

# Introduction

Blockchain

Cryptocurrency

NFT

Decentralization

# What Is a Smart Contract?

Smart contracts are computer programs that are hosted and executed on a blockchain network. Each smart contract consists of code specifying predetermined conditions that, when met, trigger outcomes. By running on a decentralized blockchain instead of a centralized server, smart contracts allow multiple parties to come to a shared result in an accurate, timely, and tamper-proof manner.

Smart contracts are a powerful infrastructure for automation because they are not controlled by a central administrator and are not vulnerable to single points of attack by malicious entities. When applied to multi-party digital agreements, smart contract applications can reduce counterparty risk, increase efficiency, lower costs, and provide new levels of transparency into processes.

## How Smart Contracts Work

Smart contracts are tamper-proof programs on blockchains with the following logic: "if/when x event happens, then execute y action." One smart contract can have multiple different conditions and one application can have multiple different smart contracts to support an interconnected set of processes. There are also multiple smart contract languages for programming, with Ethereum's Solidity being the most popular. Any developer can create a smart contract and deploy it on a public blockchain for their own purposes, e.g., a personal yield aggregator that automatically shifts their funds to the highest-earning application. However, many smart contracts involve multiple independent parties that may or may not know one another and don't necessarily trust one another. The smart contract defines exactly how users can interact with it, involving who can interact with the smart contract, at what times, and what inputs result in what outputs. The result is multi-party digital agreements that evolve from today's probabilistic state, where they will probably execute as desired, to a new deterministic state where they are guaranteed to execute according to their code.

# Communication

## Advantages

**Security** – Running the contract on decentralized blockchain infrastructure ensures there is no central point of failure to attack, no centralized intermediary to bribe, and no mechanism for either party or a central admin to use to tamper with the outcome.

**Reliability** – Having the contract logic redundantly processed and verified by a decentralized network of nodes provides strong tamper-proof, uptime, and correctness guarantees that the contract will execute on time according to its terms.

**Equitable** – Using a decentralized network to host and enforce the terms of the agreement reduces the ability of a for-profit middleman to use their position of privilege to rent-seek and siphon off value.

**Efficiency** – Automating the backend processes of the agreement—escrow, maintenance, execution, and/or settlement—means neither party has to wait for manual data to be entered, the counterparty to fulfill their obligations, or a middleman to process the transaction.

## Smart Contract Limitations

One of the inherent limitations of smart contracts is that the underlying blockchains they run on are isolated networks, meaning blockchains have no built-in connection to the outside world. Without external connectivity, smart contracts cannot communicate with external systems to confirm the occurrence of real-world events nor can they access cost-efficient computational resources. Similar to a computer without the Internet, smart contracts are extremely limited without real-world connectivity. For example, they can't know the price of an asset before executing a trade, they can't check the average monthly rainfall before paying out a crop insurance claim, and they cannot verify that goods have arrived before settling with a supplier.

Thus, the major evolution underway in the blockchain industry is programmable smart contracts that connect with real-world data and traditional systems outside a blockchain, expanding the inputs and outputs used within smart contract logic. These hybrid smart contracts use secure middleware known as an oracle to combine on-chain code with off-chain infrastructure —e.g., trigger a smart contract with external data or settle a contract off-chain on a traditional payment rail.



# What Is a Blockchain?

A blockchain is a distributed database or ledger that is shared among the nodes of a computer network. As a database, a blockchain stores information electronically in digital format. Blockchains are best known for their crucial role in cryptocurrency systems, such as Bitcoin, for maintaining a secure and decentralized record of transactions. The innovation with a blockchain is that it guarantees the fidelity and security of a record of data and generates trust without the need for a trusted third party.

One key difference between a typical database and a blockchain is how the data is structured. A blockchain collects information together in groups, known as blocks, that hold sets of information. Blocks have certain storage capacities and, when filled, are closed and linked to the previously filled block, forming a chain of data known as the blockchain. All new information that follows that freshly added block is compiled into a newly formed block that will then also be added to the chain once filled.

With many practical applications for the technology already being implemented and explored, blockchain is finally making a name for itself in no small part because of bitcoin and cryptocurrency. As a buzzword on the tongue of every investor in the nation, blockchain stands to make business and government operations more accurate, efficient, secure, and cheap, with fewer middlemen.

As we prepare to head into the third decade of blockchain, it's no longer a question of if legacy companies will catch on to the technology—it's a question of when. Today, we see a proliferation of NFTs and the tokenization of assets. The next decades will prove to be an important period of growth for blockchain.

# How Does a Blockchain Work?

The goal of blockchain is to allow digital information to be recorded and distributed, but not edited. In this way, a blockchain is the foundation for immutable ledgers, or records of transactions that cannot be altered, deleted, or destroyed. This is why blockchains are also known as a distributed ledger technology (DLT).



A new transaction is entered.



The transaction is then transmitted to a network of peer-to-peer computers scattered across the world.



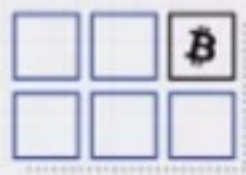
This network of computers then solves equations to confirm the validity of the transaction.



The transaction is complete.



These blocks are then chained together creating a long history of all transactions that are permanent.



Once confirmed to be legitimate transactions, they are clustered together into blocks.



# Blockchain Decentralization

What a blockchain does is to allow the data held in that database to be spread out among several network nodes at various locations. This not only creates redundancy but also maintains the fidelity of the data stored therein—if somebody tries to alter a record at one instance of the database, the other nodes would not be altered and thus would prevent a bad actor from doing so. If one user tampers with Bitcoin's record of transactions, all other nodes would cross-reference each other and easily pinpoint the node with the incorrect information. This system helps to establish an exact and transparent order of events. This way, no single node within the network can alter information held within it.

Because of this, the information and history (such as of transactions of a cryptocurrency) are irreversible. Such a record could be a list of transactions (such as with a cryptocurrency), but it also is possible for a blockchain to hold a variety of other information like legal contracts, state identifications, or a company's product inventory.

# Pros and Cons of Blockchain

For all of its complexity, blockchain's potential as a decentralized form of record-keeping is almost without limit. From greater user privacy and heightened security to lower processing fees and fewer errors, blockchain technology may very well see applications beyond those outlined above. But there are also some disadvantages.

## ✓ Pros

- Improved accuracy by removing human involvement in verification
- Cost reductions by eliminating third-party verification
- Decentralization makes it harder to tamper with
- Transactions are secure, private, and efficient
- Transparent technology
- Provides a banking alternative and a way to secure personal information for citizens of countries with unstable or underdeveloped governments

## ✗ Cons

- Significant technology cost associated with mining bitcoin
- Low transactions per second
- History of use in illicit activities, such as on the dark web
- Regulation varies by jurisdiction and remains uncertain
- Data storage limitations

# The problem with blockchains

Blockchain  
Using consensus  
smart contracts

Since the blockchain has its distributed ledger nature, each node in the network has to be able to find the same end result given the same input. Otherwise, when a node looks to validate a transaction another node makes, it would end up with a different result. This architecture is intentional, and it's designed to be deterministic intentionally.

In blockchain, the mechanism for agreeing upon a data value is called consensus, and determinism is important so that nodes can come to a consensus. You might have heard of some of them, like Proof of Work (PoW) with Nakamoto Consensus or Proof of Stake (PoS) with Byzantine Consensus. Consensus is one of the key ingredients that make blockchain work in the first place.

But we need the blockchain world to connect with the real world. We need to get the price of ETH and other cryptocurrencies into a contract so we can have DeFi. We need to get the weather data so we can have decentralized trustless insurance. We need data to use blockchain for one of its most important purposes, smart contracts. So how do we bridge the worlds with this constraint?



The oracle problem is these two pieces combined:

Blockchains alone can't access outside data.

Using centralized oracles nullifies the advantage of smart contracts — and are major security risks.

## How oracles solve this

A blockchain oracle is any device or entity that connects a deterministic blockchain with off-chain data. These oracles enter every data input through an external transaction. This way, we can be sure that the blockchain itself contains all of the information required to verify itself. This is why oracles are known as blockchain middleware: They are the bridge between the two worlds.

### **What is a decentralized oracle?**

A decentralized oracle or decentralized oracle network is a group of independent blockchain oracles that provide data to a blockchain. Every independent node or oracle in the decentralized oracle network independently retrieves data from an off-chain source and brings it on-chain. The data is then aggregated so the system can come to a deterministic value of truth for that data point. Decentralized oracles solve the oracle problem.

Chainlink is a framework for choosing your independent network of nodes to connect the real world's data to the blockchain to enable smart contracts to reach their true potential. With this, we are leveraging the same reliable decentralized infrastructure concept the blockchain has, but for blockchain oracles. If nodes/sources are hacked, depreciated, or deleted, the network of Chainlink will leverage the decentralized network and carry on.



Term	Definition
51% Attack	When more than 50% of the miners in a blockchain launch an attack on the rest of the nodes/users to attempt to steal assets or double spend.
Address	Much like a URL, a blockchain address is the location to or from which transactions occur on the blockchain.
Alt-coin	Any coin or token other than Bitcoin.
Attestation Ledger	A register or account book created for the purpose of providing support/evidence of individual transactions. Normally, an attestation ledger is used to verify that a transaction has been carried out, or to verify the authenticity of products or transactions.
Bitcoin	The first and most popular cyptocurrency based on DLT technology developed from a whitepaper written by Satoshi Nakamoto in 2008.
Block	A group of transactions entered into a blockchain; analogous to a page of a ledger or record book.
Blockchain	A mathematical structure for storing digital transactions or data in an immutable, distributed, decentralized digital ledger consisting of blocks that are linked via cryptographic signature that is nearly impossible to fake, hack or disrupt.
Blockchain (Private a.k.a. Permissioned)	A blockchain that resides on a private network of computers that is only accessible to those with permission.
Blockchain (Public a.k.a. Permissionless)	A blockchain that resides on a network of computers around the world that is accessible to everyone.
Byzantine Fault Tolerance (BFT)	A property of a distributed, decentralized system to resist complete failure even when some of the nodes fail or act maliciously.
Centralized	A system or process for which there is a singular (i.e., central) source of authority, control and/or truth.
Chain of Custody	The entire chain of documentation of ownership of a product during its lifecycle from raw materials to the final end user.
Chaincode	Another name for a smart contract.
Consensus Mechanism - Proof of Authority (PoA)	PoA is an alternative form to the PoS algorithm. Instead of staking cryptocurrency (wealth), in PoA you stake your identity. This means voluntarily disclosing who you are in exchange for the right to validate blocks. Any malicious actions you undertake as a validator will reflect back on your identity. PoA blockchains require a thorough form of KYC (Know Your Customer - a verification process that determines you actually are who you

Term	Definition
51% Attack	When more than 50% of the miners in a blockchain launch an attack on the rest of the nodes/users to attempt to steal assets or double spend.
Address	Much like a URL, a blockchain address is the location to or from which transactions occur on the blockchain.
Alt-coin	Any coin or token other than Bitcoin.
Attestation Ledger	A register or account book created for the purpose of providing support/evidence of individual transactions. Normally, an attestation ledger is used to verify that a transaction has been carried out, or to verify the authenticity of products or transactions.
Bitcoin	The first and most popular cyptocurrency based on DLT technology developed from a whitepaper written by Satoshi Nakamoto in 2008.
Block	A group of transactions entered into a blockchain; analogous to a page of a ledger or record book.
Blockchain	A mathematical structure for storing digital transactions or data in an immutable, distributed, decentralized digital ledger consisting of blocks that are linked via cryptographic signature that is nearly impossible to fake, hack or disrupt.
Blockchain (Private a.k.a. Permissioned)	A blockchain that resides on a private network of computers that is only accessible to those with permission.
Blockchain (Public a.k.a. Permissionless)	A blockchain that resides on a network of computers around the world that is accessible to everyone.
Byzantine Fault Tolerance (BFT)	A property of a distributed, decentralized system to resist complete failure even when some of the nodes fail or act maliciously.
Centralized	A system or process for which there is a singular (i.e., central) source of authority, control and/or truth.
Chain of Custody	The entire chain of documentation of ownership of a product during its lifecycle from raw materials to the final end user.
Chaincode	Another name for a smart contract.
Consensus Mechanism - Proof of Authority (PoA)	PoA is an alternative form to the PoS algorithm. Instead of staking cryptocurrency (wealth), in PoA you stake your identity. This means voluntarily disclosing who you are in exchange for the right to validate blocks. Any malicious actions you undertake as a validator will reflect back on your identity. PoA blockchains require a thorough form of KYC (Know Your Customer - a verification process that determines you actually are who you



Consensus Mechanism - Proof of Burn (PoB)	PoB allows the miners to “burn” or destroy cryptocurrency which grants them the right to add blocks in proportion to the coins destroyed. Essentially, miners burn coins/tokens to buy virtual mining rigs that give them the power to mine blocks. The more currency burned by the miner, the bigger the ensuing virtual mining rig. To burn, miners send currency to a verifiably un-spendable address. This process does not consume many resources, thus PoB is often called PoW without energy waste. Depending upon the implementation, miners are allowed to burn the native currency or the currency of an alternative chain, and in exchange, they receive a reward in the native currency of the blockchain.
Consensus Mechanism - Proof of Capacity (PoC)	PoC allows the mining devices in the network to use their available hard drive space to decide the mining rights, instead of using the mining device’s computing power (as in PoW) or the miner’s stake in the cryptocurrency (as in PoS).
Consensus Mechanism - Proof of Stake (PoS)	In PoS, miners put up (i.e., “stake”) some of the blockchain’s cryptocurrency (e.g., ether for the Ethereum blockchain) in order to increase their chances of being selected to validate a block. The stake is locked up as a deposit to ensure the miner validates the block according to the rules. If the miner violates the rules, the deposit will be “burned” or destroyed. PoS is less resource intensive than PoW since fewer miners are racing to solve the mathematical formula.
Consensus Mechanism - Proof of Work (PoW)	In PoW, transaction data (block) + a random strings of digits (nonce of block) are repeatedly applied to a (hashing) mathematical formula by miners, until a desirable outcome is found (the proof of work). Other miners then verify the proof of work by taking the alleged input string and applying it to the same formulae to see if the outcome is indeed that what was presented. If the results are the same, the transaction is verified and added to the blockchain. As many miners are racing to solve the formula which requires a great deal of computing power, PoW is resource intensive.
Consensus Mechanism (a.k.a. Consensus Protocol)	The process used to validate a transaction across a distributed blockchain network designed to achieve Byzantine Fault Tolerance.
Cryptocurrency	Digital money which uses encryption and consensus algorithms to regulate the generation of coins/tokens and transfer of funds. Cryptocurrencies are generally decentralized, operating independently of central authorities.
Cryptography	The science of securing communication using individualized codes so only the participating parties can read the messages.
DAO (Decentralized Autonomous Organization)	A governance structure without a central authority which rewards good behavior and penalizes bad behavior by a set of pre-defined rules which can only be changes by a vote, which



Genesis Block	The first or first few blocks of a blockchain
Governance	Establishment of policies and continuous monitoring of their proper implementation of an organization or system.
Hash Function	A function that receives an input of any size and returns a unique string of a uniform length.
Hyperledger Fabric	IBM's private (permissioned) blockchain toolset.
Identity	The information on an entity used by computer systems to uniquely represent a person, organization, application, or device.
Immutable/Immutability	The property of being unchangeable. Once a transaction has been added to a block and written to a blockchain, it cannot be changed and therefore is immutable.
Initial Coin Offering (ICO)	The first sale of a blockchain coin or token.
Interoperability	The ability of two or more systems to communicate and exchange data. Due to various design decisions (e.g., consensus protocol) most blockchains are not interoperable, however there are many projects that are working to connect various blockchains.
IPFS (Interplanetary File System)	A peer-to-peer hypermedia protocol for storing and sharing data in a distributed file system using content-addressing to uniquely identify each file in a global namespace connecting all computing devices.
Know Your Customer (KYC)	The legal process of a business identifying and verifying the identity of its clients. KYC requirements vary from jurisdiction to jurisdiction.
Liquidity	The ease of converting an asset (or, in this case, cryptocurrency) to cash (fiat).
Mainnet	The production version of a blockchain.
Merkle Tree/Hash Tree	In cryptography and computer science, a Merkle or hash tree is 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.
Mining	In a public blockchain, the process of verifying a transaction and writing it to the blockchain for which the successful miner is rewarded in the cryptocurrency of the blockchain.



	typically requires a stake, adding risk to the process to discourage bad actors, amongst the participants.
DApp	Software which does not rely on a central system or database but can share information amongst its users via a decentralized database, such as a blockchain
Decentralization/Decentralized	A system with no single point where the decision is made. Every node makes a decision for its own behavior and the resulting system behavior is the aggregate response.
Digital Identity (a.k.a. Self-Sovereign Identity)	The network or Internet equivalent to the real identity of a person or entity (like a business or government agency). Advocates of blockchain-based digital identity is to return ownership and control of personal information to the individuals. In any given transaction, personal information is not disclosed, but rather the information required by one party is verified by the digital identity application.
Digital Signature	A mathematical scheme for verifying digital messages or documents satisfy two requirements - they have authenticity (from a known sender) and integrity (were not altered in transit).
Digital Signature - Multi-signature	In order to increase security, mutisig addresses require more than one digital signature (and therefore multiple keys) to sign a transaction or message.
Digital Signature - Ring	A digital signature that can be performed by any of a group of people that each have keys. A property of a ring signature is that it is impossible to determine which of the group signed the transaction.
Distributed	As opposed to decentralized, a distributed system shares processing and/or data across multiple nodes, but the decisions may still be centralized and use complete system knowledge.
Distributed Ledger Technology (DLT)	The larger class of technology of which blockchain is a subset. A digital system for recording the transaction of assets in which the transactions and their details are recorded in multiple identical copies at the same time with no central data store or administration.
Double Spending	A unique problem to cryptocurrency where the same coins or tokens are spent or traded twice.
Ethereum	A public blockchain that supports smart contracts.
Fiat	Legal tender the value for which is backed by a government or governmental body (e.g., US dollars, Euros)
Fork	A collectively agreed upon software update by all nodes in a distributed network. Sometimes, the previous version continues in parallel with the new version.
Fungible	The property an item of being exchangeable with other like items. For example, USD and Euros are fungible. The value of USD can be expressed in Euros.
Gas	A fee charged to write a transaction to a public blockchain. The gas is used to reward the miner which validates the transaction.

# Overview

## Blockchain Consensus Mechanisms

Proof of work

Proof of stake

Bitcoin Whitepaper-Satoshi Nakamoto

[https://www.usssc.gov/sites/default/files/pdf/training/annual-national-training-seminar/2018/Emerging\\_Tech\\_Bitcoin\\_Crypto.pdf](https://www.usssc.gov/sites/default/files/pdf/training/annual-national-training-seminar/2018/Emerging_Tech_Bitcoin_Crypto.pdf)

Ethereum Whitepaper-Vitalik Buterin

[https://blockchainlab.com/pdf/Ethereum\\_white\\_paper-a\\_next\\_generation\\_smart\\_contract\\_and\\_decentralized\\_application\\_platform-vitalik-buterin.pdf](https://blockchainlab.com/pdf/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf)

Sharding and roll ups

DFT



The proof of work (PoW) is a common consensus algorithm used by the most popular cryptocurrency networks like bitcoin and litecoin. It requires a participant node to prove that the work done and submitted by them qualifies them to receive the right to add new transactions to the blockchain. However, this whole mining mechanism of bitcoin needs high energy consumption and a longer processing time.

### **Understanding Proof of Work**

This explanation will focus on proof of work as it functions in the bitcoin network. Bitcoin is a digital currency that is underpinned by a kind of distributed ledger known as a "blockchain." This ledger contains a record of all bitcoin transactions, arranged in sequential "blocks," so that no user is allowed to spend any of their holdings twice. In order to prevent tampering, the ledger is public, or "distributed"; an altered version would quickly be rejected by other users.

The way that users detect tampering in practice is through hashes, long strings of numbers that serve as proof of work. Put a given set of data through a hash function (bitcoin uses SHA-256), and it will only ever generate one hash. Due to the "avalanche effect," however, even a tiny change to any portion of the original data will result in a totally unrecognizable hash. Whatever the size of the original data set, the hash generated by a given function will be the same length. The hash is a one-way function: it cannot be used to obtain the original data, only to check that the data that generated the hash matches the original data.

The proof of stake (PoS) is another common consensus algorithm that evolved as a low-cost, low-energy consuming alternative to the PoW algorithm. It involves the allocation of responsibility in maintaining the public ledger to a participant node in proportion to the number of virtual currency tokens held by it. However, this comes with the drawback that it incentivizes cryptocurrency hoarding instead of spending.

Proof-of-stake reduces the amount of computational work needed to verify blocks and transactions that keep the blockchain, and thus a cryptocurrency, secure. Proof-of-stake changes the way blocks are verified using the machines of coin owners. The owners offer their coins as collateral for the chance to validate blocks. Coin owners with staked coins become "validators."

Validators are then selected randomly to "mine," or validate the block. This system randomizes who gets to "mine" rather than using a competition-based mechanism like proof-of-work.

To become a validator, a coin owner must "stake" a specific amount of coins. For instance, Ethereum will require 32 ETH to be staked before a user can become a validator.

Blocks are validated by more than one validator, and when a specific number of the validators verify that the block is accurate, it is finalized and closed.

Different proof-of-stake mechanisms may use different methods for validating blocks—when Ethereum transitions to PoS, it will use shards for transaction submissions. A validator will verify the transactions and add them to a shard block, which requires at least 128 validators to attest to.

Once shards are validated and block created, two-thirds of the validators must agree that the transaction is valid, then the block is closed.



## How Is Proof-of-Stake Different From Proof-of-Work?

Both consensus mechanisms help blockchains synchronize data, validate information, and process transactions. Each method has proven to be successful at maintaining a blockchain, although there are pros and cons to each. However, the two algorithms have very differing approaches.

Under PoS, block creators are called validators. A validator checks transactions, verifies activity, votes on outcomes, and maintains records. Under PoW, the creators are called miners. Miners solve complex mathematical problems to verify transactions; in return,

To "buy into" the position of becoming a block creator, investors need only to purchase the sufficient limit of coins or tokens required to become a validator for a PoS blockchain. For PoW, miners must invest in processing equipment and incur heavy energy charges to power the machines attempting to solve the computations.

The equipment and energy cost under PoW mechanisms are expensive, limiting access to mining and strengthening the security of the blockchain. However, PoS blockchains often allow for more scalability due to their energy efficiency.

# History of Solidity

- Vitalik Buterin, an early advocate of Bitcoin, first proposed the creation of another protocol to run more elaborate smart contracts than that of the Bitcoin network. Bitcoin's structure is simple, and for good reason – it was created primarily to be sound money.
- However, to do more complex transactions that do not require the presence of an intermediary to validate them, Buterin suggested the idea of Ethereum, a complex system of smart contracts interacting with each other.
- With Ethereum, the idea of a transaction was no longer limited to money per se, but a host of other types of arrangements that could be executed using a highly malleable and programmable protocol.
- In other words, Ethereum expanded the possibilities and brought creativity and a certain level of human-type problem-solving into the mix. As a result, this new blockchain protocol needed to be Turing complete (or as close to Turing completeness as possible) to achieve its new objectives.



# The Ethereum Virtual Machine or EVM

The Ethereum Virtual Machine (or) EVM is a virtual stack that is embedded within every fully participating node in the network, or Ethereum node, that executes contract bytecode. The EVM is a Turing complete system, which means it can perform any type of logical step associated with computational functions.

While Bitcoin provides rewards for running a transaction, Ethereum charges fees for executing software instructions. The gas mechanism in Ethereum lets users pre-pay for the instructions they want to execute on the EVM using Ether, its native currency.

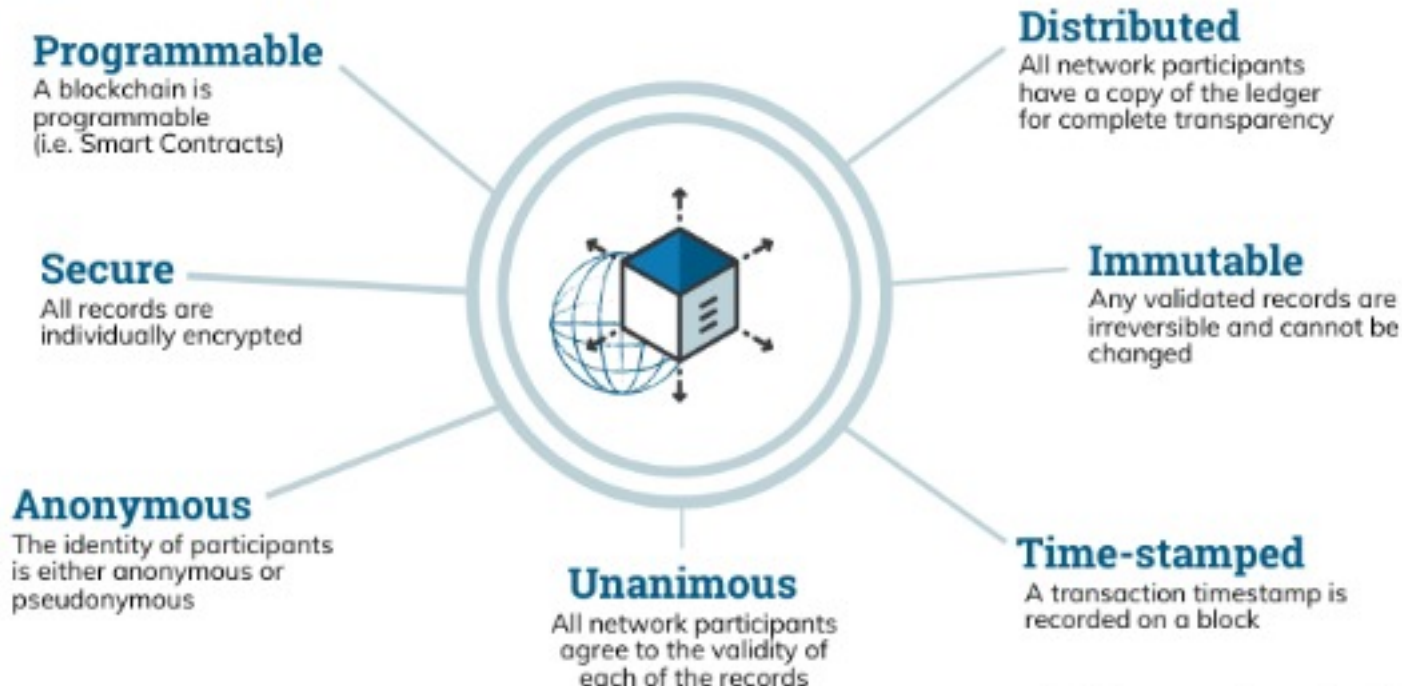
EVMs are pretty versatile in that they can be implemented using JavaScript, C++, Ruby, Python, and a variety of other languages.

The contracts that are run using the EVM are written using Solidity. So, once and for all, what is Solidity? It is a high-level programming language that is compatible with how humans express instructions – using numbers and letters instead of binary code.

Solidity smart contracts are instructions that are then compiled to the EVM's bytecode. The nodes in the Ethereum network, as mentioned, run EVM instances that permit them to agree on the execution of the same set of instructions.

# DLT

## The Properties of Distributed Ledger Technology (DLT)





# About

Blockchain is one type of a distributed ledger. Distributed ledgers use independent computers (referred to as nodes) to record, share and synchronize transactions in their respective electronic ledgers (instead of keeping data centralized as in a traditional ledger).

- This could address persistent challenges in the financial sector and change roles of financial sector stakeholders. DLT has the potential to transform various other sectors as well, like manufacturing, government financial management systems and clean energy.
- Since this technology is still nascent, the World Bank Group doesn't have general recommendations about its use for international development. We are in dialogue with standard-setting bodies, governments, central banks and other stakeholders to monitor, research and pilot applications based on blockchain and DLT.
- However, waiting for "perfect" DLT solutions could mean missing an opportunity to help shape it. To understand how DLT can address challenges in the financial sector requires both research and real-life applications and pilots.
- It also requires resolving consumer protection issues, financial integrity concerns, speed of transactions, environmental footprint, legal, regulatory and technological issues that arise with the advent of new technology.
- DLT applications will likely be incremental, and will likely first replace processes and activities that are still manual and inefficient. (Such as reference data maintenance in payment and settlement systems, trade finance, syndicated loans, and tracking provenance of agricultural products and commodities, their subsequent sale or use as financing collateral.)
- Eventually, DLT could increase efficiency and lower remittance costs, and potentially improve access to finance for unbanked populations, who are currently outside the traditional financial system.
- Messages based on WBG's fintech note on Distributed Ledger Technology and Blockchain, published December 2017.



# CODING

## Remix IDE Link

[https://remix.ethereum.org/  
#optimize=false&runs=200&evmVersion=null  
&version=soljson-v0.8.7+commit.e28d00a7.js](https://remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.7+commit.e28d00a7.js)

## Github Link

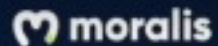
[https://github.com/gunjandadhich/  
BlockchainIOI/settings/access?guidance\\_task=](https://github.com/gunjandadhich/BlockchainIOI/settings/access?guidance_task=)

transaction

0x52bf1c651efd691140f6e4c979196b95d5adae77c7e6b755abd4aa82edbf25do

Metamask

0x3388FE6647ebe54620DA66a5deEoAaDEoe3cEc33



## Browser

\_AddressSyncStatus  
\_EthAddress  
\_Role  
\_Session  
\_User



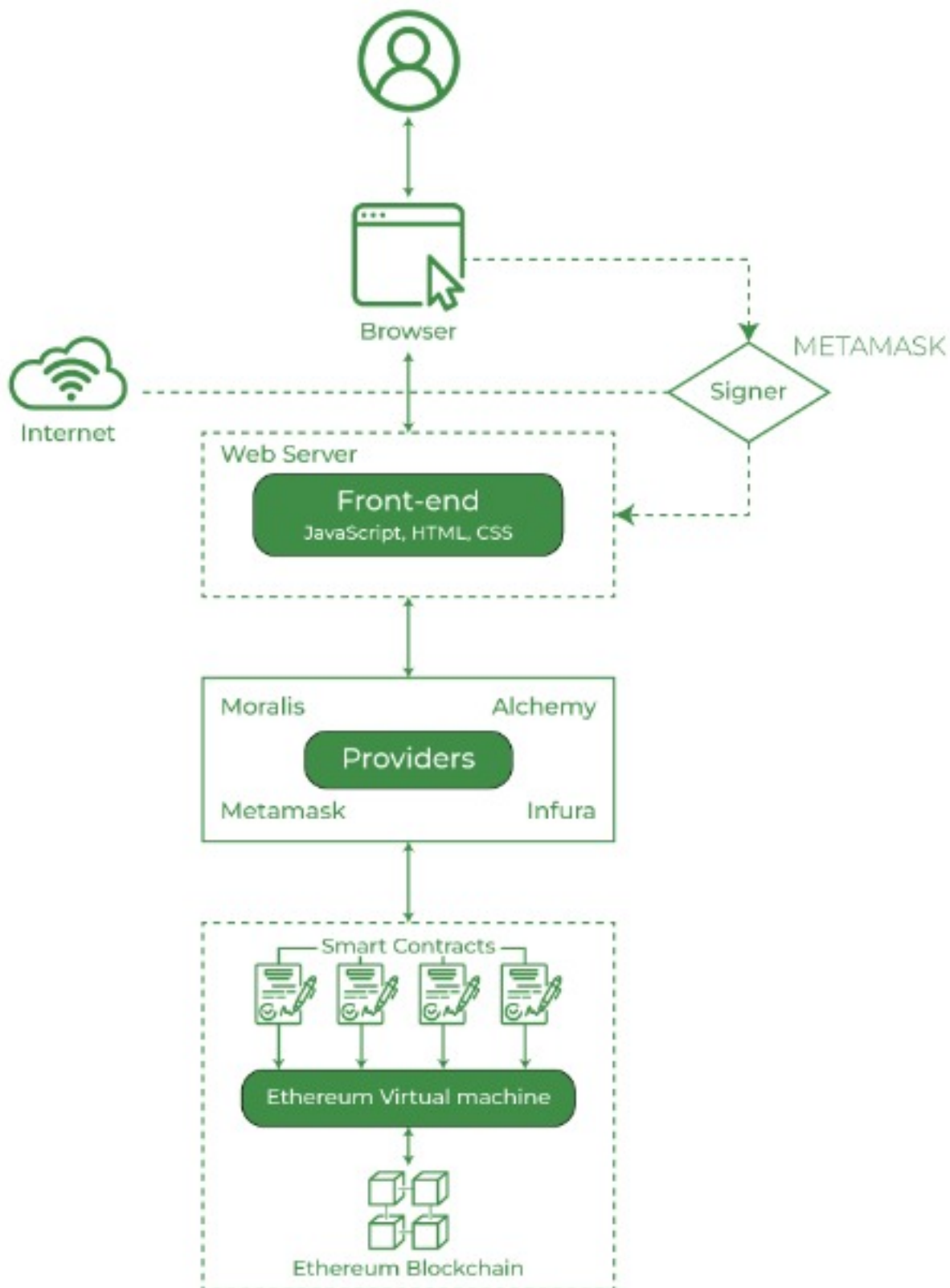
CLASS  
\_User 2 objects - Public Read and Write enabled

+ Add Row + Manage Columns ↻ Refresh ⊖ Filter 🔒 Security 📄 Edit

id	ethAddress	createdAt	password	email	bio	banner	pfp	
3	0x164836735e8a...	28 Apr 2022 at...	(hidden)	(undefined)	I like golf an...	https://ipfs.m...	https://ipfs.m...	
3	(undefined)	19 Apr 2022 at...	(hidden)	(undefined)	(undefined)	(undefined)	(undefined)	

Add a new column





# Introduction

Solidity Contracts are like a class in any other object-oriented programming language. They firmly contain data as state variables and functions which can modify these variables. When a function is called on a different instance (contract), the EVM function call happens and the context is switched in such a way that the state variables are inaccessible. A contract or its function needs to be called for anything to happen. Some basic properties of contracts are as follows :

- Constructor: Special method created using the constructor keyword, which is invoked only once when the contract is created.
- State Variables: These are the variables that are used to store the state of the contract.
- Functions: Functions are used to manipulate the state of the contracts by modifying the state variables.

**A smart contract's ABI** specifies its interface and the set of functions accessed from outside the smart contract. The ABI is used only for defining the events of the contract and function signatures, such as names of the function, return types, and argument types. However, it does not define their implementation.



# Visibility Modifiers

Solidity provides four types of visibilities for functions and state variables. Functions have to be specified by any of the four visibilities but for state variables `external` is not allowed.

**External:** External functions can be called by other contracts via transactions. An external function cannot be called internally. For calling an external function within the contract `this.function_name()` method is used. Sometimes external functions are more efficient when they have large arrays of data.

**Public:** Public functions or variables can be called both externally or internally via messages. For public static variables, a getter method is created automatically in solidity.

**Internal:** These functions or variables can be accessed only internally i.e. within the contract or the derived contracts.

**Private:** These functions or variables can only be visible for the contracts in which they are defined. They are not accessible to derived contracts also.

```
// Solidity program to demonstrate
// visibility modifiers
pragma solidity ^0.5.0;

// Creating a contract
contract contract_example {

// Declaring private
// state variable
uint private num1;

// Declaring public
// state variable
uint public num2;

// Declaring Internal
// state variable
string internal str;

// Defining a constructor
constructor() public {
    num2 = 10;
}
```



# Constructors

A constructor is a special method in any object-oriented programming language which gets called whenever an object of a class is initialized. It is totally different in case of Solidity, Solidity provides a constructor declaration inside the smart contract and it invokes only once when the contract is deployed and is used to initialize the contract state. A default constructor is created by the compiler if there is no explicitly defined constructor.

## Creating a constructor

A Constructor is defined using a constructor keyword without any function name followed by an access modifier. It's an optional function which initializes state variables of the contract. A constructor can be either internal or public, an internal constructor marks contract as abstract.

## Syntax:

```
constructor() <Access Modifier> {  
}
```

```
// Solidity program to demonstrate
// creating a constructor
pragma solidity ^0.5.0;

// Creating a contract
contract constructorExample {

    // Declaring state variable
    string str;

    // Creating a constructor
    // to set value of 'str'
    constructor() public {
        str = "GeeksForGeeks";
    }

    // Defining function to
    // return the value of 'str'
    function getValue(
    ) public view returns (
    string memory) {
        return str;
    }
}
```



# Variables

## Declaration of Variables

In Solidity declaration of variables is a little bit different, to declare a variable the user has to specify the data type first followed by access modifier.

### Syntax:

<type> <access modifier> <variable name> ;

## Types of Variables

Solidity is a statically typed language i.e. each declared variable always has a default value based on its data type, which means there is no concept of 'null' or 'undefined'. Solidity supports three types of variables:

- 1. State Variables:** Values of these variables are permanently stored in the contract storage. Each function has its own scope, and state variables should always be defined outside of that scope.
- 2. Local Variable:** Values of these variables are present till the function executes and it cannot be accessed outside that function. This type of variable is usually used to store temporary values.
- 3. Global Variables:** These are some special variables that can be used globally and give information about the transactions and blockChain properties. Some of the global variables are listed below :

# Array

Arrays are data structures that store the fixed collection of elements of the same data types in which each and every element has a specific location called index. Instead of creating numerous individual variables of the same type, we just declare one array of the required size and store the elements in the array and can be accessed using the index. In Solidity, an array can be of fixed size or dynamic size. Arrays have a continuous memory location, where the lowest index corresponds to the first element while the highest represents the last

## Creating an Array

To declare an array in Solidity, the data type of the elements and the number of elements should be specified. The size of the array must be a positive integer and data type should be a valid Solidity type

## Syntax:

```
<data type> <array name>[size] = <initialization>
```

## Types of Array

**Fixed-size Arrays** - The size of the array should be predefined. The total number of elements should not exceed the size of the array. If the size of the array is not specified then the array of enough size is created which is enough to hold the initialization.



```
// Solidity program to demonstrate
// creating a fixed-size array
pragma solidity ^0.5.0;

// Creating a contract
contract Types {

    // Declaring state variables
    // of type array
    uint[6] data1;

    // Defining function to add
    // values to an array
    function array_example() public returns (
    int[5] memory, uint[6] memory){

        int[5] memory data
        = [int(50), -63, 77, -28, 90];
        data1
        = [uint(10), 20, 30, 40, 50, 60];

        return (data, data1);
    }
}
```

```
// Solidity program to demonstrate
// creating a dynamic array
pragma solidity ^0.5.0;

// Creating a contract
contract Types {

    // Declaring state variable
    // of type array. One is fixed-size
    // and the other is dynamic array
    uint[] data
    = [10, 20, 30, 40, 50];
    int[] data1;

    // Defining function to
    // assign values to dynamic array
    function dynamic_array() public returns(
    uint[] memory, int[] memory){

        data1
        = [int(-60), 70, -80, 90, -100, -120, 140];
        return (data, data1);
    }
}
```

1. Accessing Array Elements: The elements of the array are accessed by using the index. If you want to access  $i$ th element then you have to access  $(i-1)$ th index.

2. Length of Array: Length of the array is used to check the number of elements present in an array. The size of the memory array is fixed when they are declared, while in case the dynamic array is defined at runtime so for manipulation length is required.

3. Push: Push is used when a new element is to be added in a dynamic array. The new element is always added at the last position of the array.

4. Pop: Pop is used when the last element of the array is to be removed in any dynamic array.



# Enums & Struct

**Enums** are the way of creating user-defined data types, it is usually used to provide names for integral constants which makes the contract better for maintenance and reading. Enums restrict the variable with one of a few predefined values, these values of the enumerated list are called enums. Options are represented with integer values starting from zero, a default value can also be given for the enum. By using enums it is possible to reduce the bugs in the code.

Syntax:

```
enum <enumerator_name> {  
    element 1, element 2, ..., element n  
}
```

**Restriction** The explicit conversions check the value ranges at runtime and a failure causes an exception. Enums needs at least one member.

## Struct

Structs in Solidity allows you to create more complicated data types that have multiple properties. You can define your own type by creating a struct.

They are useful for grouping together related data.

Structs can be declared outside of a contract and imported in another contract. Generally, it is used to represent a record. To define a structure struct keyword is used, which creates a new data type.

# web3.eth

The web3-eth package allows you to interact with an Ethereum blockchain and Ethereum smart contracts.

## Note on checksum addresses

All Ethereum addresses returned by functions of this package are returned as checksum addresses. This means some letters are uppercase and some are lowercase. Based on that it will calculate a checksum for the address and prove its correctness. Incorrect checksum addresses will throw an error when passed into functions. If you want to circumvent the checksum check you can make an address all lower- or uppercase.

## Example

```
web3.eth.getAccounts(console.log);
```

```
>
```

```
["0x11f4d0A3c12e86B4b5F39B213F7E19D048276DAe", "0x85F43D8a49eeB85d32Cf465507DD71d507100Cid"]
```

```
var Web3 = require('web3');
var web3 = new Web3('http://localhost:8545');
// or
var web3 = new Web3(new
Web3.providers.HttpProvider('http://localhost:8545'));

// change provider
web3.setProvider('ws://localhost:8546');
// or
web3.setProvider(new
Web3.providers.WebsocketProvider('ws://
localhost:8546'));

// Using the IPC provider in node.js
var net = require('net');
var web3 = new Web3('/Users/myuser/Library/Ethereum/
geth.ipc', net); // mac os path
// or
var web3 = new Web3(new
Web3.providers.IpcProvider('/Users/myuser/Library/
Ethereum/geth.ipc', net)); // mac os path
// on windows the path is: "\\.\pipe\geth.ipc"
// on linux the path is: "/users/myuser/.ethereum/
geth.ipc"
```



# WEB2 VS WEB3

Many Web3 developers have chosen to build dapps because of Ethereum's inherent decentralization:

Anyone who is on the network has permission to use the service – or in other words, permission isn't required.

No one can block you or deny you access to the service.

Payments are built in via the native token, ether (ETH).

Ethereum is turing-complete, meaning you can program pretty much anything

Web3 has some limitations right now:

Scalability – transactions are slower on web3 because they're decentralized. Changes to state, like a payment, need to be processed by a node and propagated throughout the network.

UX – interacting with web3 applications can require extra steps, software, and education. This can be a hurdle to adoption.

Accessibility – the lack of integration in modern web browsers makes web3 less accessible to most users.

Cost – most successful dapps put very small portions of their code on the blockchain as it's expensive.

## PRACTICAL COMPARISONS

Web2	Web3
Twitter can censor any account or tweet	Web3 tweets would be uncensorable because control is decentralized
Payment service may decide to not allow payments for certain types of work	Web3 payment apps require no personal data and can't prevent payments
Servers for gig-economy apps could go down and affect worker income	Web3 servers can't go down – they use Ethereum, a decentralized network of 1000s of computers as their backend

# Usecase

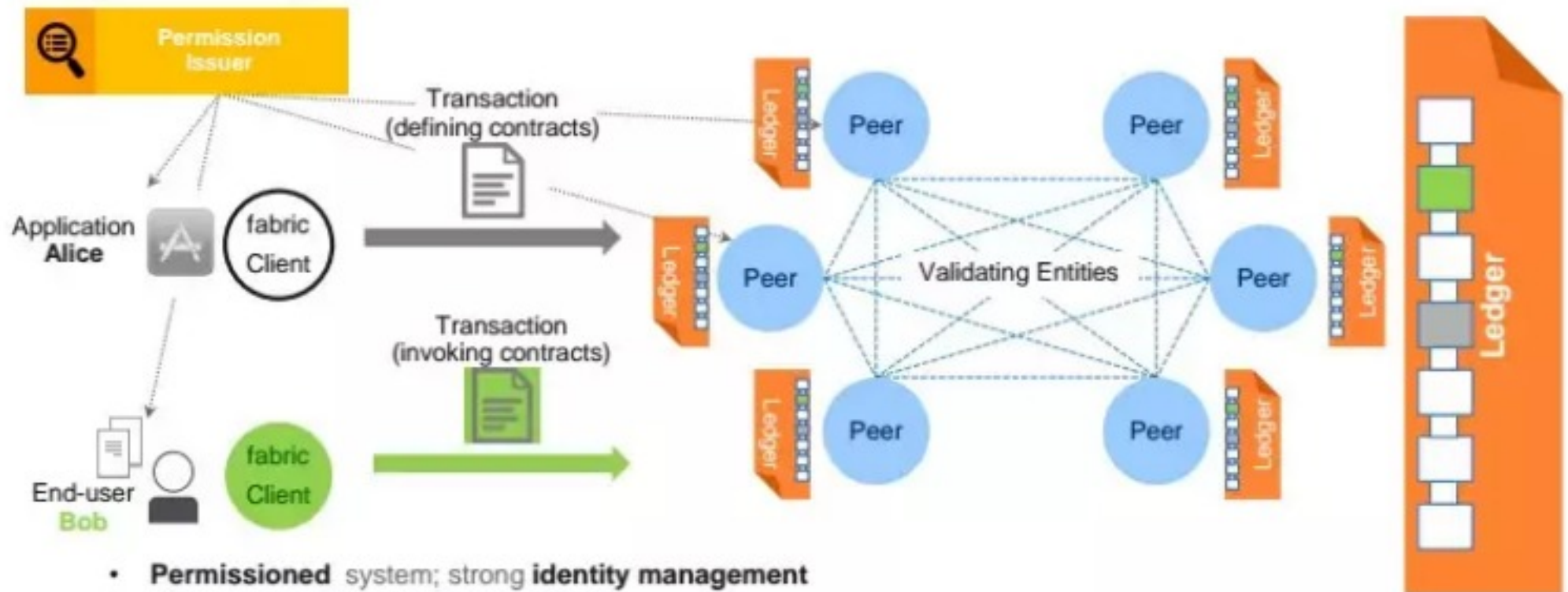
Smart Contract

Dapp

Hyperledger fabric

# Framework

## Hyperledger-fabric model



- **Permissioned** system; strong **identity management**
- Distinct roles of **users**, and **validators**
- Users **deploy** new pieces of code (chaincodes) and **invoke** them through **deploy & invoke** transactions
- Validators evaluate the effect of a transaction and reach consensus over the new version of the **ledger**
- **Ledger** = total order of transactions + hash (global state)
- **Pluggable consensus** protocol, currently PBFT & Sieve



In a public blockchain, anyone is free to join and participate in the core activities of the blockchain network.

A private blockchain allows only selected and verified participants; the operator has the rights to override, edit, or delete entries on the blockchain.

A permissioned blockchain has properties of both private and public blockchains.

Permissioned blockchains have seen an increase in popularity thanks to their ability to allocate specific permissions to various users on the network.

# Public Blockchain

A public blockchain is one where anyone is free to join and participate in the core activities of the blockchain network. Anyone can read, write, and audit the ongoing activities on a public blockchain network, which helps achieve the self-governed, decentralized nature often touted when blockchain is discussed.

## Advantages

A public network operates on an incentivizing scheme that encourages new participants to join and keep the network agile. Public blockchains offer a particularly valuable solution from the point of view of a truly decentralized, democratized, and authority-free operation.

## Disadvantages

The primary disadvantage to secured public blockchains is the heavy energy consumption required to maintain them. The concern is a consensus mechanism that requires participants to compete to validate the information and receive a reward for letting the network use their processing power. Not all blockchain networks use an energy-intensive validation process, so not all use enormous amounts of electricity.

# Private Blockchain

Participants can join a private blockchain network only through an invitation where their identity or other required information is authentic and verified. The validation is done by the network operator(s) or by a clearly defined set protocol implemented by the network through smart contracts or other automated approval methods.

Private blockchains control who is allowed to participate in the network. If the network is capable of mining, its private nature could control which users can execute the consensus protocol that decides the mining rights and rewards. Additionally, only select users might maintain the shared ledger. The owner or operator has the right to override, edit, or delete the necessary entries on the blockchain as required or as they see fit.

## **Advantages**

A private blockchain is not decentralized. It is a distributed ledger that operates as a closed database secured with cryptographic concepts and the organization's needs. Only those with permission can run a full node, make transactions, or validate/authenticate the blockchain changes.

By reducing the focus on protecting user identities and promoting transparency, private blockchains prioritize efficiency and immutability—the state of not being able to be changed.

These are important features in supply, logistics, payroll, finances, accounting, and many other enterprise and business areas.

## **Disadvantages**

While purposefully designed for enterprise applications, private blockchains lose out on many of the valuable attributes of permissionless systems simply because they are not widely applicable. They are instead built to accomplish specific tasks and functions.

In this respect, private blockchains are susceptible to data breaches and other security threats. This is because there is generally a limited number of validators used to reach a consensus about transactions and data if there is a consensus mechanism.

In a private blockchain, there may not be consensus but only the immutability of entered data unless an operator or administrator can make changes.



# Decentralized application

Pro

A decentralized app (also known as a dApp or dapp) operates on a blockchain or peer-to-peer network of computers. It enables users to engage in transactions directly with one another as opposed to relying on a central authority.

Dapps have their backend code (smart contracts) running on a decentralized network and not a centralized server. They use the blockchain for data storage and smart contracts for their app logic.

A smart contract is like a set of rules that live on-chain for all to see and run exactly according to those rules. Imagine a vending machine: if you supply it with enough funds and the right selection, you'll get the item you want. And like vending machines, smart contracts can hold funds much like your blockchain account. This allows code to mediate agreements and transactions.

Once dapps are deployed on the Blockchain network you can't change them. Dapps can be decentralized because they are controlled by the logic written into the contract, not an individual or a company.

A dapp can have frontend code and user interfaces written in any language (just like an app) to make calls to its backend. Furthermore, its frontend can get hosted on decentralized storage such as IPFS.

Decentralized - dapps operate on Ethereum, an open public decentralized platform where no one person or group has control

Deterministic - dapps perform the same function irrespective of the environment in which they get executed

Turing complete - dapps can perform any action given the required resources

Isolated - dapps are executed in a virtual environment known as Ethereum Virtual Machine so that if the smart contract has a bug, it won't hamper the normal functioning of the blockchain network

On smart contracts



**The InterPlanetary File System (IPFS) is a protocol and peer-to-peer network for storing and sharing data in a distributed file system**

A dapp can have frontend code and user interfaces written in any language (just like an app) to make calls to its backend. Furthermore, its frontend can get hosted on decentralized storage such as IPFS.

Here's what happens when you add a file to IPFS — whether you're storing that file on your own local node or one operated by a pinning service or IPFS-enabled app.

- 1) When you add a file to IPFS, your file is split into smaller chunks, cryptographically hashed, and given a unique fingerprint called a content identifier (CID). This CID acts as a permanent record of your file as it exists at that point in time.
- 2) When other nodes look up your file, they ask their peer nodes who's storing the content referenced by the file's CID. When they view or download your file, they cache a copy — and become another provider of your content until their cache is cleared.
- 3) A node can pin content in order to keep (and provide) it forever, or discard content it hasn't used in a while to save space. This means each node in the network stores only content it is interested in, plus some indexing information that helps figure out which node is storing what.
- 4) If you add a new version of your file to IPFS, its cryptographic hash is different, and so it gets a new CID. This means files stored on IPFS are resistant to tampering and censorship — any changes to a file don't overwrite the original, and common chunks across files can be reused in order to minimize storage costs.
- 5) However, this doesn't mean you need to remember a long string of CIDs — IPFS can find the latest version of your file using the IPNS decentralized naming system, and DNSLink can be used to map CIDs to human-readable DNS names.



**Zero downtime** – Once the smart contract is deployed on the blockchain, the network as a whole will always be able to serve clients looking to interact with the contract. Malicious actors, therefore, cannot launch denial-of-service attacks targeted towards individual dapps.

**Privacy** – You don't need to provide real-world identity to deploy or interact with a dapp.

**Resistance to censorship** – No single entity on the network can block users from submitting transactions, deploying dapps, or reading data from the blockchain.

**Complete data integrity** – Data stored on the blockchain is immutable and indisputable, thanks to cryptographic primitives. Malicious actors cannot forge transactions or other data that has already been made public.

**Trustless computation/verifiable behavior** – Smart contracts can be analyzed and are guaranteed to execute in predictable ways, without the need to trust a central authority. This is not true in traditional models; for example, when we use online banking systems, we must trust that financial institutions will not misuse our financial data, tamper with records, or get hacked.

**Maintenance** – Dapps can be harder to maintain because the code and data published to the blockchain are harder to modify.

**Performance overhead** – There is a huge performance overhead, and scaling is really hard. To achieve the level of security, integrity, transparency, and reliability that Ethereum aspires to, every node runs and stores every transaction.

**Network congestion** – When one dapp uses too many computational resources, the entire network gets backed up. Currently, the network can only process about 10-15 transactions per second.

**User experience** – It may be harder to engineer user-friendly experiences because the average end-user might find it too difficult to set up a tool stack necessary to interact with the blockchain in a truly secure fashion.

**Centralization** – User-friendly and developer-friendly solutions built on top of the base layer of Ethereum might end up looking like centralized services anyways.