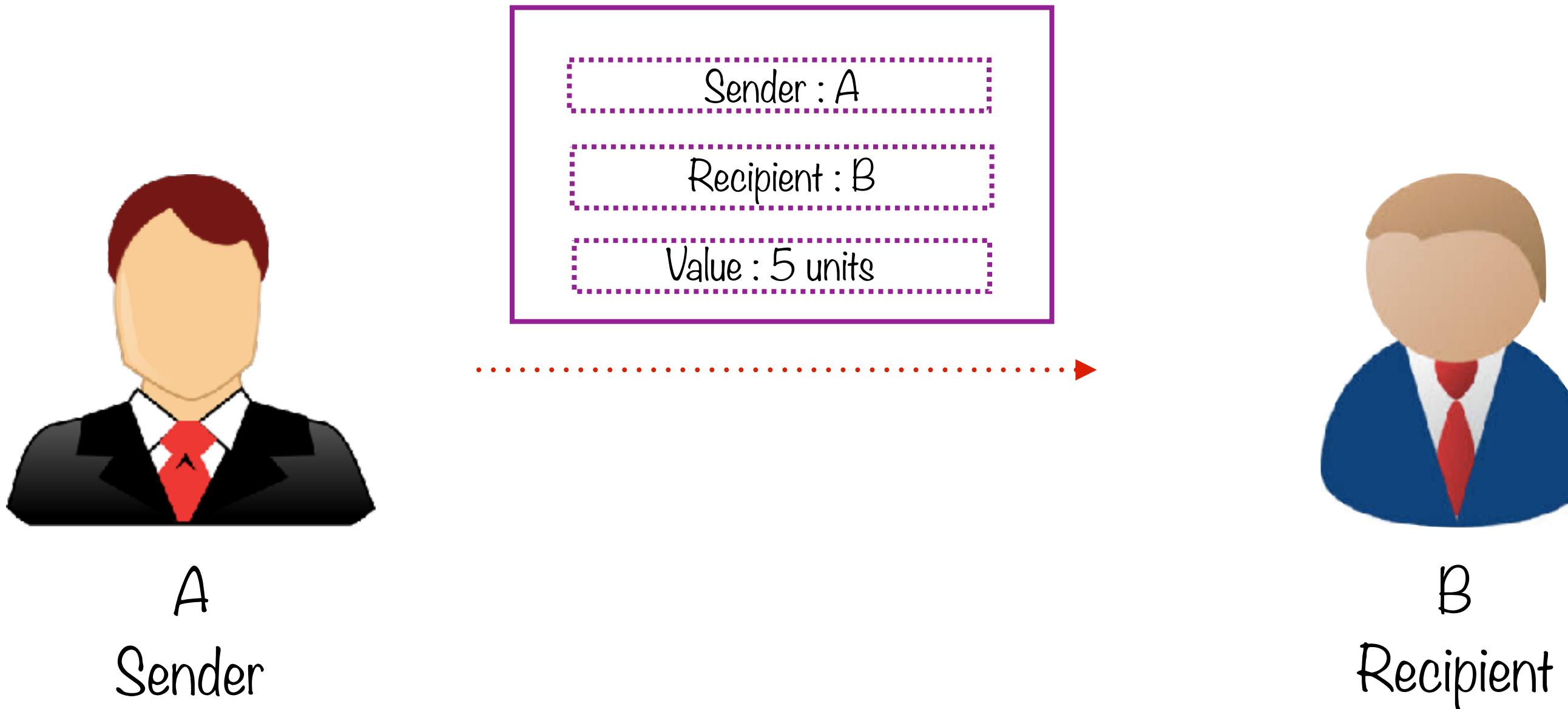
B
Recipient



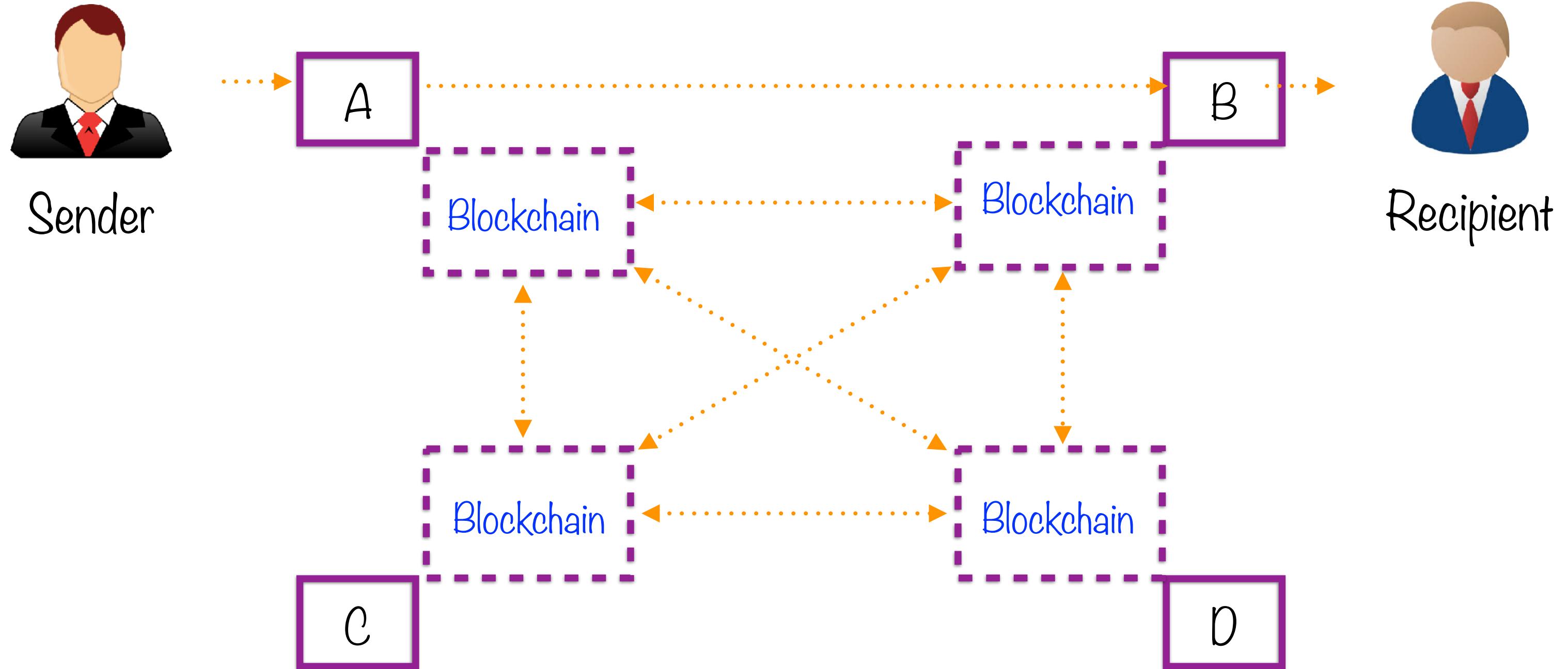
Accounts in a Blockchain Network



Transaction Verification

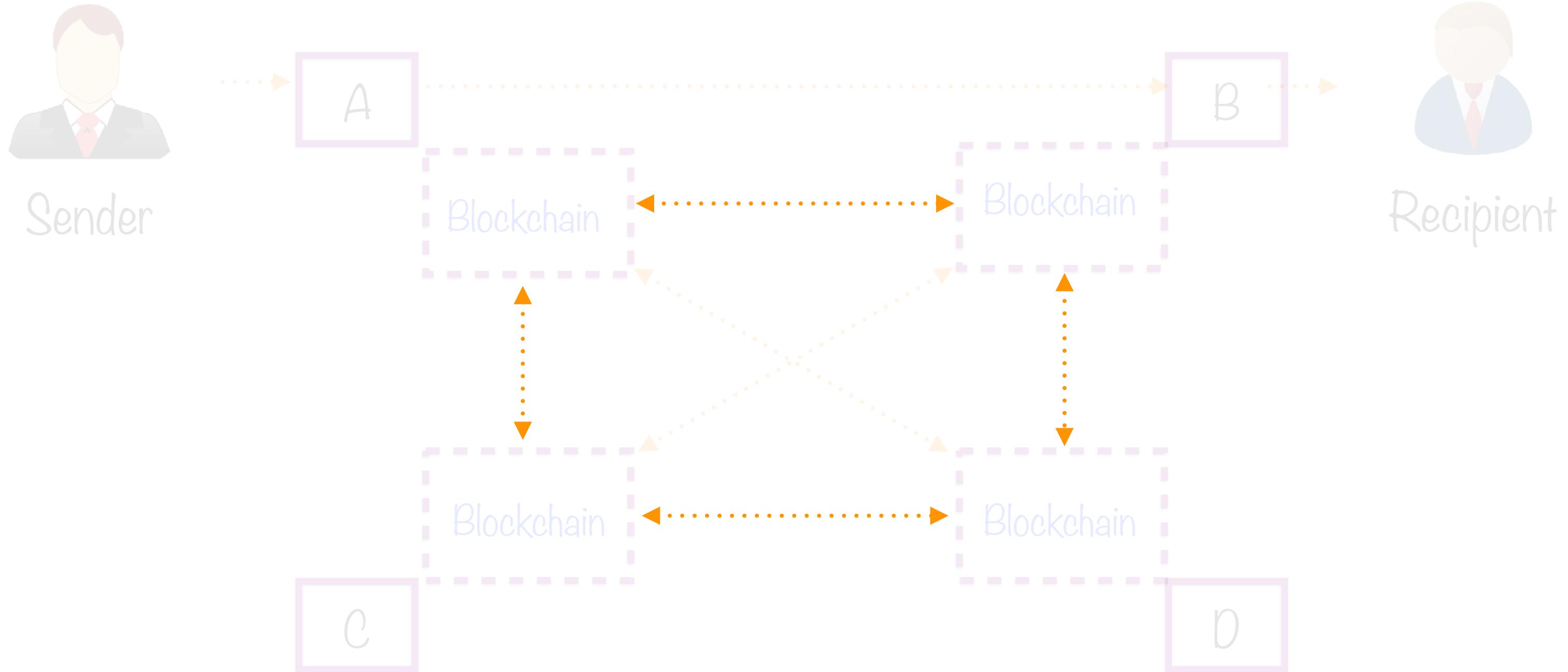


Accounts in a Blockchain Network



Every transaction is broadcast across the network

Accounts in a Blockchain Network



Every transaction is broadcast across the network

Consensus

Many transactions may be broadcast simultaneously

Many miners might pick these transactions up in differing order

How is the order determined?

- Via a consensus algorithm



Two Consensus Algorithms

Proof-of-work

Currently used in Ethereum to achieve trustless consensus

Proof-of-Stake

Ethereum actively moving to switch to this algorithm instead

Consensus by Proof-of-work

Two Consensus Algorithms

Proof-of-work

Currently used in Ethereum to achieve trustless
consensus

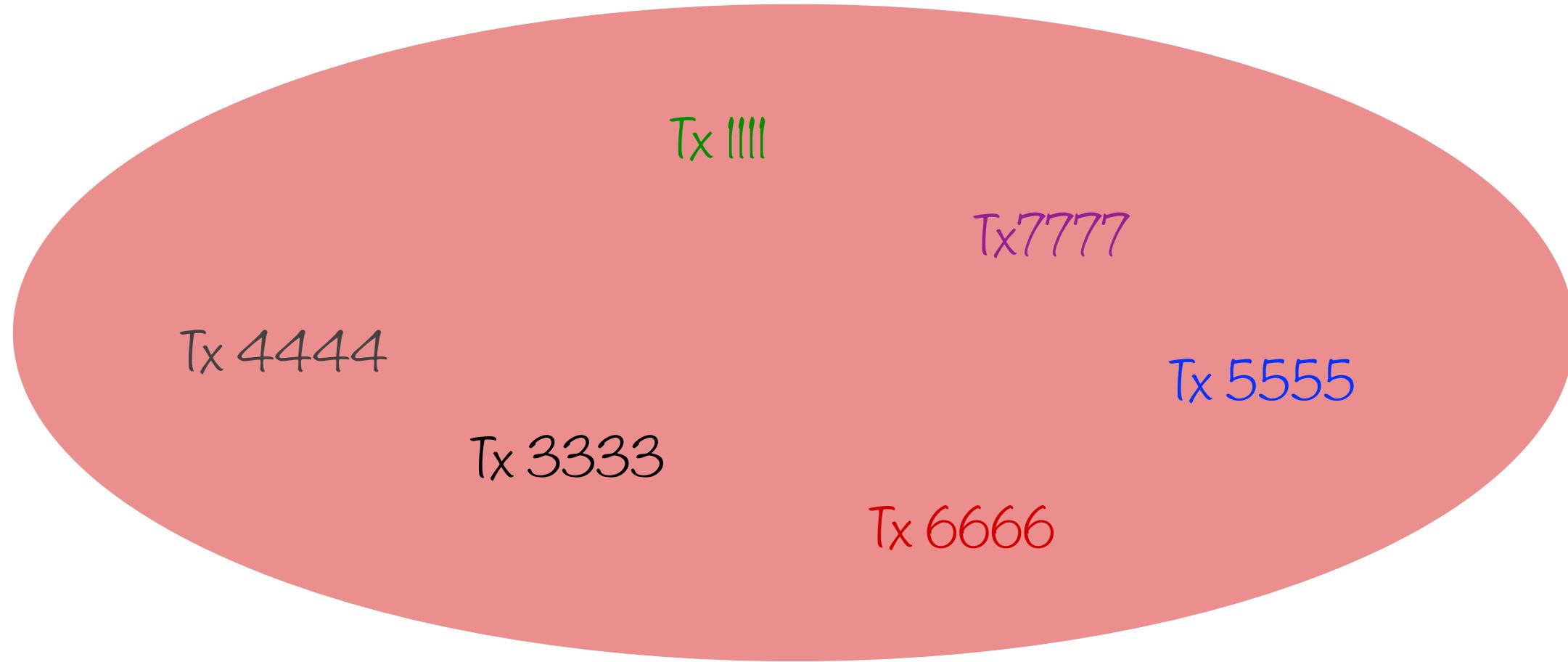
Proof-of-Stake

Ethereum actively moving to switch to this
algorithm instead

Determining order of transactions in blockchain is a right

To gain that right, do some work and prove that you did it

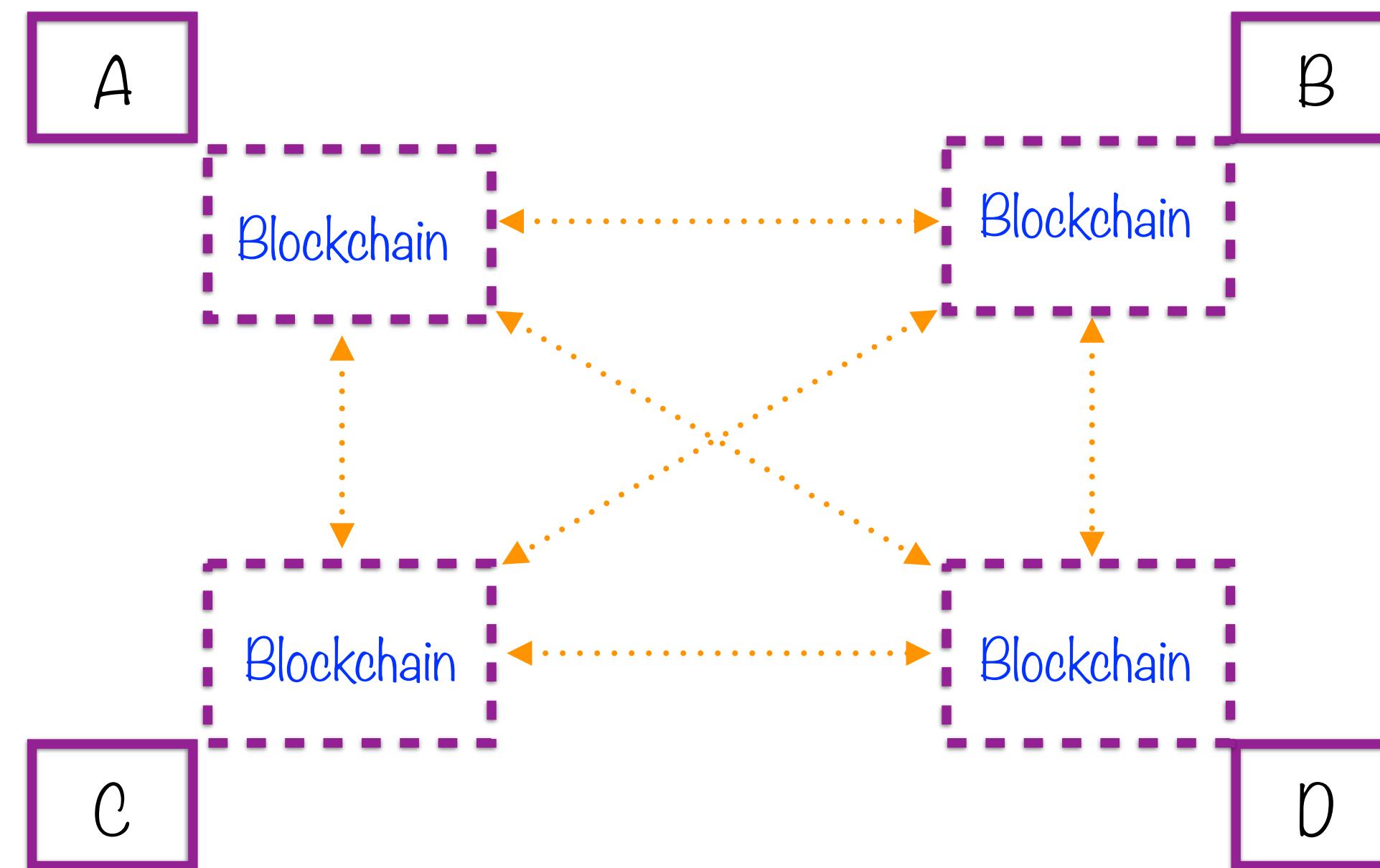
Unverified Transaction Pool



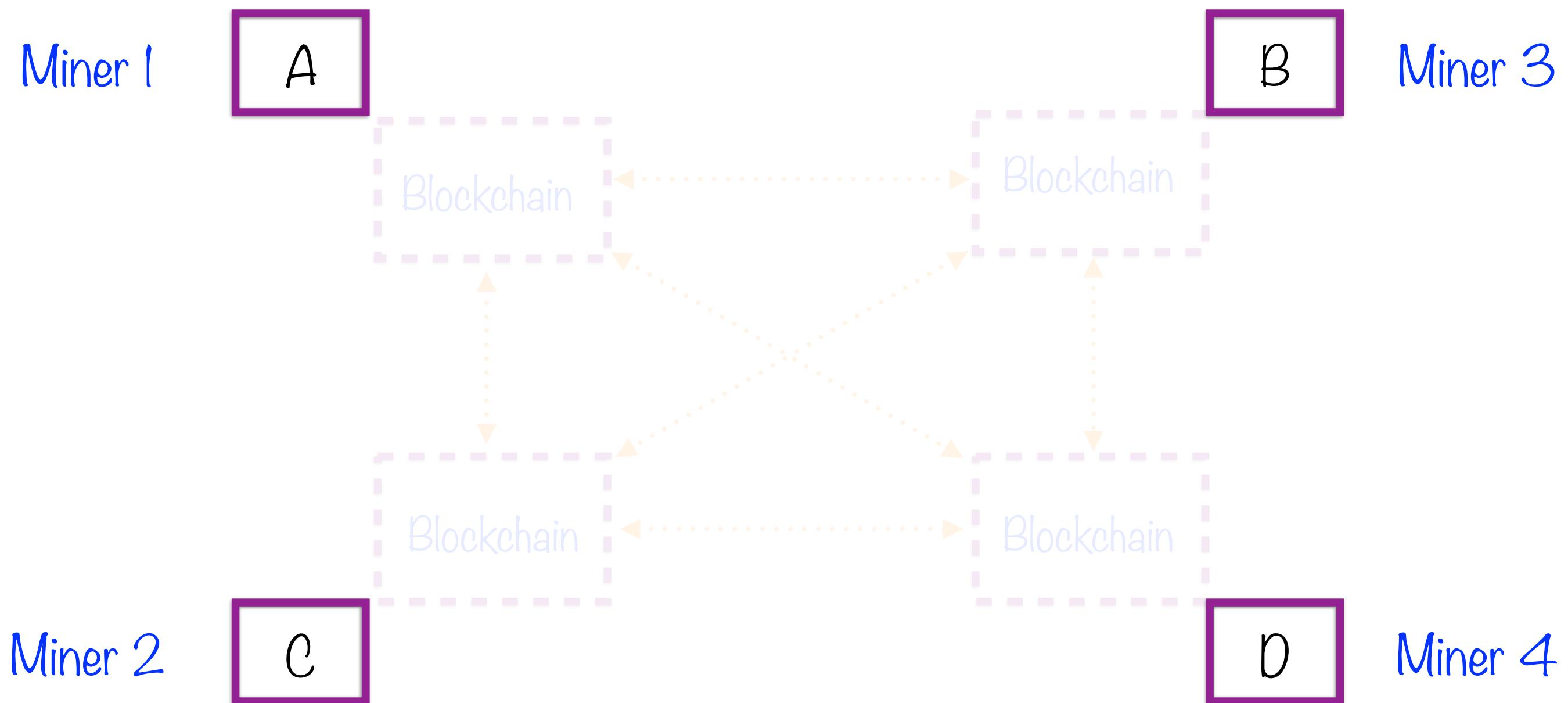
Initiated transaction requests are broadcast to the entire blockchain network

The process of generating proof-of-work is called mining and the participants are called miners

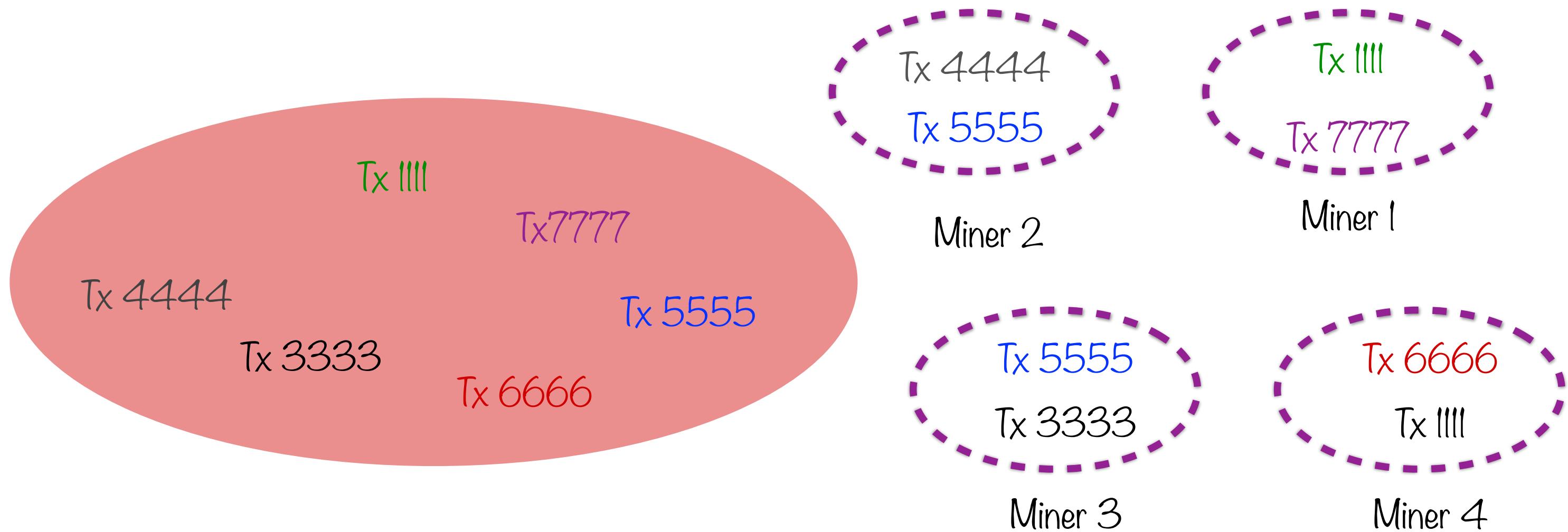
Nodes in a Blockchain Network



Nodes in a Blockchain Network

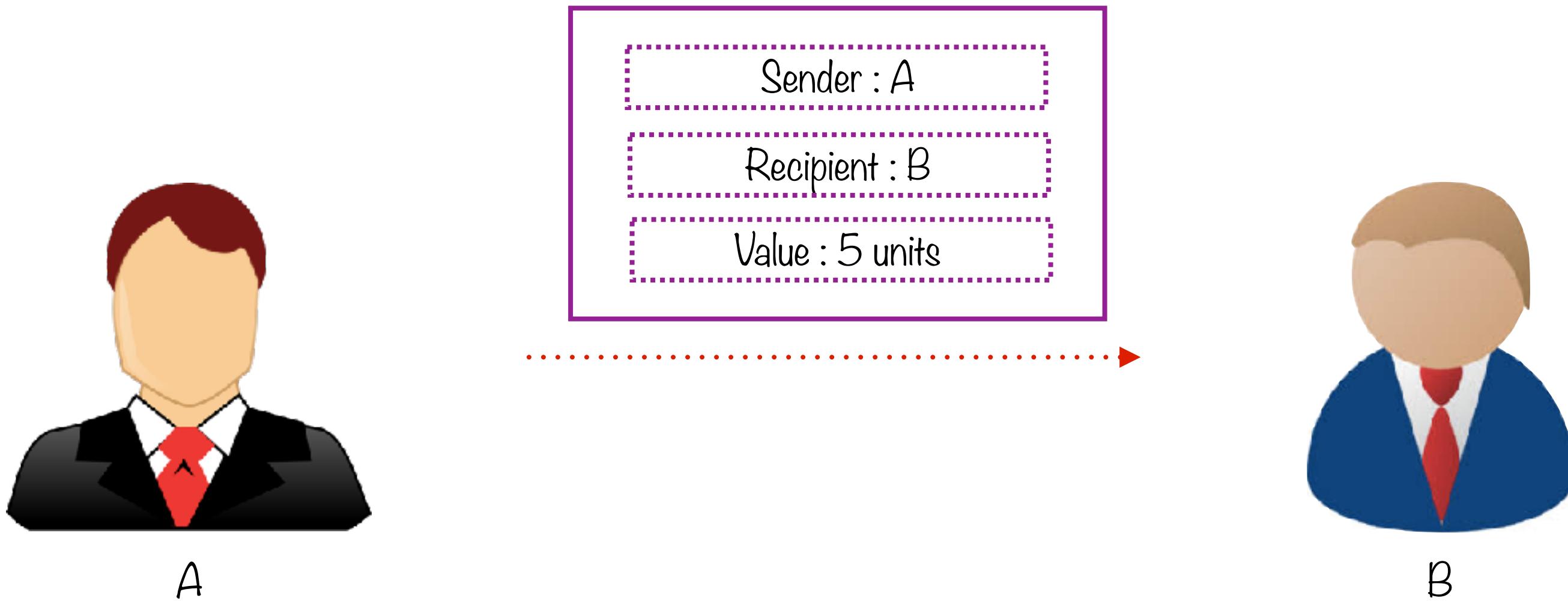


Unverified Transaction Pool



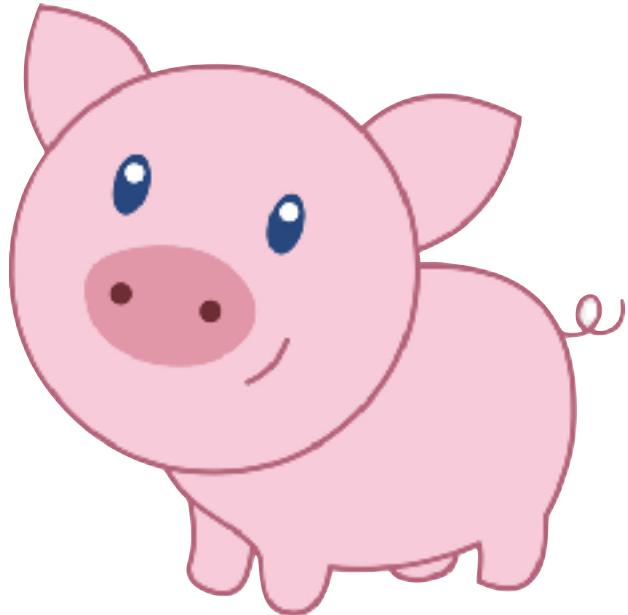
Miners add the transactions to their blocks

Transaction Verification



In order to incentivize miners to accept a transaction, the initiator must pay a fee

Gas

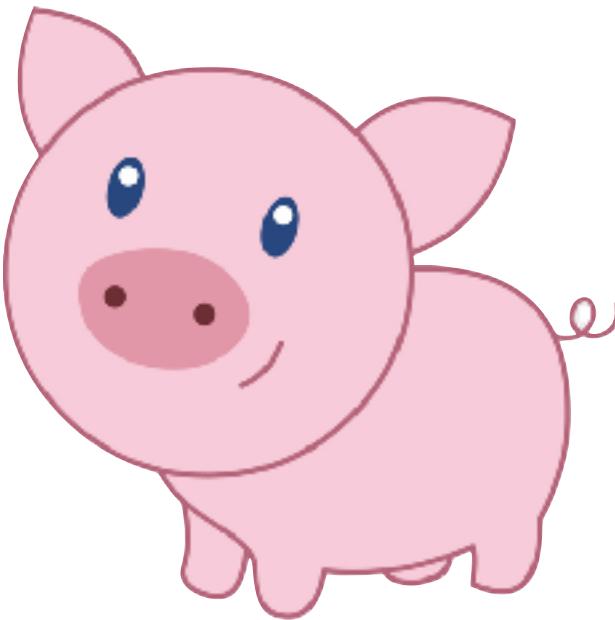


Computation work need to verify a transaction is called
Gas

Each transaction contains the **Gas Price** that the initiator
is willing to pay

Each transaction also contains a hard absolute **Gas Limit**

Gas



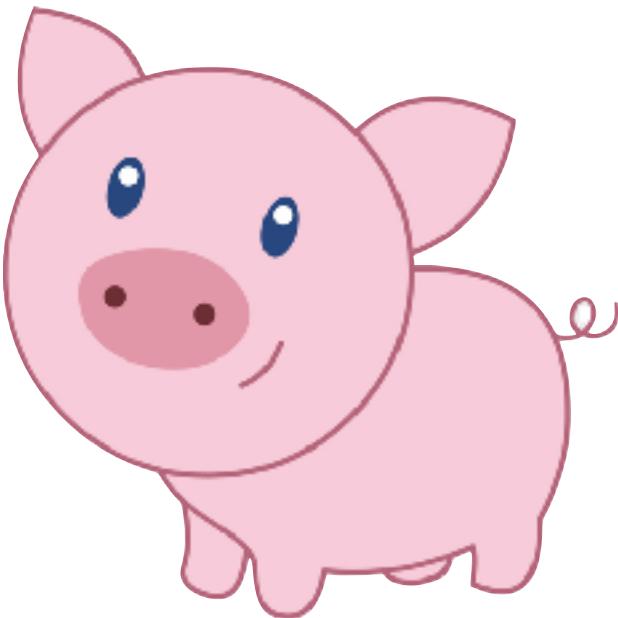
Every node must be running the Ethereum Virtual Machine (EVM)

EVM is a common runtime environment

Runs during verification protocol

EVM measures gas consumption

Miner's Prize



TotalCost = gasUsed \times gasPrice

The miner that verifies a transaction collects this prize

This miner also determines the next block added to the chain

Proof-of-work

Miners compete to verify transactions

In order to win transaction prizes

Verifying transactions == Determining order and state
of each node's blockchain

Proof-of-work

Verifying transactions is intentionally made very difficult

Miners must incur significant costs to do so

Deters malicious agents seeking to control state of system

Proof-of-work

Each initiated transaction goes into an unverified transaction pool

Miners take transactions out of that pool

Each miner verifies validity of transaction

Then does work and submits proof

What's the Work?

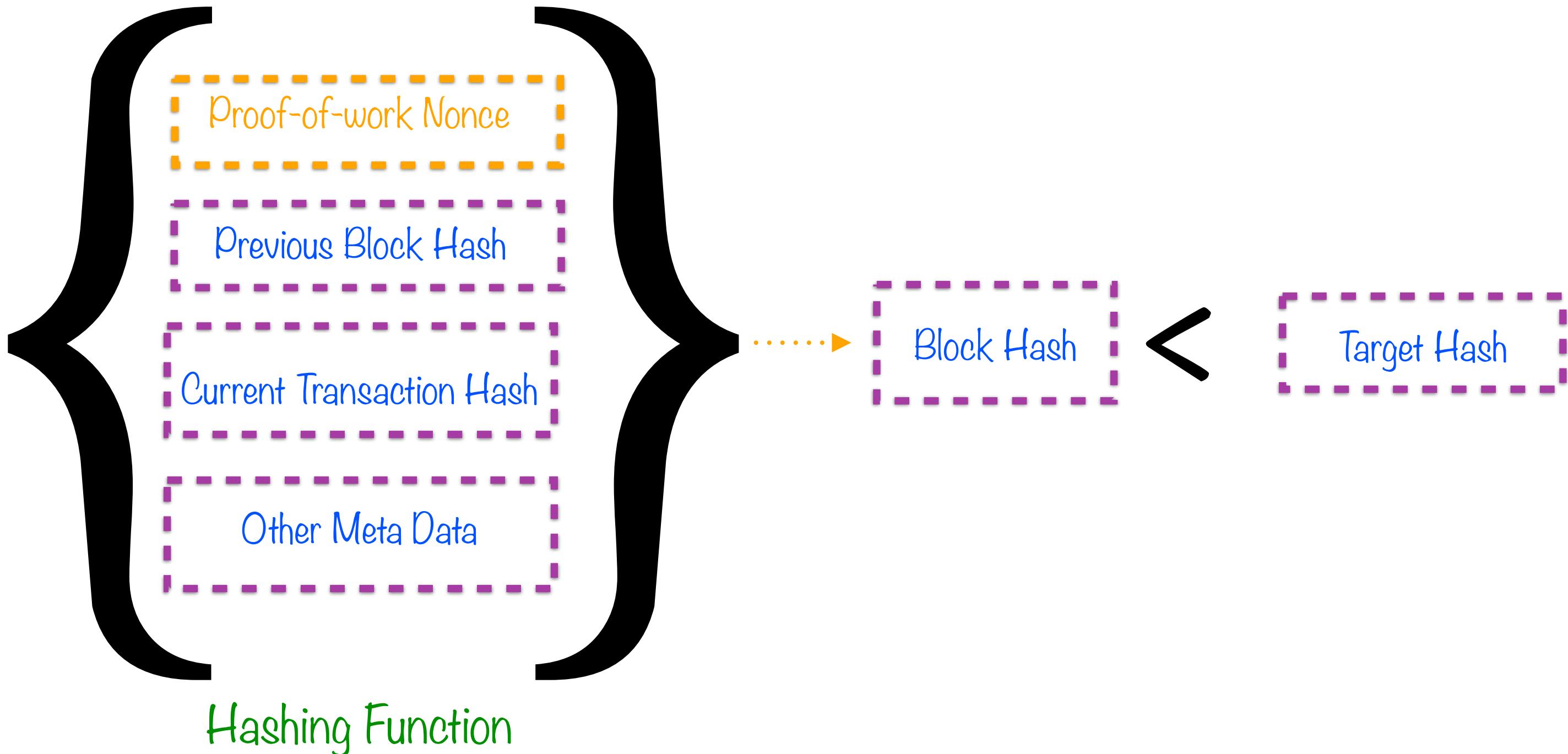
Work: Find the proof-of-work nonce

Each miner has to find a nonce that

- satisfies an arbitrary, hard condition
- can only be found by trial-and-error

Finding the Proof-of-workNonce

Find Proof-of-workNonce



Find nonce that satisfies this arbitrary, difficult condition

Nonce (Number used Once)

A one-off - something meant for use exactly once

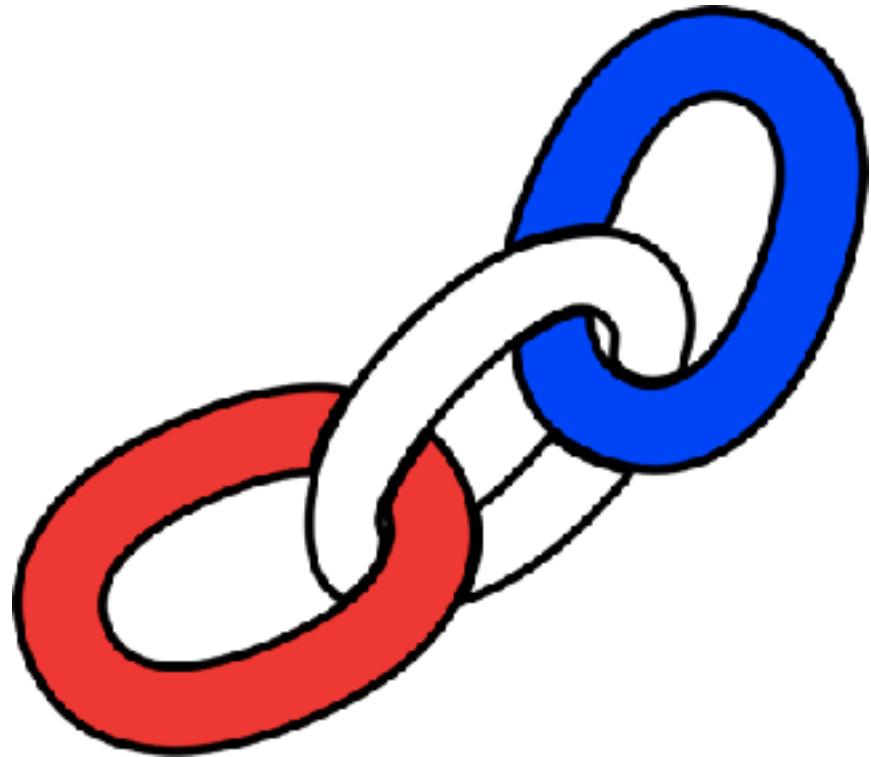
Transaction Nonce

A unique integer ID, incremented by 1, that identifies each transaction initiated from a particular account

Proof-of-work Nonce

A random number used while computing the Merkle Hash (which determine

Distributed Ledger



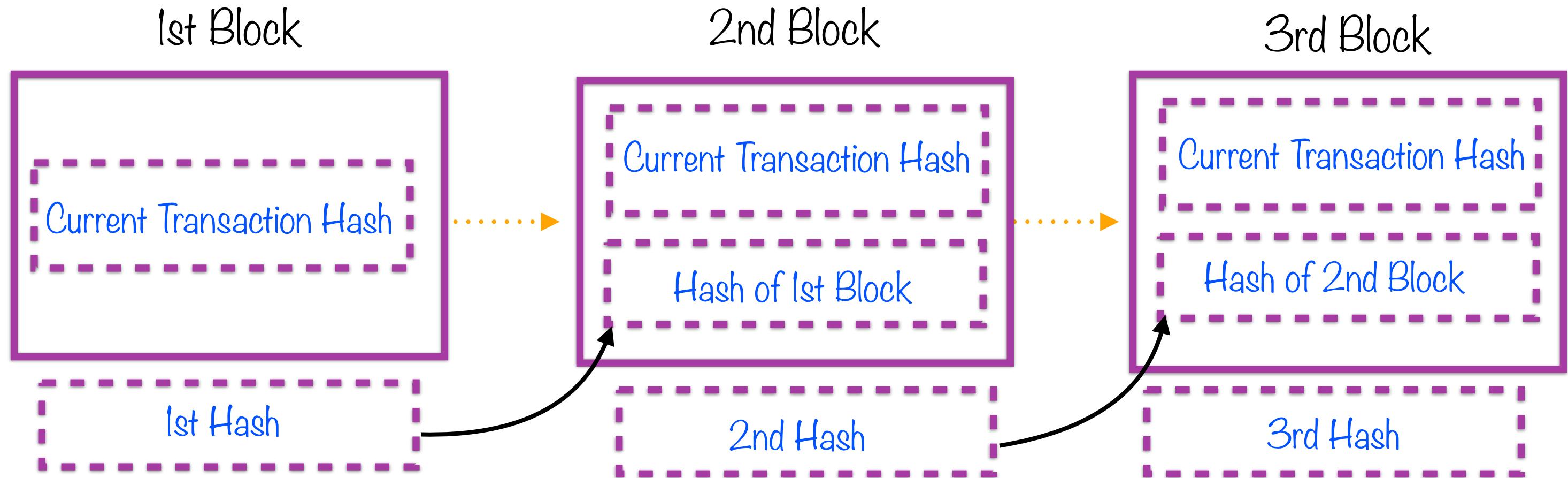
Include hash of each record while computing hash of next record

Divide ledger into **blocks** to simplify

Ledger is now a **chain of blocks**

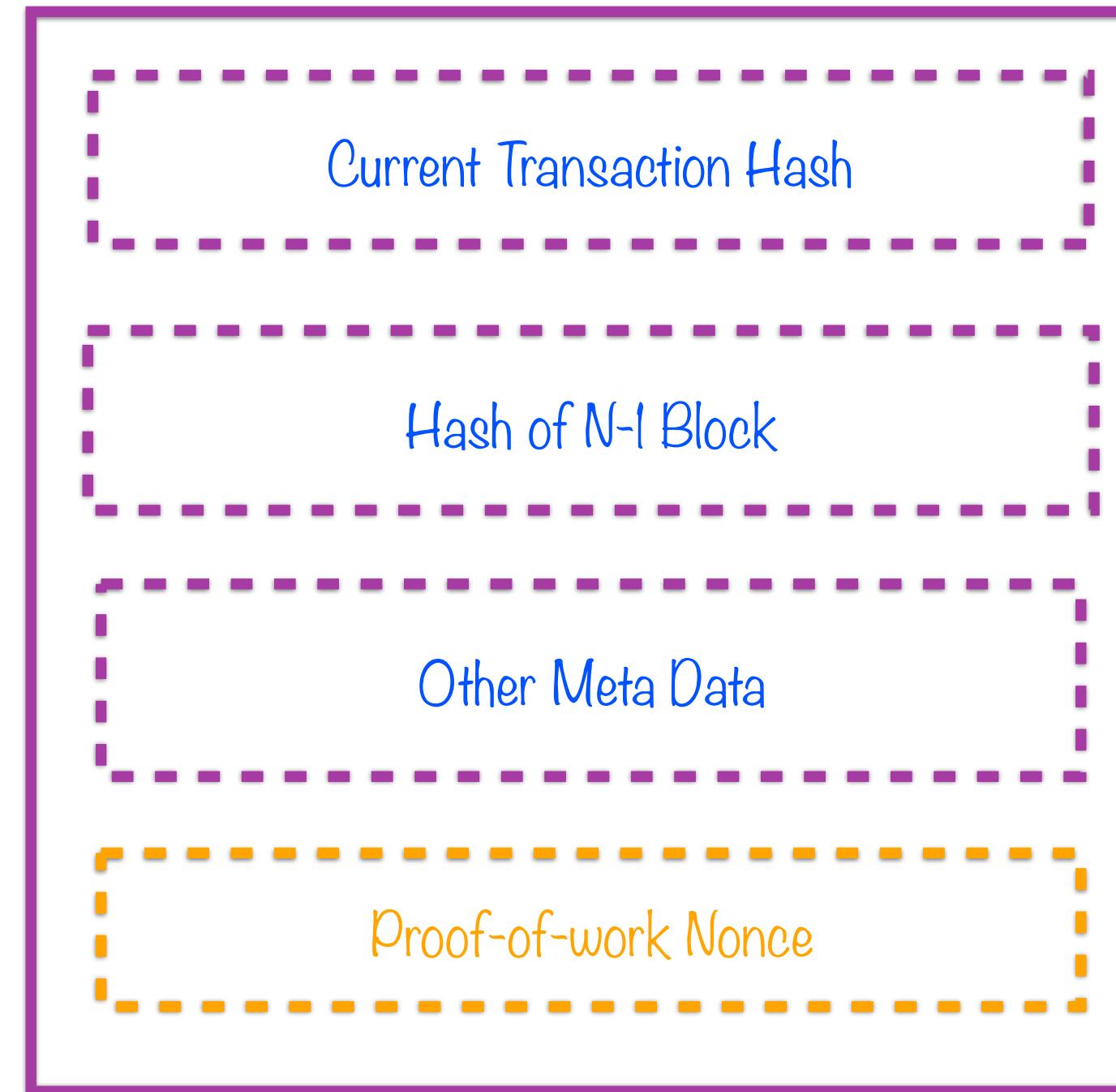
Hence, the name “**blockchain**”

Chain of Blocks

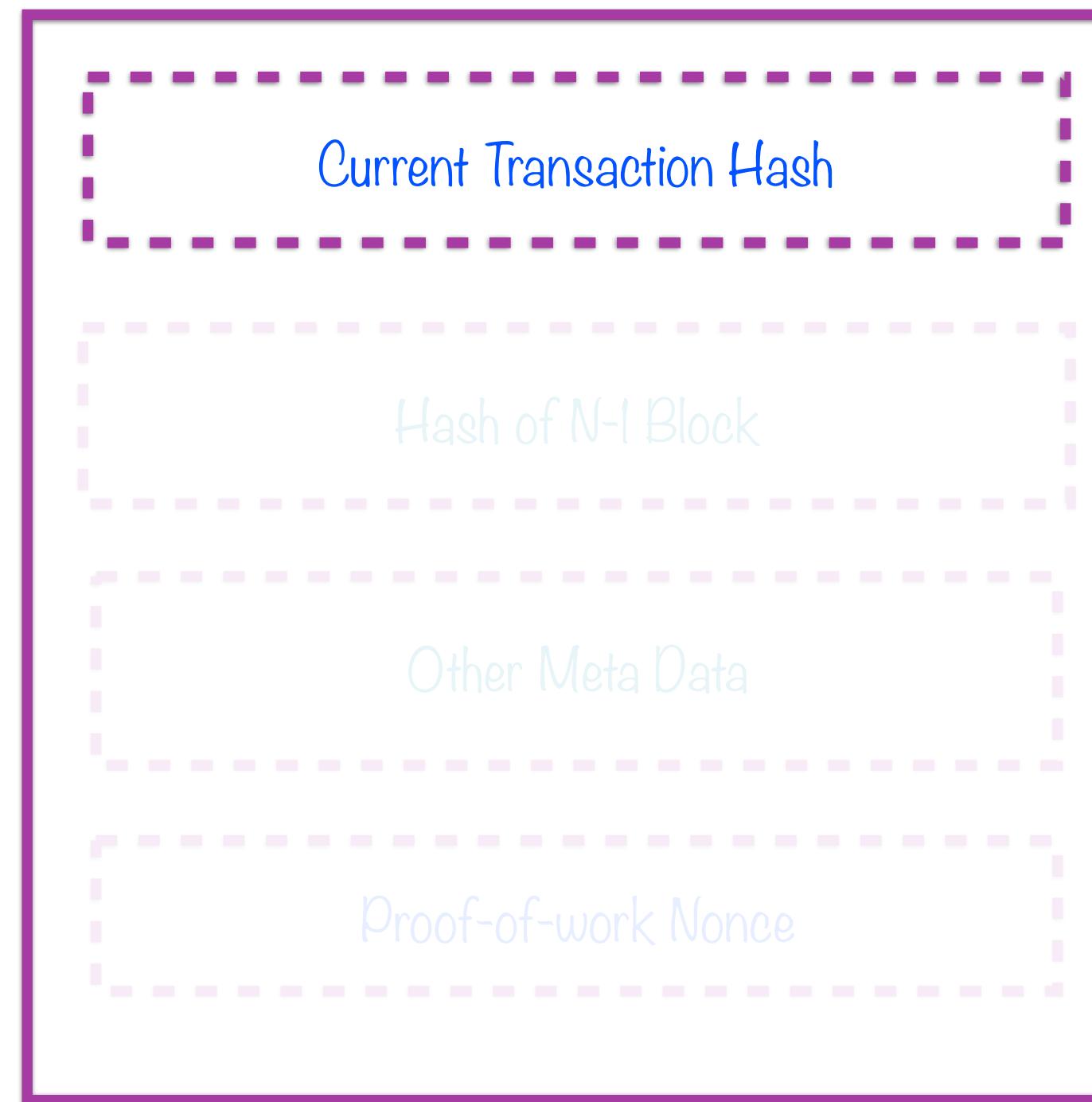


Each block contains hash of the preceding one in chain

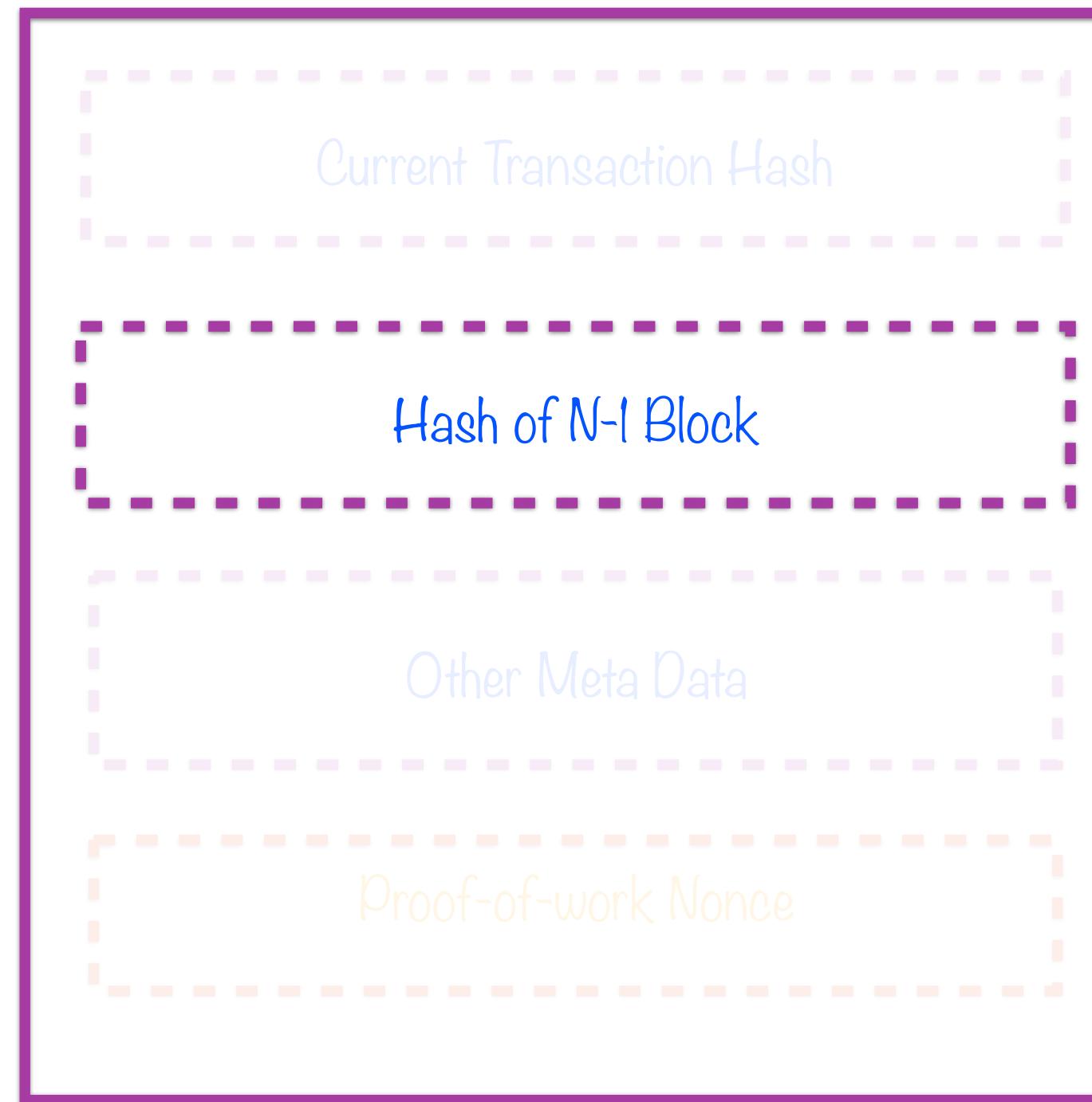
Generation of Block Hash



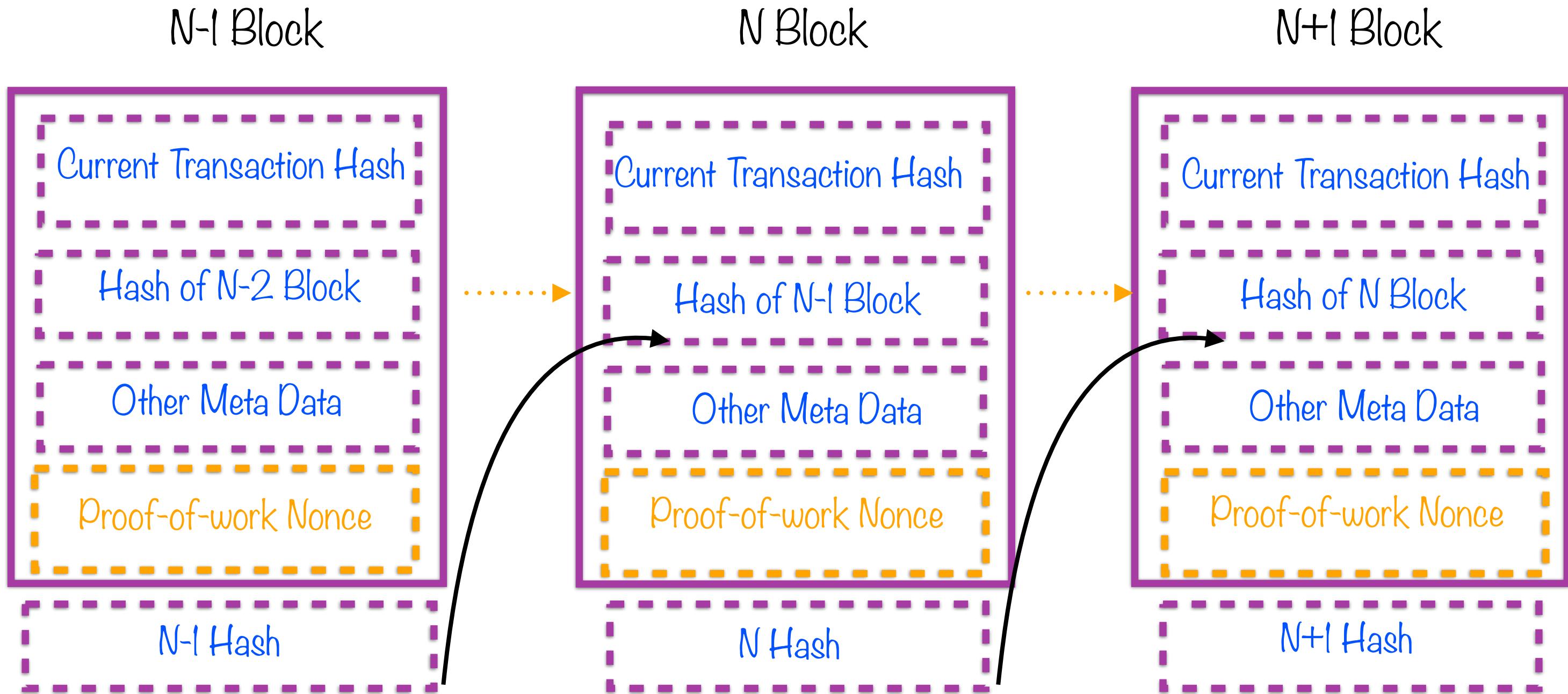
Generation of Block Hash



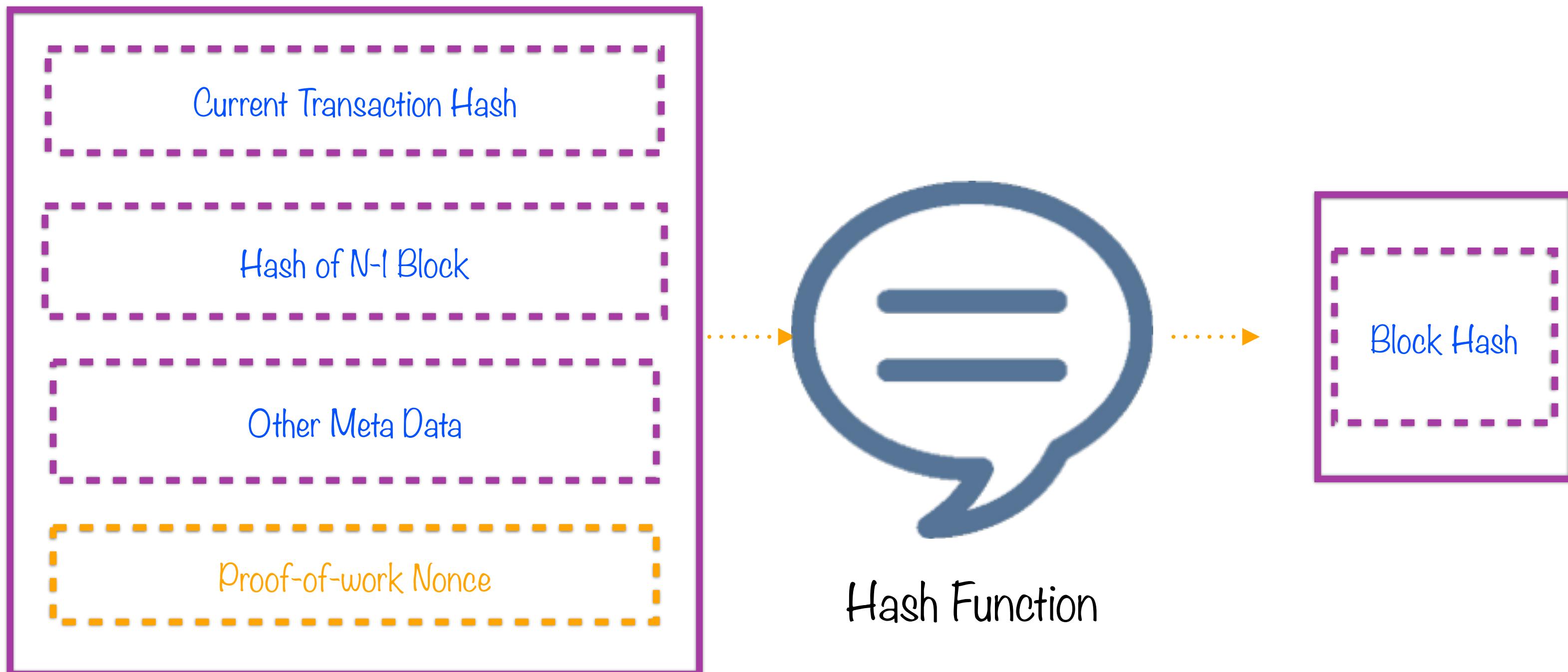
Generation of Block Hash



Generation of Block Hash



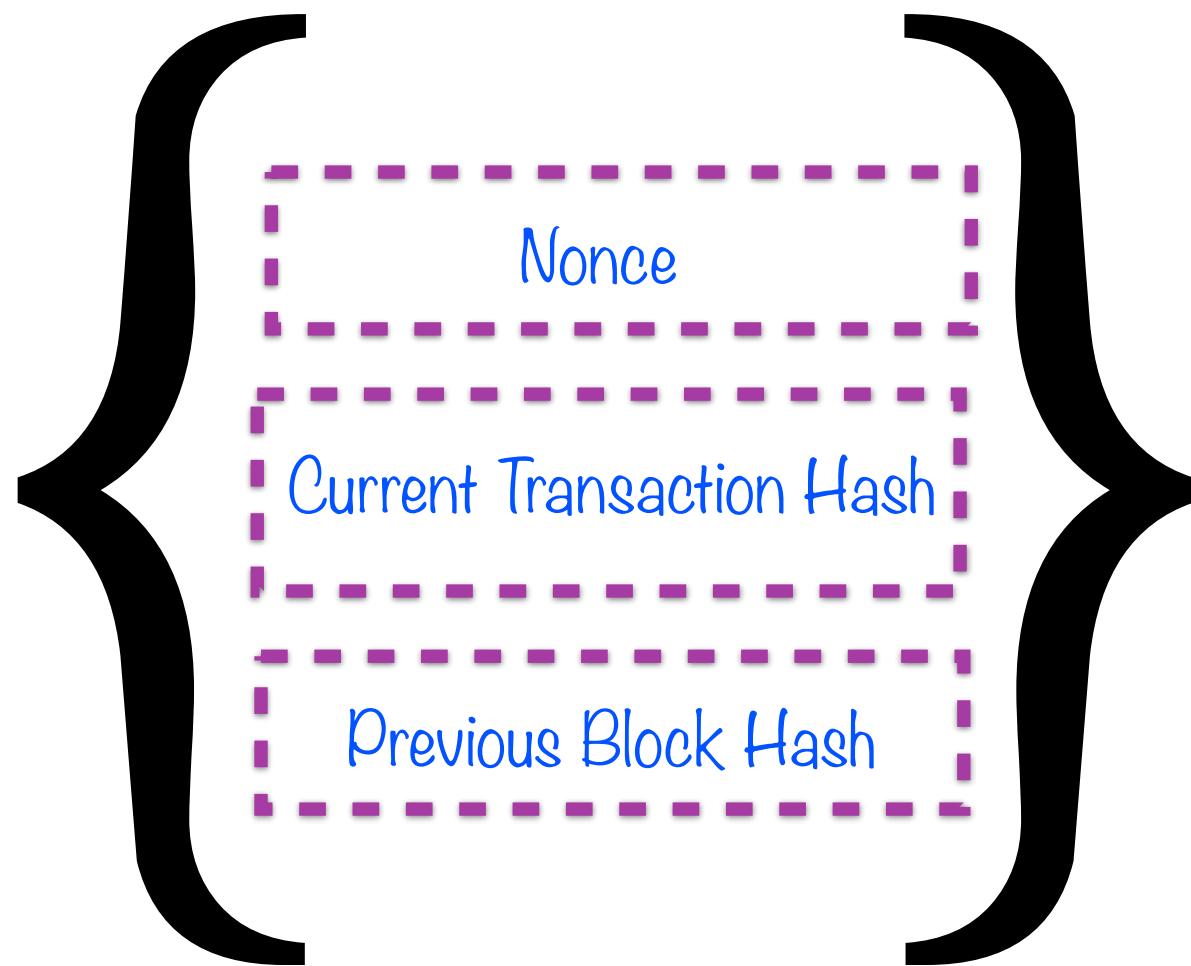
Generation of Block Hash



The Inputs should be hashed in such a way
that

Target Hash > Block Hash

i.e the new hash contains desired number of
zeros in the prefix



Hashing Function

The Target Hash > New Hash i.e the new hash contains desired number of zeros in the prefix

The Target Hash

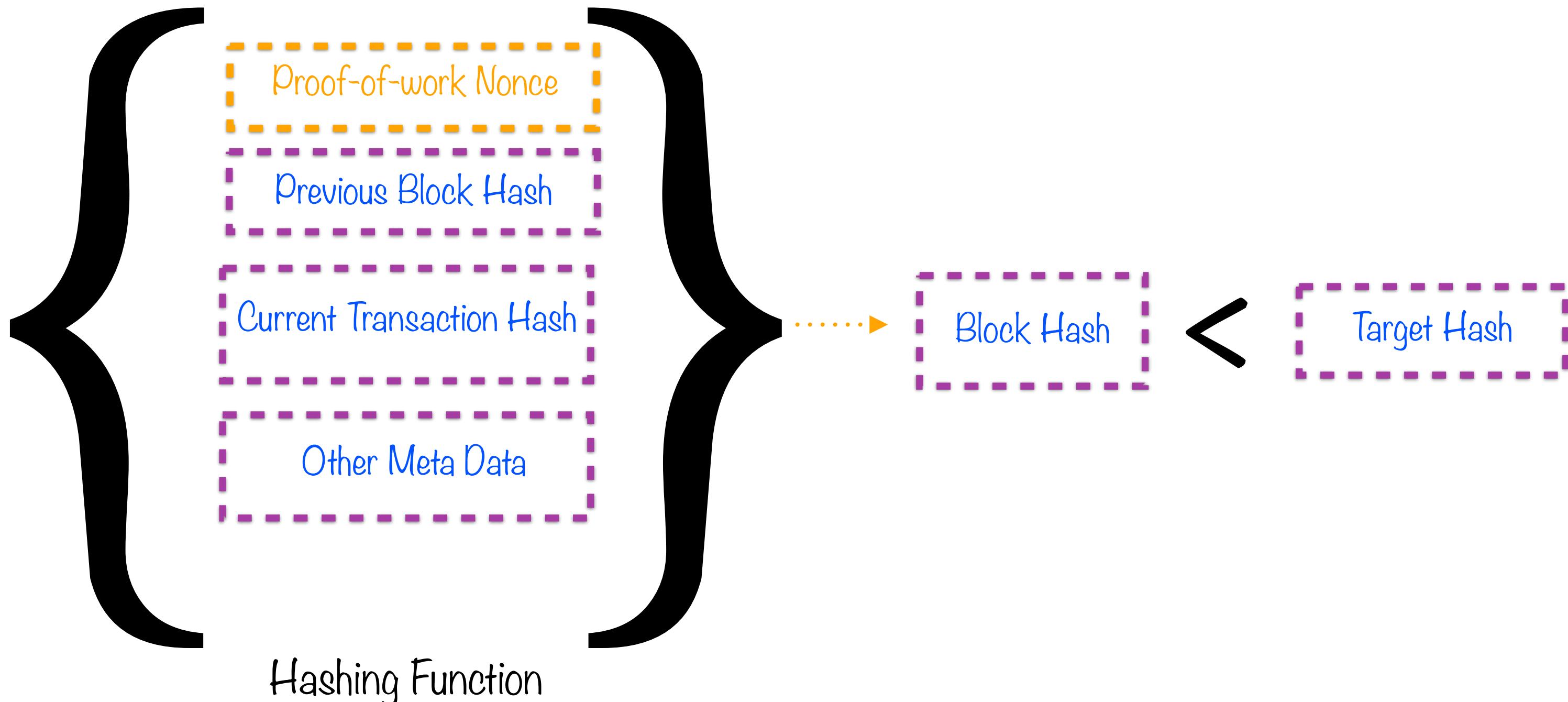
00000999

>

New Hash

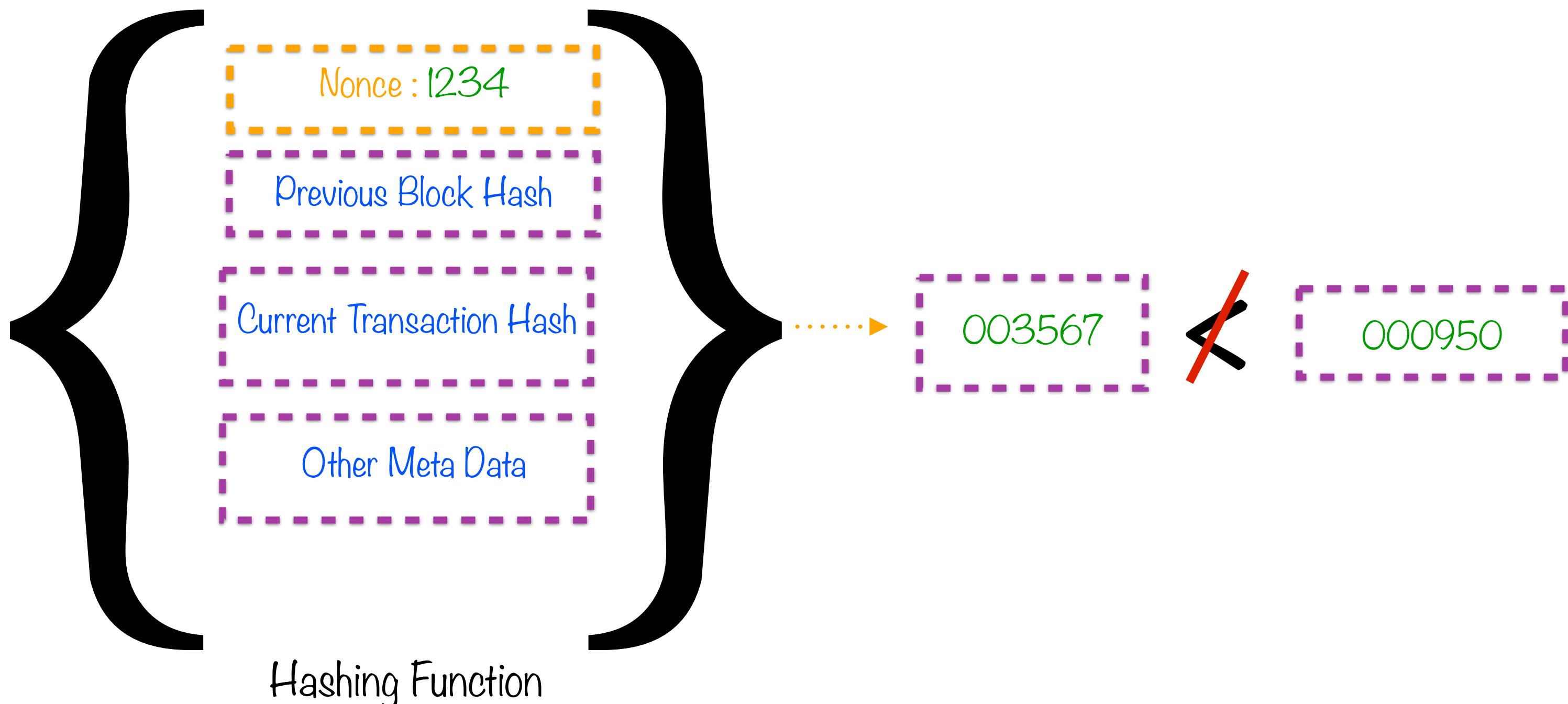
00000997

Find Proof-of-workNonce



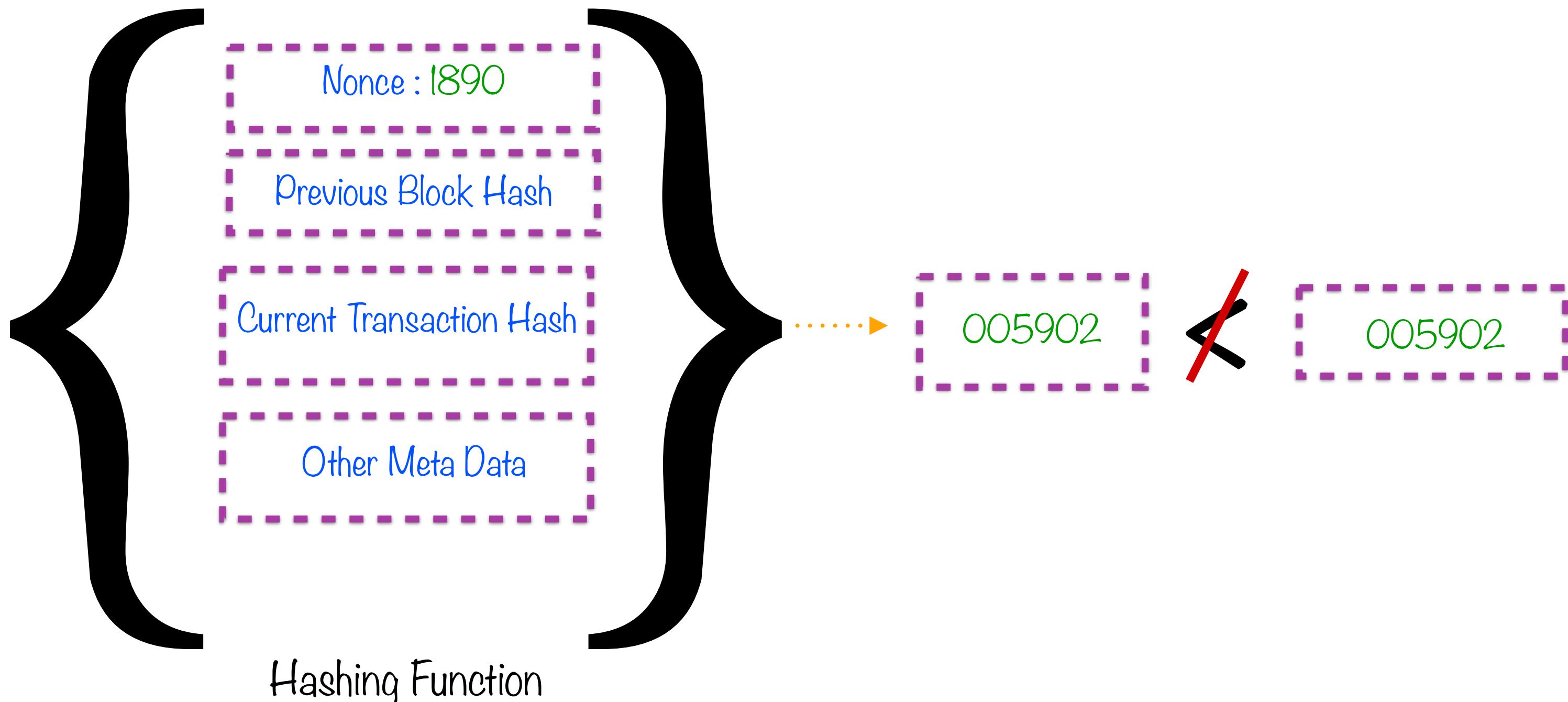
Find nonce that satisfies this arbitrary, difficult condition

Find Proof-of-workNonce



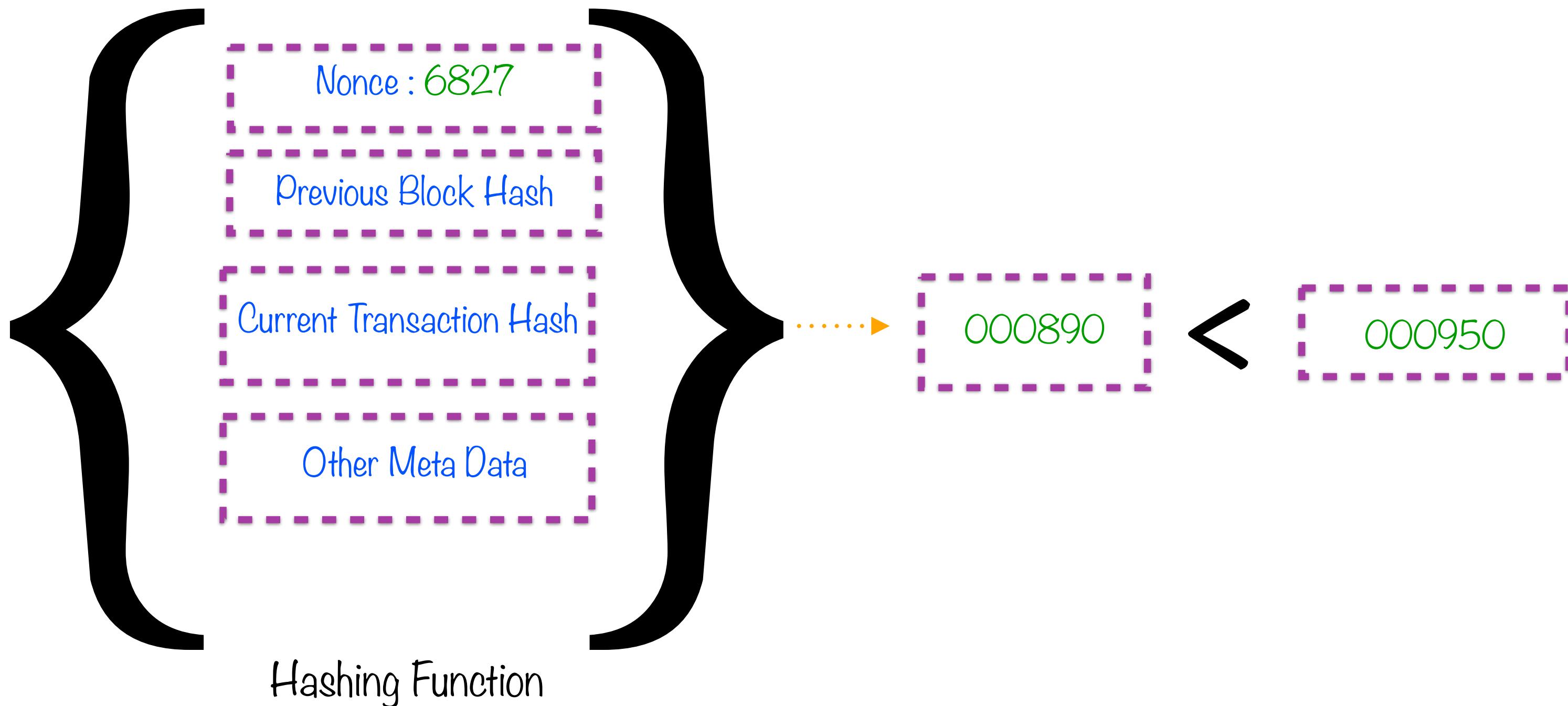
Can only be solved by trial-and-error (not intelligent computation)

Find Proof-of-workNonce



Can only be solved by trial-and-error (not intelligent computation)

Find Proof-of-workNonce



Keep trying until nonce that satisfies condition is found

After the Win

Claiming the Prize

Each miner is searching for its own nonce

(Running its own race)

Once found, broadcasts to all nodes

Once verified, miner claims transaction prize

- $\text{gasPrice} \times \text{gasUsed}$

The Rest Fall in Line

Other nodes receive broadcast claiming prize

They cease searching for their own nonces

Switch to verifying victory of claimant

Update their blockchain accordingly

The Rest Fall in Line

What if they ignore the message?

At risk of falling out of sync with peers

Will lose chance of claiming future prizes

The Rest Fall in Line

Verification of nonce is almost trivial
Finding the nonce is very very hard

Beyond Proof-of-work

Two Consensus Algorithms

Proof-of-work

Currently used in Ethereum to achieve
trustless consensus

Proof-of-Stake

Ethereum actively moving to switch to
this algorithm instead

Two Consensus Algorithms

Proof-of-work

Currently used in Ethereum to achieve
trustless consensus

Proof-of-Stake

Ethereum actively moving to switch to
this algorithm instead

Flaws of Proof-of-work

Small number of miners control the consensus

Can “corner” the process

If control $> 51\%$, they could emit and verify fraudulent transactions

Flaws of Proof-of-work

Every miner seeks the nonce
Wastes tremendous energy doing
so
All but one waste all their effort

Proof-of-Stake

Ethereum will shortly switch to this consensus algorithm

Meant to be less vulnerable to centralization

Details beyond scope of this course

Blockchain Capabilities

Allows multiple parties to initiate transactions

There is no need for a trusted, central authority

Includes mechanisms to ensure transactions are verified and secure

Transactions are immutable and can be verified independently

Each participant in the network has access to the shared ledger

Using AWS Blockchain Template for Ethereum

AWS support for Ethereum

AWS Blockchain Templates allow one to quickly create and deploy blockchain networks on AWS infrastructure. We will create a **private network**

[https://docs.aws.amazon.com/blockchain-templates/latest/developerguide/
blockchain-templates-ethereum.html](https://docs.aws.amazon.com/blockchain-templates/latest/developerguide/blockchain-templates-ethereum.html)

AWS - hopefully cover these in ~5 mins

EC2: EC2 == VM

Availability Zone

Key Pair: Private/Public keys needed to connect to EC2 instance

VPC: VPC == Network of related hosts

Subnet: Logical subdivision of network

CIDR: Format to specify range of IP addresses (for a subnet)

Route Table: Directs traffic into and out of subnet

Elastic Container Service: Docker container orchestration on AWS

Why Merkle Tree?

Finding inconsistent Tx becomes an $O(\log n)$ operation

Demo

Creating and configuring a VPC

AWS - hopefully cover these within 2 mins

Security Groups

Demo

Creating and configuring security groups

AWS - hopefully cover these within 2 mins

IAM

Demo

Setting up IAM roles

Bastion Hosts - 1 minute intro

Also known as jump host

Demo

Creating a bastion host

Metamask

Ethereum Client which is a Chrome plugin. It allows one to:

- create Ethereum accounts
- Connect to an Ethereum network
- Initiate transactions on the network

Demo

Setting up MetaMask

Set up Ethereum network using AWS CloudFormation

AWS CloudFormation: Infrastructure as Code

Ethereum stack creation: <https://docs.aws.amazon.com/blockchain-templates/latest/developerguide/blockchain-templates-create-stack.html>

Sets up EthStats (to view stats about our blockchain network) and EthExplorer (to view our blocks and transactions)

Demo

Using AWS CloudFormation to set up Ethereum

Set up proxy server

We use FoxyProxy (a Chrome plugin) to set up proxy server.
This will be used to redirect requests from our browser, through the bastion host, to EthStats and EthExplorer (thus allowing us to view their web UI)

These would be useful:

- what a proxy server does
- What is a SOCKS proxy?

Demo

Connecting to Ethereum using a proxy server

Deploying a Simple Smart Contract App to the Network

Smart Contracts

Smart Contract

Mechanism that allows common contractual clauses to be specified, verified or enforced even in the absence of trust between contracting parties and in the absence of a third party

Smart Contract



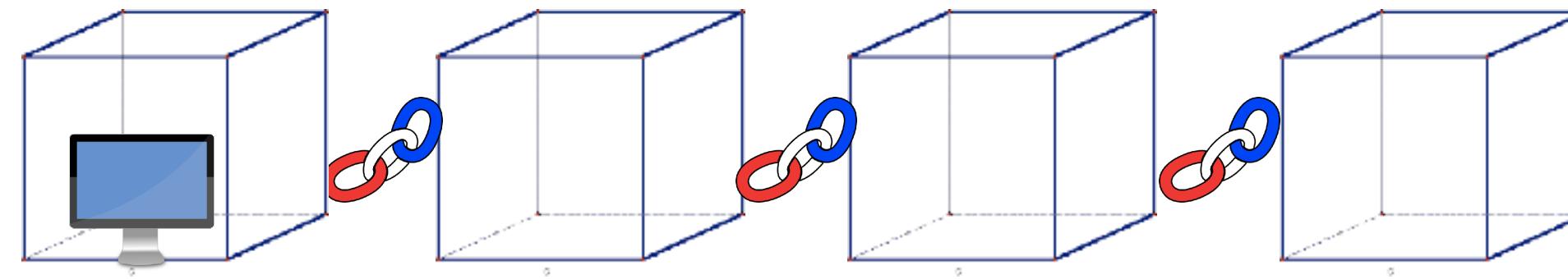
Smart Contracts



Contracts

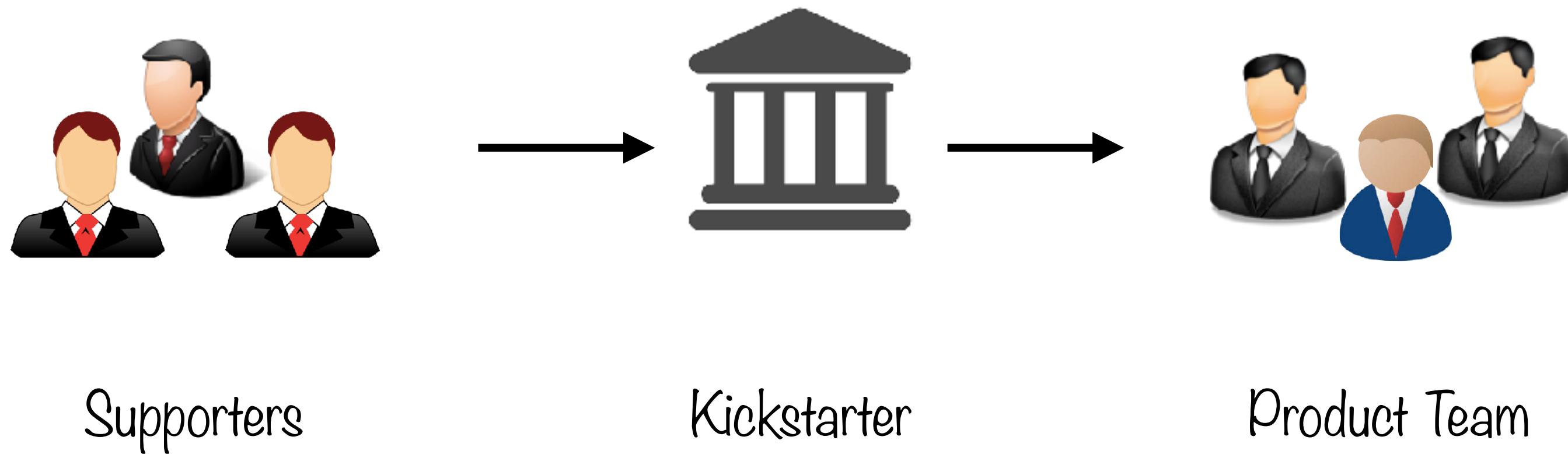
Smart Contracts are just like Contracts in the real world with one unique difference is that they are digital

Smart Contract



Smart contracts are tiny computer program that is stored inside a blockchain

Intuitions behind Smart Contracts



Intuitions behind Smart Contracts

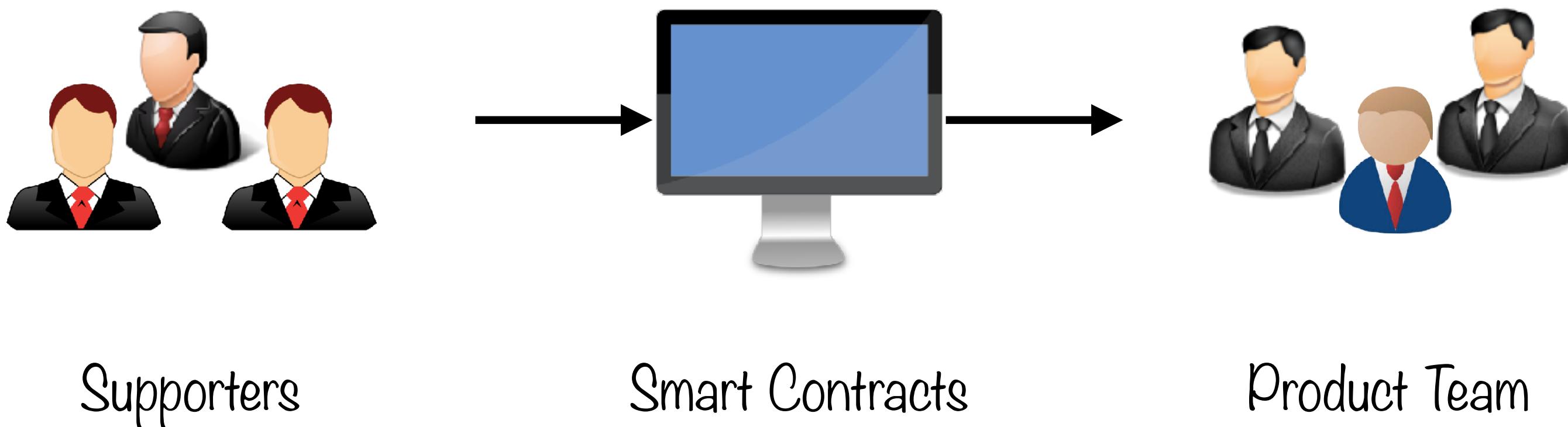
Kickstarter is a fund raising platform
Kickstarter is a third party sits between Product Team and Supporters
Product team goes to Kickstarter, creates a project

Intuitions behind Smart Contracts

If project gets successfully funded,
Project Team expects Kickstarter to give
their money
And supporters want their money to go into
the project

Above problem has one major limitation
that Supporters and Product Team have to
trust on third party

Intuitions behind Smart Contracts



Smart Contracts

Program the Smart Contracts such that it holds all the received funds until goal is reached

Supporters of the project can transfer their money to Smart Contracts

If the projects is success, Smart Contracts passes money to the Product team

Smart Contracts

If the project fails, money automatically goes back to supporters

Because Smart Contracts are stored inside blockchain, everything is distributed

With this technique no one is in control of money

Properties of Smart Contracts

Immutable

Distributed

Properties of Smart Contracts

Immutable

Distributed

Once Smart Contracts created, it can never be changed again. So no-one
can tamper Smart Contracts

Properties of Smart Contracts

Immutable

Distributed

The output of the contract is validated by everyone on the network. So a single person can not force the contract to release the funds.

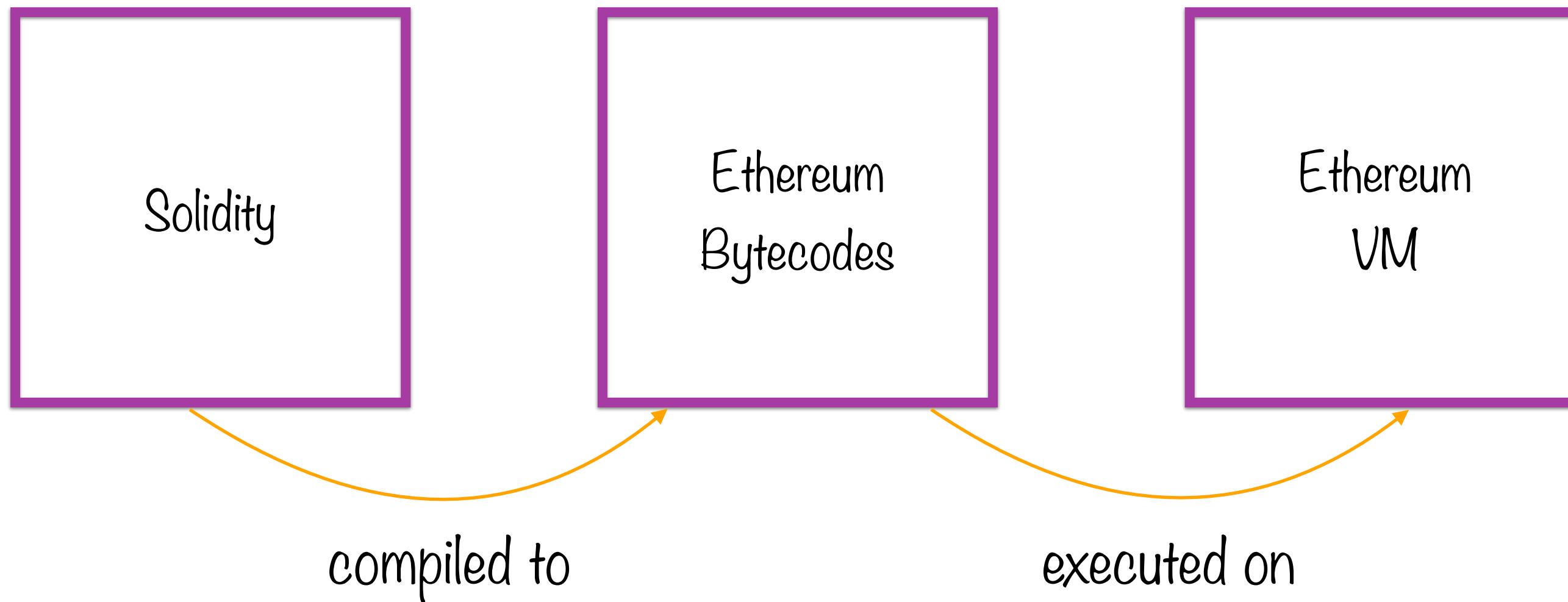
Smart Contracts are programmed using
special programming language called
Solidity

Solidity Programming Language

Solidity

Solidity is a high-level language whose syntax is similar to that of JavaScript and it is designed to compile to code for the Ethereum Virtual Machine.

Sequence of execution of Solidity



Features of Solidity

Comments

Types

Events

Functions

Function Modifiers

Inheritance

Features of Solidity

Comments

Types

Events

Functions

Function Modifiers

Inheritance

Single and multi-line comments supported

Features of Solidity

Comments

Types

Events

Functions

Function Modifiers

Inheritance

Data types supported by Solidity

Like any traditional language it supports booleans, integers and strings

No support for floating point variables as of yet

Also supports structs and enumerations as well as byte arrays that can hold data of any type

Solidity support mappings which are in essence key-value stores

Features of Solidity

Comments

Types

Events

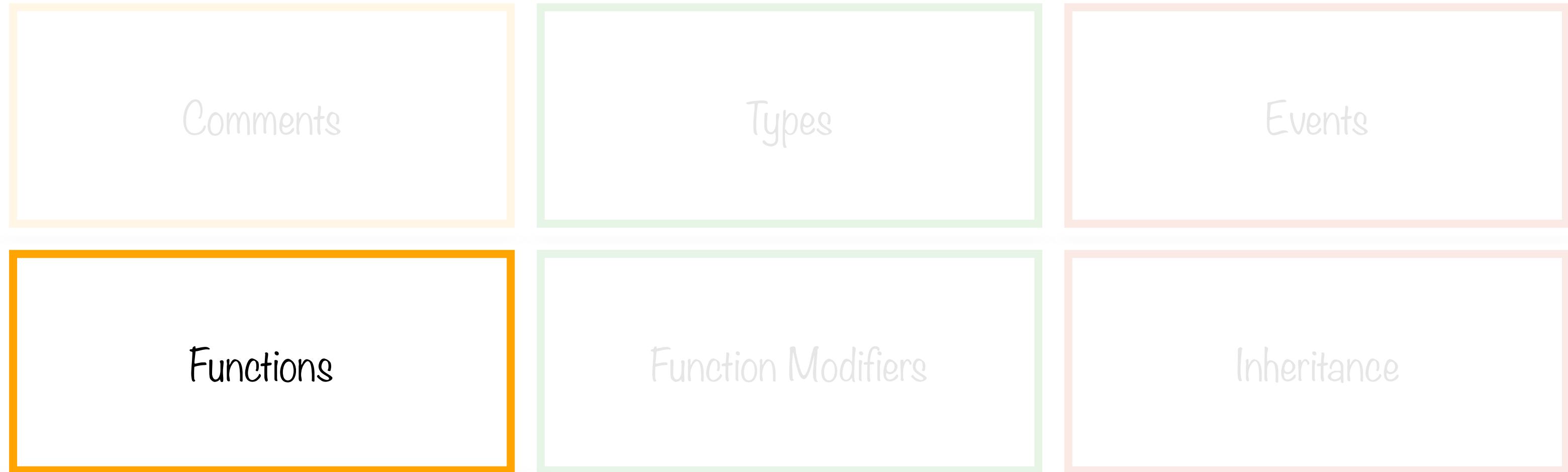
Functions

Function Modifiers

Inheritance

Events are the way Solidity provides in order for accessing the transaction logs

Features of Solidity



Solidity distinguishes two types of functions, constant and transactional

Features of Solidity

Comments

Types

Events

Functions

Function Modifiers

Inheritance

Functional Modifiers

Function Modifiers are used to change the behaviour of a function

Used to check if a condition is satisfied before a function can be executed

Modifiers are inheritable properties of functions

Each function can belong to multiple modifiers

Features of Solidity

Comments

Types

Events

Functions

Function Modifiers

Inheritance

Similar to other programming languages, Solidity supports inheritance

Solidity bytecode and opcode

Bytecode

Bytecode, is an instruction executed
by a software interpreter

Understood by Ethereum Virtual
Machine (EVM)

Opcode

Low level human readable instructions
of the program
Opcodes have their hexadecimal
counterparts

eg. ADD is 0x01

MUL is 0x02

MOD is 0x06

Demo

Building a simple smart contract app

Any operations in blockchain run as transaction and for every transaction there is a cost associated with it

Gas

Gas is a unit that measures the amount of computational effort that it will take to execute certain operations.

Gas is fuel

Every operation in EVM requires a pre-defined amount of gas

Transactions must provide enough fuel

First, to cover its data: intrinsic gas

Second, to cover its entire computation

Unused gas is kept by transaction originator

Transactions that run out of gas are reverted

Transaction still included in a block
but fee is paid

Elements of Gas

Gas Price

Gas Limit

Elements of Gas

Gas Price

Gas Limit

Gas Price

It is the amount that the sender willing to pay per unit of gas in order for transaction to execute

Gas Price

One can also pay less for their transaction

This can be done by varying a variable called gas price

Gas price also determines the final cost or the transaction cost

Gas price is not fixed

Gas price expressed in a unit called Ether

Either

Ether is the name of the currency used within Ethereum

It is used to pay for computation within the EVM

wei is the subunit for ether currency

1 ether = 10^{18} wei

Ether can also be bought for regular currency

1 ether = 291.92 USD

Elements of Gas

Gas Price

Gas Limit

Gas Limit

The gas limit is the maximum limit of gas the sender is willing to set for a particular transaction

Gas Limit

By default the minimum gas limit for all transactions = 21000 gas

Sending ether from one account to another = 21000 gas

Interacting with smart contract = 21000 gas + all executed opcode gas

Following points must be considered while specifying gas limit:

- Different operations will have different gas costs
- Miners will stop executing the moment the gas runs out
- If there is any gas left over, it will be immediately refunded to the account of operation generator

Demo

Building a smart contract app to simulate land sales and purchases

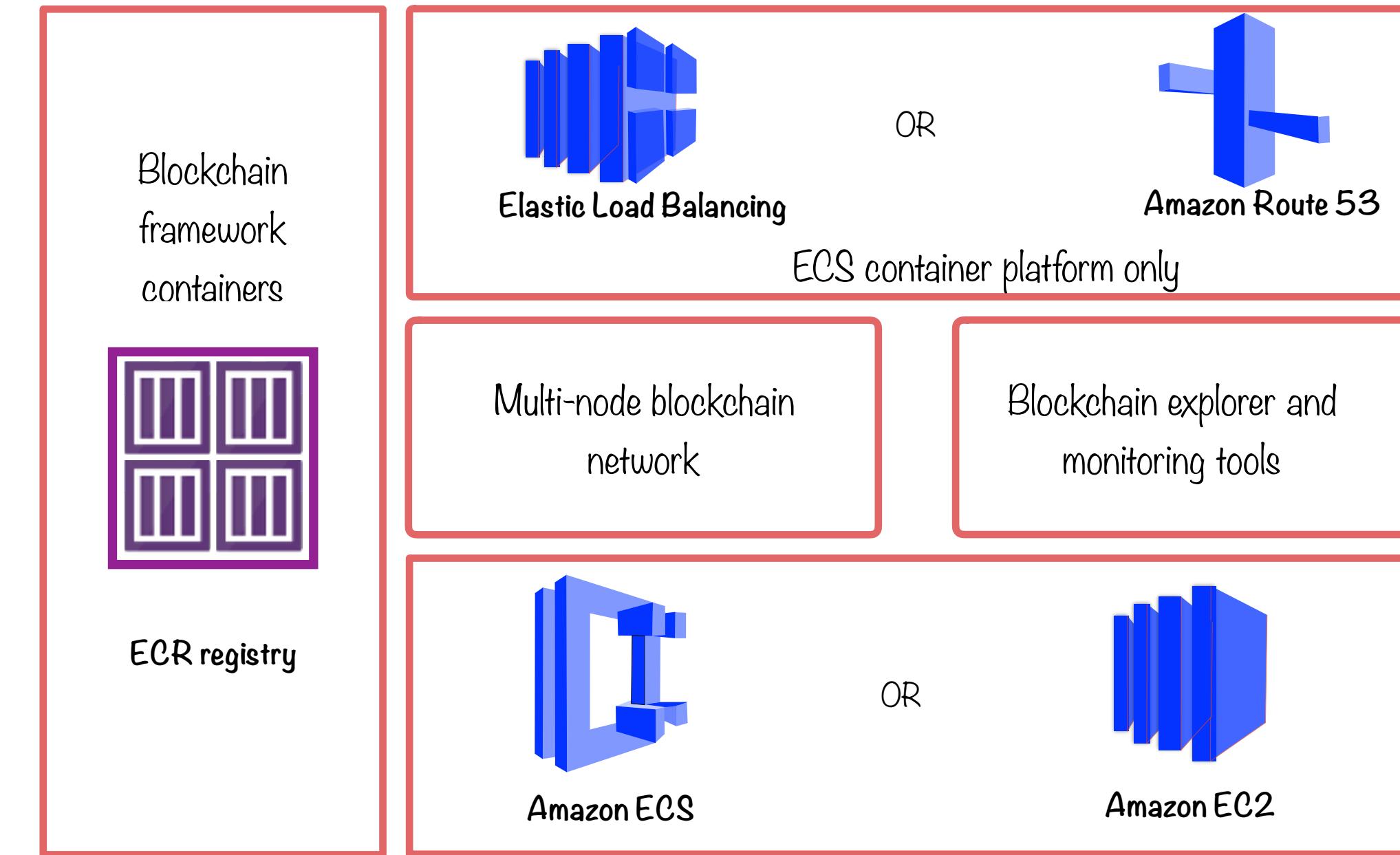
Demo

Running transactions on smart contract apps

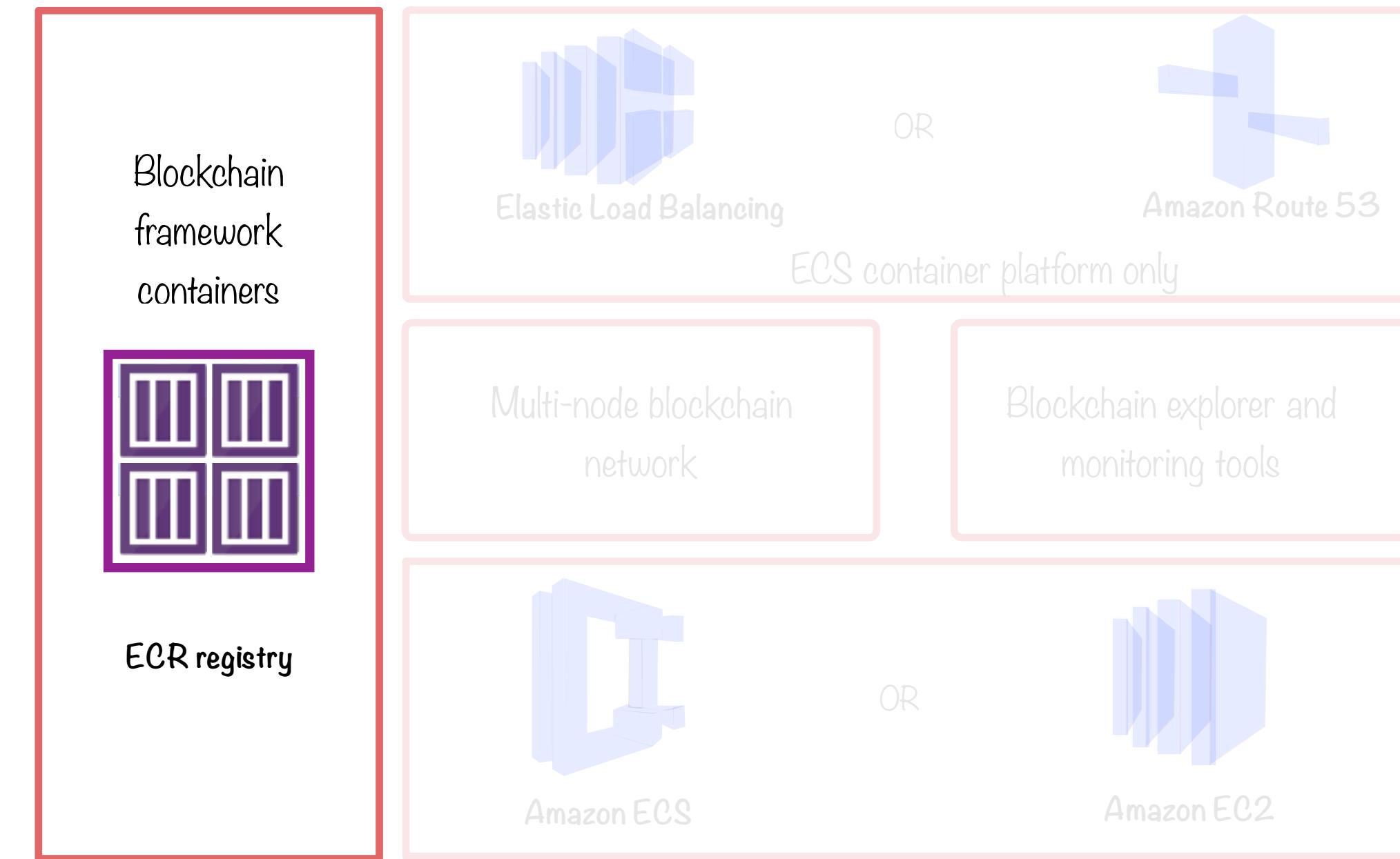
Demo

Cleaning up cloud resources

Blockchain Network Components

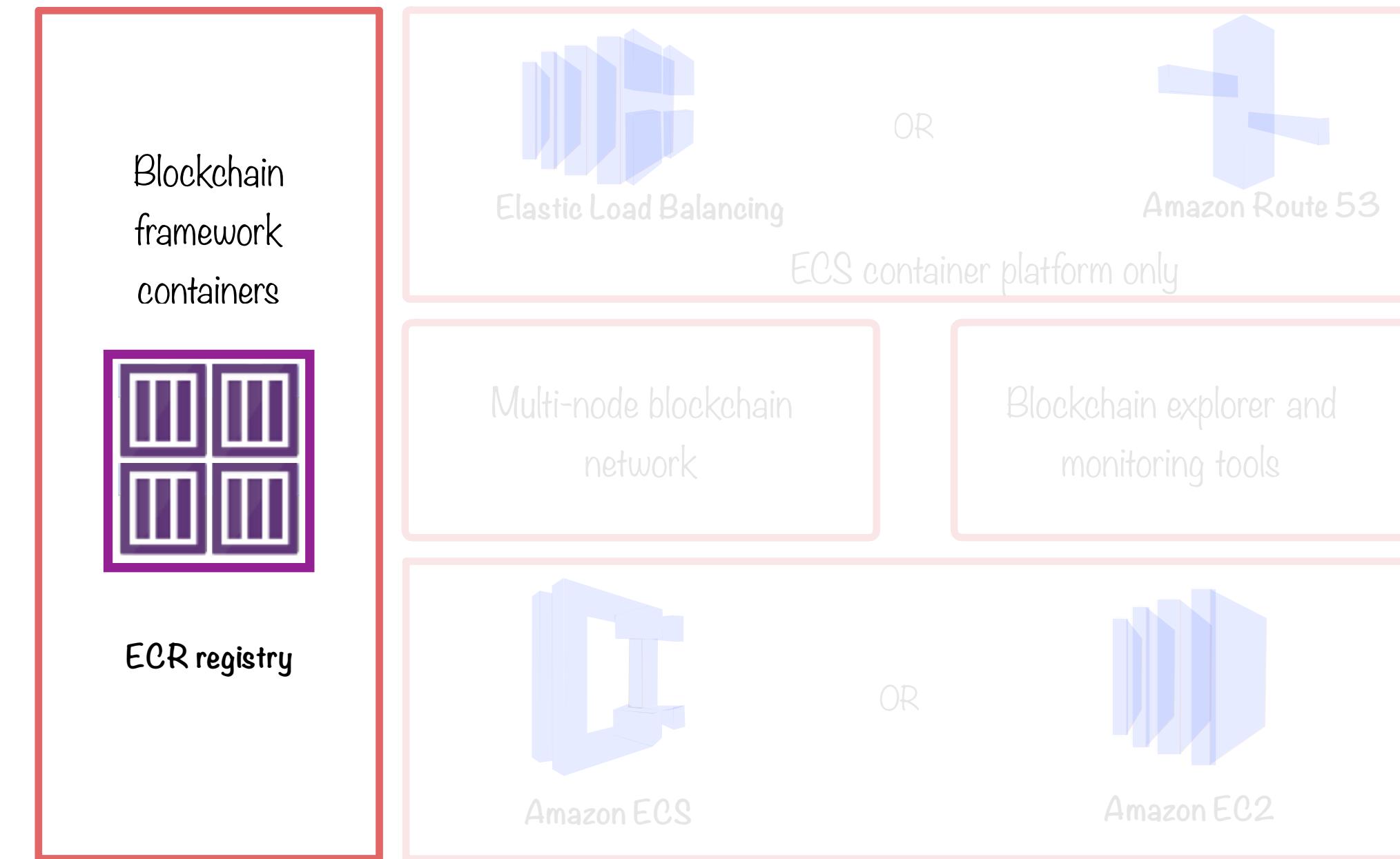


Docker Containers run Blockchain Software



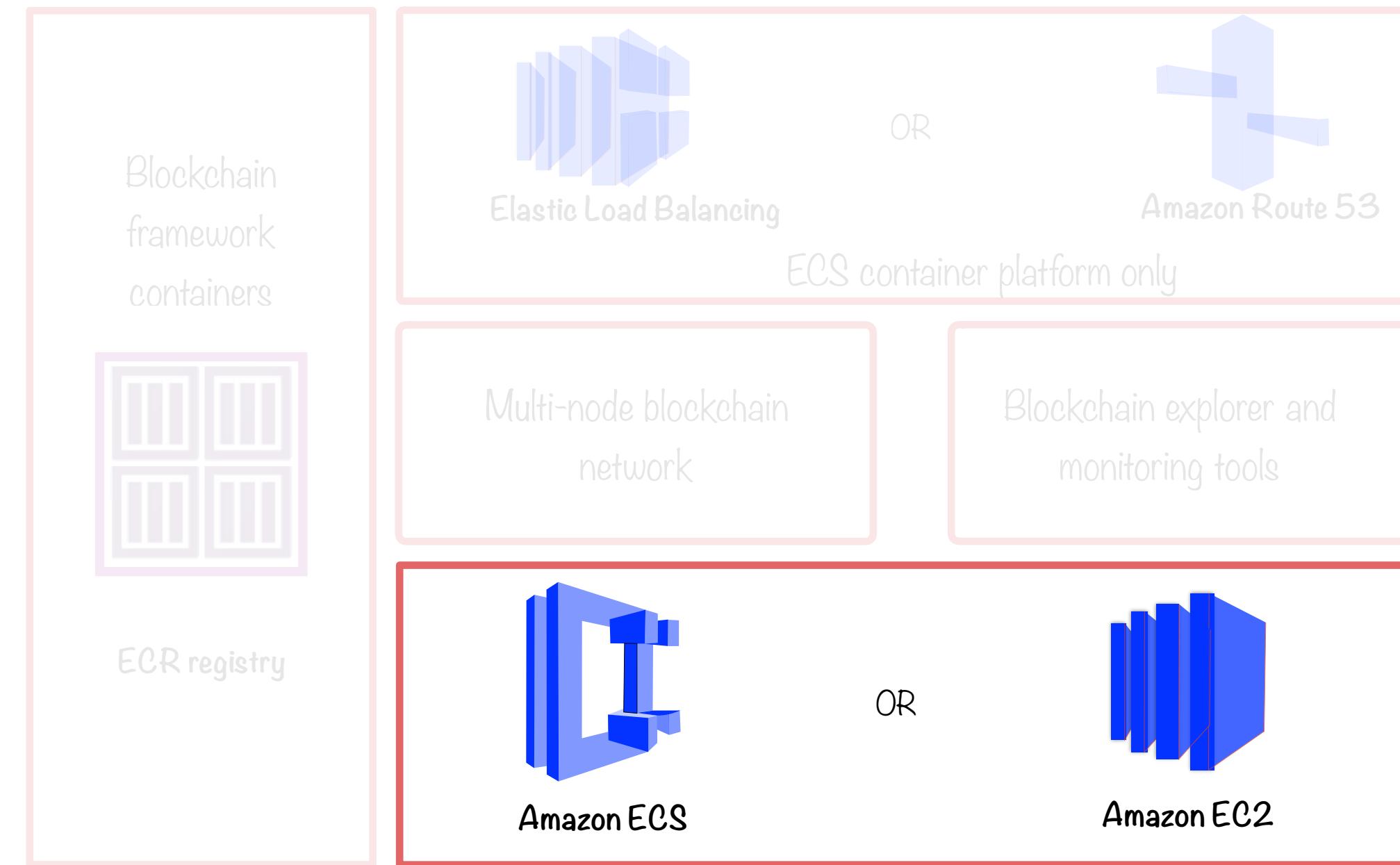
Blockchain software is present on Docker containers

Docker Containers run Blockchain Software



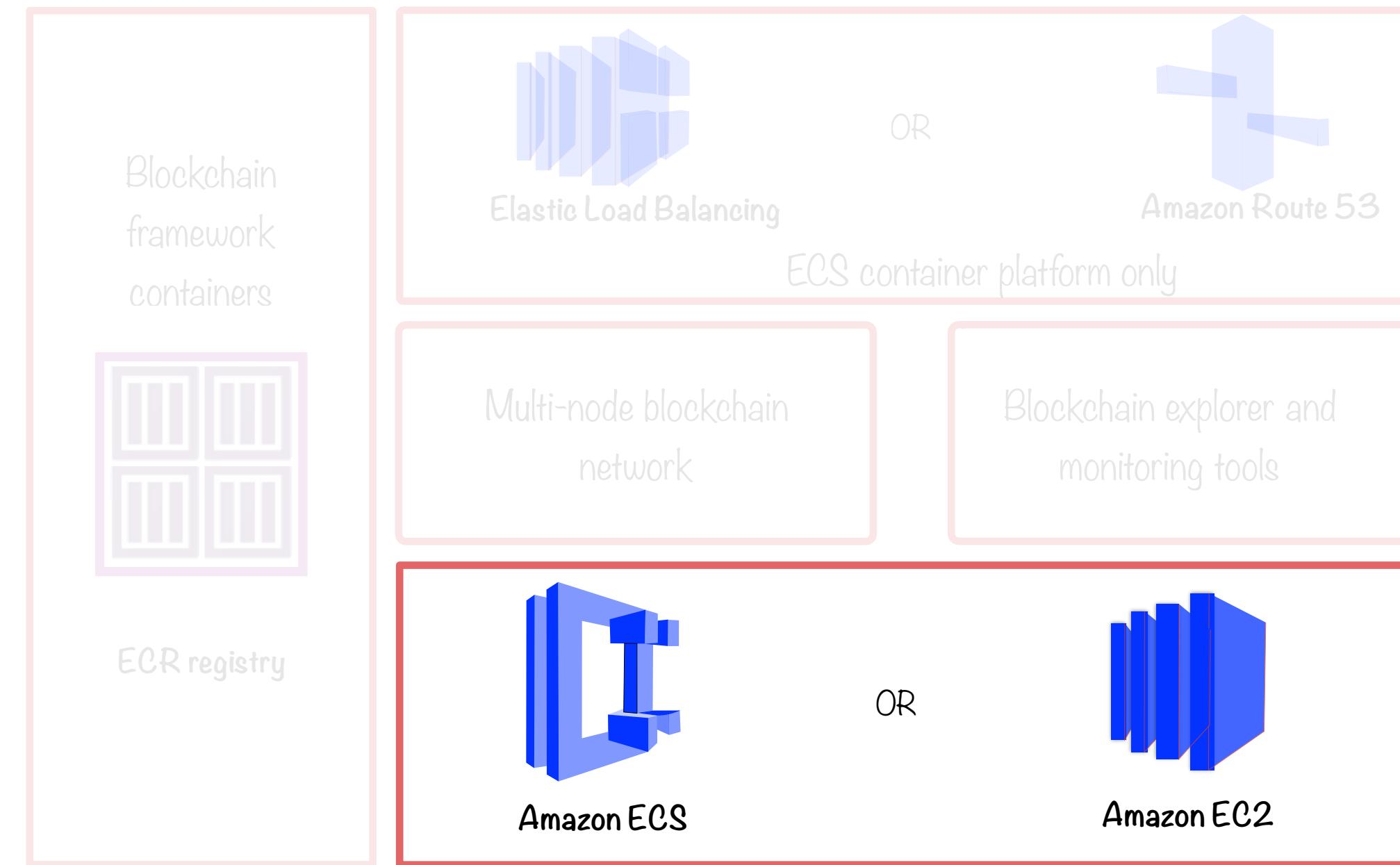
For Ethereum these container nodes run the EVM

ECS Setup or Docker Local Setup



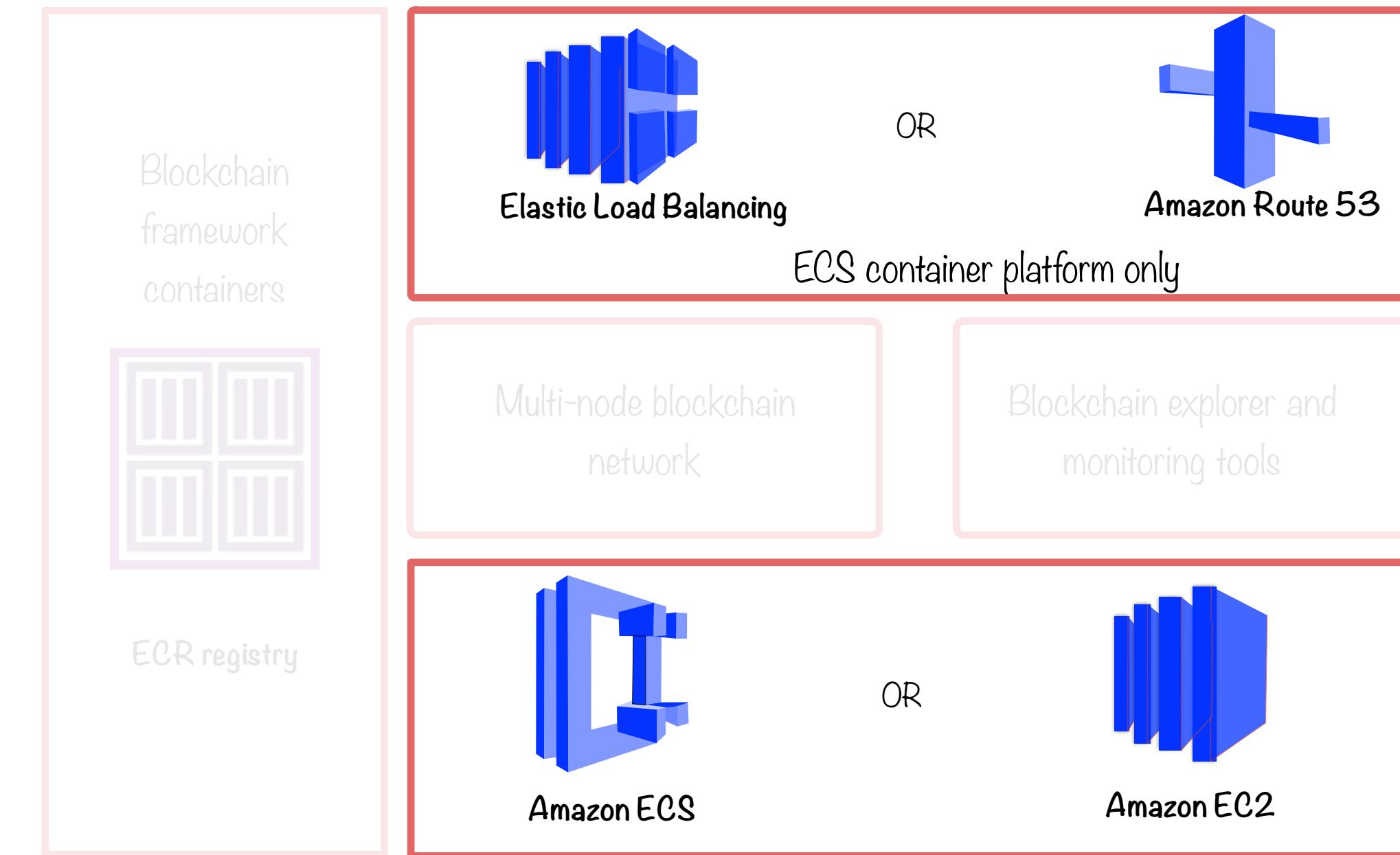
Ethereum containers can be on multiple ECS nodes

ECS Setup or Docker Local Setup

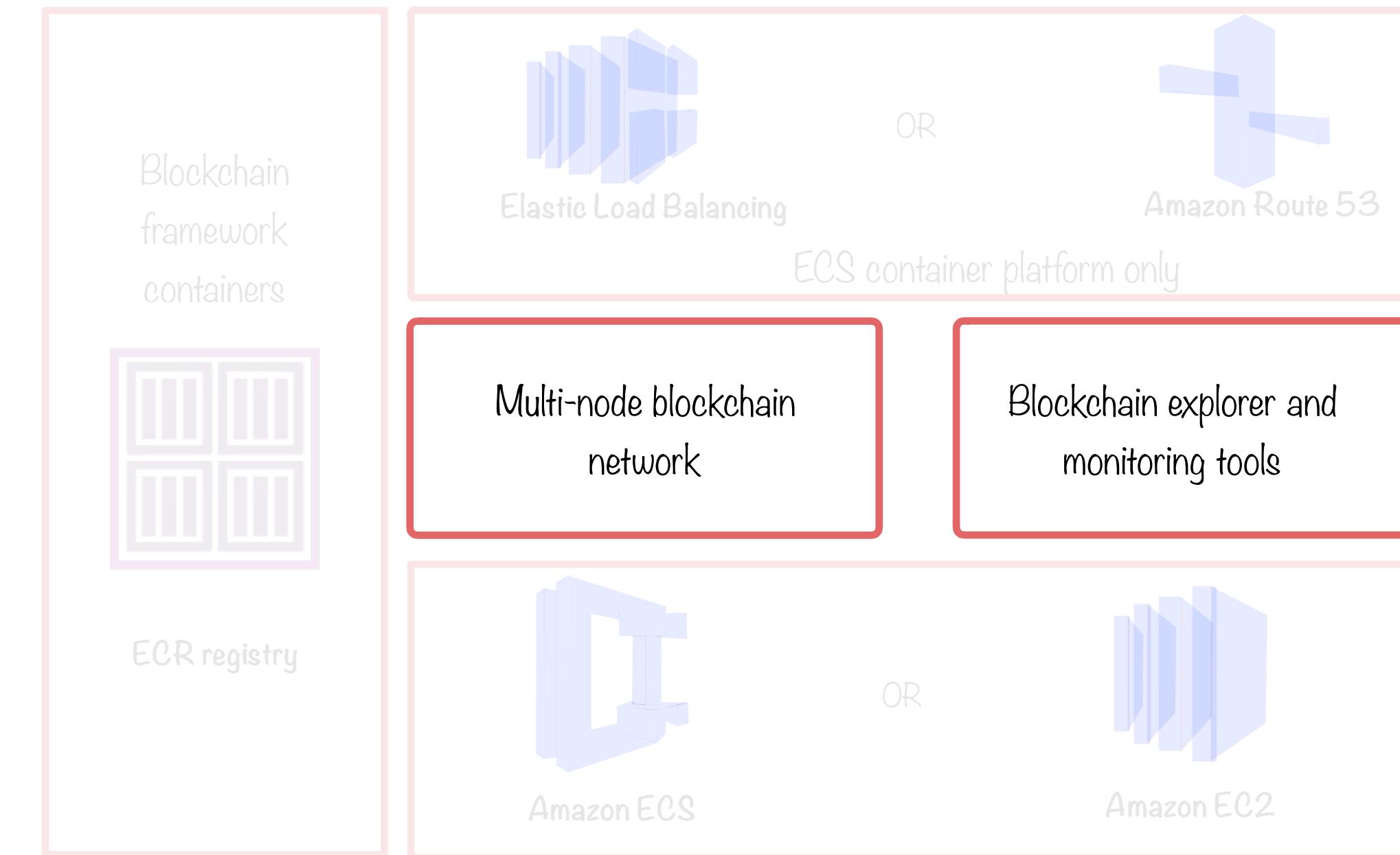


Simplified set up - have all Ethereum containers on the same EC2 instance

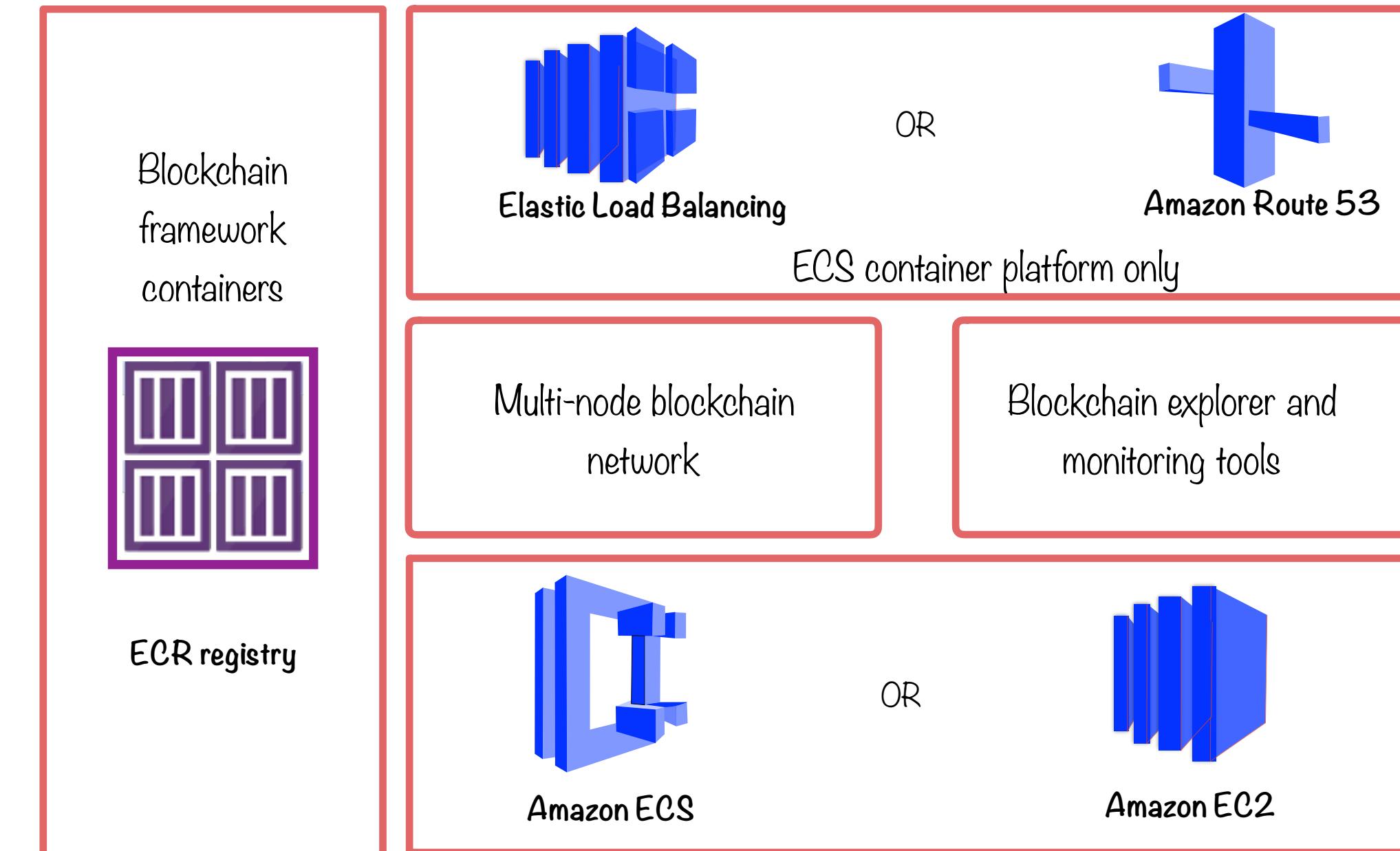
Load Balancers Route Traffic to Nodes



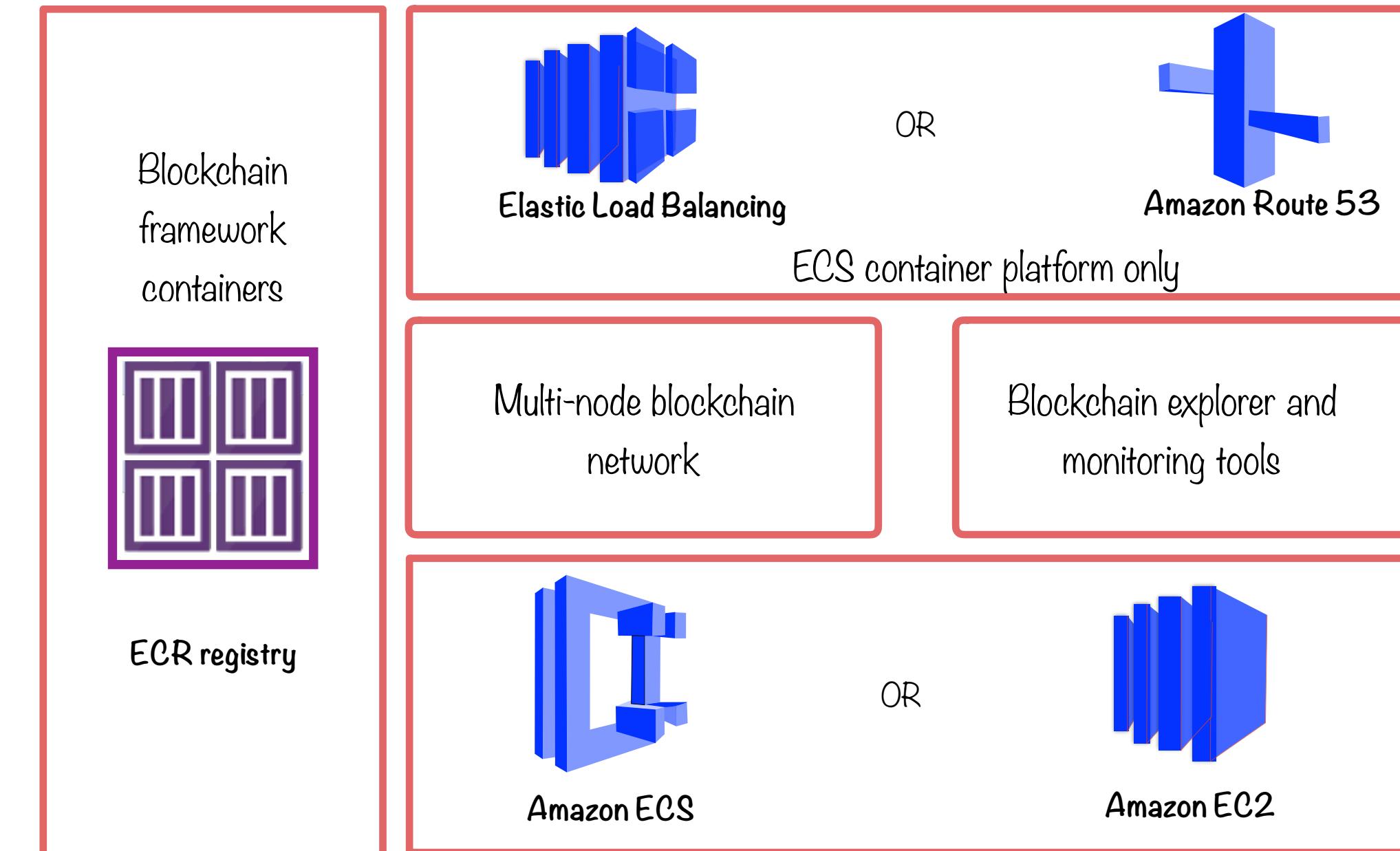
Monitoring Tools to Manage Network



Blockchain Network Components

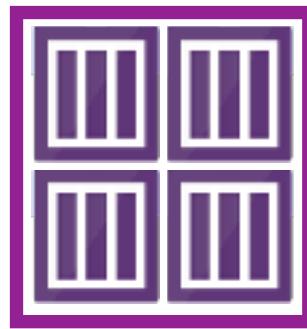


Blockchain Network Components

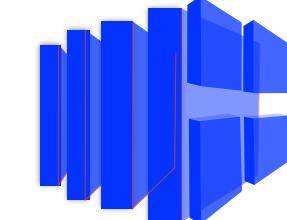


Blockchain Network Components

Blockchain
framework
containers

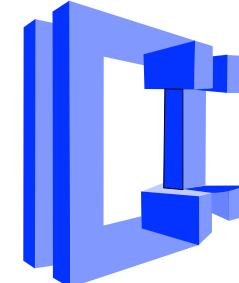


ECR registry



Elastic Load Balancing

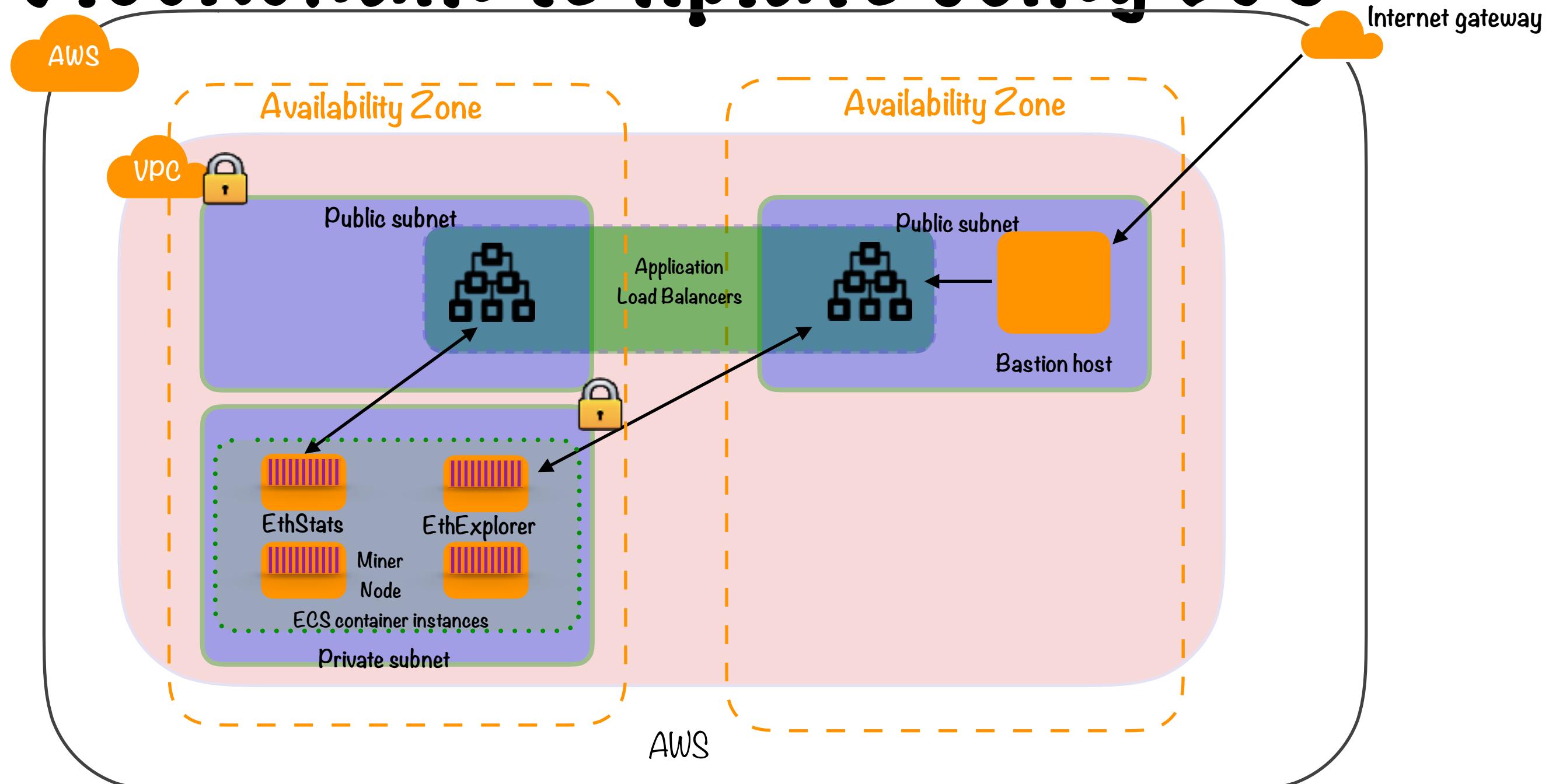
Multi-node blockchain
network



Amazon ECS

Blockchain explorer and
monitoring tools

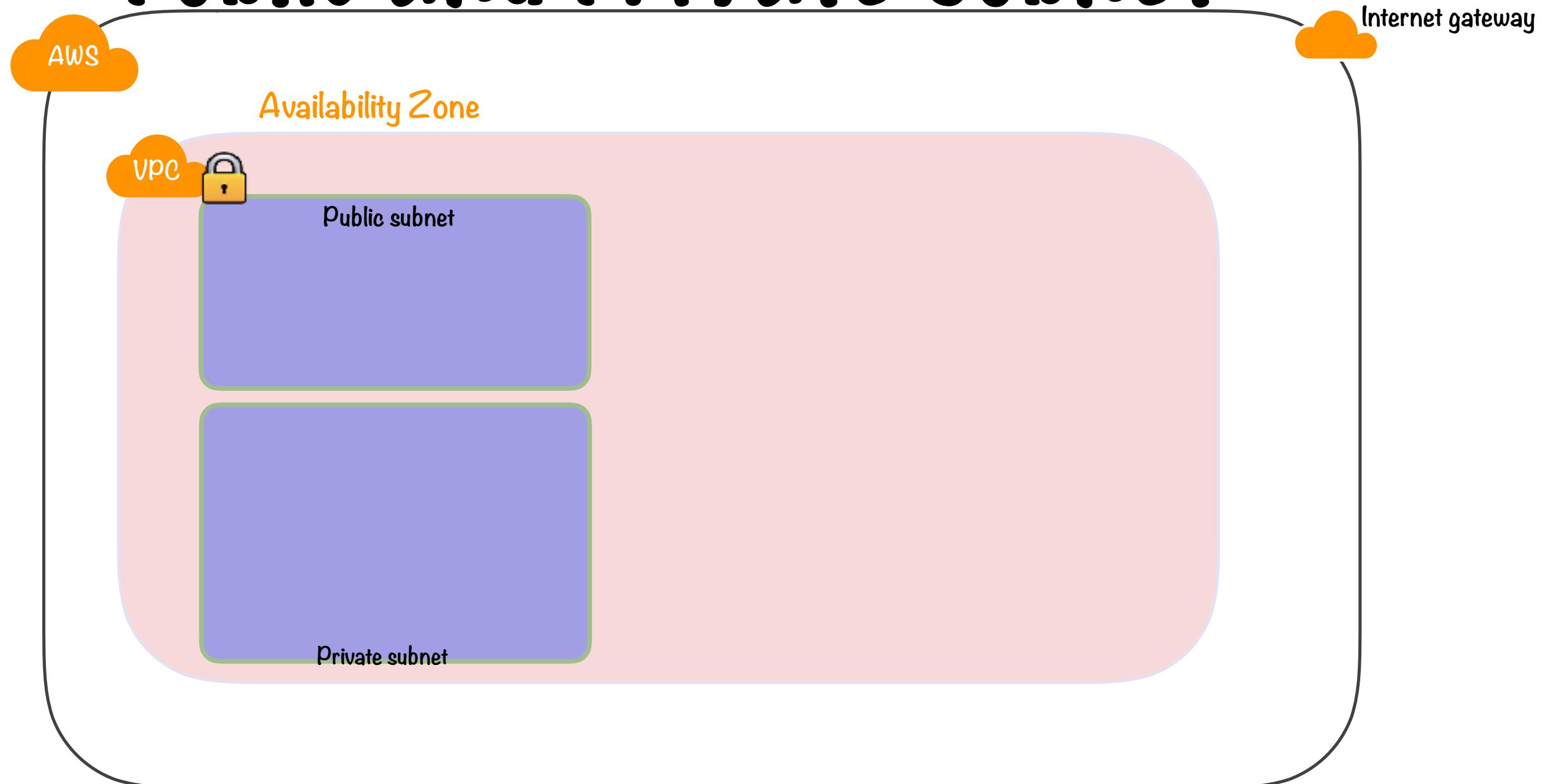
Blockchain Template using ECS



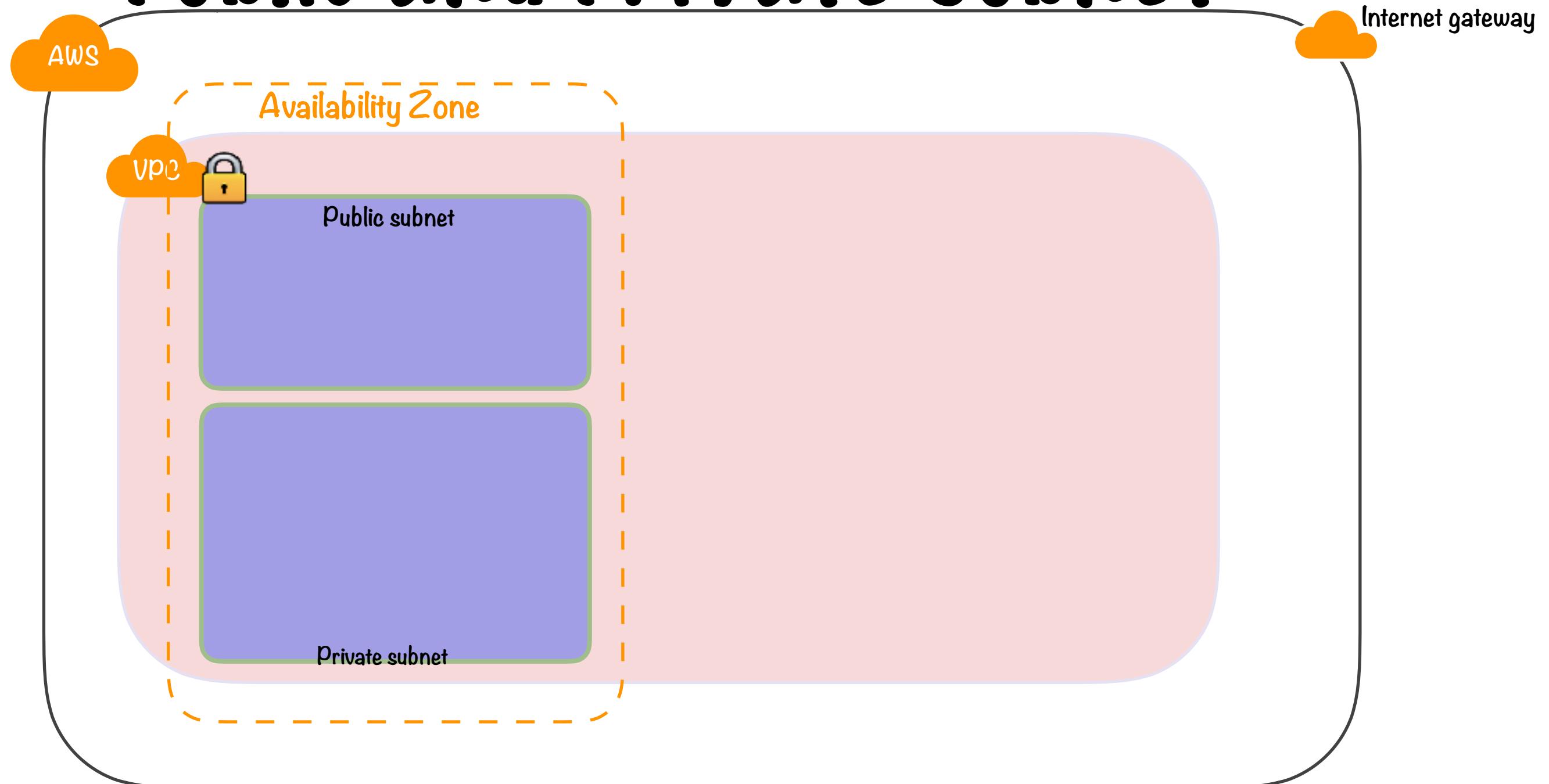
VPC to Hold all EC2 Instances



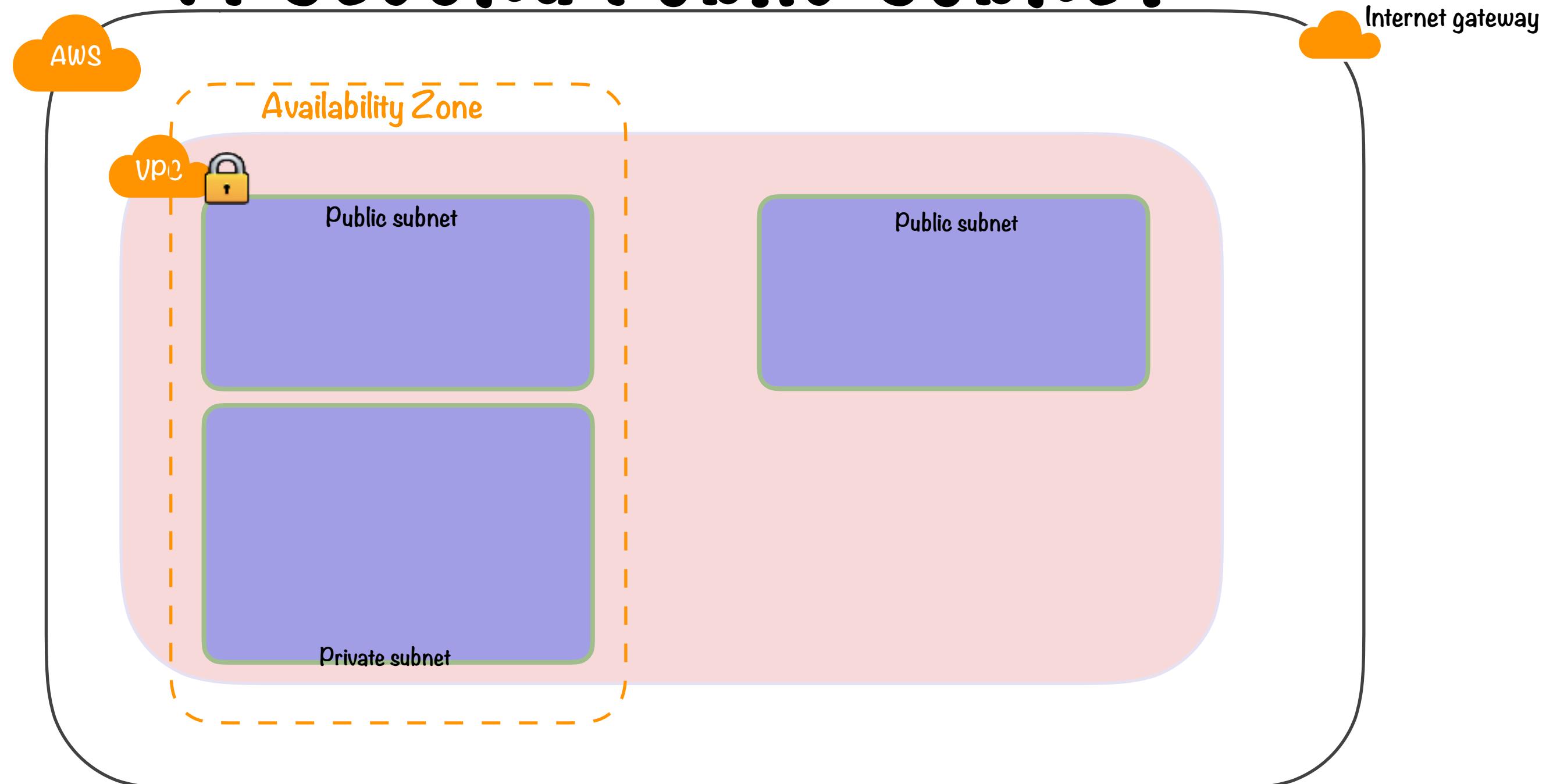
Public and Private Subnet



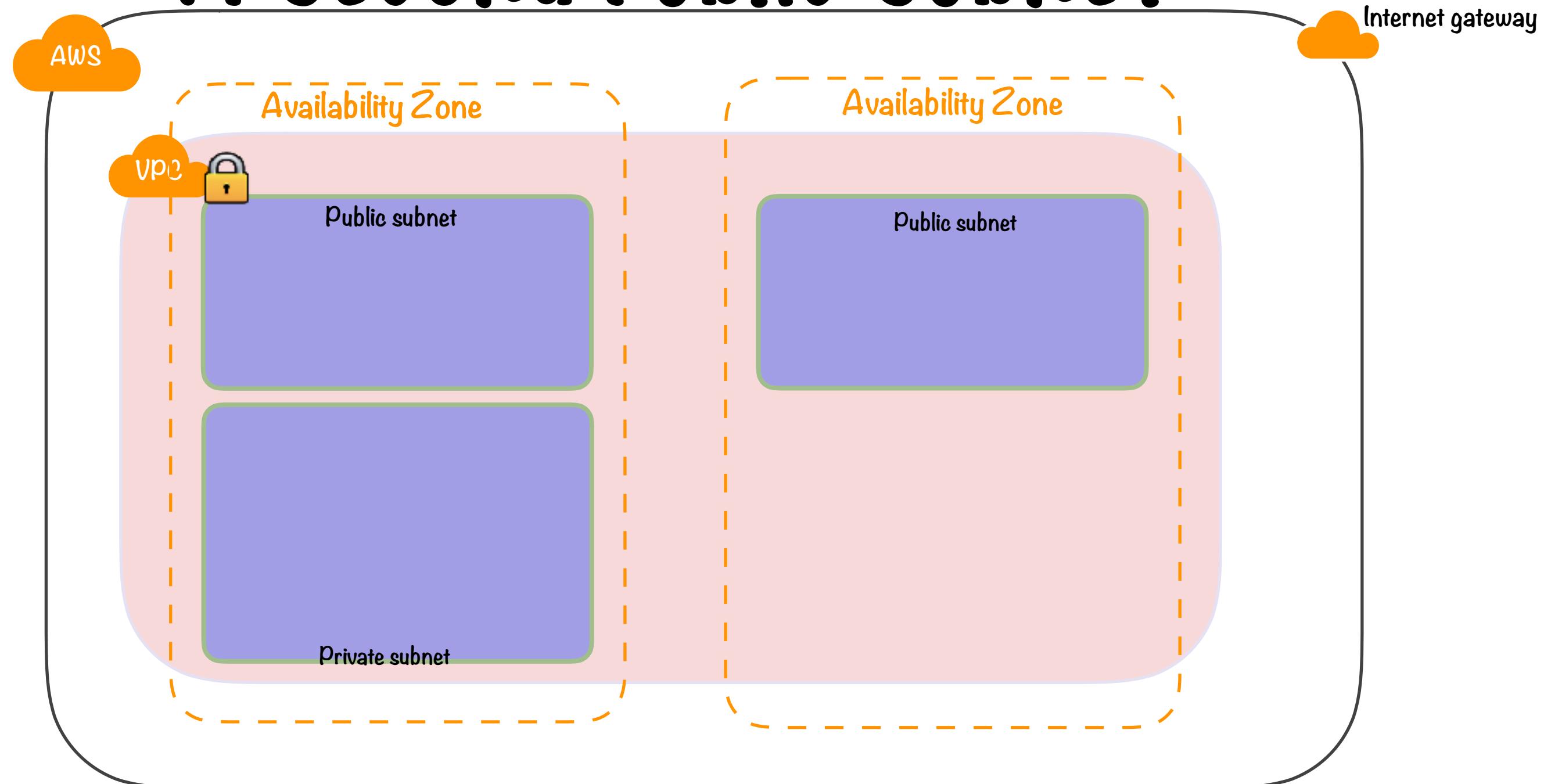
Public and Private Subnet



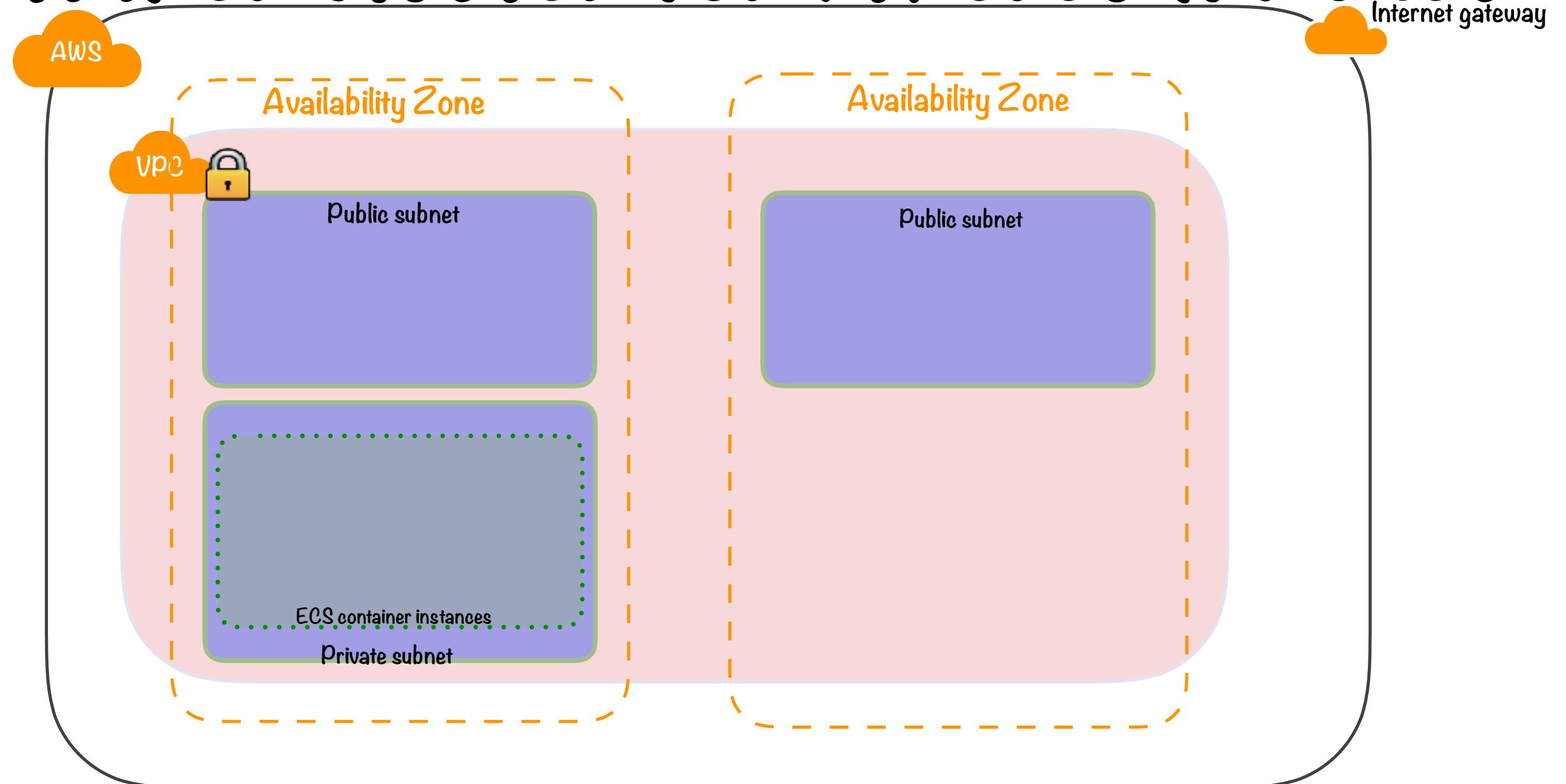
A Second Public Subnet



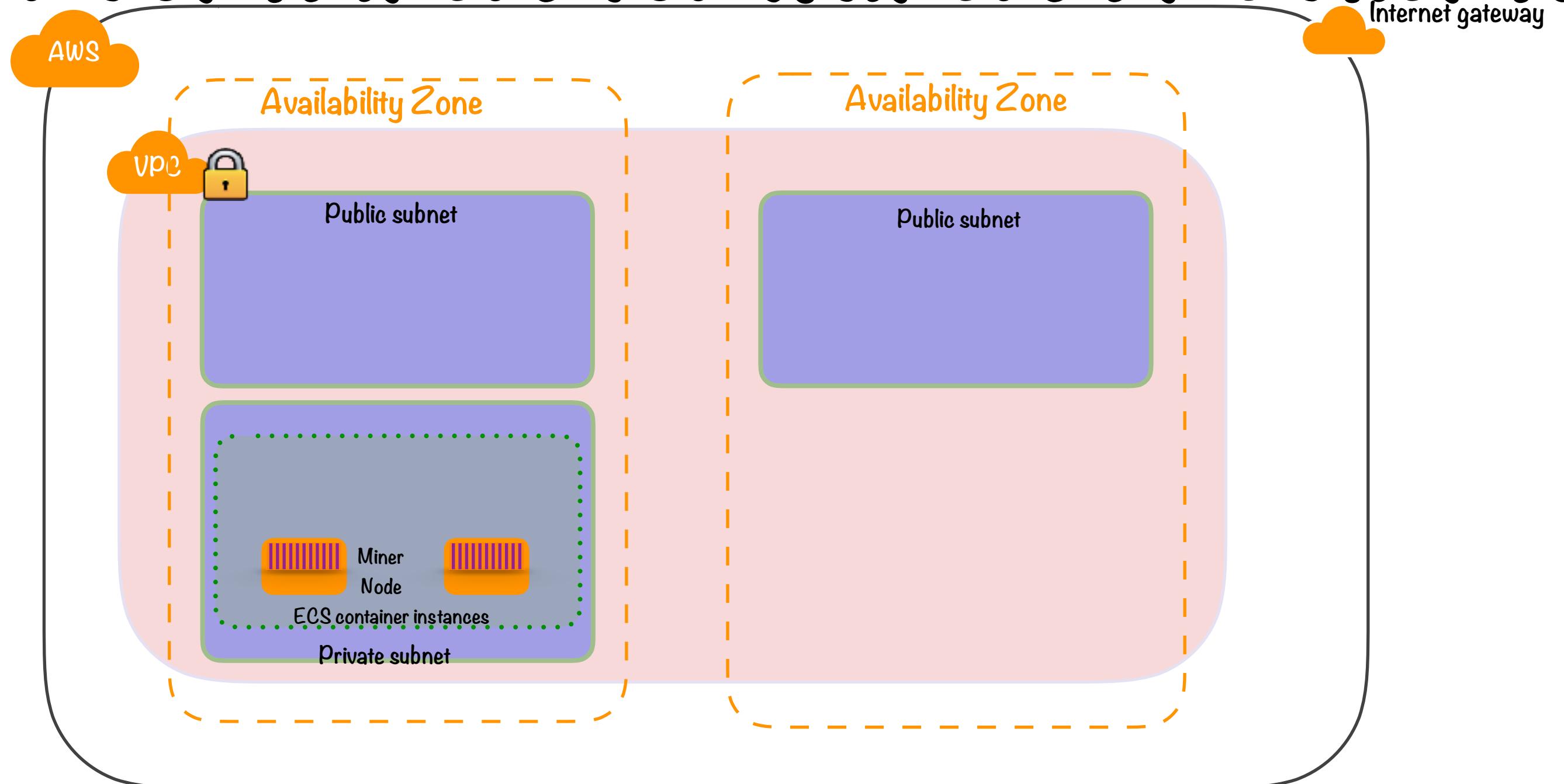
A Second Public Subnet



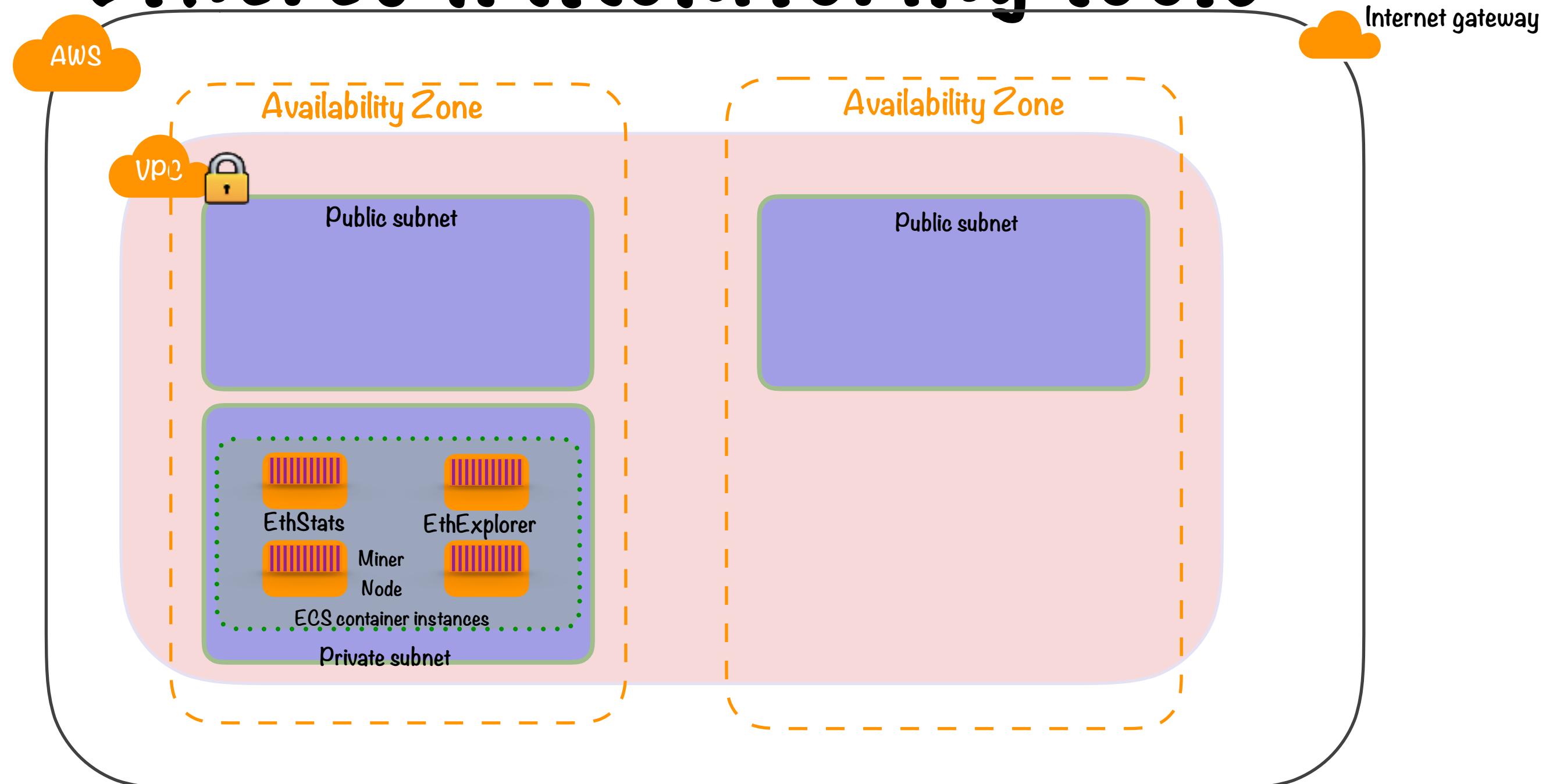
Container Cluster for Ethereum Nodes



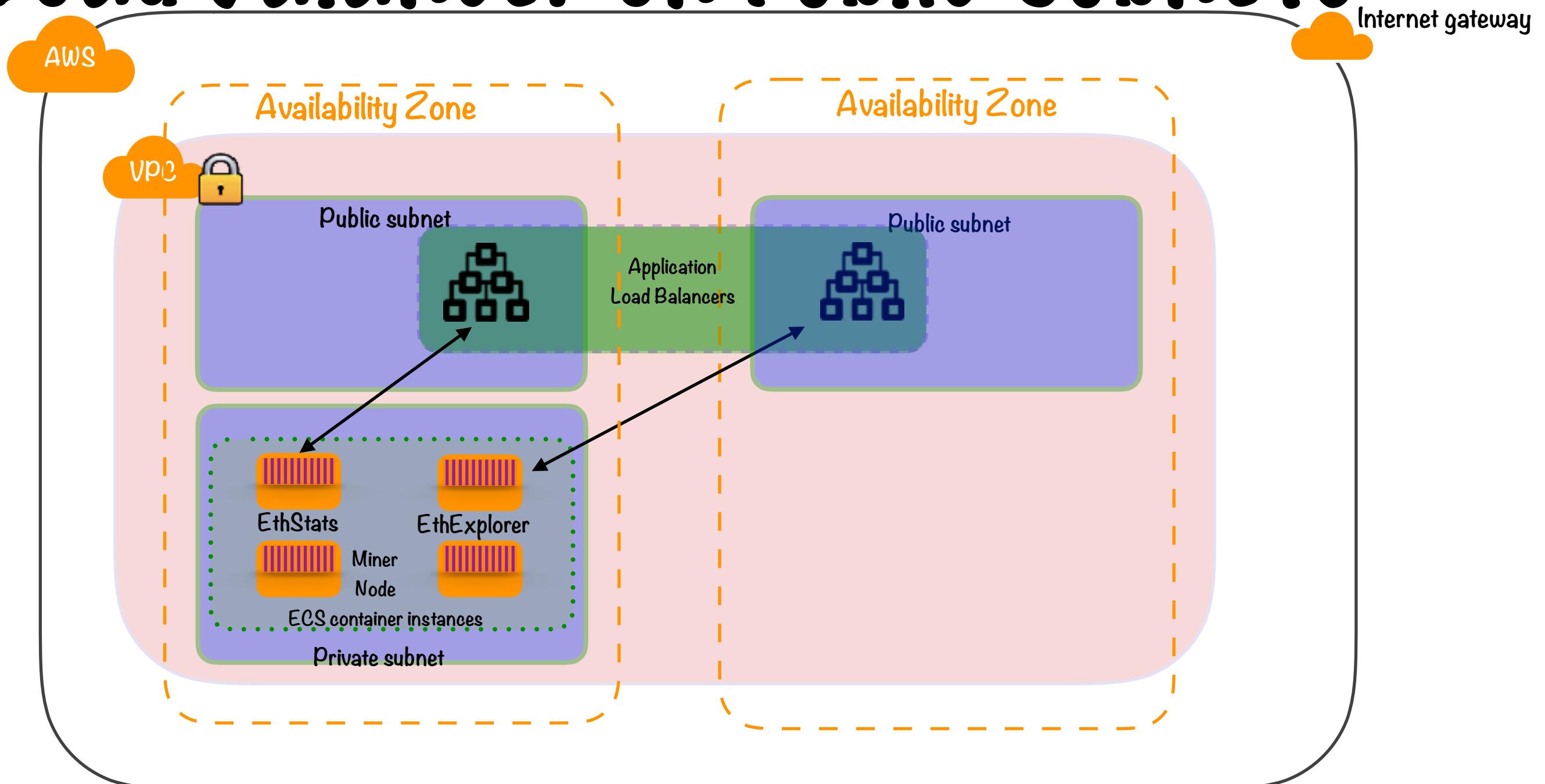
Docker Containers for Miners and Clients



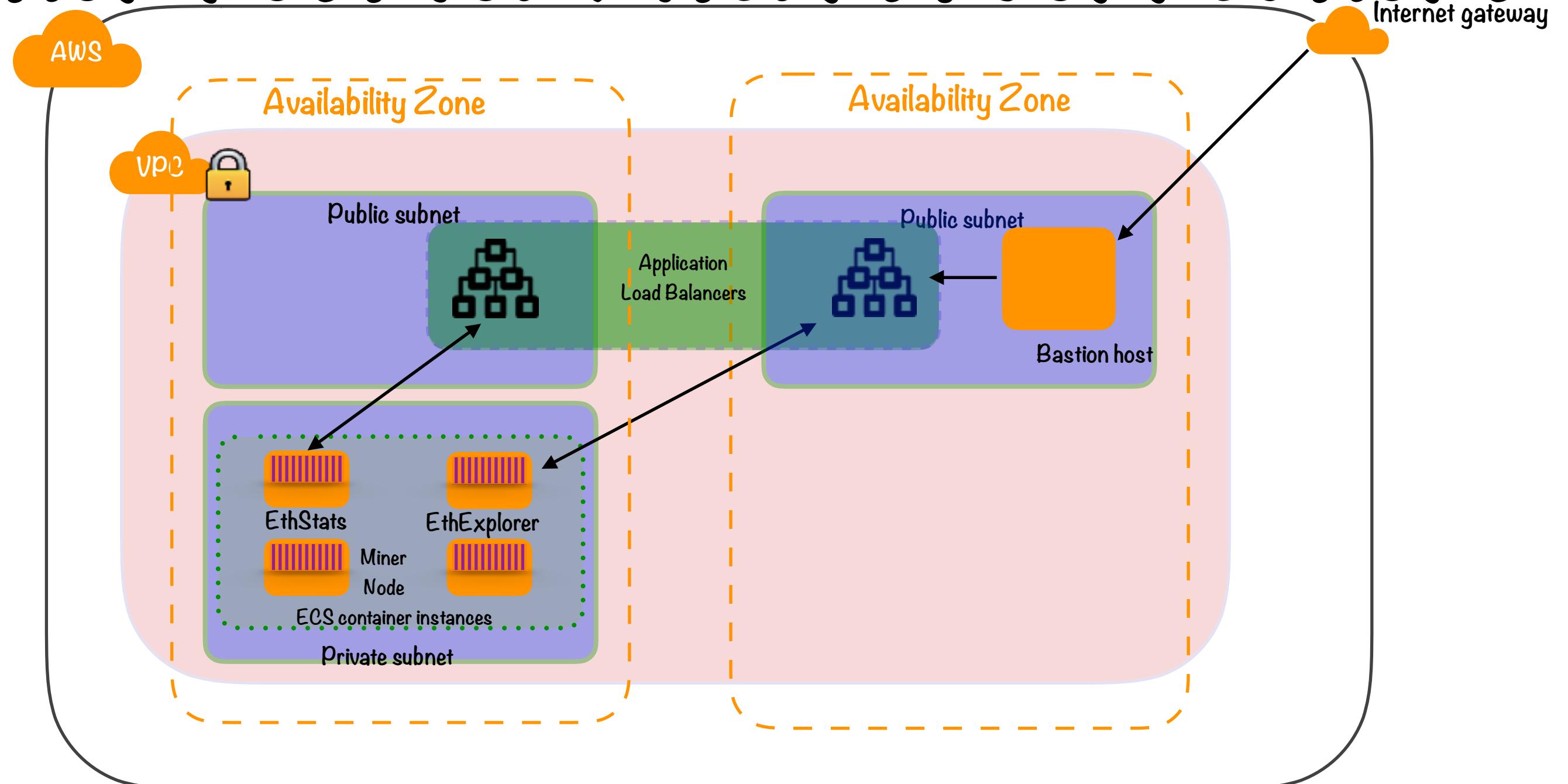
Ethereum Monitoring Tools



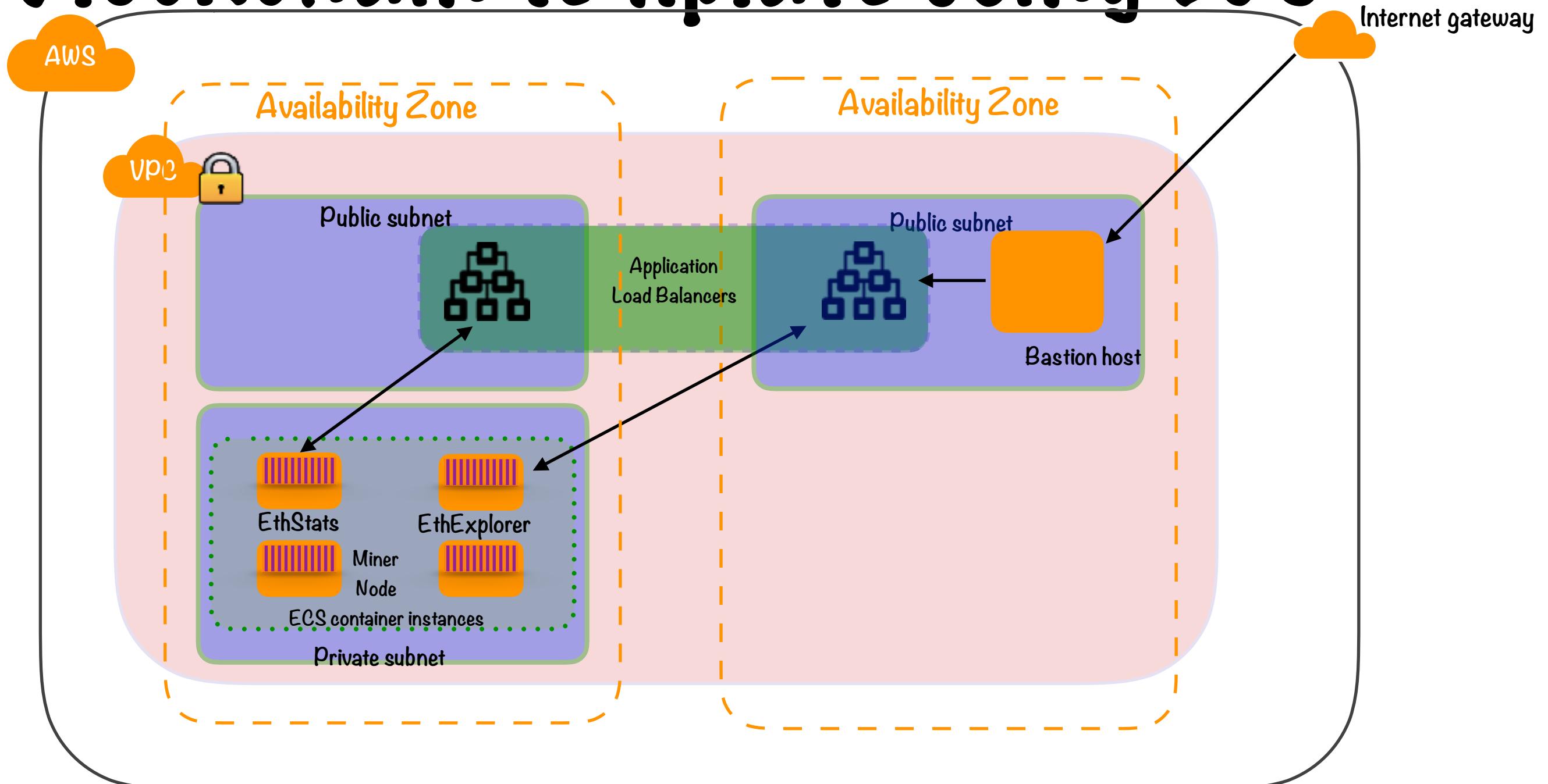
Load Balancer on Public Subnets



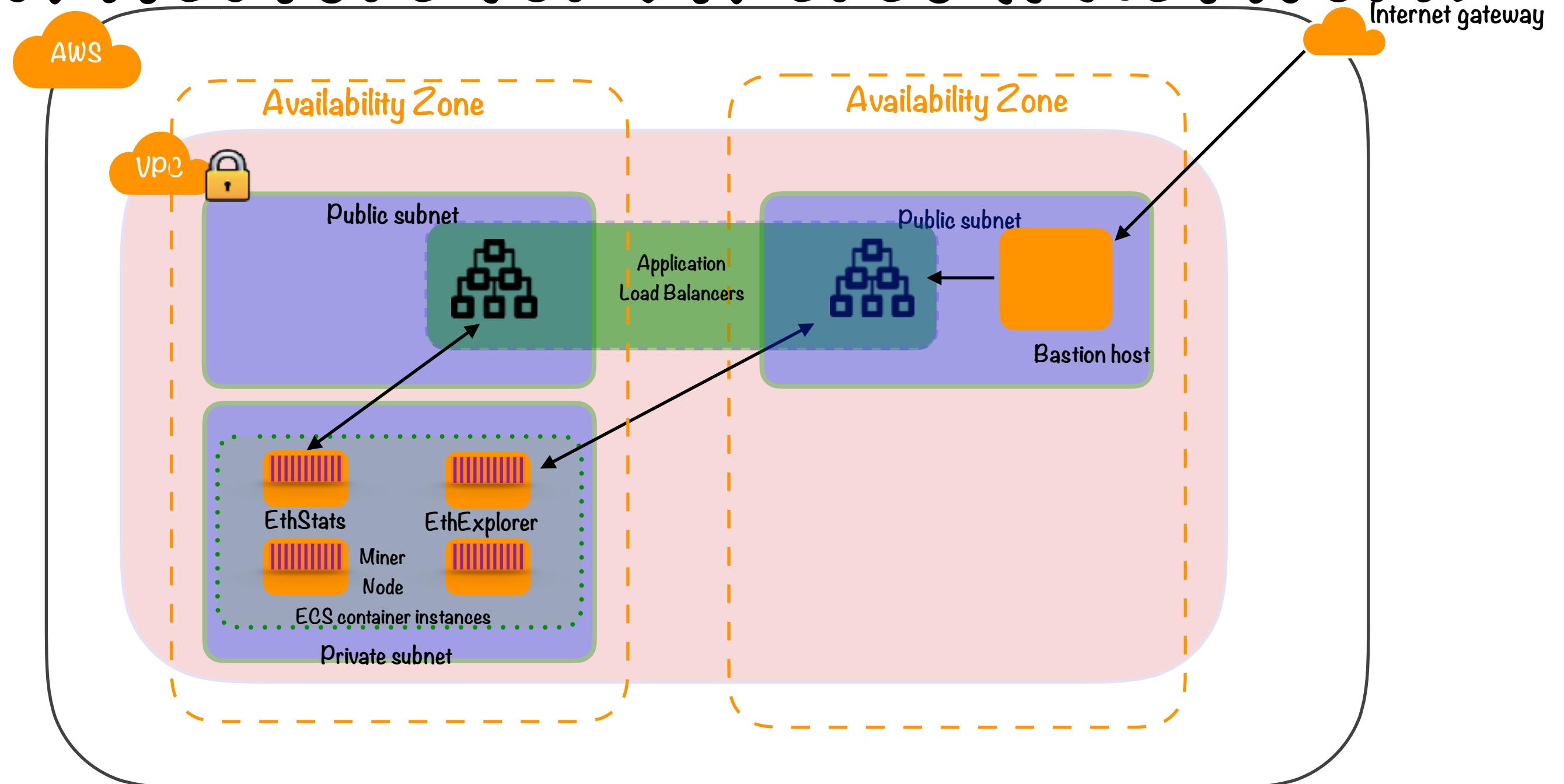
Bastion Host for External Connections



Blockchain Template using ECS



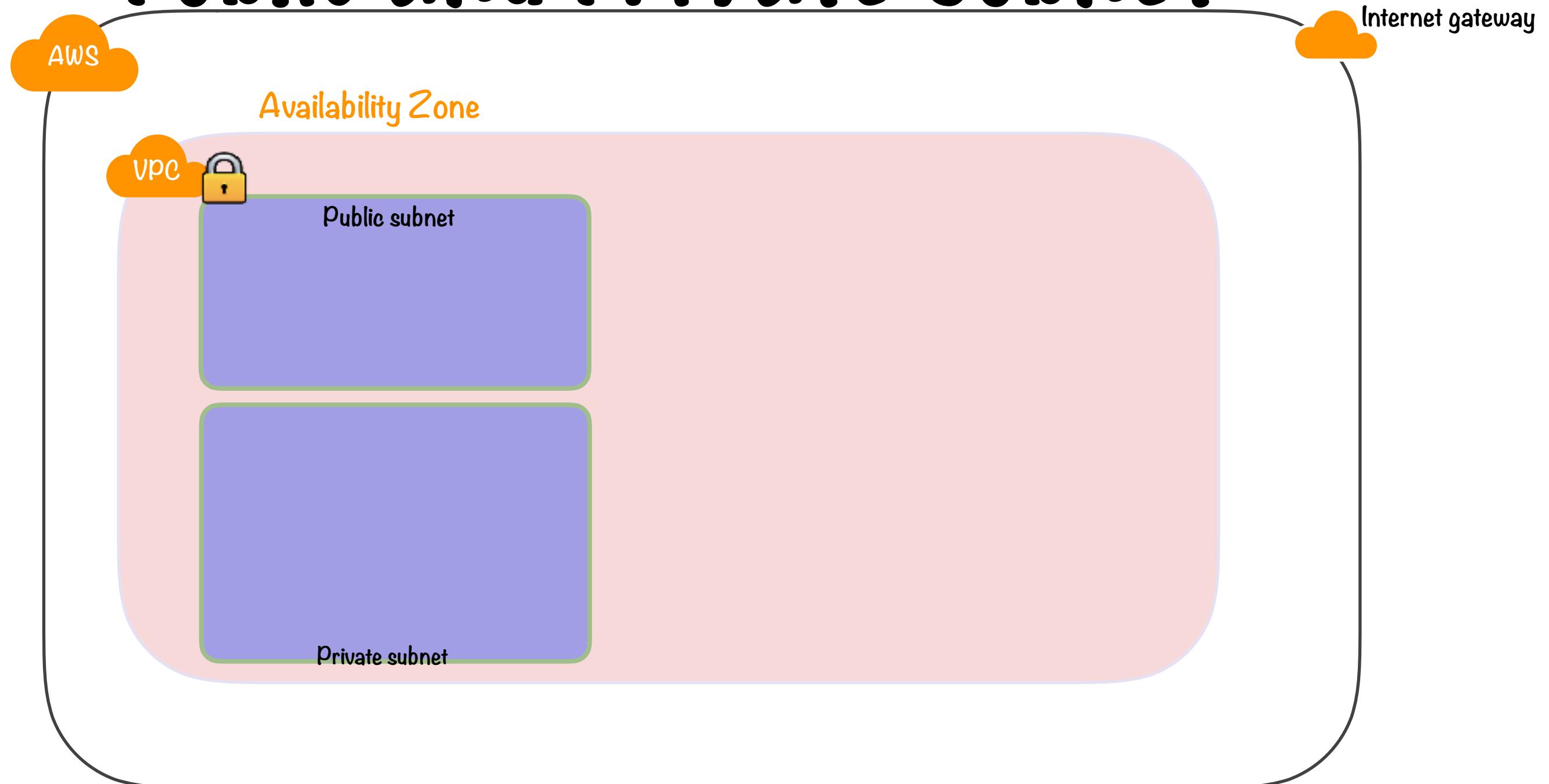
Architecture for Ethereum Network



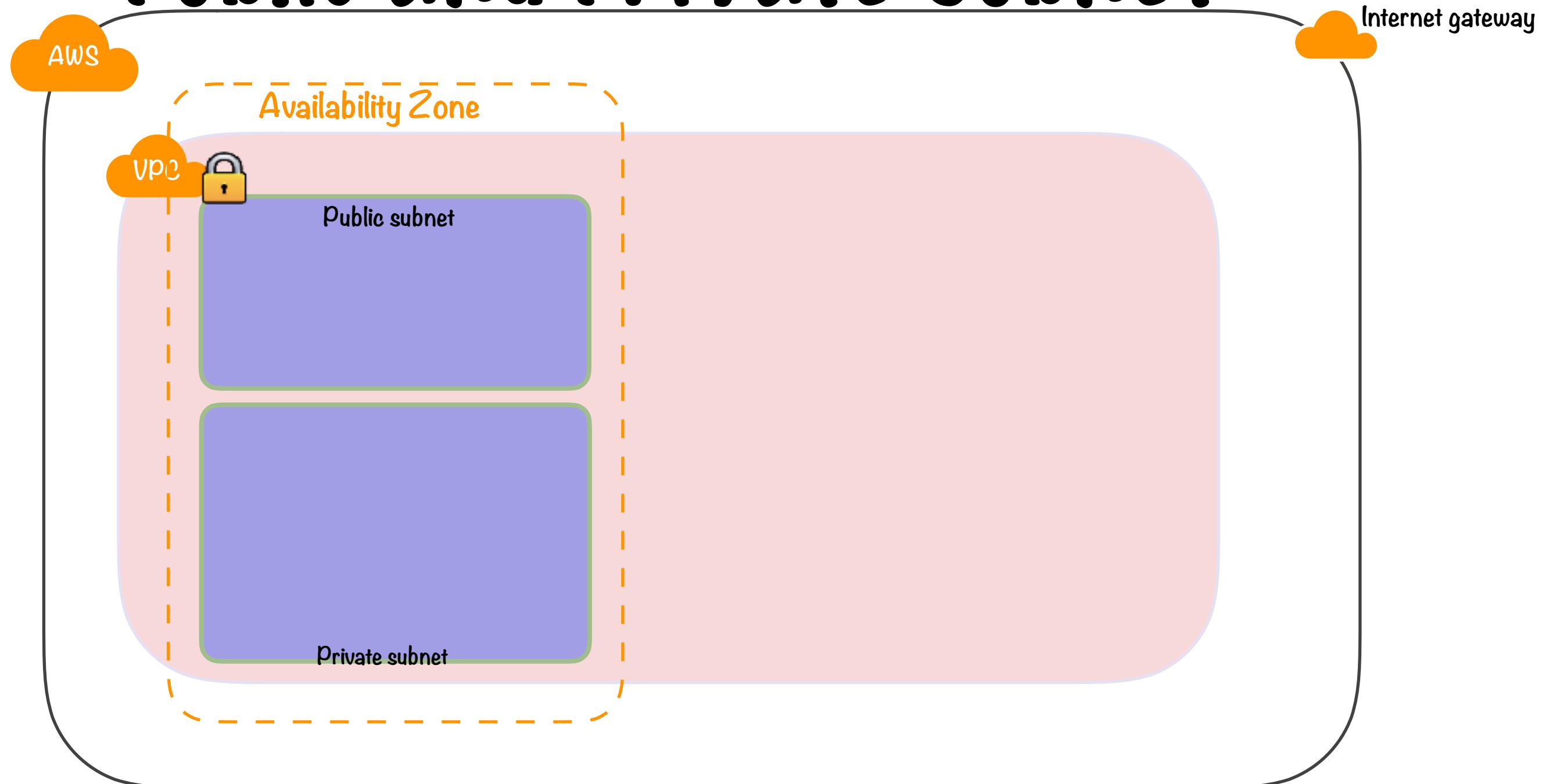
VPC to Hold all EC2 Instances



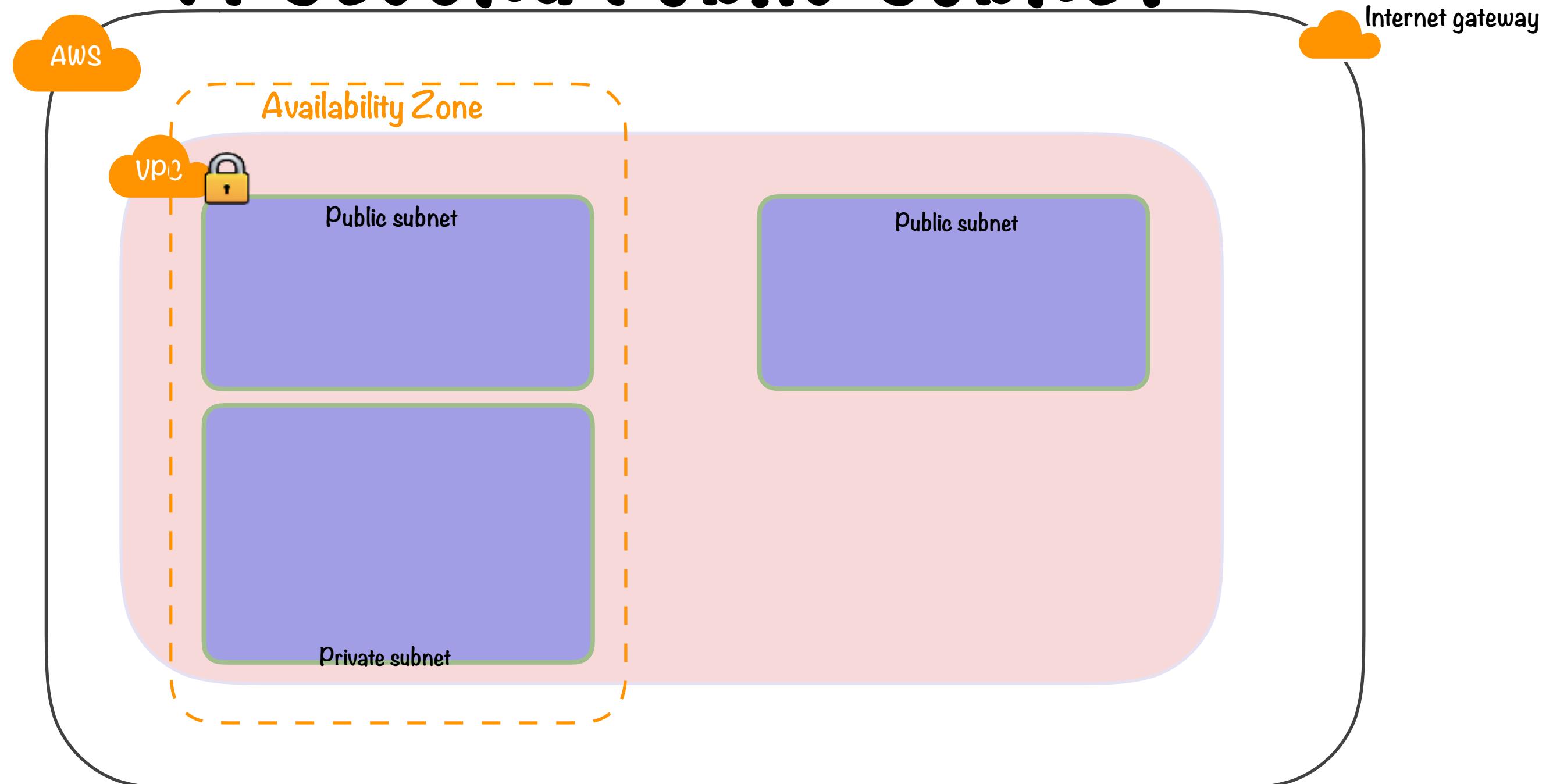
Public and Private Subnet



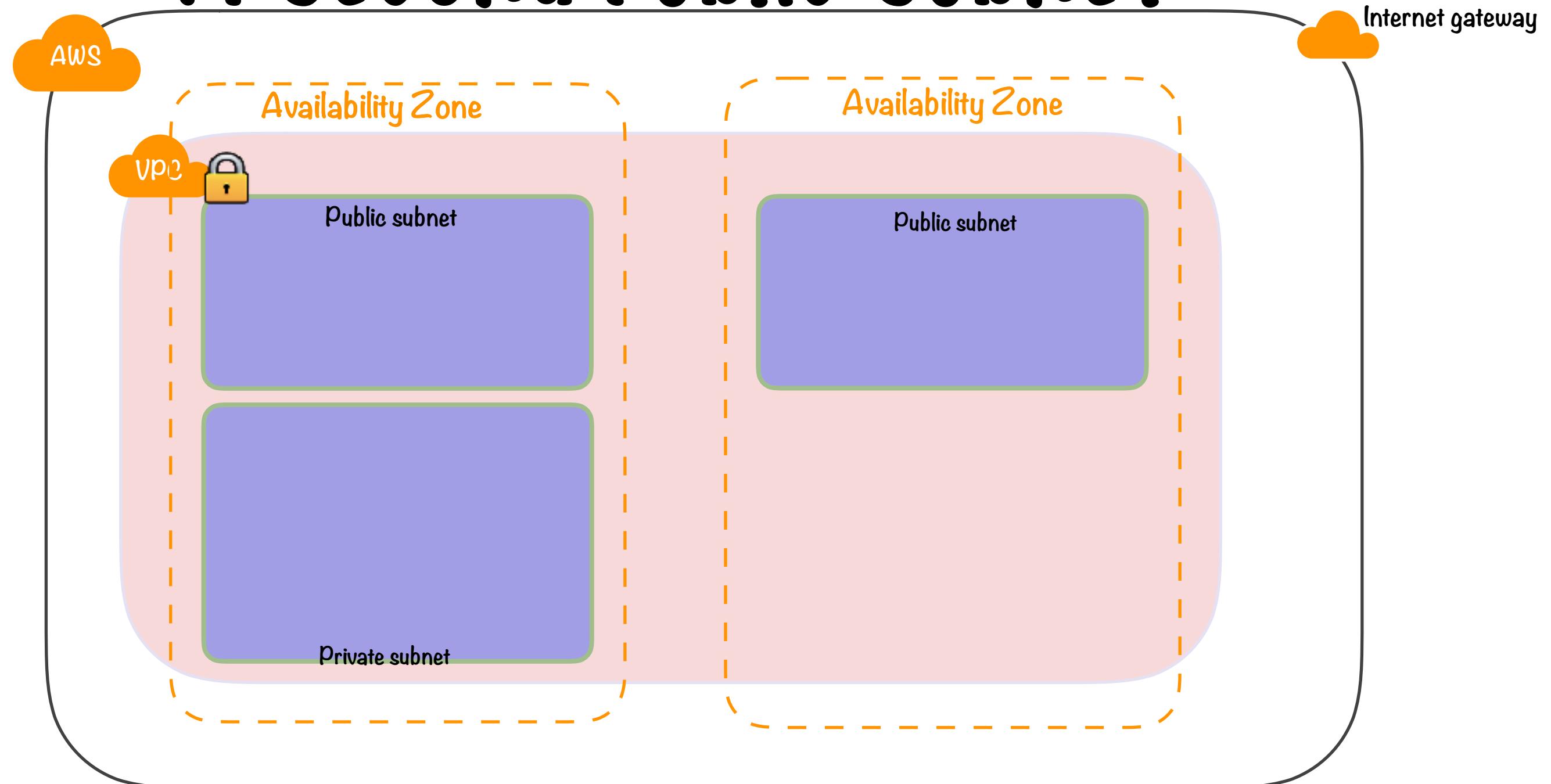
Public and Private Subnet



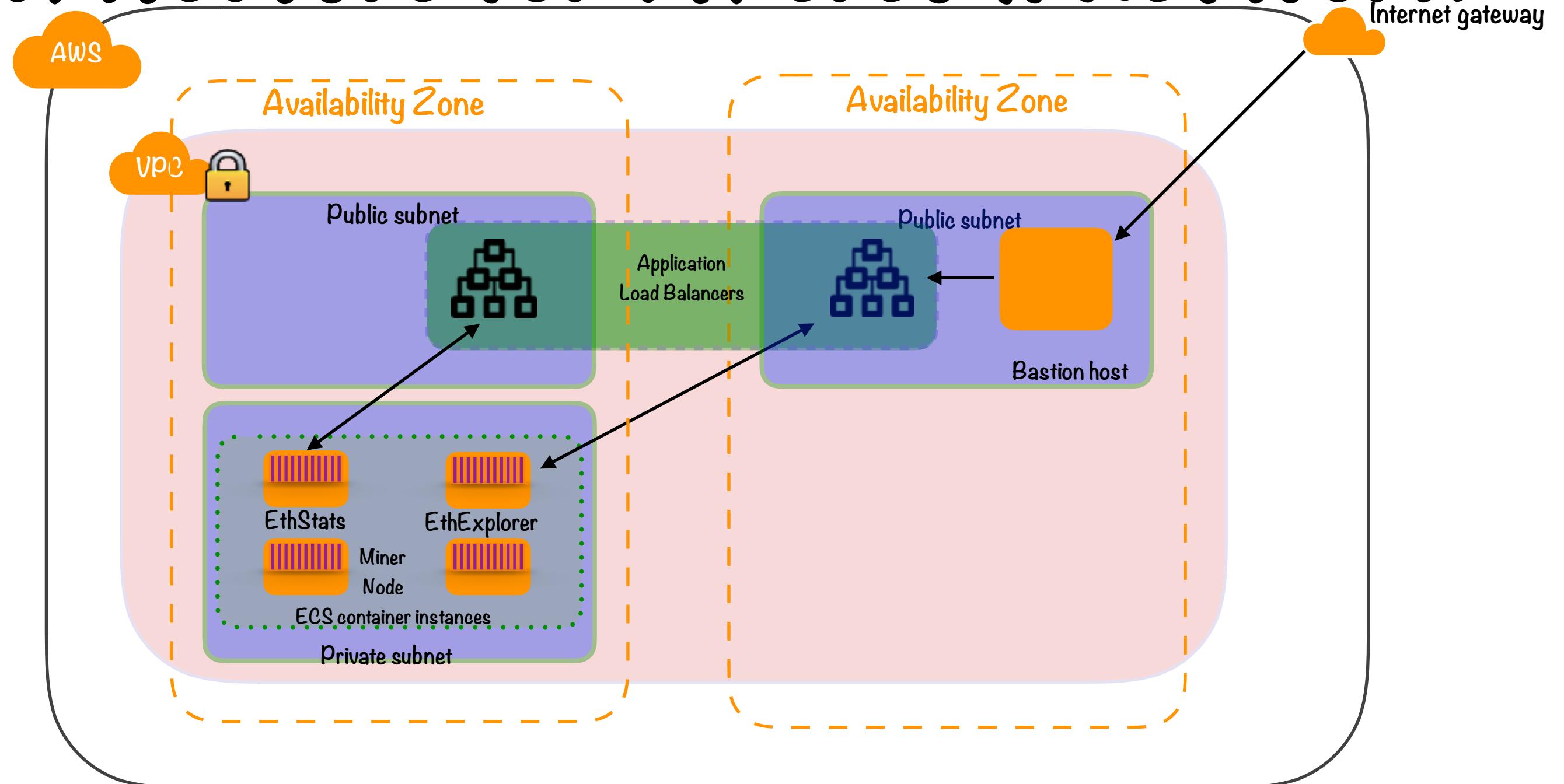
A Second Public Subnet



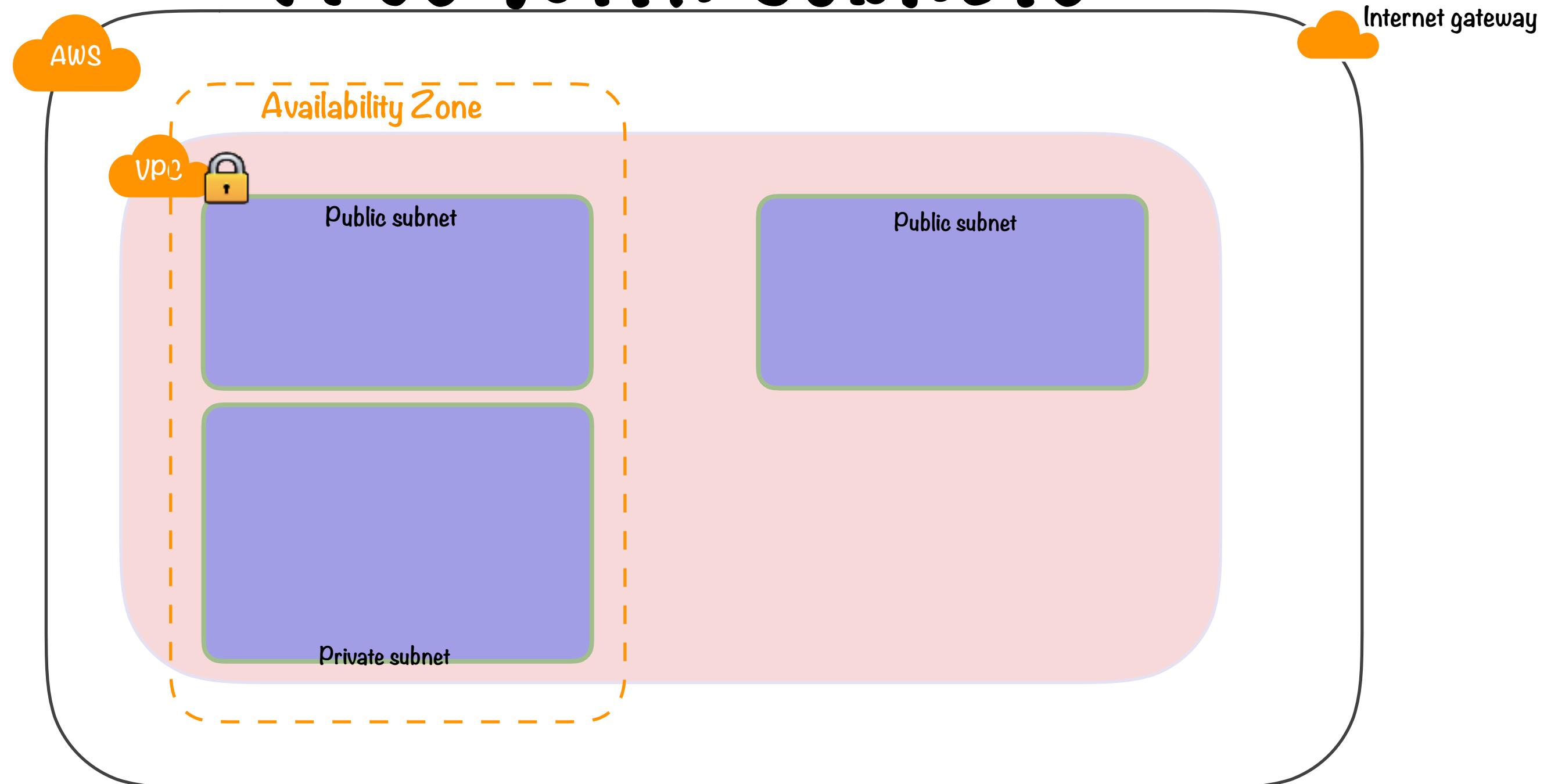
A Second Public Subnet



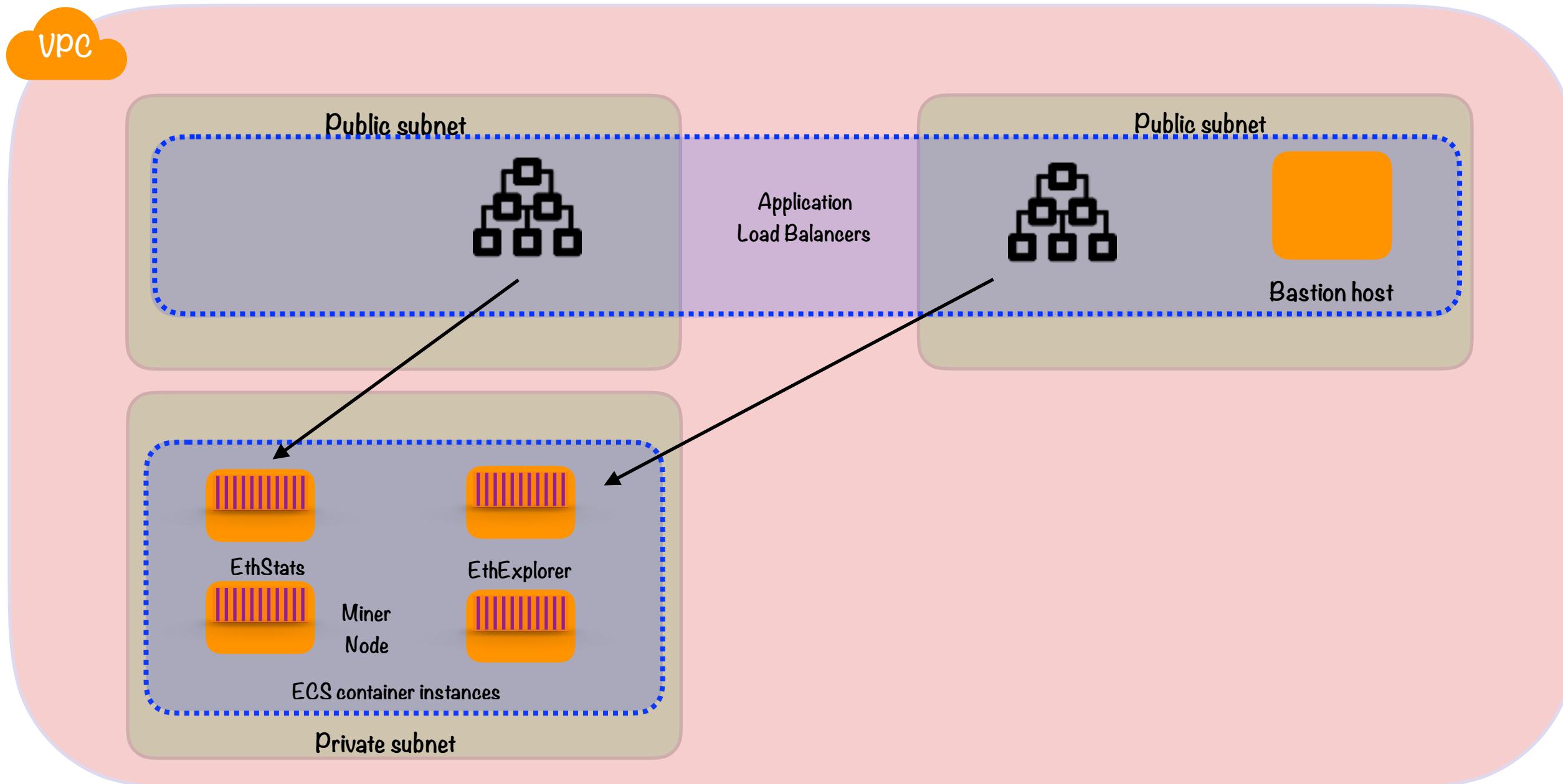
Architecture for Ethereum Network



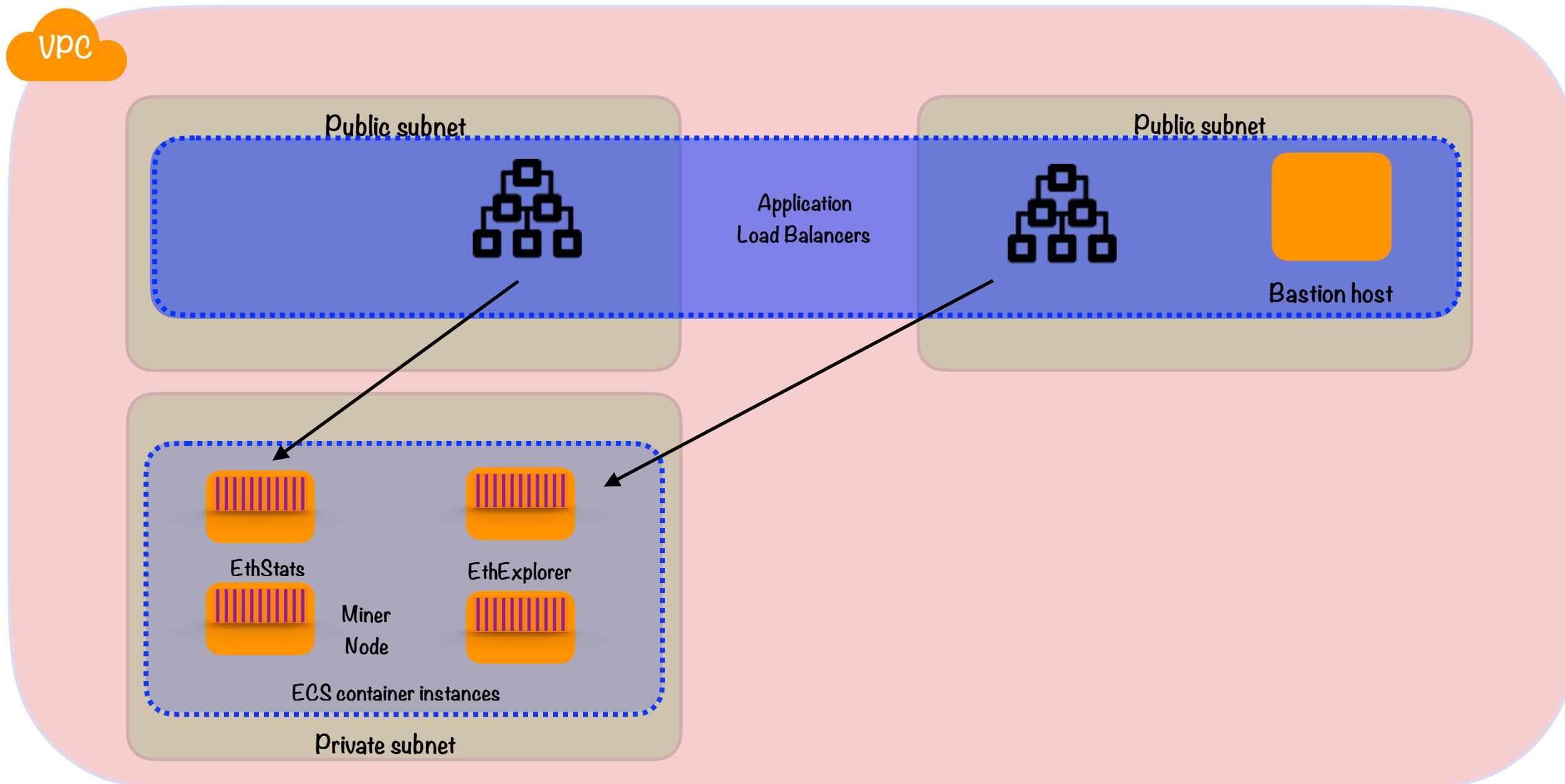
VPCs With Subnets



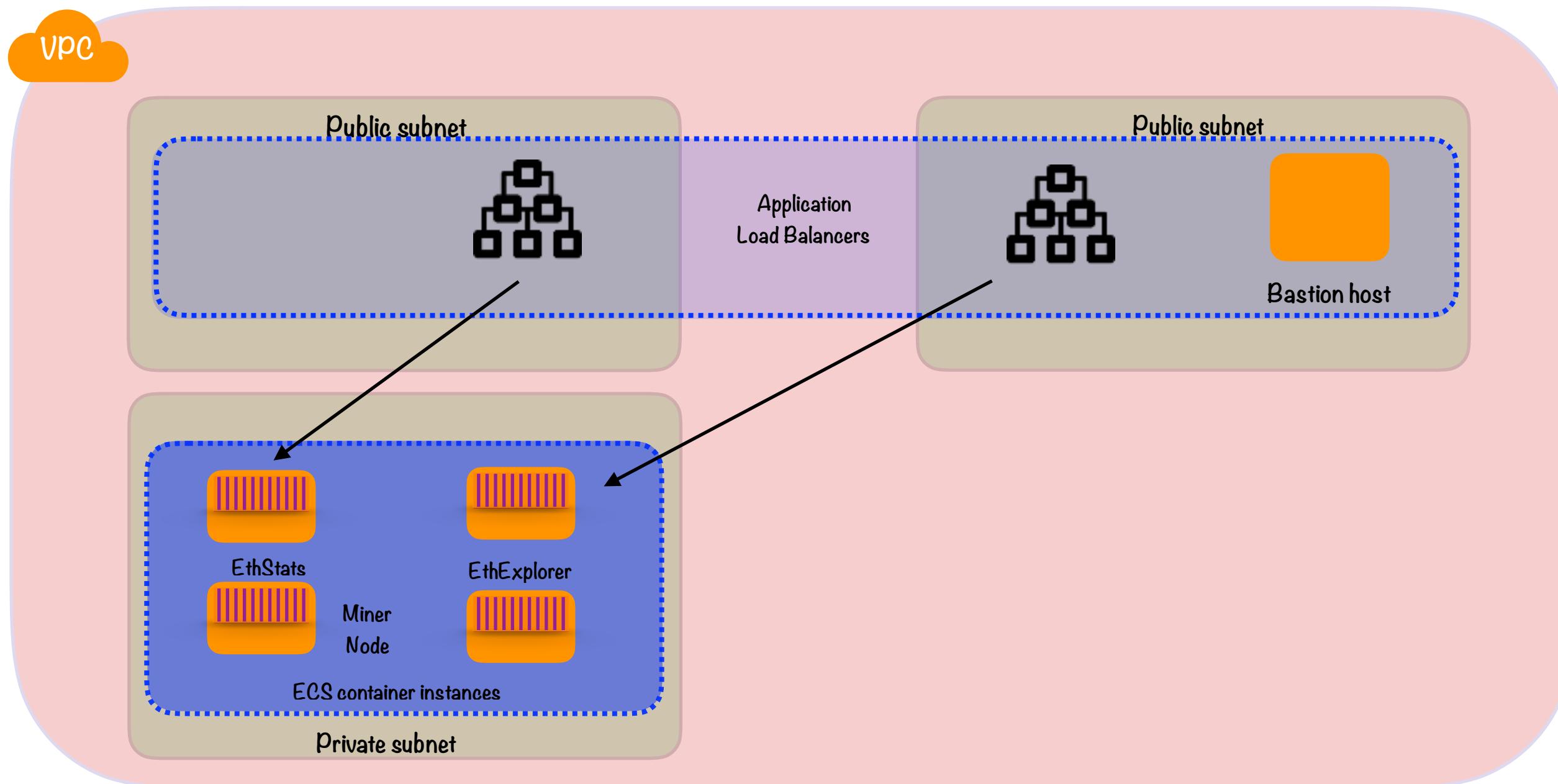
VPCs With Subnets



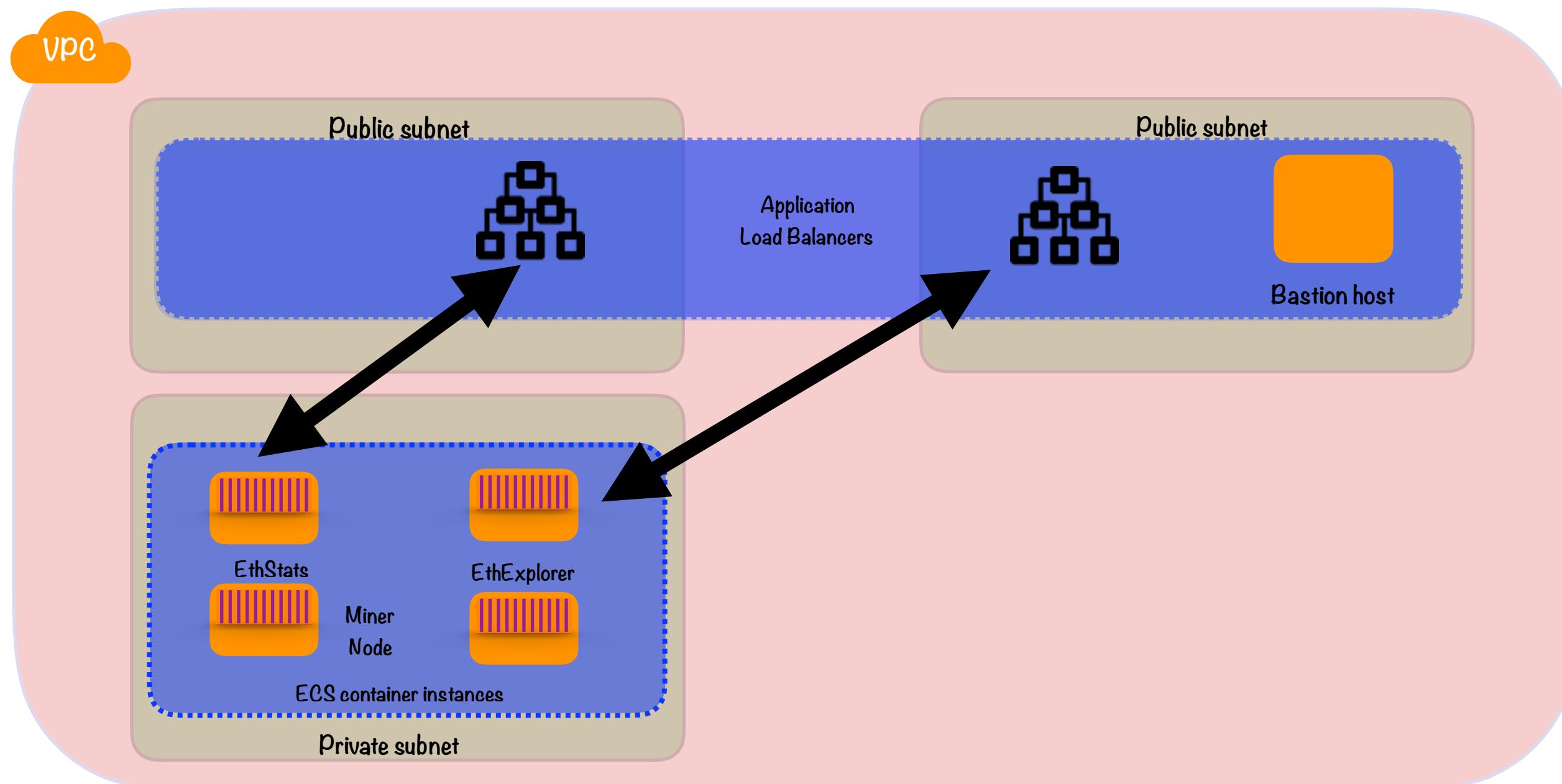
Security Group for The Load Balancer + Bastion Host



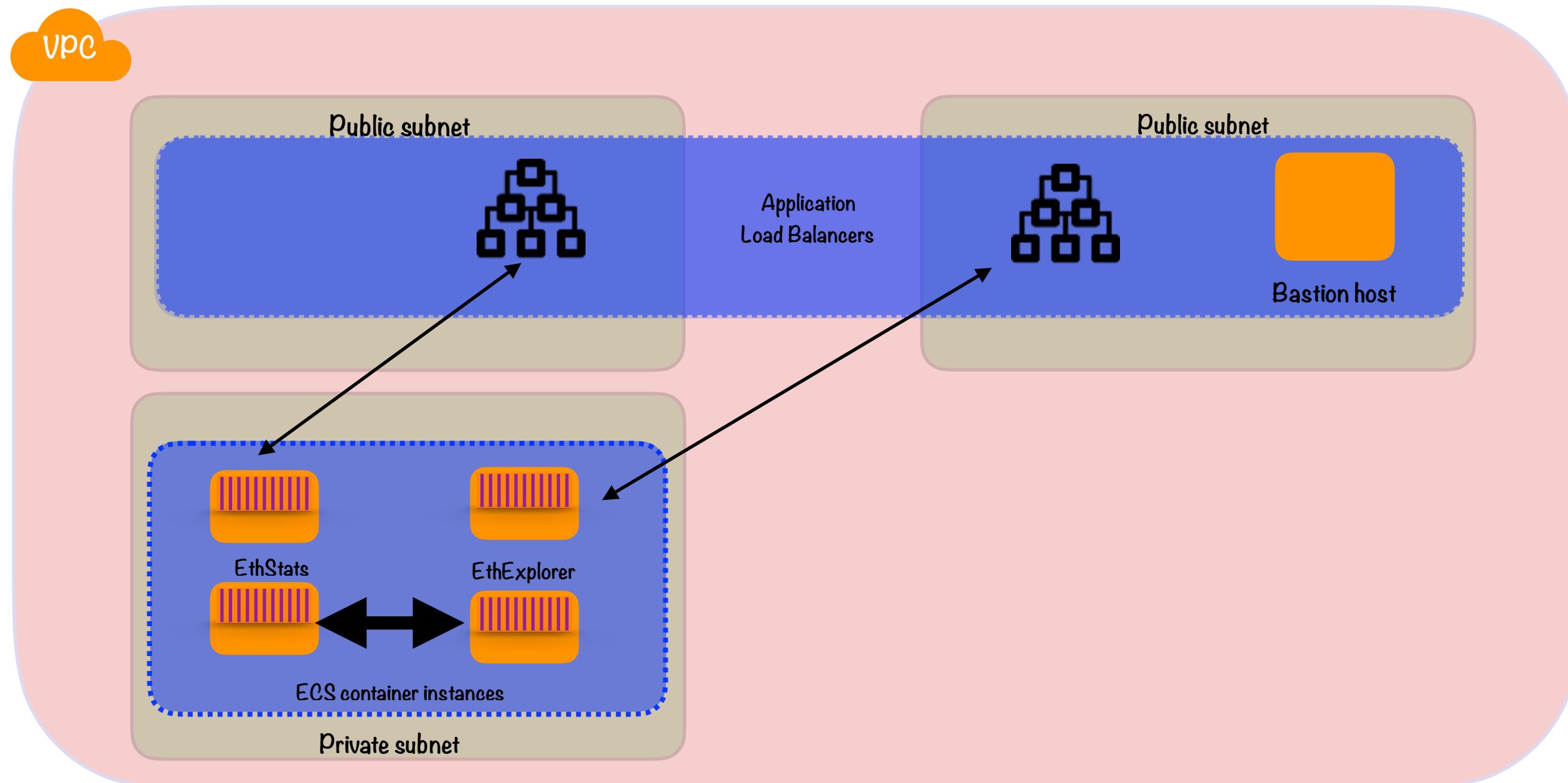
Security Group for The Ethereum Nodes



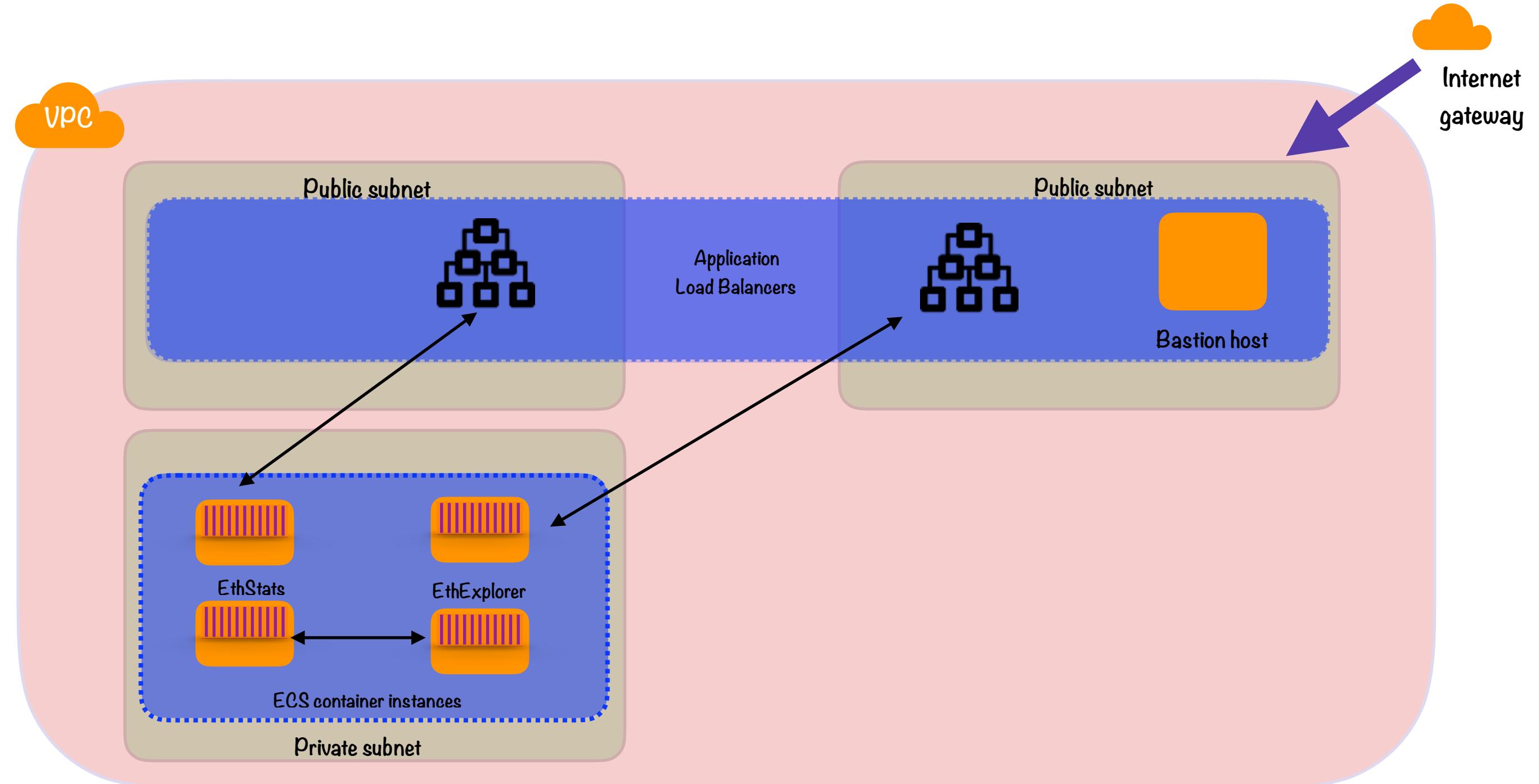
Load Balancer Communicates with Nodes



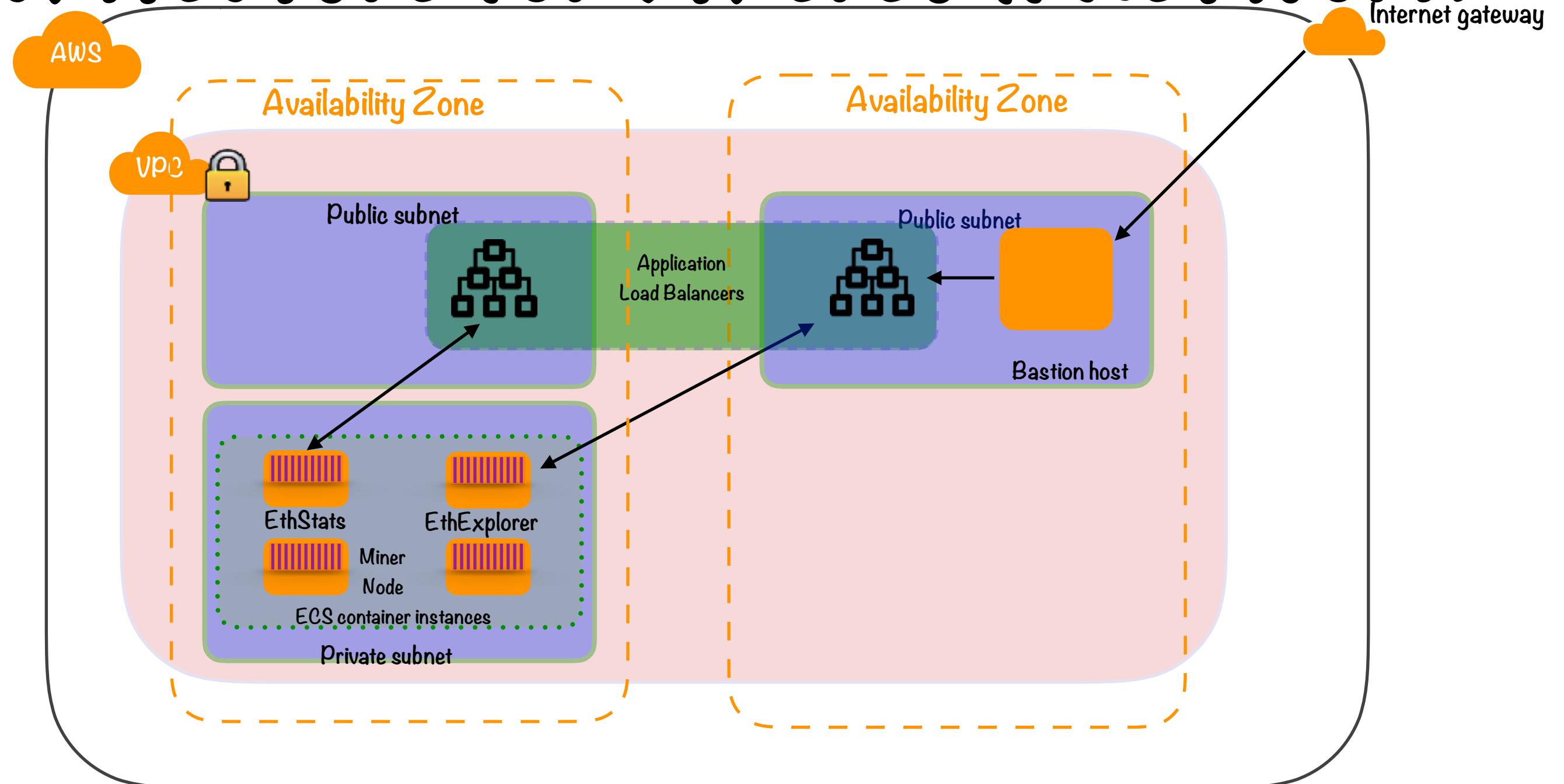
Ethereum Nodes Communicate With Each Other



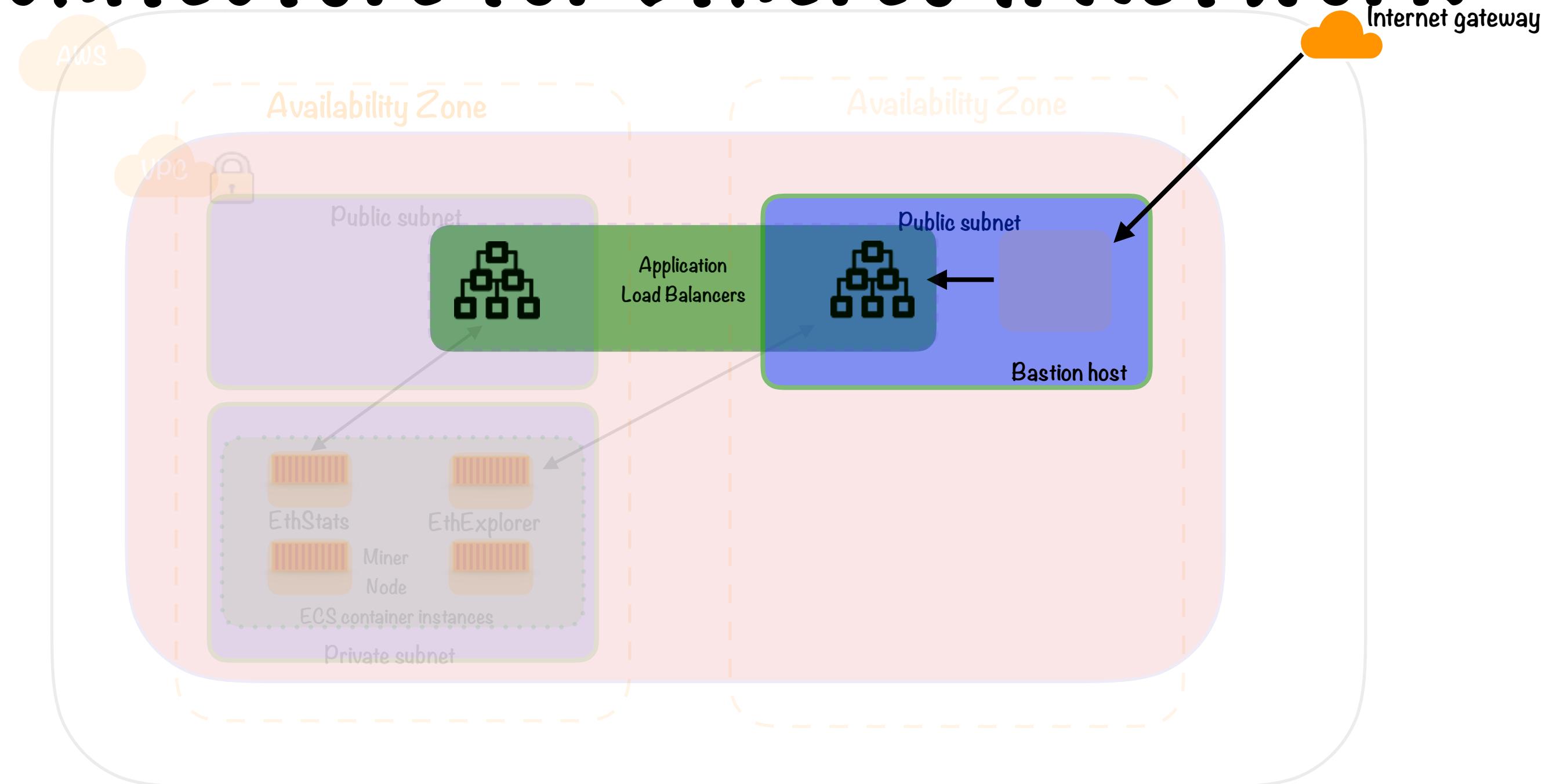
SSH to The Bastion Host From External Hosts



Architecture for Ethereum Network

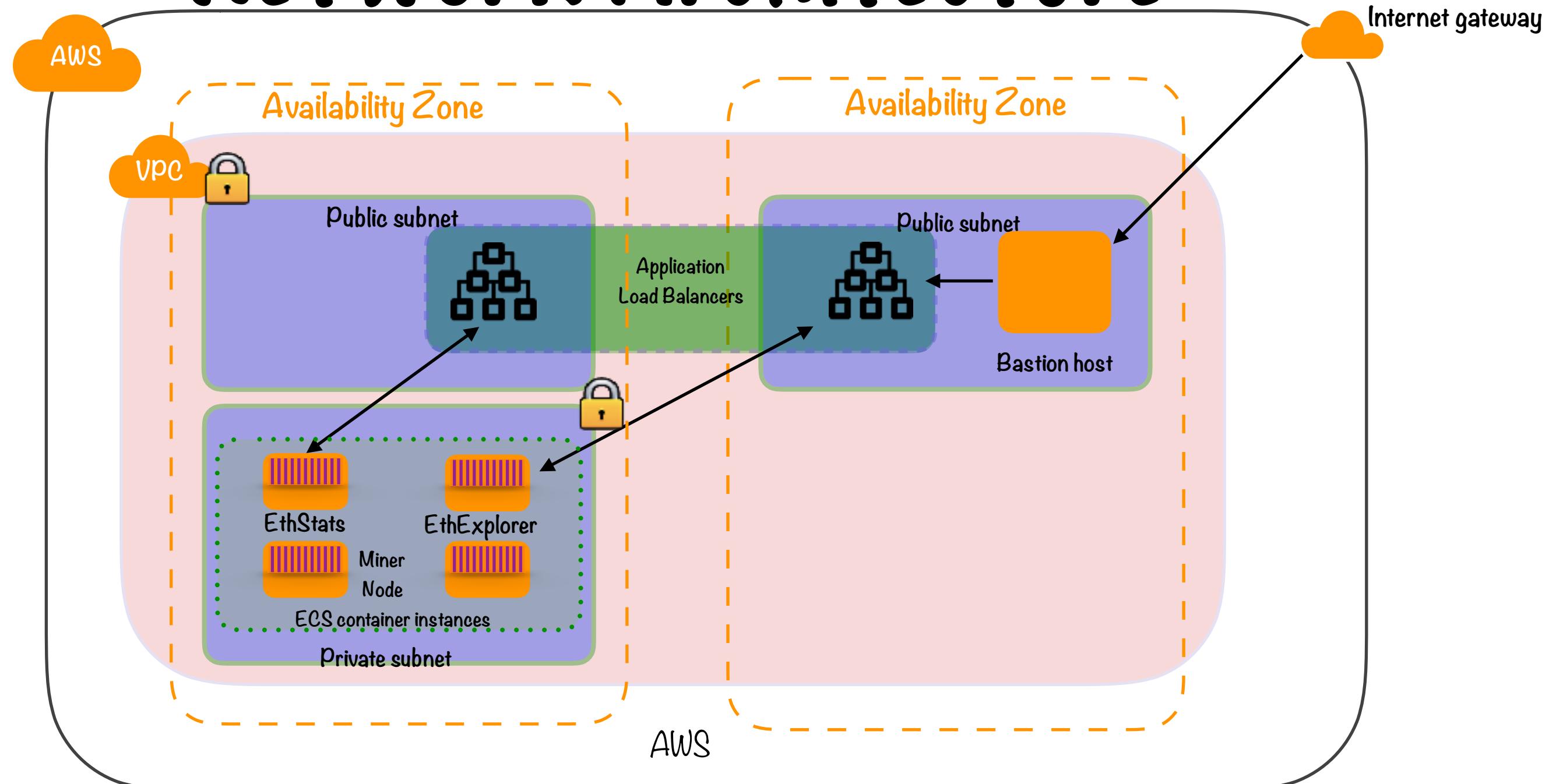


Architecture for Ethereum Network

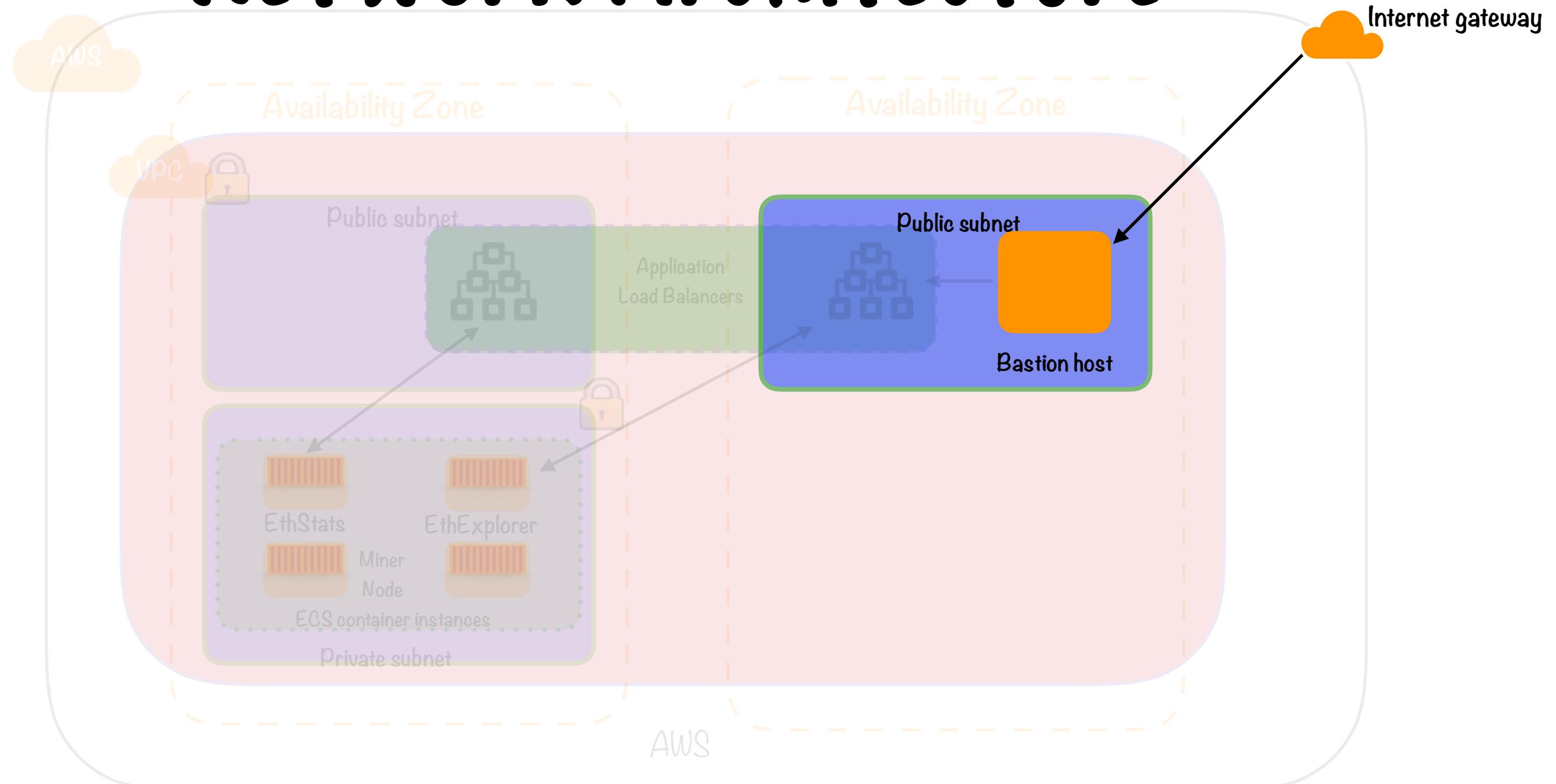


Blockchain Network Architecture on AWS

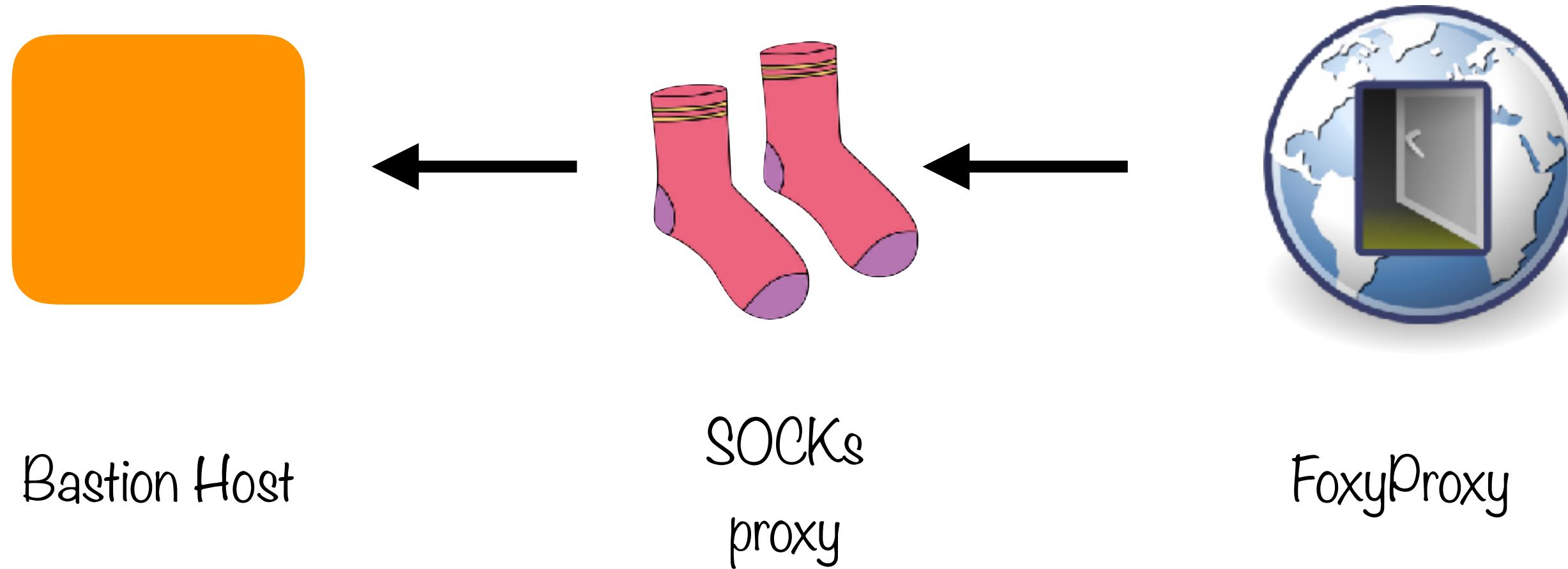
Network Architecture



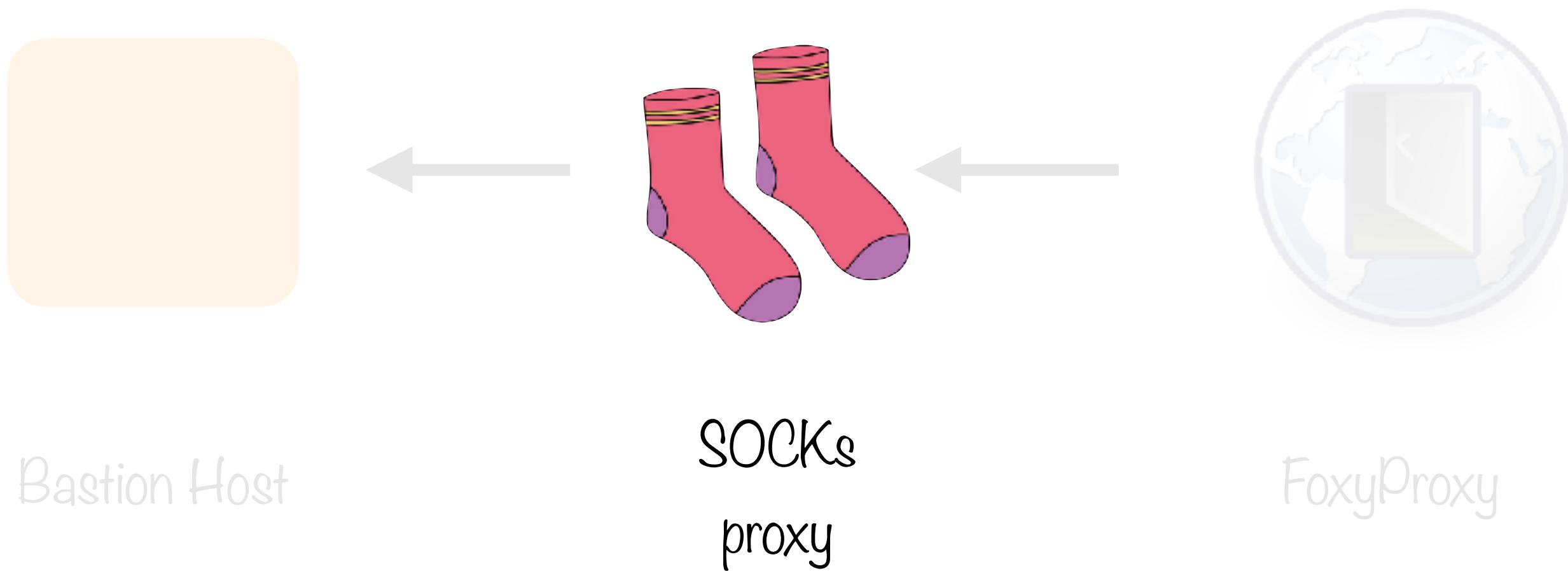
Network Architecture



Connecting to the Bastion Host

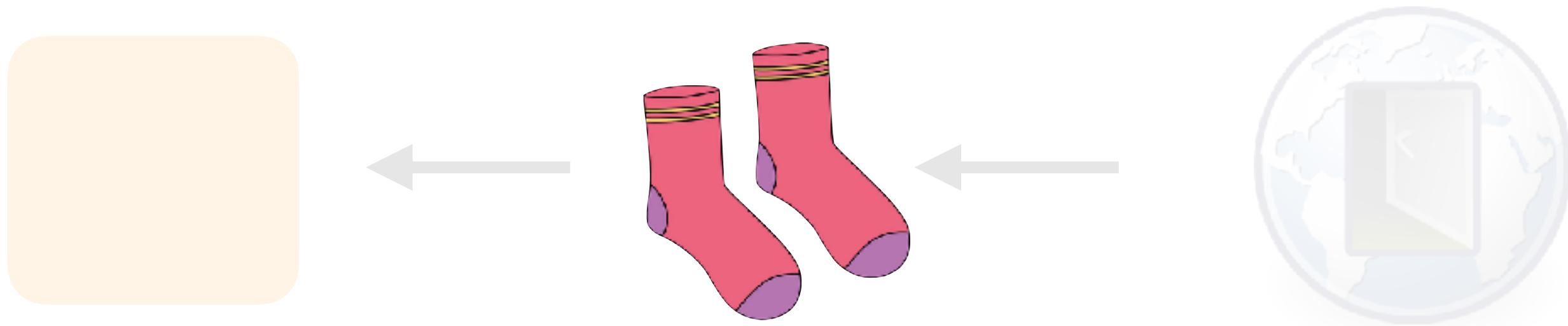


SOCKs Proxy



A proxy acts as an intermediary to connect from our local machine to our AWS network

SOCKs Proxy



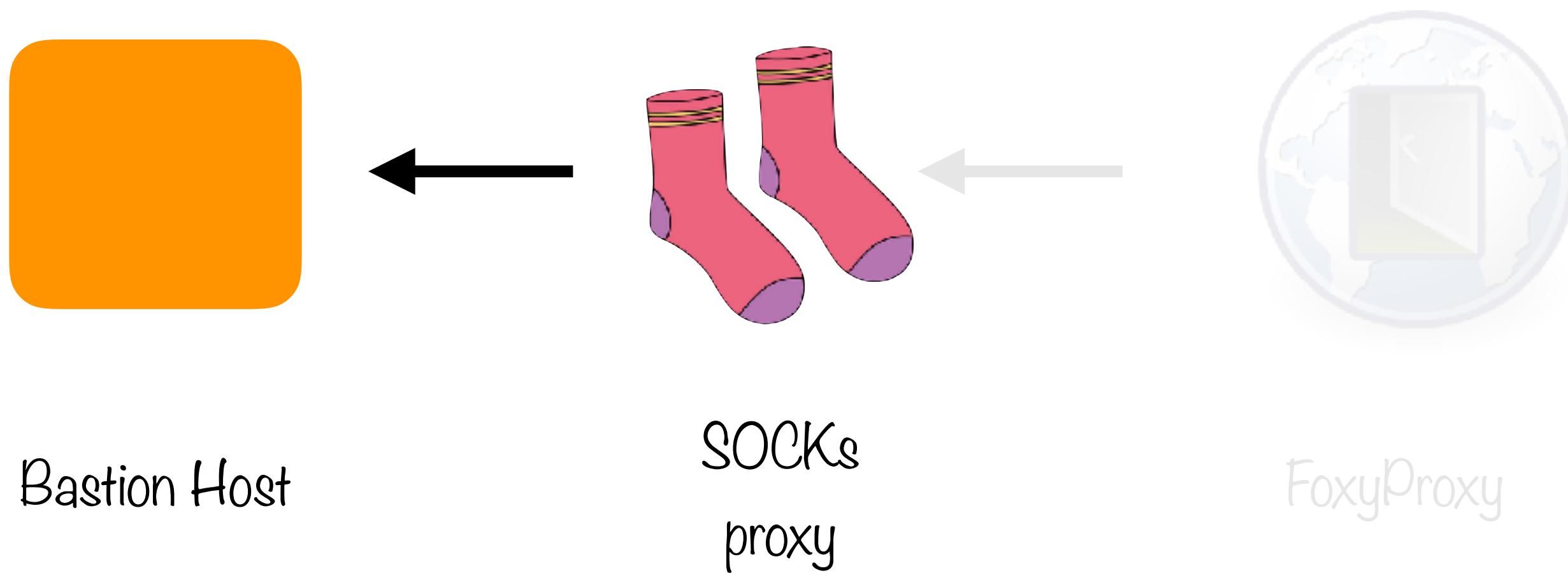
Bastion Host

SOCKs
proxy

FoxyProxy

SOCKS is a general purpose proxy server which we install on our local machine

SOCKs Proxy



Establishes a secure connection to our bastion host

FoxyProxy



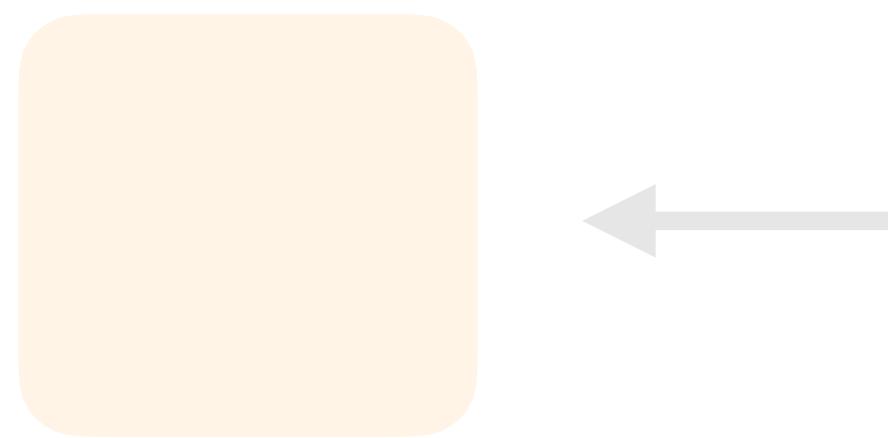
A browser extension which allows you to switch across different proxy servers

FoxyProxy



Matches URL patterns to pick the right proxy server

FoxyProxy



Bastion Host



SOCKs
proxy



FoxyProxy

Installed as a Chrome extension

Summary

A blockchain is a ledger of transactions which is distributed, immutable and verifiable

Ethereum is an open blockchain-based platform which supports Smart Contracts (written in Solidity)

AWS Blockchain Templates automate several tasks for provisioning an Ethereum network

Ethereum transactions incur a fee proportional to their computational complexity (calculated in gas)