Blockchain and Cryptocurrencies

Week 8 - Smart Contracts

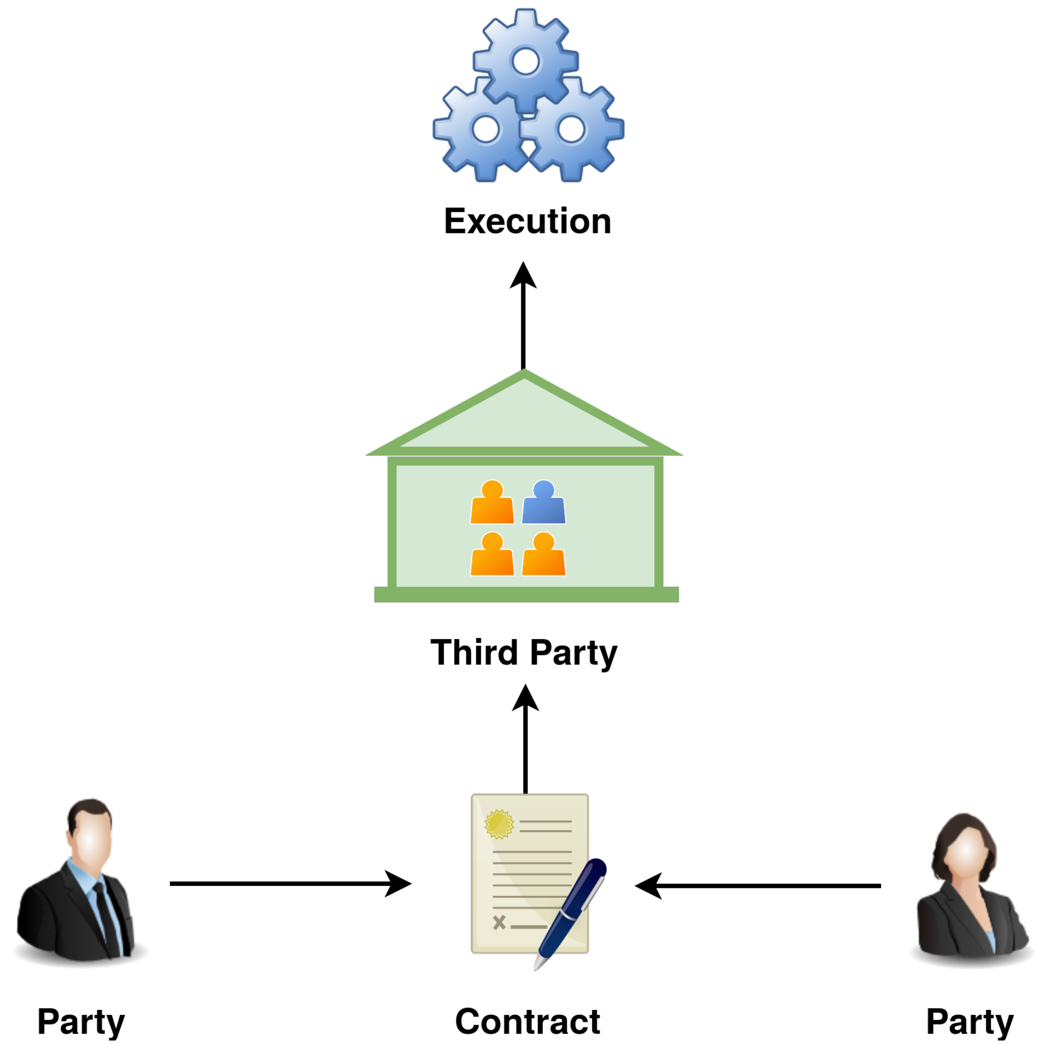
Dr. Thi Thu Ha Doan   Prof. Dr. Peter Thiemann

Albert-Ludwigs-Universität Freiburg, Germany

SS 2020

Contents

Traditional Contracts



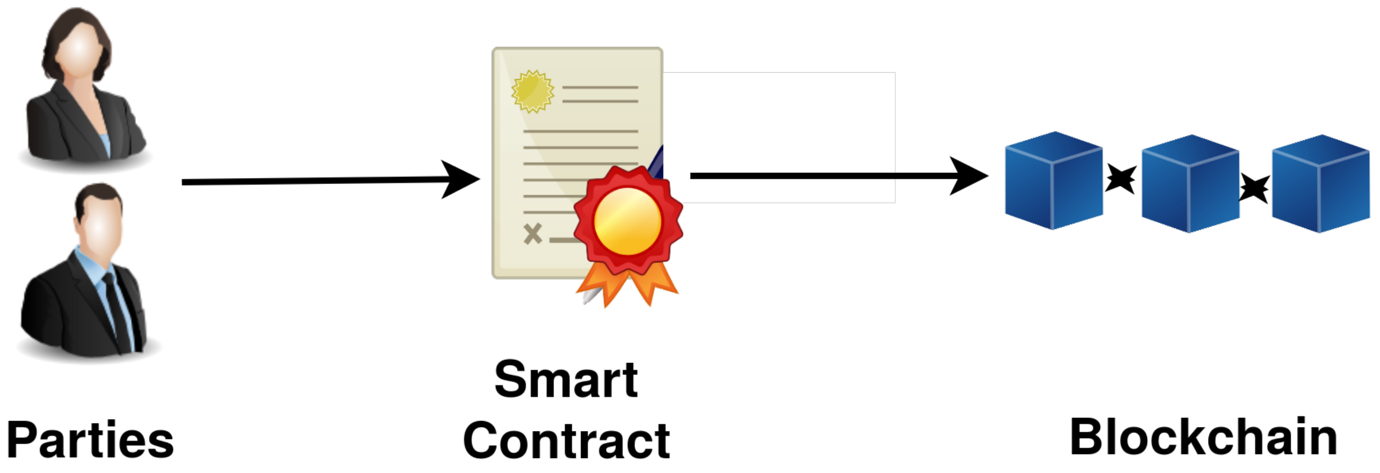
Definition

- contract parties
- common goal
- obligations of each party
- further contractual clauses
- declaration of will

Smart Contracts

A **smart contract** is a program that is stored and automatically executed on a blockchain.

- First mentioned 1997 by Nick Szabo
- Contract in form of code (Code is law)
- Self-executing and self-enforcing rules



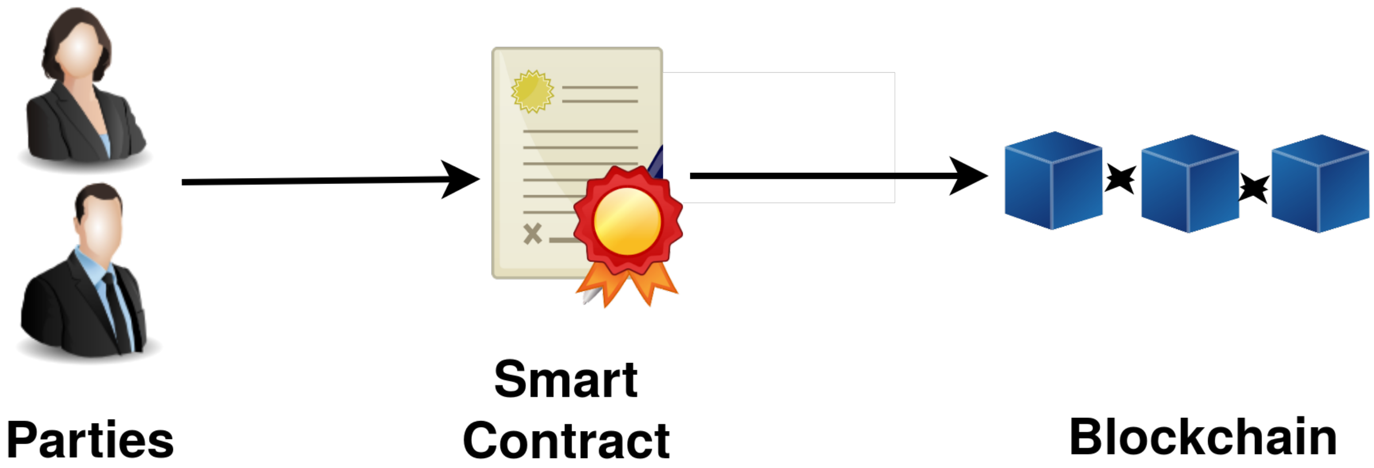
Smart Contracts

A **smart contract** is a program that is stored and automatically executed on a blockchain.

- First mentioned 1997 by Nick Szabo
- Contract in form of code (Code is law)
- Self-executing and self-enforcing rules

Goals:

- Trusted execution without third parties
- Security and low transaction costs



Smart Contracts

Properties

- Stored on a blockchain ⇒ **immutable**
- Executed and validated by the network ⇒ **distributed**
- transparent
  - access, interact and verify (everyone)
  - consensus about the result
  - maintain application state

Smart Contracts

Properties

- Stored on a blockchain $\Rightarrow$ **immutable**
- Executed and validated by the network $\Rightarrow$ **distributed**
- transparent
  - access, interact and verify (everyone)
  - consensus about the result
  - maintain application state

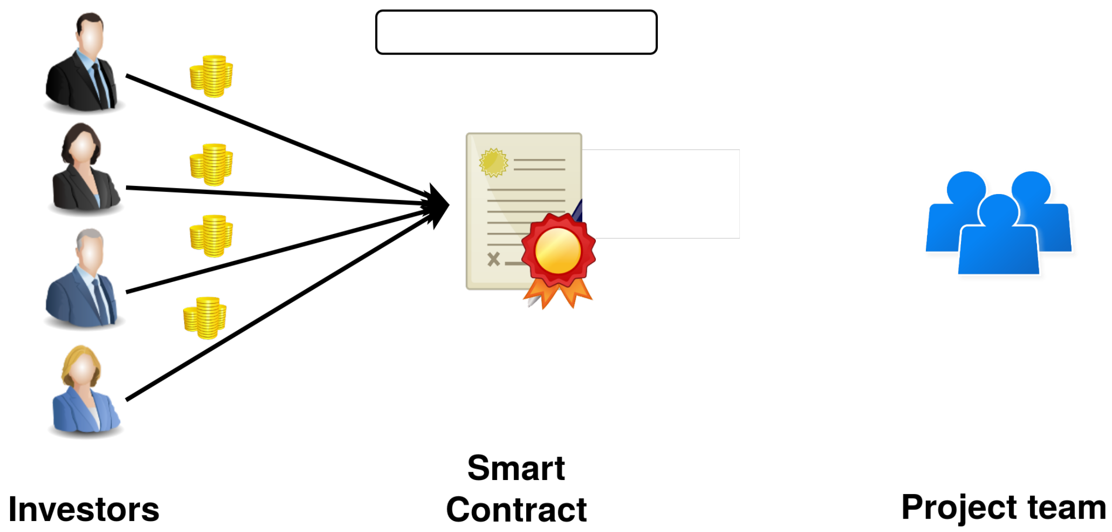
Functions

- send and receive coins
- interact with other smart contracts

Example: Crowdfunding

To raise money from a group of investors to fund a project

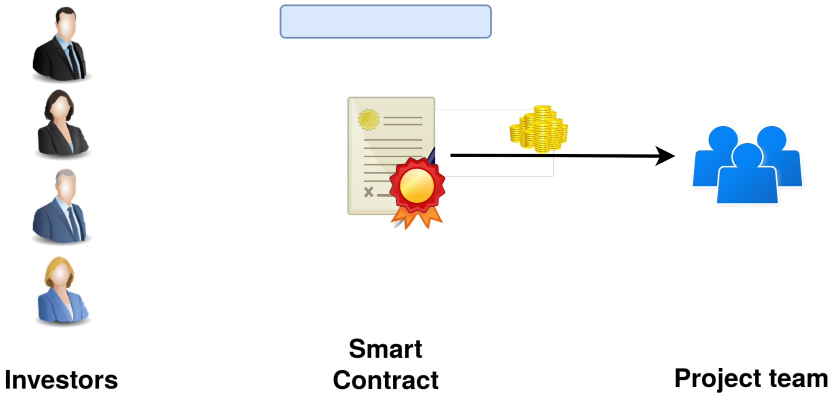
- set up a funding goal and a deadline
- start to collect money



Example: Crowdfunding

Terms and Conditions

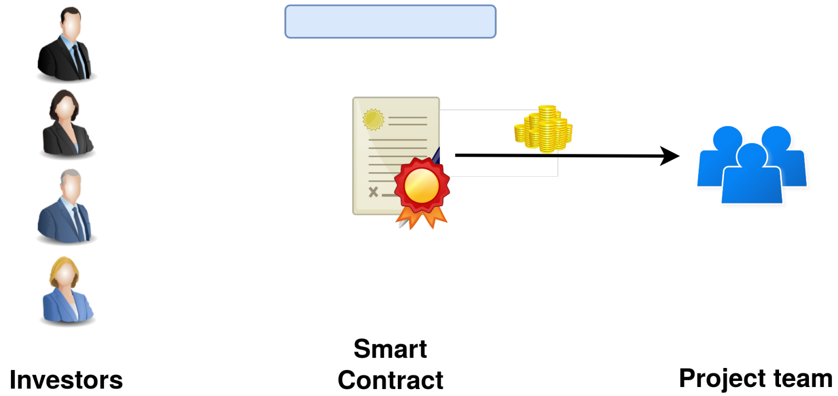
1. if the funding goal is met, transfer the money to the project team



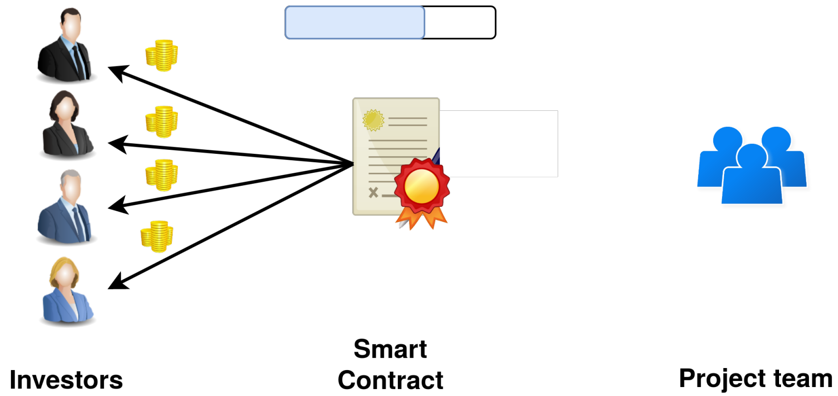
Example: Crowdfunding

Terms and Conditions

1. if the funding goal is met, transfer the money to the project team



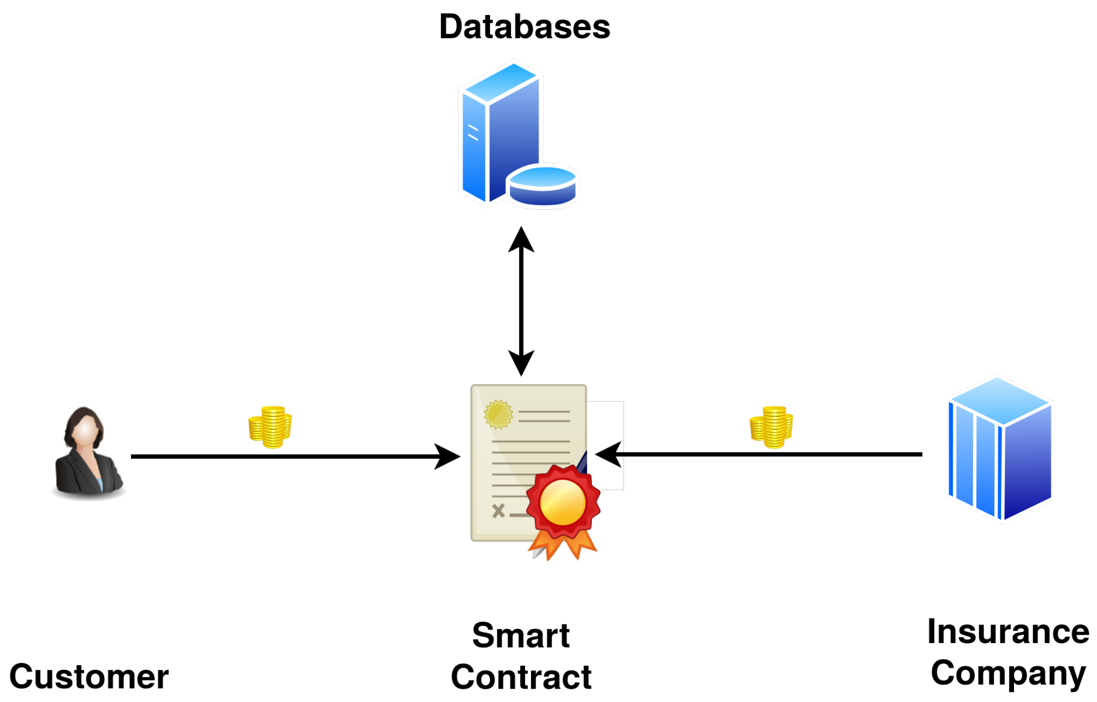
2. if the project fails to reach the funding goal, return the money to the investors



Example: Flight Delay Insurance

Ensure that a customer is compensated for a delayed flight

- compensation condition: flight delayed for more than two hours
- linked to the databases that record the flight status

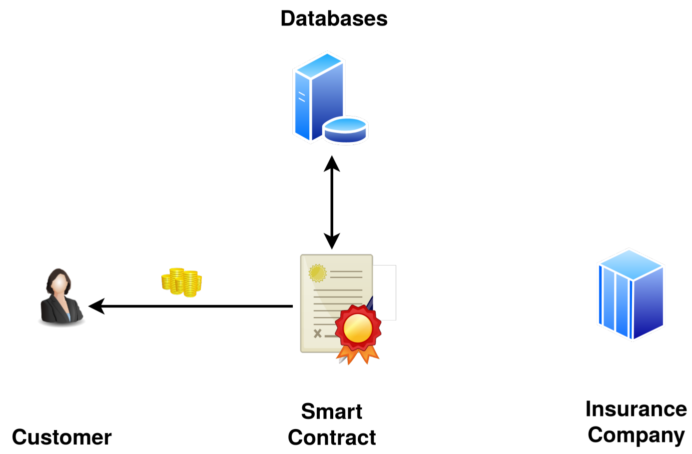


https://www.axa.com/en/magazine/axa-goes-blockchain-with-fizzy

Example: Flight Delay Insurance

Terms and Conditions

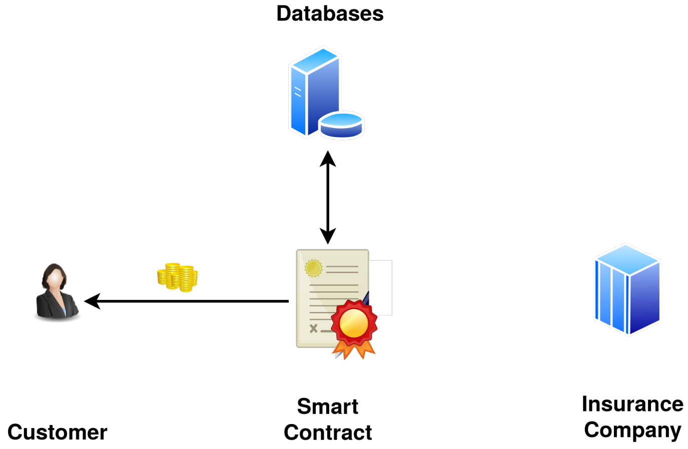
1. if flight delayed for more than two hours, customer is automatically compensated



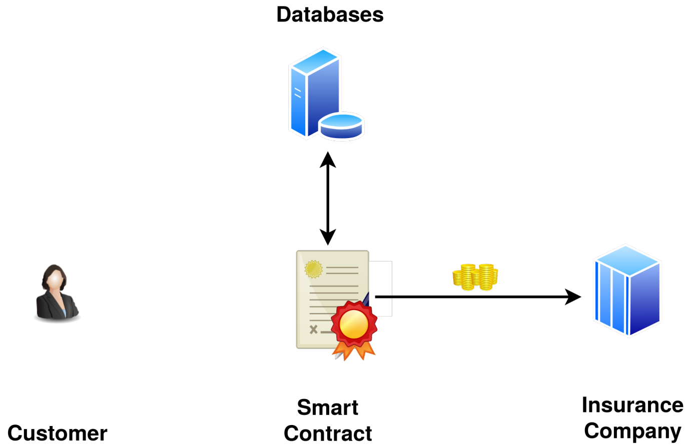
Example: Flight Delay Insurance

Terms and Conditions

1. if flight delayed for more than two hours, customer is automatically compensated



2. if less than two hours delay, transfer premium to the insurance company



Store

Smart contract's variables/states are stored on the blockchain

- storage space of a smart contract is limited by cost

Gas

Traditional transaction fees

- Fee to compensate the network
- Fixed for each kind of transaction
- Bitcoin: depending on the size of the transaction

Gas

Traditional transaction fees

- Fee to compensate the network
- Fixed for each kind of transaction
- Bitcoin: depending on the size of the transaction

Transaction fees for smart contracts

Effort of the network:

- time to run (number of steps)
- storage needed

Gas

Traditional transaction fees

- Fee to compensate the network
- Fixed for each kind of transaction
- Bitcoin: depending on the size of the transaction

Transaction fees for smart contracts

Effort of the network:

- time to run (number of steps)
- storage needed

Gas

- each operation and each storage allocation requires a certain amount of gas to be executed.
- fee is calculated from gas price **offered by the caller**
  Total fee = gas_Used * gas_Price

Gas

Example: Gas consumption

Add two numbers: ADD 3 4

- load or store a number: 10 gas
- adding two numbers: 20 gas

$$\text{Total gas} = 10 + 10 + 20 + 10$$

Gas

Gas Limit

Maximum amount of gas a caller is willing to pay (deposited like a prepayment)

- each operation has different gas cost
- reasonable gas limit is needed
- when the gas runs out
  - network stops executing the code,
  - **contract reverts back to its original state**

Gas

Gas Limit

Maximum amount of gas a caller is willing to pay (deposited like a prepayment)

- each operation has different gas cost
- reasonable gas limit is needed
- when the gas runs out
  - network stops executing the code,
  - **contract reverts back to its original state**

Gas limit too low

- must still pay the fee for the computational effort even if the execution fails

Gas

Gas Limit

Maximum amount of gas a caller is willing to pay (deposited like a prepayment)

- each operation has different gas cost
- reasonable gas limit is needed
- when the gas runs out
  - network stops executing the code,
  - **contract reverts back to its original state**

Gas limit too low

- must still pay the fee for the computational effort even if the execution fails

Gas limit too high

- the block gas limit
- unused gas is returned

Contents

Ethereum Blockchain

- Distributed computing platform (blockchain 2.0)
- First and most well-known smart contract platform
- Released 2015, co-founded by Vitalik Buterin and Gavin Wood
- Goal: decentralized software development platform
- Native cryptocurrency: 1 Ether $= 10^{18}$ Wei $= 10^{9}$ Gwei

Ethereum's Features

Block time

About 15 seconds (compared to 10 minutes BTC)

Minting

New coins generated at a constant rate (unless hard forked)

Mining

- Proof-of-work, but different algorithm Ethash for which ASICs are less advantageous
- Migration to proof-of-stake planned

Transaction fees

Calculated by computational effort, bandwidth use, and storage needs

Account-based

Coins are represented by account balances (rather than UTxOs). Accounts can be controlled by a private key or a smart contract.

Pros and cons of account-based blockchains

Advantages

- More flexible transactions. Dependance on existing state and on external input allows for oracles and other logic to influence the outcome.
- Transactions do not explicitly refer to the resulting state $\Rightarrow$ smaller transaction sizes.

Disadvantages

- Transactions affecting the same account(s) will have to be scheduled.
- Account models encourage address reuse, which enables linking transactions to a single owner, thus compromising privacy.

Ethereum Fee Schedule

Appendix G. Fee Schedule

The fee schedule G is a tuple of 31 scalar values corresponding to the relative costs, in gas, of a number of abstract operations that a transaction may effect.

Name	Value	Description*
G_{zero}	0	Nothing paid for operations of the set W_{zero}.
G_{base}	2	Amount of gas to pay for operations of the set W_{base}.
$G_{verylow}$	3	Amount of gas to pay for operations of the set $W_{verylow}$.
G_{low}	5	Amount of gas to pay for operations of the set W_{low}.
G_{mid}	8	Amount of gas to pay for operations of the set W_{mid}.
G_{high}	10	Amount of gas to pay for operations of the set W_{high}.
$G_{extcode}$	700	Amount of gas to pay for an EXTCODESIZE operation.
$G_{extcodehash}$	400	Amount of gas to pay for an EXTCODEHASH operation.
$G_{balance}$	400	Amount of gas to pay for a BALANCE operation.
G_{sload}	200	Paid for a SLOAD operation.
$G_{jumpdest}$	1	Paid for a JUMPDEST operation.
G_{sset}	20000	Paid for an SSTORE operation when the storage value is set to non-zero from zero.
G_{sreset}	5000	Paid for an SSTORE operation when the storage value's zeroness remains unchanged or is set to zero.
R_{sclear}	15000	Refund given (added into refund counter) when the storage value is set to zero from non-zero.
$R_{selfdestruct}$	24000	Refund given (added into refund counter) for self-destructing an account.
$G_{selfdestruct}$	5000	Amount of gas to pay for a SELFDESTRUCT operation.
G_{create}	32000	Paid for a CREATE operation.
$G_{codedeposit}$	200	Paid per byte for a CREATE operation to succeed in placing code into state.
G_{call}	700	Paid for a CALL operation.
$G_{callvalue}$	9000	Paid for a non-zero value transfer as part of the CALL operation.
$G_{callstipend}$	2300	A stipend for the called contract subtracted from $G_{callvalue}$ for a non-zero value transfer.
$G_{newaccount}$	25000	Paid for a CALL or SELFDESTRUCT operation which creates an account.
G_{exp}	10	Partial payment for an EXP operation.
$G_{expbyte}$	50	Partial payment when multiplied by $\lceil \log_{256}(exponent) \rceil$ for the EXP operation.
G_{memory}	3	Paid for every additional word when expanding memory.
G_{txcreate}	32000	Paid by all contract-creating transactions after the *Homestead* transition.
$G_{txdatazero}$	4	Paid for every zero byte of data or code for a transaction.
$G_{txdatanonzero}$	68	Paid for every non-zero byte of data or code for a transaction.
$G_{transaction}$	21000	Paid for every transaction.
G_{log}	375	Partial payment for a LOG operation.
G_{logdata}	8	Paid for each byte in a LOG operation's data.
G_{logtopic}	375	Paid for each topic of a LOG operation.
G_{sha3}	30	Paid for each SHA3 operation.
$G_{sha3word}$	6	Paid for each word (rounded up) for input data to a SHA3 operation.
G_{copy}	3	Partial payment for *COPY operations, multiplied by words copied, rounded up.
$G_{blockhash}$	20	Payment for BLOCKHASH operation.
$G_{quaddivisor}$	20	The quadratic coefficient of the input sizes of the exponentiation-over-modulo precompiled contract.

https://ethereum.github.io/yellowpaper/paper.pdf

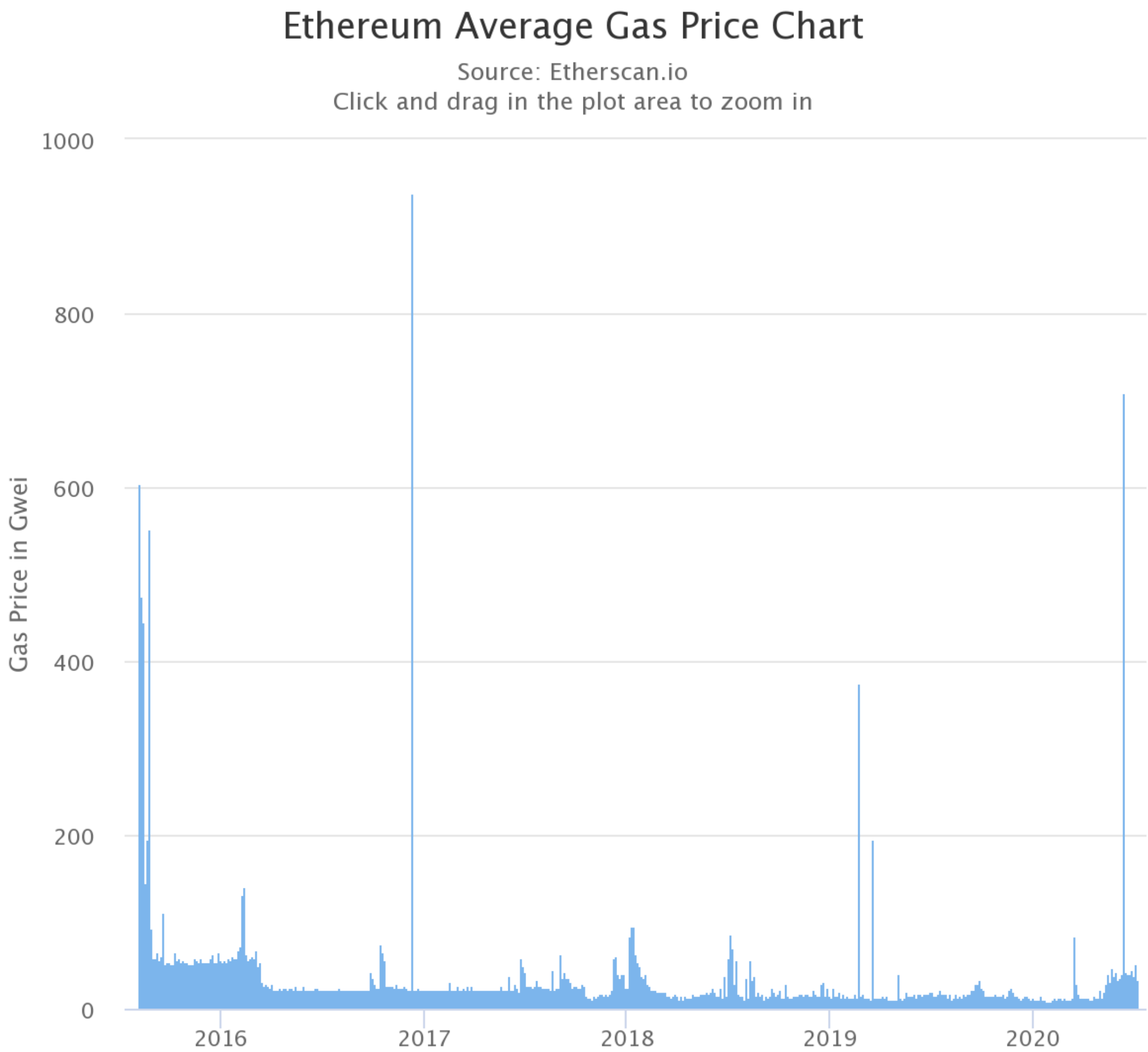
Gas Price

The gas price offered determines how quickly the network processes the contract.
It is up to the miner to accept an offer.
Some websites offer predictions:

Predictions for Gas Used = 21000	Gas Price 32 Gwei	Gas Price 66 Gwei
% of last 200 blocks accepting this gas price	48.1481481481	100
Transactions At or Above in Current Txpool	109	11
Mean Time to Confirm (Blocks)	41	2
Mean Time to Confirm (Seconds)	555	27
Transaction fee (ETH)	0.000672	0.001386
Transaction fee (Fiat)	$0.15926	$0.32848

https://ethgasstation.info/calculatorTxV.php

Ethereum Average Gas Price Chart



https://etherscan.io/chart/gasprice

Ethereum Virtual Machine (EVM)

- Execution platform of Ethereum smart contracts
- Stack-based (like BTC), but Turing-complete
- EVM bytecode: Special instructions for dealing with blockchain requirements (e.g., hashing, signing)
- State space components: account state, world state, storage state, block information, runtime environment information

EVM state space

Accounts

1. Externally owned accounts, controlled by private key
2. Contract accounts, associated with contract code

- Messages between external accounts: transfer of value
- Messages from external account $\rightarrow$ contract: invoke the contract's code

Components of account state

- Nonce
- Balance (in Wei)
- CodeHash

EVM state space (II)

World State

- mapping between 160-bit address $\rightarrow$ account state
- maintained in Modified Merkle Patricia Trie (a mix of a Merkle tree and a prefix tree)

Storage State

- account specific information at run time
- e.g., the runtime stack

EVM state space (III)

Block information — to support a transaction

- Blockhash – The hash of the most recently completed block
- Coinbase – The address of the recipient
- Timestamp – The current block's timestamp
- Number – The number of the current block
- Difficulty – The difficulty of the current block
- Gaslimit – The gas limit that is attached to the current block

Runtime Environment Information

- Gas price – Current gas price as specified by the initiator of the transaction
- Codesize – The size of the transaction codebase
- Caller – The address of the account that is executing the transaction
- Origin – The address of the transaction's original sender

High-Level Contract Languages

Several contract languages compile to EVM byte code

- Solidity (most common, some resemblance to JavaScript),
- Serpent (deprecated),
- LLL (low-level Lisp-like language),
- Vyper (derived from Python).

Solidity in a Nutshell

- Object-oriented, high-level language for implementing smart contracts.
- Design influenced by C++, Python and JavaScript
- Statically typed, supports inheritance, libraries and complex user-defined types.
- Compiles to EVM.

Example

```
pragma solidity >=0.4.0 <0.6.0;

contract SimpleStorage {
    uint storedData;

    function set(uint x) public {
        storedData = x;
    }

    function get() public view returns (uint) {
        return storedData;
    }
}
```

- keeps a single unsigned integer of persistent data
- get and **set** methods to retrieve and modify
- callable by anyone with the contract's address

Example Overlay Currency

```
contract Coin {
    address public minter;
    mapping (address => uint) public balances;

    event Sent(address from, address to, uint amount);

    constructor() public {
        minter = msg.sender;
    }

    function mint(address receiver, uint amount) public {
        require(msg.sender == minter);
        require(amount < 1e60);
        balances[receiver] += amount;
    }

    function send(address receiver, uint amount) public {
        require(amount <= balances[msg.sender], "Insufficient balance.");
        balances[msg.sender] -= amount;
        balances[receiver] += amount;
        emit Sent(msg.sender, receiver, amount);
} }
```

Example Overlay Currency (II)

- address data type: 160 bit addresses, no arithmetic
- public instructs the compiler to generate an access method (like a getter) for a state variable
- mapping (address => uint)public balances;
  another public state variable, initialized to all zero, lookup only—no iterator!
- event
  can be emitted from the contract
  UIs can listen to events without much cost
- the constructor remembers the creator of the contract msg.sender
- mint
  only the creator can mint, the amount should not be ridiculous, minting is on behalf of a single receiver
- send
  callable by anyone, provided sufficient funds available

Smart Contracts Benefits

- Transparency
- Automated
- Accuracy
- Security
- Speed
- Efficiency
- Trust
- Clear Communication
- Storage & Backup.
- Paper Free
- ...

Challenges of Smart Contracts

- privacy and security: no effective way to guarantee the security of smart contract code
- performance: gas and store limits, resource-constrained running environment, limited online resources
- programming languages and tools: basic, limitations

Smart Contract Applications

- Financial Services & Insurance
- Supply chain management
- Mortgage transactions
- Automated payments
- Real Estate
- Engagement contracts
- Transport & Logistics
- Supply Chain Transparency
- Medical Research
- Creating Smart Contracts
- ...

Smart Contract Platforms

- Ethereum
- EOS
- Tezos
- Cardano
- Stellar
- NEO
- ...