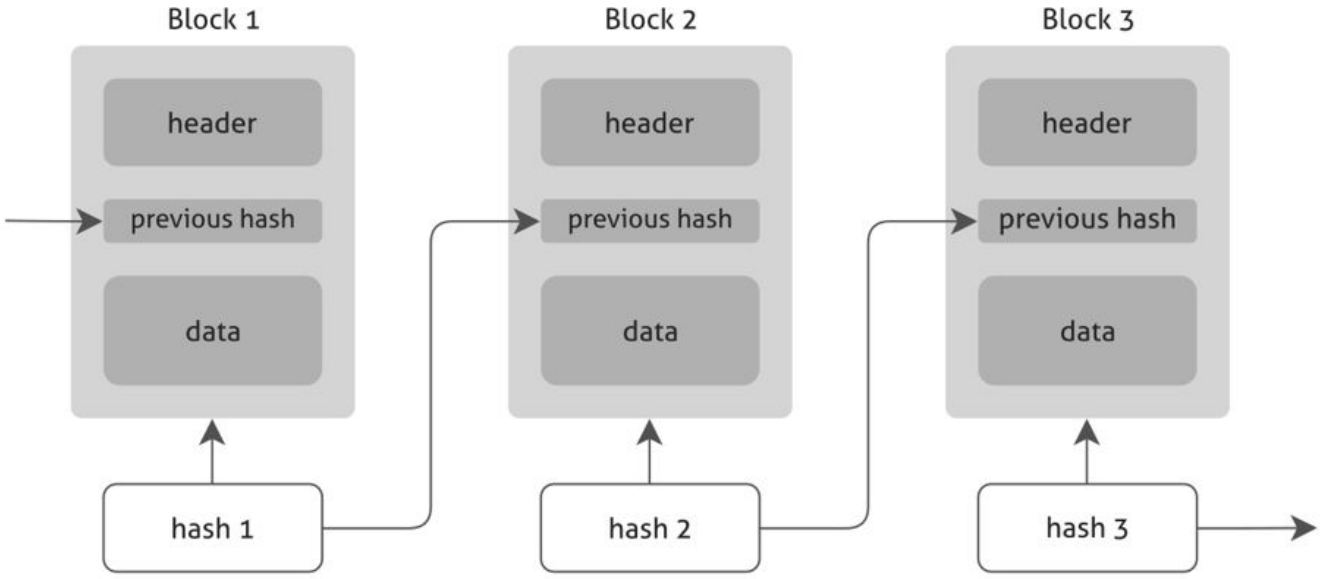
Ethereum Smart Contracts

- State of the art -

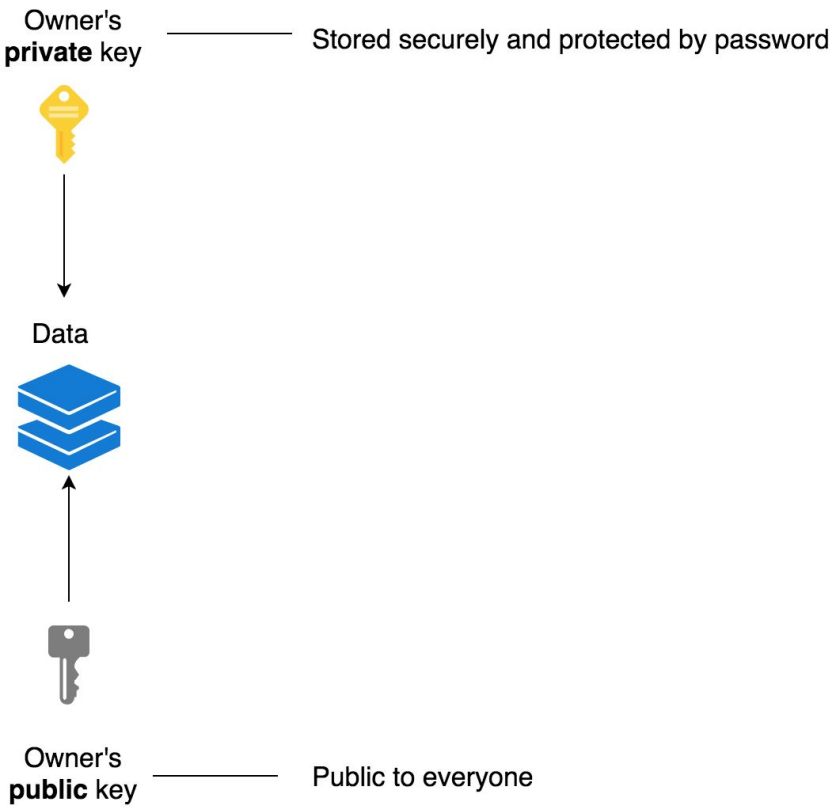
1. Introduction

1.1. Blockchain

There are many concepts involving the blockchain technology. As the name implies, blockchain are blocks of data chained together using their hash data, signed using a private key - user's key. These blocks are generally limited to the the size of 1MB and store as many transactions as possible in this limit.



Blocks store transactions Using asymmetric cryptography the system ensures that everybody can access our public-key while the private key remains secret. In order to ensure authenticity whenever we want to sign data, we use the private-key to encode it, as everyone can decode using our public-key and securely certify it, but nobody can change the data and sign it again, because it will be rejected.



The mechanism that links the blocks ensures the the Blockchain remains inviolable. When a new block will be generated a hash is generated from the current block's data, including its signature. Then, the new block is generated and the hash from the previous block is combined with the data from the current block and a new hash is generated and signed with user's private-key and so on.

If someone tries to change the data into a block to forge a transaction, he must generate all subsequent blocks so fast before the entire network starts to invalidate all these regenerated blocks.

1.2. Smart Contracts

Smart contract is just a phrase used to describe computer code that can facilitate the exchange of money, content, property, shares, or anything of value. When running on the blockchain a smart contract becomes like a self-operating computer program that

automatically executes when specific conditions are met. Because smart contracts run on the blockchain, they run exactly as programmed without any possibility of censorship, downtime, fraud or third party interference.

Smart contracts help you exchange money, property, shares, or anything of value in a transparent, conflict-free way while avoiding the services of a middleman.
The best way to describe smart contracts is to compare the technology to a vending machine. Ordinarily, you would go to a lawyer or a notary, pay them, and wait while you get the document. With smart contracts, you simply drop a bitcoin into the vending machine (i.e. ledger), and your escrow, driver's license, or whatever drops into your account. More so, smart contracts not only define the rules and penalties around an agreement in the same way that a traditional contract does, but also automatically enforce those obligations.

Smart contracts can:

- Function as 'multi-signature' accounts, so that funds are spent only when a required percentage of people agree
- Manage agreements between users, say, if one buys insurance from the other
- Provide utility to other contracts (similar to how a software library works)
- Store information about an application, such as domain registration information or membership records.

Smart Contracts vs. Traditional Contracts

Traditional Contracts

a) Vast amount of printed documents
b) Heavily rely on third parties for putting it in motion and enforcement (i.e. banks or national authorities)

c) In case of lack of enforcement, the need to turn to the judicial system

Smart Contracts

a) Entirely digital

b) Self-executing

c) The code itself defines obligations of the Parties

2. Ethereum

Ethereum is a **decentralized platform that runs smart contracts**: applications that run exactly as programmed without any possibility of downtime, censorship, fraud or third-party interference.

These apps run on a custom built blockchain, an enormously powerful shared global infrastructure that can move value around and represent the ownership of property.
This enables developers to create markets, store registries of debts or promises, move funds in accordance with instructions given long in the past (like a will or a futures contract) and many other things that have not been invented yet, all without a middleman or counterparty risk.

The project was bootstrapped via an ether presale in August 2014 by fans all around the world. It is developed by the Ethereum Foundation, a Swiss non-profit, with contributions from great minds across the globe.

2.1 Bitcoin vs. Ethereum

Like Bitcoin, Ethereum is a distributed public blockchain network. Although there are some significant technical differences between the two, the most important distinction to note is that Bitcoin and Ethereum differ substantially in purpose and capability. Bitcoin offers one particular application of blockchain technology, a peer to peer electronic cash system that enables online Bitcoin payments. While the Bitcoin

blockchain is used to track ownership of digital currency (bitcoins), the Ethereum blockchain focuses on running the programming code of any decentralized application.

By contrast, ethereum replaces bitcoin's more restrictive language (a scripting language of a hundred or so scripts) and replaces it with a language that allows developers to write their own programs.

Ethereum allows developers to program their own smart contracts, or 'autonomous agents', as the ethereum white paper calls them. The language is 'Turing-complete', meaning it supports a broader set of computational instructions.

In the Ethereum blockchain, instead of mining for bitcoin, miners work to earn Ether, a type of crypto token that fuels the network. Beyond a tradeable cryptocurrency, Ether is also used by application developers to pay for transaction fees and services on the Ethereum network.
There is a second type of token that is used to pay miners fees for including transactions in their block, it is called gas, and every smart contract execution requires a certain amount of gas to be sent along with it to entice miners to put it in the blockchain.

2.2 Ethereum Blockchain

Ethereum blockchain has a very similar structure to Bitcoin’s. It is a shared record of the entire transaction history. Every node on the network stores a copy of this history.
The big difference with ethereum is that its nodes store the most recent state of each smart contract, in addition to all of the ether transactions.

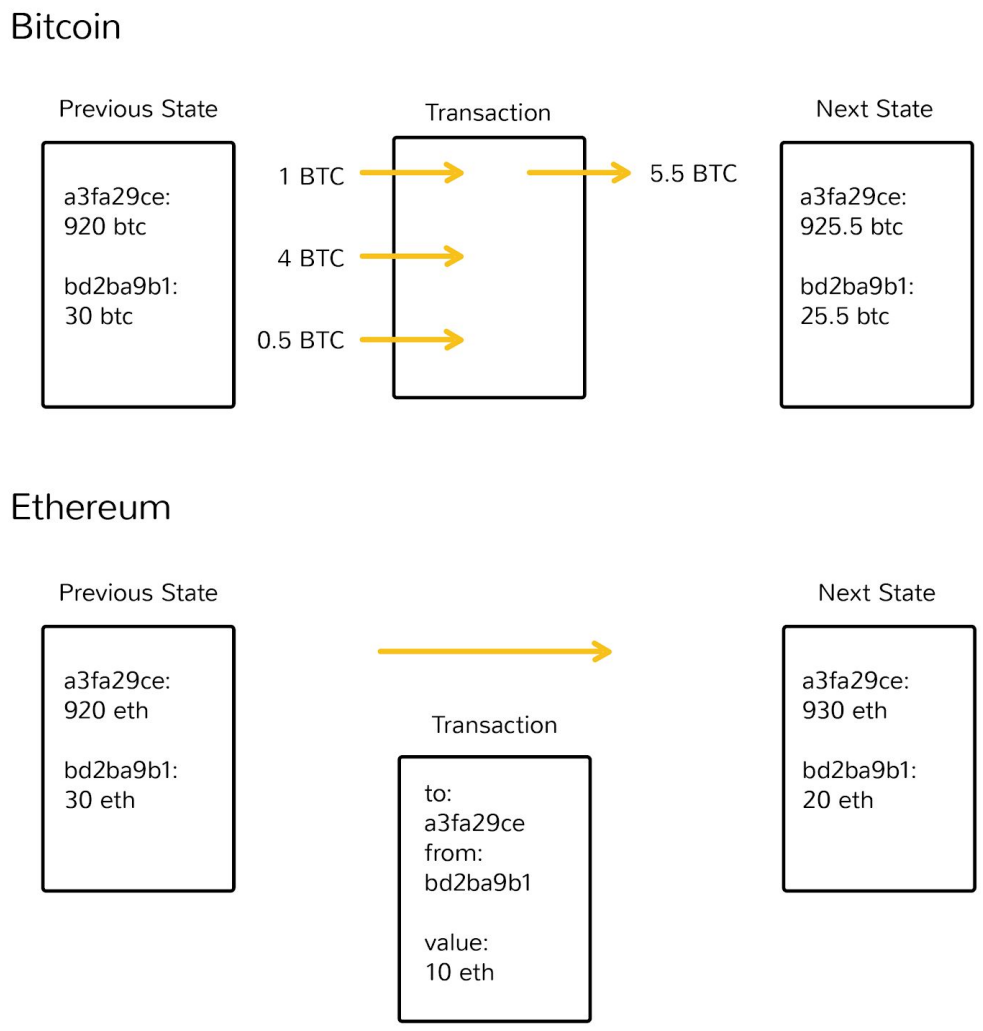
For each Ethereum application, the network needs to keep track of the “state”, or the current information of all of these applications, including each user's balance, all the smart contract code and where it's all stored.

Bitcoin uses unspent transaction outputs to track who has how much bitcoin. Every time a bitcoin transaction is made, the network 'breaks' the total amount as if it was paper

money, issuing back bitcoins in a way that makes the data behave similarly to physical coins or change.

To make future transactions, the bitcoin network must add up all your pieces of change, which are classed as either 'spent' or 'unspent'.

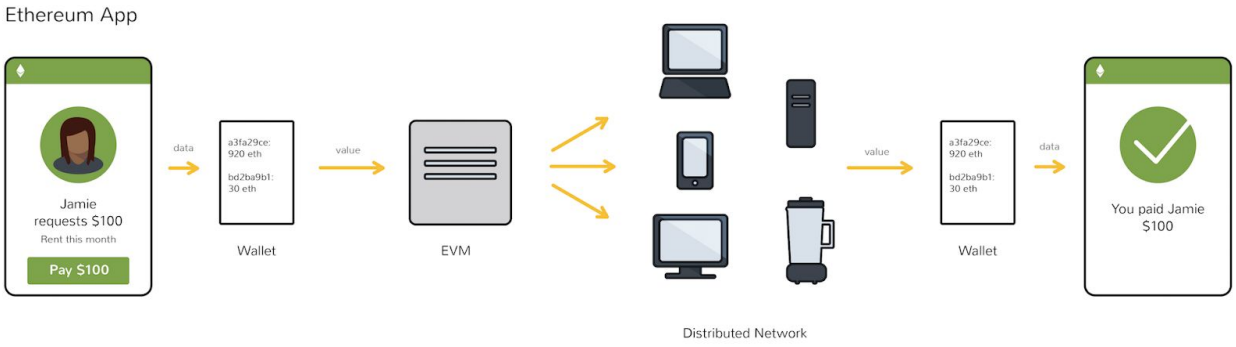
Ethereum, on the other hand, uses accounts. Like bank account funds, ether tokens appear in a wallet, and can be ported (so to speak) to another account. Funds are always somewhere, yet don't have what you might call a continued relationship.



2.3 Ethereum Virtual Machine

With ethereum, every time a program is used, a network of thousands of computers processes it. Contracts written in a smart contract-specific programming languages are compiled into 'bytecode', which a feature called the 'ethereum virtual machine' (EVM) can read and execute.

All the nodes execute this contract using their EVMs.



Every node in the network holds a copy of the transaction and smart contract history of the network, in addition to keeping track of the current 'state'. Every time a user performs some action, all of the nodes on the network need to come to agreement that this change took place.

The goal here is for the network of miners and nodes to take responsibility for transferring the shift from state to state, rather than some authority such as PayPal or a bank. Bitcoin miners validate the shift of ownership of bitcoins from one person to another. The EVM executes a contract with whatever rules the developer initially programmed.

Actual computation on the EVM is achieved through a stack-based bytecode language (the ones and zeroes that a machine can read), but developers can write smart contracts

in high-level languages such as Solidity and Serpent that are easier for humans to read and write.

Miners are the ones that are preventing bad behavior, ensuring that no one is spending their money more than once and rejecting smart contracts that haven't been paid for. There are a few thousand ethereum nodes out there, and every node is compiling and executing the same code.

3. Alternatives to Ethereum Smart Contracts

3.1 NEO

NEO has been called the “Chinese Ethereum” of the cryptocurrency world, due to how similar people say they are. It is the first open-source blockchain network to launch in China. Both NEO and ETH offer a decentralized network and platform for smart contracts to be activated without any third party meddling. They’re both great at doing that as well, but they do have some significant differences.

For starters, NEO is much more scalable than Ethereum is. It can process more transactions at once, significantly reducing wait times for its users. While Ethereum’s developers are working on expanding the network’s scale, it doesn’t come close to NEO’s 1,000 transactions per second.

Also, NEO supports multiple programming languages. While Ethereum’s language (Solidity) is similar to popular ones like JavaScript, it still requires users to learn a new language for programming in the network. NEO allows developers to use C#, Java, or other mainline languages to write smart contracts. This means that as NEO’s presence grows, it may start to gather more developers in the long run due to its ease of access.

NEO wants to prevent any random hard forks like the Ethereum/Ethereum classic debacle, so it is coded to avoid this entirely. While soft forks are still possible, hard forks are not.

Finally, NEO coins are not mined. Instead, token holders are given NEO GAS, which is a dividend of NEO coins and can be received in any suitable NEO Wallet which supports GAS.

3.2 EOS

EOS wants to be the best of all worlds in cryptocurrency. Essentially, EOS combines the security of Bitcoin and the smart contracts and dApp support from Ethereum (among other technologies) to create the ultimate scalable blockchain platform.

Anything a decentralized application dev team may need, EOS plans to have it for them. Shared databases, authentication systems, account recovery, cloud storage and hosting, potentially infinite scaling, all paid for by staking money in EOS tokens. Companies can create monetization and service strategies for their users, all with the provided framework.

The EOS community participates by voting on which applications are running properly or if any changes are needed to the source code. Nothing is done without the people's approval on EOS.

Blocks are structured into "cycles" that are sequentially verified, reducing latency on the network and keeping performance high at all times. EOS tokens provide no function other than as a stakeholder for both developers and community members.

4. Tools and resources

4.1 Solidity

Solidity Language itself is a tool that used to generate machine-level code that can execute on the EVM. It is a language with a compiler which takes high-level human-readable code and breaks it down into simple instructions like “put data into a register”, “add data from two registers”, “jump back to instruction at memory point x”, which form the basis of any microprocessor executable program.

When an ethereum block is “mined”, the smart-contract deployments and function calls within that block - meaning those lined up to happen within the last block duration - get executed on the node that mines the block, and the new state changes to any storage spaces or transactions within that smart-contract actually occur on that miner node. Then the new block gets propagated out to all the other nodes and each node tries to independently verify the block, which includes doing those same state changes to their local copy of the blockchain also. This is where it will fail if the smart-contract acts non-deterministically. If the other nodes cannot come to a consensus about the state of blockchain after the new block and its contracts get executed, the network could literally halt.

This is why Ethereum smart-contracts (and smart-contracts in general in any blockchain system) must be deterministic: so that the network of nodes can always validate and maintain consensus about the new blocks coming in, in order to continue running.

3.2 Truffle

Truffle is a developer environment, testing framework used to test and deploy smart contracts written in Solidity into Ethereum Blockchains. It allows developers to spin up smart contract project at the click of a button and provides you with a project structure,

files, and directories that make deployment and testing much easier (or else you would have to configure these yourself).
To use it you first need nodejs. Truffle ensures:

- Built-in smart contract compilation, linking, deployment and binary management.
- Automated contract testing with Mocha and Chai.
- Configurable build pipeline with support for custom build processes.
- Scriptable deployment & migrations framework.
- Network management for deploying to many public & private networks.
- Interactive console for direct contract communication.
- Instant rebuilding of assets during development.
- External script runner that executes scripts within a Truffle environment

3.3 Ganache

Ganache allows you to create a private Ethereum blockchain for you to run tests, execute commands, and inspect state while controlling how the chain operates. It gives you the ability to perform all actions you would on the main chain without the cost. Many developers use this to test their smart contracts during development. It provides convenient tools such as advanced mining controls and a built-in block explorer.

3.4 PyEthereum

This is the Python core library of the Ethereum project
Pyethereum is the core blockchain related logic: transactions, blocks, contract VM etc.

Pyethapp uses both pydevp2p for the p2p logic along with Pyethereum to create a complete Ethereum client.