OST
Eastern Switzerland
University of Applied Sciences

Distributed Systems & Blockchain (DS1)

Bitcoin/Blockchain III

Thomas Bocek

1 May 2021

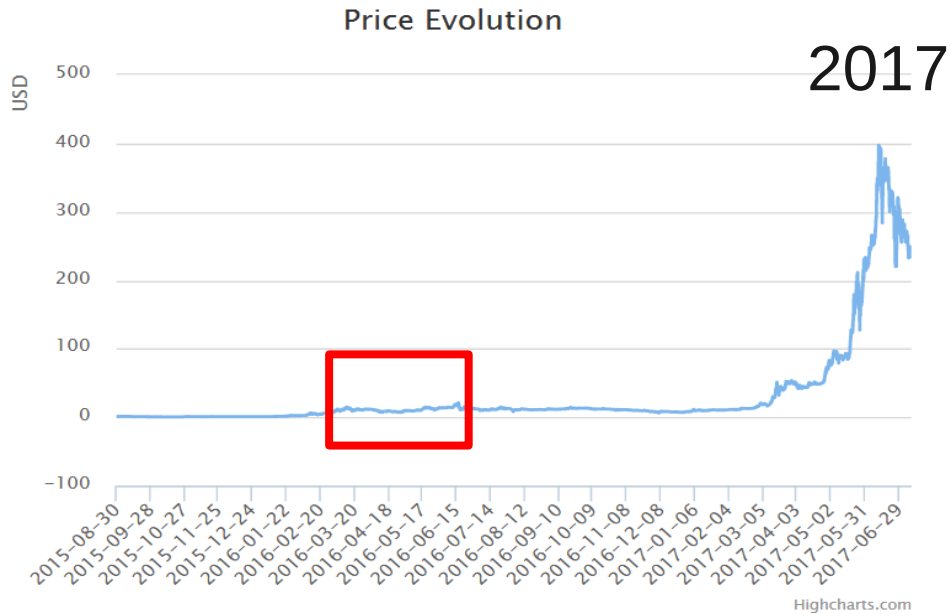
Ethereum History

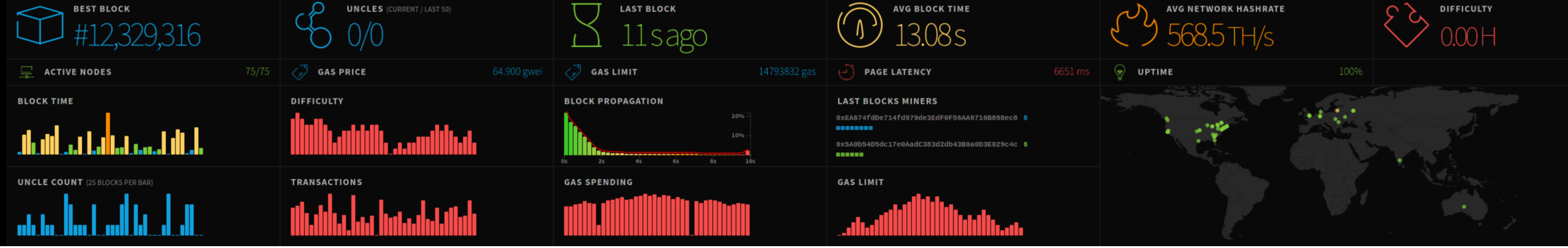
- Olympic (past) – released 09.05.2015
  - Last Ethereum Proof-of-Concept series
  - “Olympic will feature a total prize fund of up to 25,000 ether” (now 70m USD)
- Frontier (past) - released 30.07.2015
  - Main public network, “Beta”/use at your own risk
- Homestead (past) - released 14.03.2016
  - Public network considered “stable”, integrate critical protocol changes
- Byzantium - released 16.10.2017
  - Initial supports for ZKP (private smart contracts), reducing mining reward to 3 ETH
- Constantinople – released 27.02.2019
  - Bitshifting, mining reward 2 ETH (delay difficulty bomb)
- Istanbul - released 08.12.2019, gas cost changes, DDoS resilience, Zcash interop
- Muir Glacier- released 01.01.2020 (delay difficulty bomb)
- Berlin – released 14.04.2021 (gas cost)
- Ethereum 2.0 / Serenity (future)
  - Scalability, proof-of-stake, Sharding, ZKP

OST

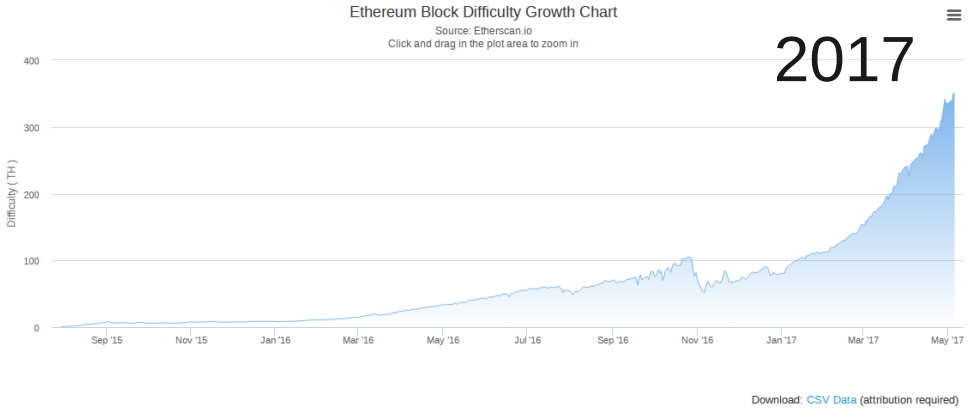
Ethereum Stats

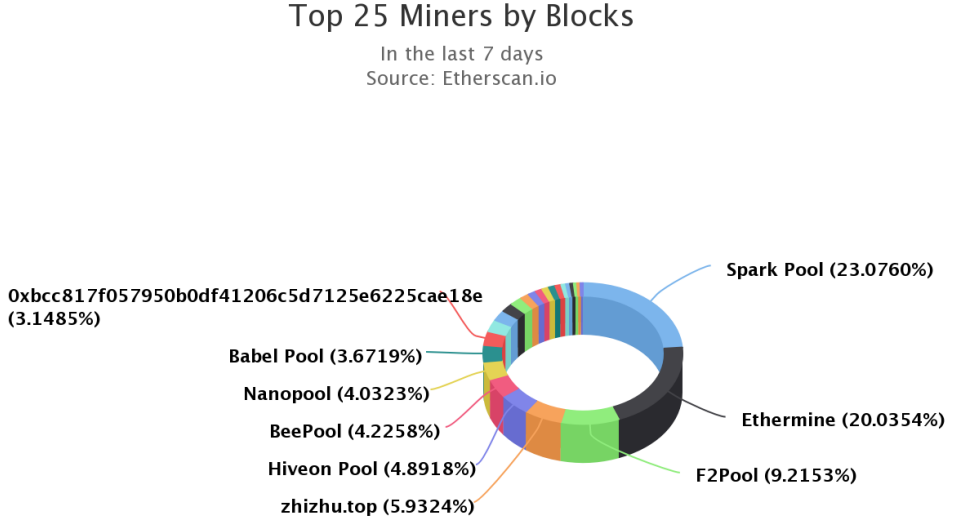
- Basic Stats
  - 2nd in market cap ~ 312b USD
  - Daily transactions (highest ~18.2 TPS, 22.4.2021)
    now ~1250k per day
  - Node count (~6'000 nodes)
  - Blocksize ~50KB
  - Accounts (150mio)

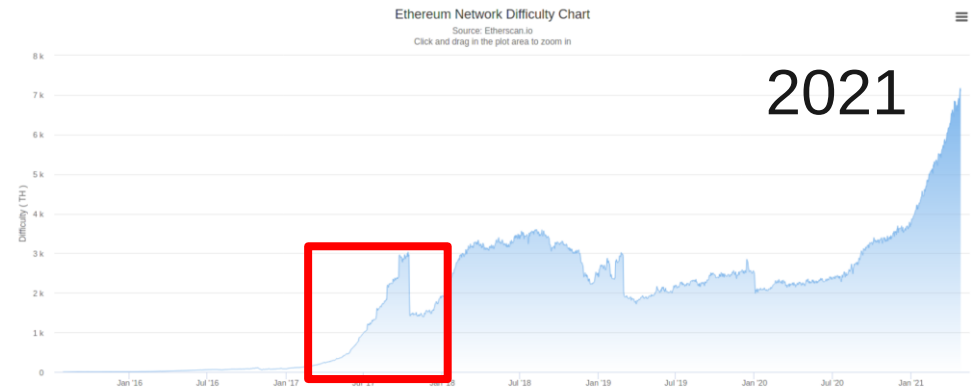




- 115mio ETH supply (stats)
- Increasing difficulty
  - Mining in pools
  - Ethermine + Sparkpool ~43%





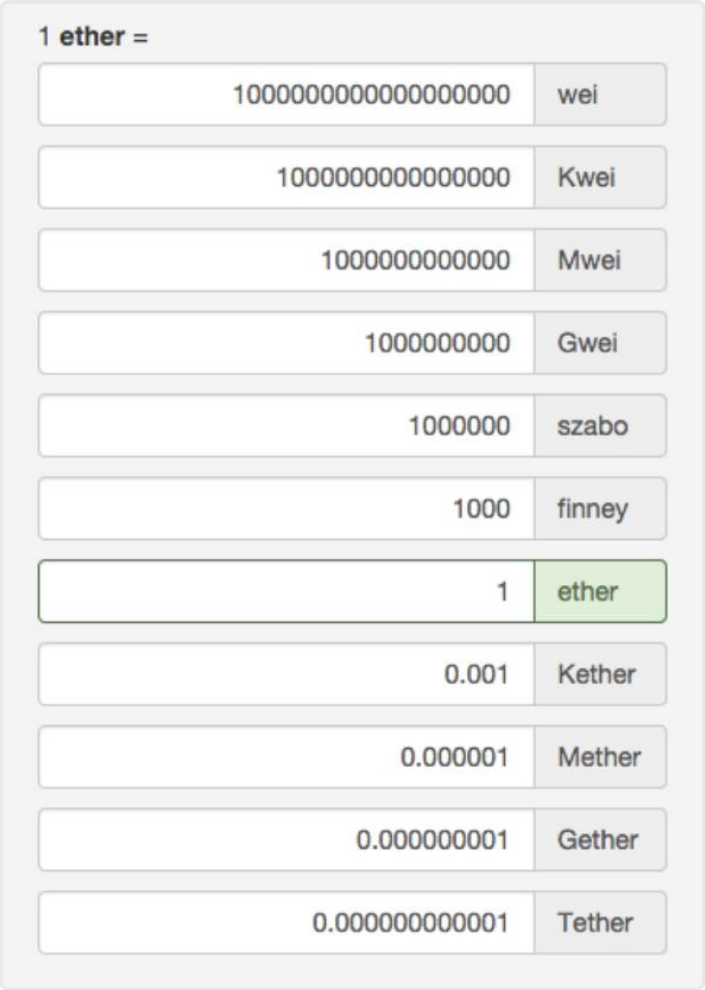


OST

Blocktime and Gas

- Gas Price set by Miner
  - Gas price ~59 gwei
- Miner decides which transaction at which gas price to include
  - Market for TX
- Gas price too low, longer waiting time until TX will be included

- Units:



OST

Blocktime and Gas

- Block time: ~14-15s
  - Ice age
- Smart Contracts are turing complete
  - Every instruction needs to be paid for (example)
  - Gas price ~59 gwei
- Gas price / Gas limit by miners
  - If you run out of gas, state is reverted, ETH gone

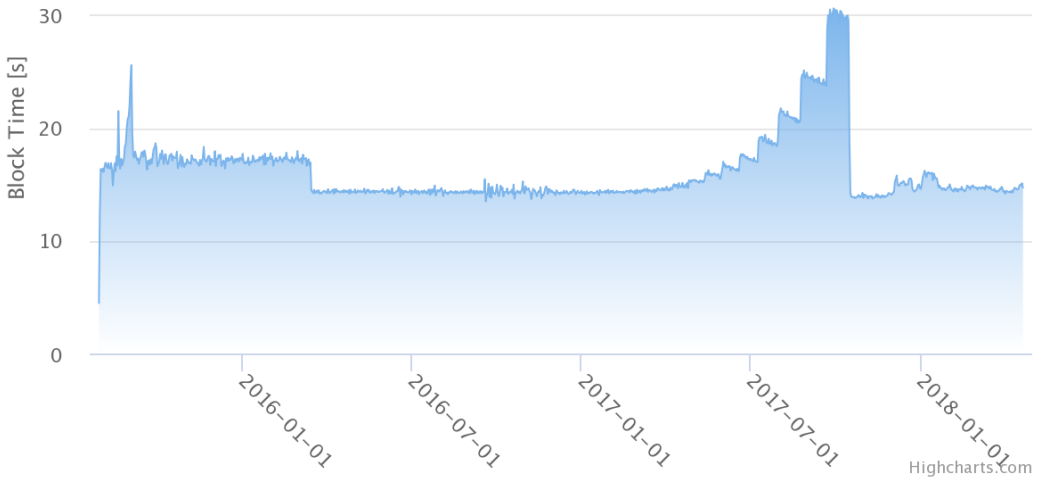
W_{zero} = {STOP, RETURN}
W_{base} = {ADDRESS, ORIGIN, CALLER, CALLVALUE, CALLDATASIZE, CODESIZE, GASPRICE, COINBASE, TIMESTAMP, NUMBER, DIFFICULTY, GASLIMIT, POP, PC, MSIZE, GAS}
$W_{verylow}$ = {ADD, SUB, NOT, LT, GT, SLT, SGT, EQ, ISZERO, AND, OR, XOR, BYTE, CALLDATALOAD, MLOAD, MSTORE, MSTORE8, PUSH*, DUP*, SWAP*}
W_{low} = {MUL, DIV, SDIV, MOD, SMOD, SIGNEXTEND}
W_{mid} = {ADDMOD, MULMOD, JUMP}
W_{high} = {JUMPI}
$W_{extcode}$ = {EXTCODESIZE}

source

Appendix G. Fee Schedule

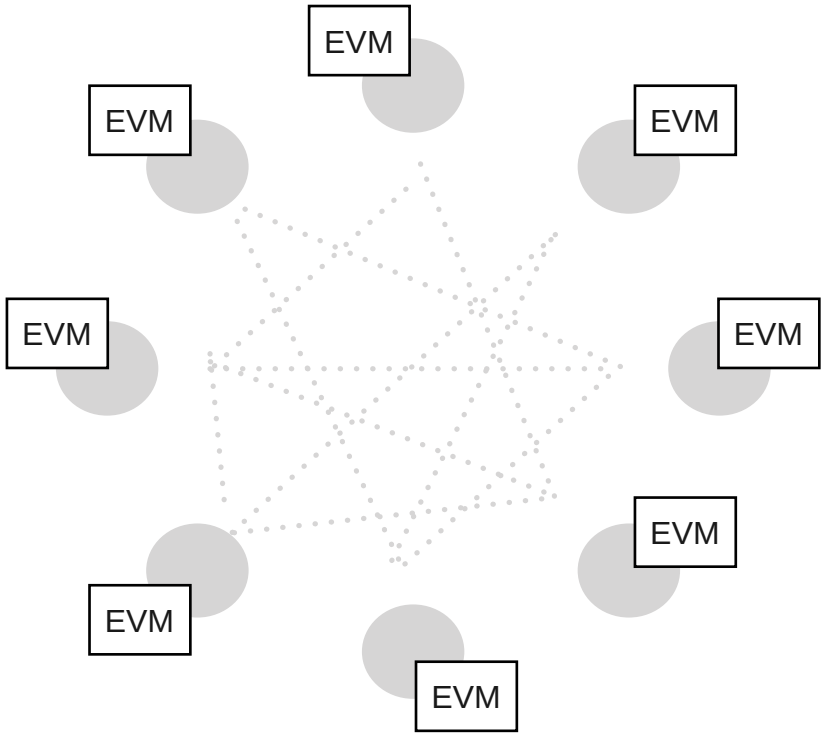
The fee schedule G is a tuple of 31 scalar values corresponding to the relative costs, in gas, of a number of abstract operations that a transaction may effect.

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 operations of the set $W_{extcode}$.
$G_{balance}$	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.
$R_{suicide}$	24000	Refund given (added into refund counter) for suiciding an account.
$G_{suicide}$	5000	Amount of gas to pay for a SUICIDE 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 SUICIDE operation which creates an account.
G_{exp}	10	Partial payment for an EXP operation.
$G_{expbyte}$	10	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.



Ethereum smart contract

- For now, proof of work (PoW)
  - But difficult with ASIC (memory-hard), starts at 1GB RAM for full node and miner (2GB RAM should be enough)
- Computation and storage on EVM is "very expensive“: every contract is run on every full Ethereum node
  - Result on every node is the same
  - Global computer, always running, always correct
- Account-based
  - 2 types: externally controlled, contract
    - Both can have and send ether
    - External accounts: controlled by private keys
  - Contract accounts never executed on their own
    - Contract accounts: controlled by code
    - All action fired from externally controlled accounts



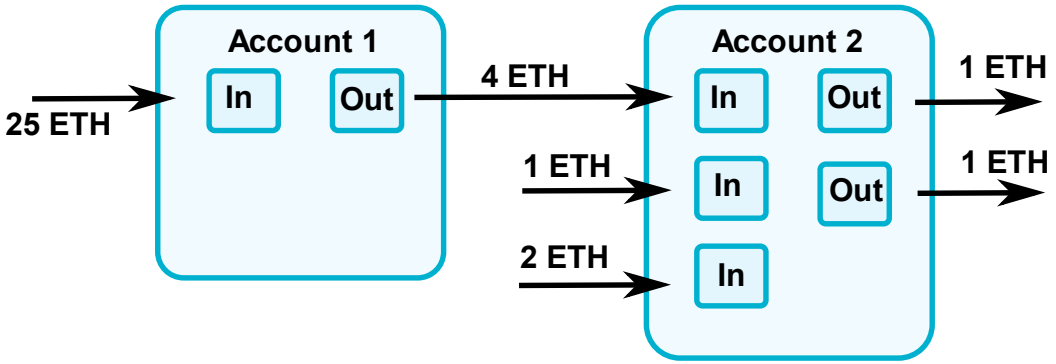
ethereum

OST

Account vs UTXO - Introduction

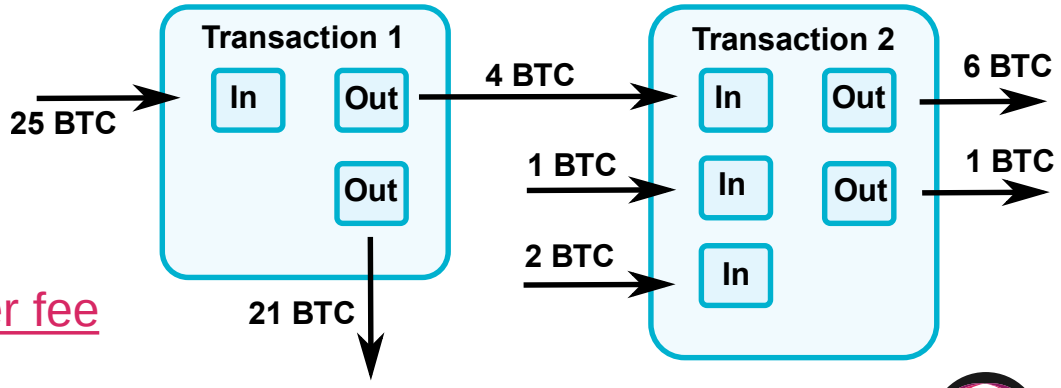
Account-based

- Global state stores a list of accounts with balances and code
- Transaction is valid if the sending account has enough balance
  - Balance on sender is deducted, new balance
- If the receiving account has code, the code runs, and state may be changed
  - Signature must match sending account



UTXO-based

- Every referenced input must be valid and not yet spent
- Total value of the inputs must equal or exceed the total value of the outputs
  - You always spend all outputs
- Transaction must have a signature matching the owner of the input for every input
  - Script determines if input is valid



OST

Account vs UTXO – Pros and Cons

Account-based

- Large space savings
  - One input – one output – one signature
- Simplicity
  - Easier to code and understand
  - Easier for smart contracts (stateful scripting language)

UTXO-based

- Higher degree of privacy
  - New address for each TX
- No replay attacks – no nonce required
- Allows transactions to be processed in parallel
  - If outputs / inputs are separate

OST

Solidity - http://solidity.readthedocs.io

- Version Pragma - reject compiled with specific compiler versions

```
// SPDX-License-Identifier:
pragma solidity ^0.8.4; //not before 0.8.4, not after 0.9.0
```

- Comments

```
// This is a single-line comment.
/*
This is a
multi-line comment.
*/
```

- Contract with State Variables (state change is expensive!)

```
pragma solidity ^0.8.4;
//minimal contract
contract Example1 {
    uint256 counter;
}
```

https://learnxinyminutes.com/docs/solidity/

https://solidity.readthedocs.io

OST

Solidity - http://solidity.readthedocs.io

- Types
  - bool: true and false, int / **uint** (int8, int16, …, int256), **address**: 20 bytes, fixed size arrays: bytes1, bytes2, bytes3, ..., bytes32, variable sized: bytes (push), strings
- Structs

```
struct Account {
  string name;
  uint256 amount;
}
```

- Mapping

```
mapping (address => uint256) accounts; //mapping with basic types
mapping (uint256 => Account) accounts; //mapping with structs
```

OST

Solidity – Basics

- Arrays

```
string[] names

uint256 newLength = names.push("John");
```

- Another contract

```
pragma solidity ^0.8.4;

contract Example2 {

    struct Account {

       string name;

       uint256 amount;

    }
    uint256 public counter;

    mapping (uint256 => Account) accounts;

}
```

- State changing/non-state changing

```
function get(uint256 nr) public view returns (string
memory) {

    return accounts[nr].addr;

}

function set(uint256 nr, string memory name) public
{

    require(owner == msg.sender);

    accounts[counter++] = Account(name, nr);

}
```

- Read state variables
  - Constant function does not modify state
  - Reads “for free”

OST

Solidity – ownership

- Preconditions
  - Special Variables and Functions

```
require(owner == msg.sender);
```

  - Set owner – old syntax

```
pragma solidity ^0.4.0;
contract Example3 {
    address owner;
    function Example3() {
        owner = msg.sender;
    }
}
```

- Constructor – new syntax

```
constructor(string memory addr) {
    owner = msg.sender;
}
```

- Events – notifications

```
pragma solidity ^0.8.4;
contract Example4 {
    event Message(string msg);
```

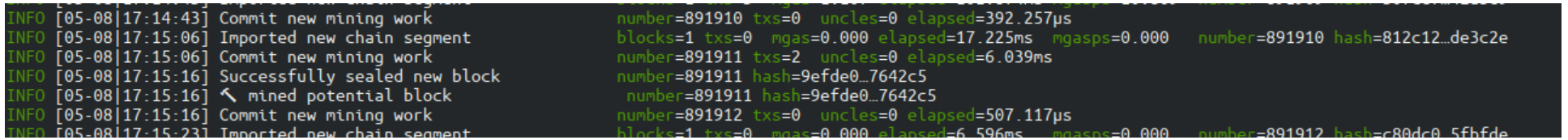
- E.g. in the Geth console

```
at = browser_example1_sol_example2Contract.at("0x…")

var event = at.Message(function(error, result){console.log(result.args.msg)});
```

OST

Solidity Deployment

- Create address (geth) – or openethereum, or MEW, or MetaMask…
  - Ethereum address = create random private key → derive public key → hash it, take last 20bytes
    - geth --rinkeby account new
- Important sites
  - https://rinkeby.etherscan.io/ (blockchain explorer)



  - No mining (use twitter with https://www.rinkeby.io)
- Start geth
  - geth --rinkeby --syncmode light --rpc --rpccorsdomain '*' --rpcapi "eth,net,web3,personal" --allow-insecure-unlock --unlock 0 --password <(echo 123456)
  - geth attach http://127.0.0.1:8545 or
  - Connect Remix Solidity IDE to geth / or use JavaScript Ethereum blockchain in Remix

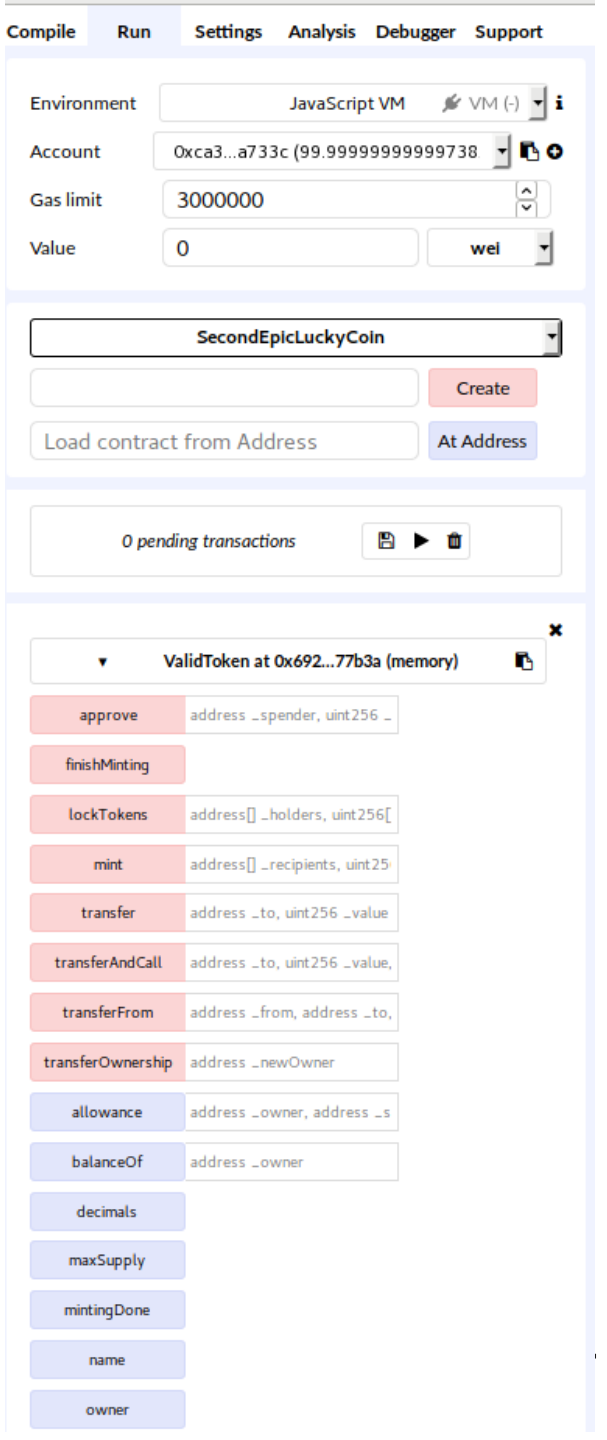
OST

Solidity IDE

- Create your first contract

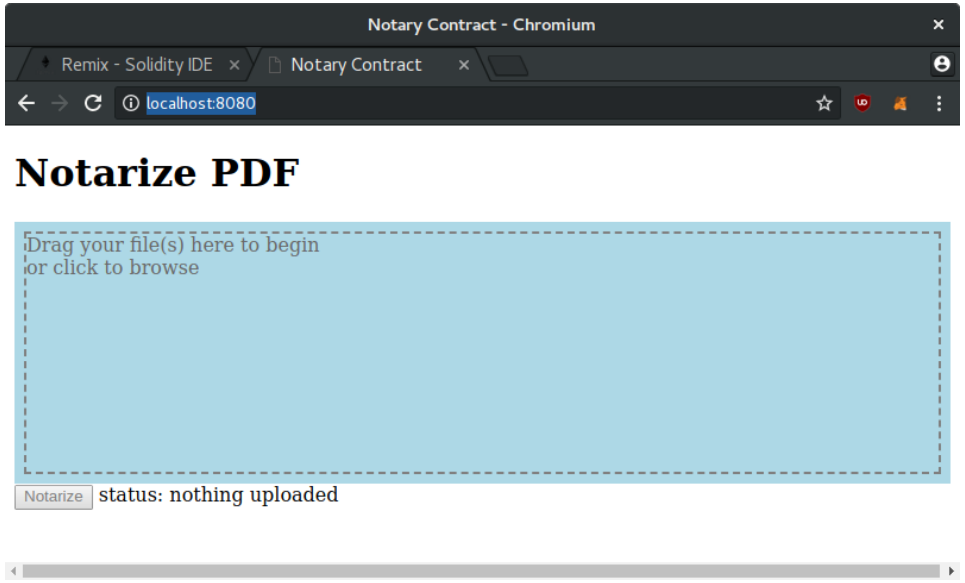
```
pragma solidity ^0.8.4;

//minimal contract

contract Example1 {

    uint counter;

}
```

- Remix – Solidity IDE – Environment: Web3 Provider
  - http://localhost:8545
- «Deploy» contract
- Mixed content - workaround: use http
  - **https://remix.ethereum.org**
- Select Web3 Provider (geth is your Web3 Provider)
  - Alternatively: use scripts, MetaMask, or JavaScript EVM



Example

- Installation
  - npm install
  - ./node_modules/.bin/webpack
  - ./node_modules/.bin/webpack serve
- Open Browser: http://localhost:8080/



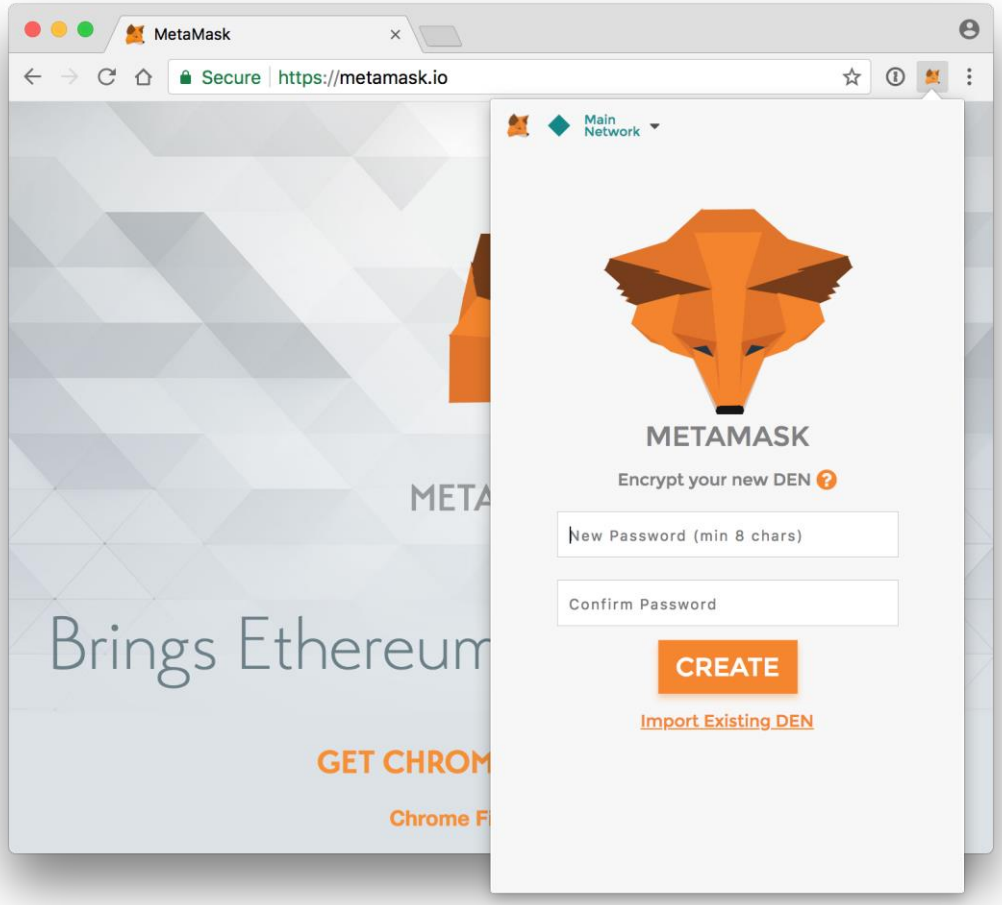

```
draft@home: ~/git/VSS-web3js
File  Edit  View  Search  Terminal  Help
draft@home:~/git/VSS-web3js$ ./node_modules/.bin/webpack-dev-server
i 「wds」: Project is running at http://localhost:8080/
i 「wds」: webpack output is served from /
i 「wdm」: Hash: c4c7c0d3279286de6649
Version: webpack 4.7.0
Time: 1139ms
Built at: 2018-05-06 12:57:52
                         Asset       Size  Chunks             Chunk Names
main.c4c7c0d3279286de6649.js    947 KiB    main  [emitted]  main
                  index.html  395 bytes          [emitted]
Entrypoint main = main.c4c7c0d3279286de6649.js
[./node_modules/ansi-html/index.js] 4.16 KiB {main} [built]
[./node_modules/loglevel/lib/loglevel.js] 7.68 KiB {main} [built]
[./node_modules/strip-ansi/index.js] 161 bytes {main} [built]
[./node_modules/url/url.js] 22.8 KiB {main} [built]
[./node_modules/vue/dist/vue.esm.js] 286 KiB {main} [built]
[./node_modules/webpack-dev-server/client/index.js?http://localhost:8080] (webpack)-dev-server/client?http://localhost:8080 7.75 KiB {main} [built]
[./node_modules/webpack-dev-server/client/overlay.js] (webpack)-dev-server/client/overlay.js 3.58 KiB {main} [built]
[./node_modules/webpack-dev-server/client/socket.js] (webpack)-dev-server/client/socket.js 1.05 KiB {main} [built]
[./node_modules/webpack/hot sync ^\.\/log$] (webpack)/hot sync nonrecursive ^\.\/log$ 170 bytes {main} [built]
[./node_modules/webpack/hot/emitter.js] (webpack)/hot/emitter.js 77 bytes {main} [built]
[./node_modules/webpack/hot/log.js] (webpack)/hot/log.js 1010 bytes {main} [optional] [built]
[./src/App.vue] 908 bytes {main} [built]
[./src/App.vue?vue&type=template&id=7ba5bd90] 194 bytes {main} [built]
[0] multi (webpack)-dev-server/client?http://localhost:8080 ./src 40 bytes {main} [built]
[./src/index.js] 129 bytes {main} [built]
    + 63 hidden modules
Child html-webpack-plugin for "index.html":
     1 asset
    Entrypoint undefined = index.html
    [./node_modules/html-webpack-plugin/lib/loader.js!./index.html] 527 bytes {0} [built]
    [./node_modules/lodash/lodash.js] 527 KiB {0} [built]
    [./node_modules/webpack/buildin/global.js] (webpack)/buildin/global.js 489 bytes {0} [built]
    [./node_modules/webpack/buildin/module.js] (webpack)/buildin/module.js 497 bytes {0} [built]
i 「wdm」: Compiled successfully.
```


OST

MetaMask

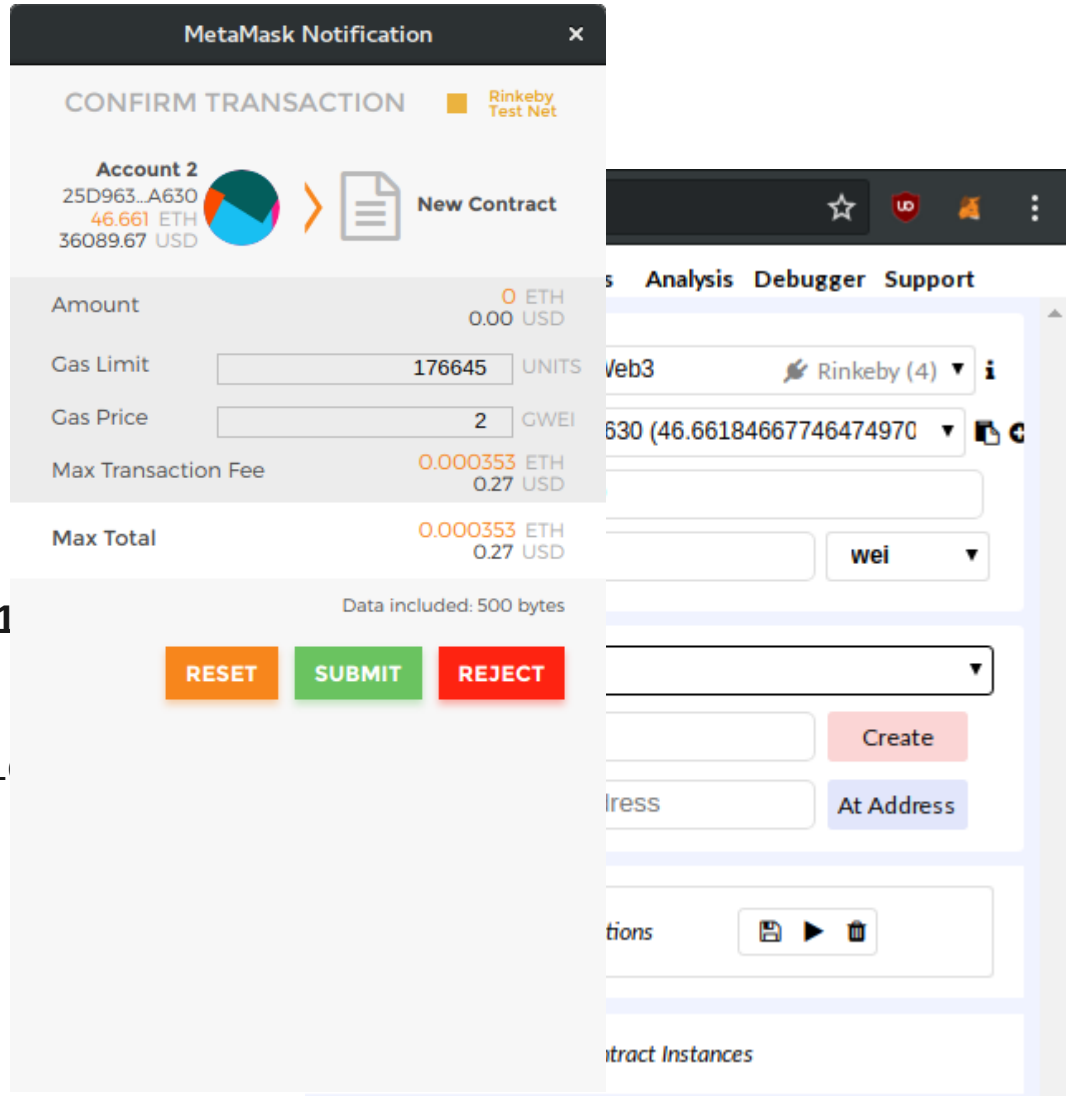
- MetaMask
  - Browser plugin to make Ethereum transactions in browsers
  - Manage your key pairs and sign blockchain transactions
  - MetaMask injects javascript library - ethers.js
  - Uses infura

- Remix IDE: https://remix.ethereum.org
  - Use Notary.sol from https://github.com/tbocek/FS21/blob/main/ethereum/Notary.sol
  - Alternatively, use IntelliJ solidity plugin and deploy via geth, parity, or a local test blockchain



Create Contract

- Connect to MetaMask
  - Injected Web3
  - If you see your accounts, good!
- Create contract!
  - Add address to the code (manually)
- Store value
  - 0x654cf88c**4cff3e62696f7cbb55fbba922b2a75897a01 347e371e9398b658c4affa611aa**
  - Hash is: 0x4cff3e62696f7cbb55fbba922b2a75897a010347e371 398b658c4affa611aa
  - What is 0x654cf88c?
- First four bytes of the Keccak (SHA-3) hash of the signature of the function.
  - Keccak(store(bytes32))



ERC20

- ERC20 is a technical standard for smart contracts on the Ethereum blockchain for implementing tokens
  - proposed on November 19, 2015
  - Mai 2021: more than 392’868 ERC20 token contracts: …
- ERC20 Token (Simplified)
  - No allowance / approval / transferFrom, also no Approval event
  - Creator gets 1000 coins
  - Before 0.8.0 - SafeMath

```
abstract contract SimpleERC20 {
    function totalSupply() public virtual returns (uint256);
    function balanceOf(address who) public virtual returns (uint256);
    function transfer(address to, uint256 value) public virtual returns (bool);
    event Transfer(address indexed from, address indexed to, uint256 value);
}

    string public constant name = "VSS-TOKEN";
    string public constant symbol = "VST";
    uint8 public constant decimals = 18;

    mapping(address => uint256) balances;
    uint256 totalSupply_;

    constructor() {
        totalSupply_ = 1000 * (10**18);
        balances[msg.sender] = totalSupply_;
    }
…
```


OST

Security Considerations

- List of the top smart contract vulnerabilities
  1. Reentrancy (DAO 3.5M, 50M)
  2. Access Control (Parity 150K, 30M)
  3. Arithmetic Issues (DAO)

```
function withdraw(uint _amount) {
      require(balances[msg.sender] >= _amount);
      msg.sender.call.value(_amount)();
      balances[msg.sender] -= _amount;
}

Contract AA {
function attack() {
      A a = A(addressOfA);
      a.withdraw(100);
}
function () payable {
      A a = A(addressOfA);
      a.withdraw(100);
}
}
 function initContract() public {
      owner = msg.sender;
 }

 function withdraw(uint _amount) {
      require(balances[msg.sender] - _amount > 0);
      msg.sender.transfer(_amount);
      balances[msg.sender] -= _amount;
 }
```


OST

Security Considerations

- List of the top smart contract vulnerabilities
  1. Reentrancy (DAO 3.5M, 50M)
  2. Access Control (Parity 150K, 30M)
  3. Arithmetic Issues (DAO)
  4. Unchecked Return Values For Low Level Calls
  5. Denial of Service (Parity 500K, 300M)
  6. Bad Randomness (400K)
     - Future blockhash as random source, but miner can influence it.

```
function withdraw(uint256 _amount) public {
      require(balances[msg.sender] >= _amount);
      balances[msg.sender] -= _amount;
      etherLeft -= _amount;
      msg.sender.send(_amount);
}

function getEtherBalanceIndex(address _address) internal
returns (uint256) {
        for (uint256 i = 0; i < investors.length; i++) {
            if (investors[i] == _address) {
                return i;
            }
        }
        return investors.length;
    }

 function play() public payable {
       require(msg.value >= 1 ether);
       if (block.blockhash(blockNumber) % 2 == 0) {
             msg.sender.transfer(this.balance);
       }
 }
```


OST

Security Considerations

- List of the top smart contract vulnerabilities
  1. Reentrancy (DAO 3.5M, 50M)
  2. Access Control (Parity 150K, 30M)
  3. Arithmetic Issues (DAO)
  4. Unchecked Return Values For Low Level Calls
  5. Denial of Service (Parity 500K, 300M)
  6. Bad Randomness (400K)
  7. Front-Running
  8. Time Manipulation - A malicious miner sets own timestamp ~ 900s
  9. Short Address Attack – not yet exploited
  10. Unknown Unknowns

1. A smart contract publishes an RSA number (N = prime1 x prime2).
2. A call to its submitSolution() public function with the right prime1 and prime2 rewards the caller.
3. Alice successfully factors the RSA number and submits a solution.
4. Someone on the network sees Alice's transaction (containing the solution) waiting to be mined and submits it with a higher gas price.
5. The second transaction gets picked up first by miners due to the higher paid fee. The attacker wins the prize.

We believe more security audits or more tests would have made no difference. The main problem was that reviewers did not know what to look for.

Christoph Jentzsch

OST

Security Considerations

- Multiple Exchanges Suspend ERC20 Token Trading Due To Potential BatchOverflow Bug
  - Integer overflows in multiple contracts
  - Poloniex suspended all ERC20 token deposits and withdrawals,
  - OKEX's decision to halt ERC20 deposits

```
255   function batchTransfer(address[] _receivers, uint256 _value) public whenNotPaused returns (bool) {
256     uint cnt = _receivers.length;
257     uint256 amount = uint256(cnt) * _value;
258     require(cnt > 0 && cnt <= 20);
259     require(_value > 0 && balances[msg.sender] >= amount);
260
261     balances[msg.sender] = balances[msg.sender].sub(amount);
262     for (uint i = 0; i < cnt; i++) {
263         balances[_receivers[i]] = balances[_receivers[i]].add(_value);
264         Transfer(msg.sender, _receivers[i], _value);
265     }
266     return true;
267   }
268 }
```


OST