HS19

DISTRIBUTED SYSTEMS ADVANCED & BLOCKCHAIN

Repetition VSS II

T. Bocek

thomas.bocek@hsr.ch

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Admin

- **Mini project – challenge task, 2 group members: send to thomas.bocek@hsr.ch until 25.9.2019**
  - Work on challenge task during exercise hours and at home
  - ~~Supervisor will be here and in the office 1.169~~
  - Supervisor will be here and floating around somewhere at HSR (write me an email!)
- **3 requirements to be met**
  1. Fully decentralized game
  2. Waiting room, chat, (group of 3: random number for e.g., card game)
  3. Fully decentralized gambling (cryptocurrency)
- **To write exam all 3 requirements must be fulfilled**
- **Milestone 1: send source code to until 06.11.2019**
- **Final hand-in with presentation and demo: 11.12.2019**
- **Exercises this week**
  - Organize with your group
  - Setup infrastructure
  - Discussion/decision about what game to implement
  - Initial discussion about which technologies to use

Distributed Systems - In the *News*

- **18.09.2019: Serverless: 15% slower and 8x more expensive**
  - Highly controversial topic
  - S3, C# API with elastic beanstalk (pre-configured Linux) with CloudFront (164$)
- **AWS Lambda: deployment in 20sec**
- **Performance (same location), average time per request**
  - Serverless: 0.48324 (15% slower)
  - Beanstalk: 0.41966

Elastic Compute Cloud		**$317.57**
EU (Ireland)		**$3.30**
US West (Oregon)		**$314.27**
Amazon Elastic Compute Cloud running Linux/UNIX		$195.13
$0.020 per On Demand Linux t1.micro Instance Hour	744 Hrs	$14.88
$0.023 per On Demand Linux t2.small Instance Hour	1,492.015 Hrs	$34.32
$0.044 per On Demand Linux m1.small Instance Hour	2,183.546 Hrs	$96.08
$0.067 per On Demand Linux m3.medium Instance Hour	744 Hrs	$49.85
EBS		$10.07
$0.05 per 1 million I/O requests - US West (Oregon)	4,142,563 IOs	$0.21
$0.05 per GB-month of Magnetic provisioned storage - US West (Oregon)	24.000 GB-Mo	$1.20
$0.05 per GB-Month of snapshot data stored - US West (Oregon)	11.781 GB-Mo	$0.59
$0.10 per GB-month of General Purpose SSD (gp2) provisioned storage - US West (Oregon)	80.710 GB-Mo	$8.07
Elastic Load Balancing - Application		$51.01
$0.008 per used Application load balancer capacity unit-hour (or partial hour)	99.219 LCU-Hrs	$0.79
$0.0225 per Application LoadBalancer-hour (or partial hour)	2,232 Hrs	$50.22
Elastic Load Balancing - Classic		$58.06
$0.008 per GB Data Processed by the LoadBalancer	282.515 GB	$2.26
$0.025 per LoadBalancer-hour (or partial hour)	2,232 Hrs	$55.80

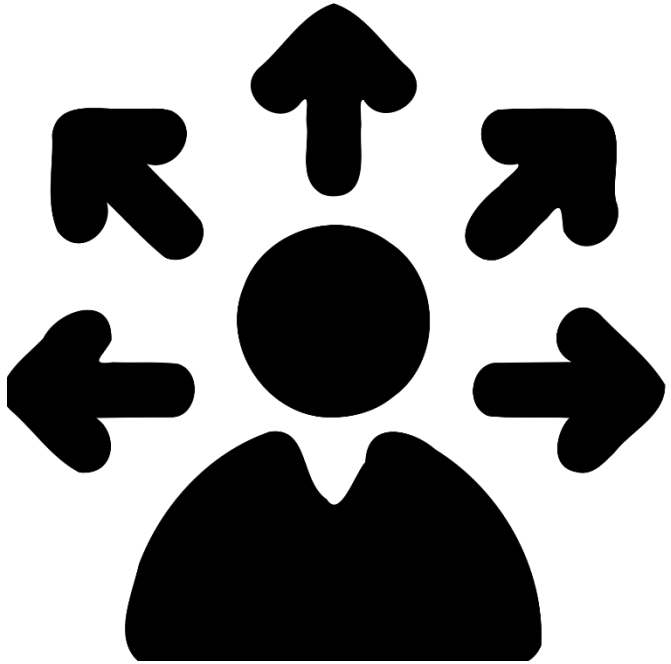
Distributed Systems - In the *News*

- **Pricing: “Pay for what you use”**
  - API gateway, price per million requests: 3.5$
  - 10m requestest/day → 35$ per day, lambda around 10$ per day → 1350$ per month
- **Serverless: know your use-case**
  - No devops, save money?
- **Discussion: https://news.ycombinator.com/item?id=21046547**

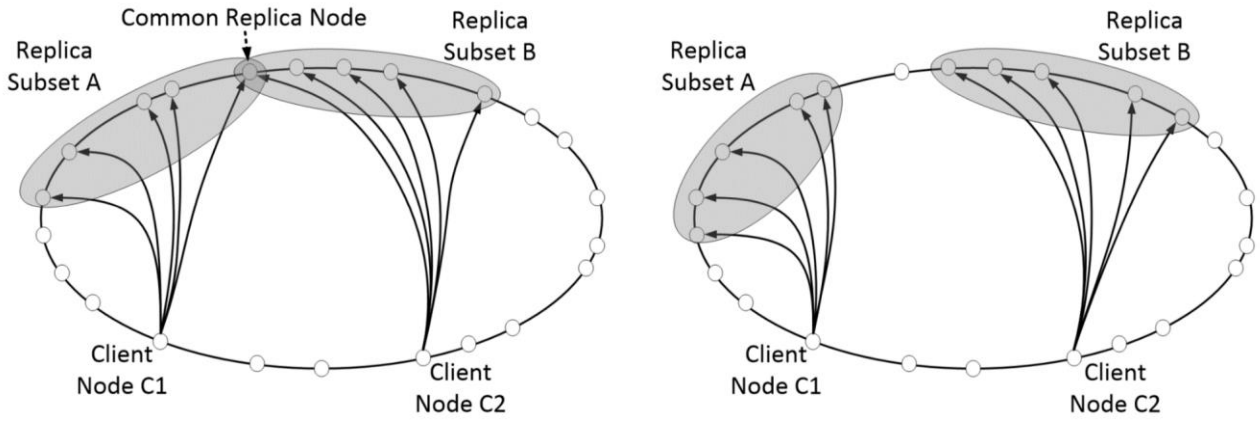
- **16.09.2019: Parliament gives green light for anti-piracy legislation**
  - Providers of pirated content over the internet will face legal consequences
  - “Critics argue that amended Swiss law falls short of punishing consumers violating intellectual property rights.”
- **25.09.2019: Bitcoin Dips Below $8K in First Since June**
  - “margin calls and contract liquidations on Bitmex”

VSS W10: Consistency

- **DHTs have weak consistency**
  - Peer A put $X_{A.1}$, Peer B gets $X_{A.1}$ modifies it puts $X_{B.2}$
  - Same time: Peer C gets $X_{A.1}$ modifies it puts $X_{C.2}$
    - Which one is stored B.2 of B or C.2 of X?
- **Consistency generic issue in distributed systems**
  - Coordinator required:
    - easy solution: centralized
    - Interesting solution: decentralized, in case failed peer, pick another peer
- **Coordinator needs to be defined**
  - Election, example Paxos



- **Consistency in DHTs – vDHT**
  - Paxos and DHTs [1], [2]
  - vDHT: CoW, versions, 2PC, replication, software transactional memory (STM) → for consistent updates. Works for light churn
    - No locking, no timestamps (replication time may have an influence)
    - Every update – new version
      1. get latest version, check if all replica peers have latest version, if not wait and try again
      2. put prepared with data and short TTL, if status is OK on all replica peers, go ahead, otherwise, remove the data and go to step 1.
      3. put confirmed, don't send the data, just remove the prepared flag
- **In case of heavy churn, API user needs to resolve**



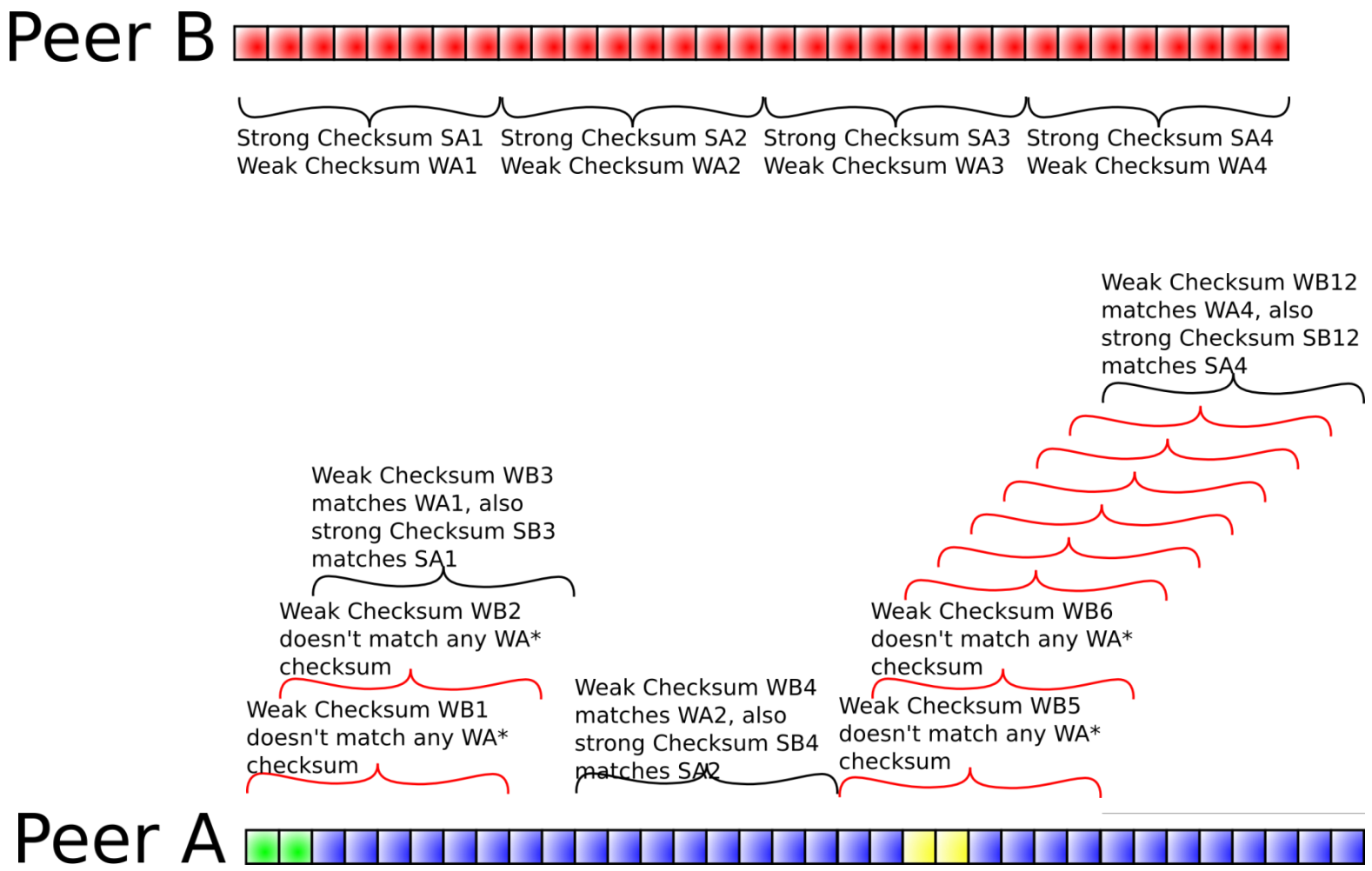
VSS W10: RPC – Application Layer

- **Simple Distributed Architecture**
  - Call a function/procedure on another machine
  - Often: wait for result (synchronous)
- **Already solved?**
  - SOAP – lots of XML
  - CORBA - heavyweight
  - DCOM, COM+ - mostly windows
  - RMI – mostly Java
  - JSON, XML – no protocol description
- **Goal**
  - Language and platform neutral way for serializing structured data for communication protocols
  - Simplicity!
- **Binary vs. Text**
  - JSON: {“account_fee”:”1234”} – 22 bytes
    - Parsing needs time
    - Human readable – easy to debug
  - Binary: 0x2, 0x1234 – e.g. length+data – 3 bytes
    - Less parsing
    - Needs a protocol, hard to debug
- **Protocol Buffers, Apache Thrift, Apache Avro**
  - Interface Description Language (IDL)
  - Binary format (performance)

URL: b.socrative.com/student/

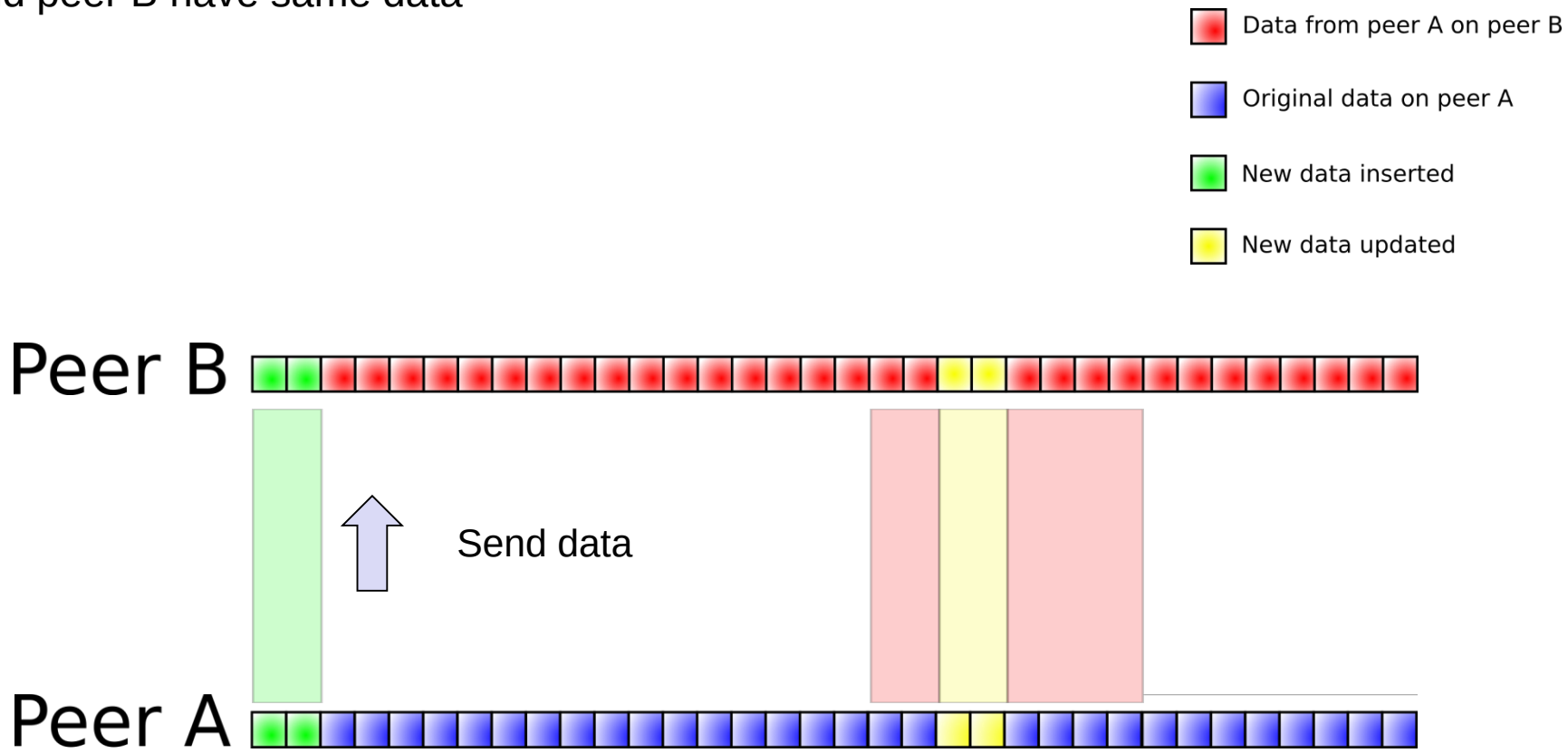
Room: HSR

VSS W11: Rsync - Example



VSS W11: Rsync - Example

- **Peer A sends 2 + 8 blocks to peer B**
  - Peer A and peer B have same data

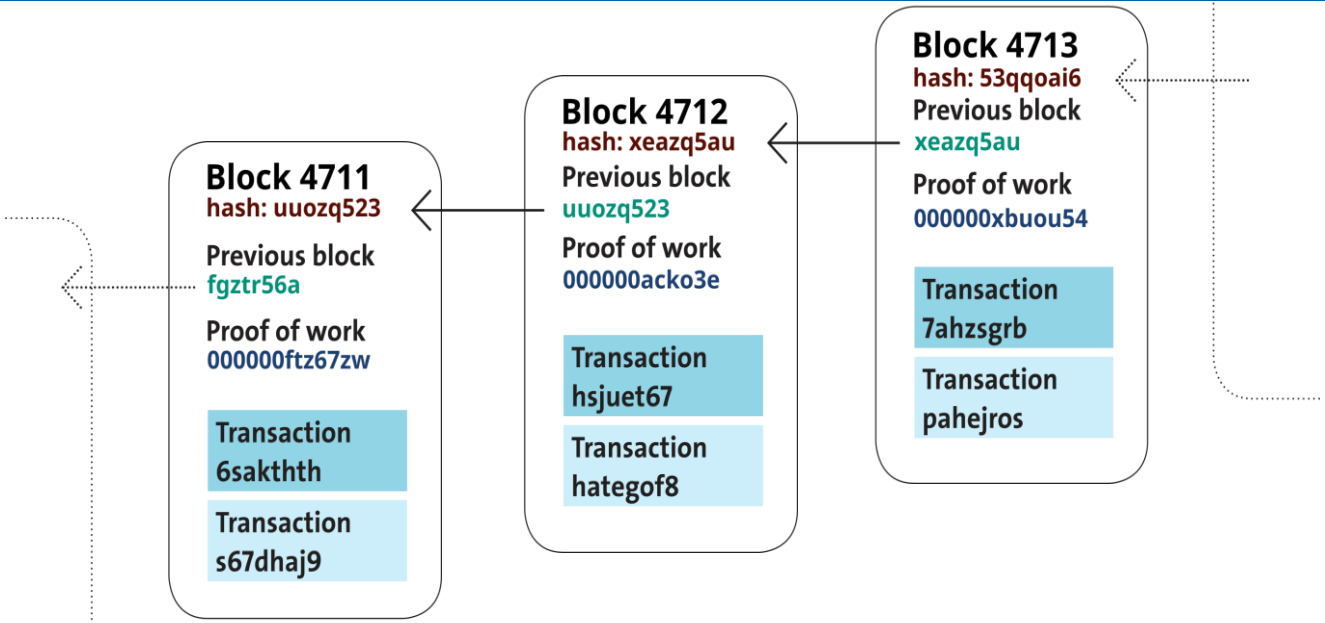


URL: b.socrative.com/student/

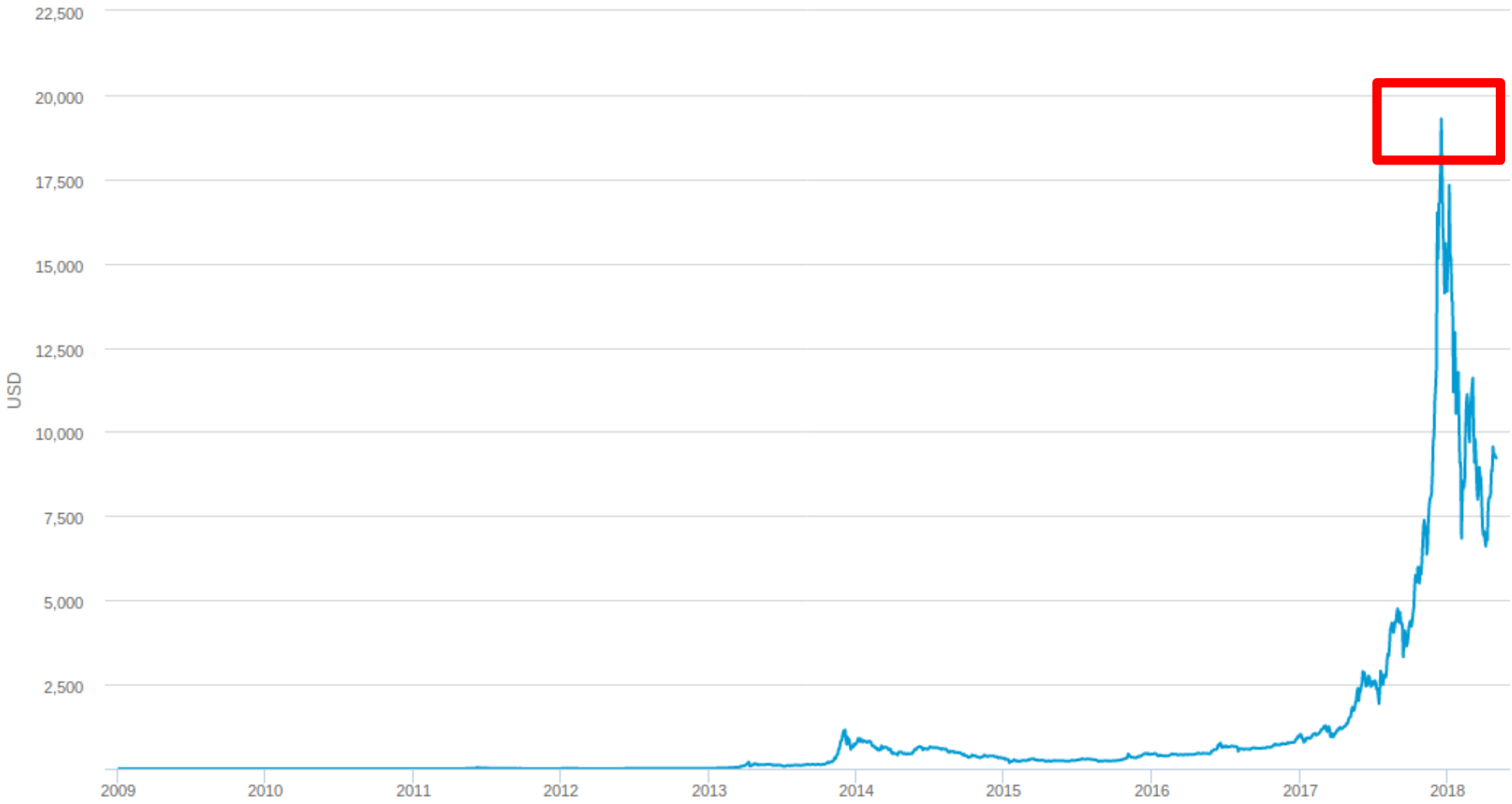
Room: HSR

VSS W12: Building Block

- **Hash function**
  - Cryptographical one-way function
  - Hash from a document - easy
  - Document from a hash is hard! (brute force)
- **Access to coins public / private keys**
  - Private key lost == funds lost
- **Transactions in blocks**
- **Create a valid block ~ 10min (Bitcoin)**
  - Find nonce that SHA256 hash has enough bits set to zero - brute force (crypto puzzle)
  - Creation of blocks is called mining (reward)
  - Waste of energy?
    - Country closest to Bitcoin in terms of electricity consumption: Austria



VSS W12: Bitcoin‘s Market Capitalization in USD



2009 ***2011*** ***2014*** ***2016*** ***2018***

VSS W12: Bitcoin - Protocol

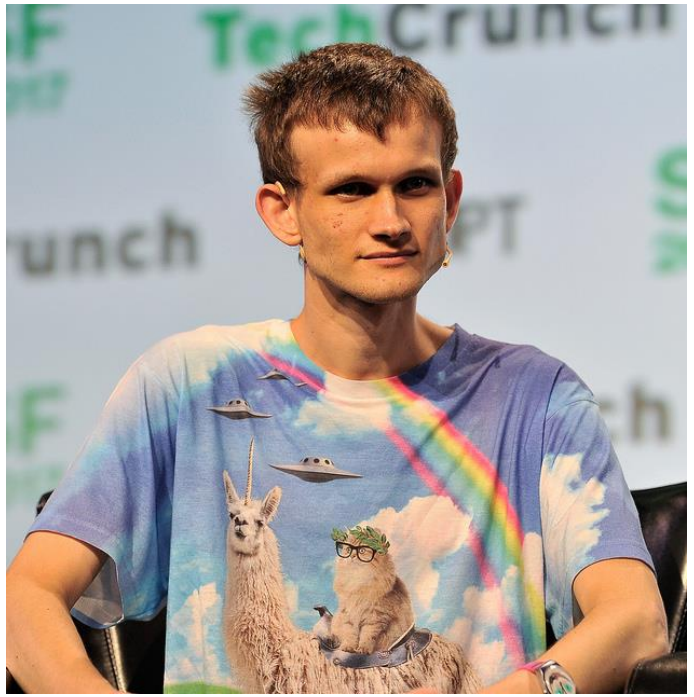
- **TX in details**

version		01 00 00 00
input count		01
input	previous output hash (reversed)	48 4d 40 d4 5b 9e a0 d6 52 fc a8 25 8a b7 ca a4 25 41 eb 52 97 58 57 f9 6f b5 0c d7 32 c8 b4 81
	previous output index	00 00 00 00
	script length	8a
	scriptSig	47 30 44 02 20 2c b2 65 bf 10 70 7b f4 93 46 c3 51 5d d3 d1 6f c4 54 61 8c 58 ec 0a 0f f4 48 a6 76 c5 4f f7 13 02 20 6c 66 24 d7 62 a1 fc ef 46 18 28 4e ad 8f 08 67 8a c0 5b 13 c8 42 35 f1 65 4e 6a d1 68 23 3e 82 01 41 04 14 e3 01 b2 32 8f 17 44 2c 0b 83 10 d7 87 bf 3d 8a 40 4c fb d0 70 4f 13 5b 6a d4 b2 d3 ee 75 13 10 f9 81 92 6e 53 a6 e8 c3 9b d7 d3 fe fd 57 6c 54 3c ce 49 3c ba c0 63 88 f2 65 1d 1a ac bf cd
	sequence	ff ff ff ff
output count		01
output	value	62 64 01 00 00 00 00 00
	script length	19
	scriptPubKey	76 a9 14 c8 e9 09 96 c7 c6 08 0e e0 62 84 60 0c 68 4e d9 04 d1 4c 5c 88 ac
block lock time		00 00 00 00

http://www.righto.com/2014/02/bitcoins-hard-way-using-raw-bitcoin.html

VSS W13: Bitcoin / Ethereum

- **“Issues” with Bitcoin**
  - Slow development, implementing new features difficult
    - SegWit vs. BTU (segregated witness vs. increasing block size voting)
  - Bitcoin Script limited
    - Lightning network
- **Ethereum (1 ETH ~330$)**
  - 1 ETH ~ 170$
  - Generalized blockchain (loops, arithemitcs, etc.)
  - «Blockchain 2.0» with smart contracts
  - White paper released in December 2013, release July 2015
  - Ethereum foundation located in Zug (initiator known) - non-profit foundation
  - Mining reward ~ block every ~14s – ~2 ETH

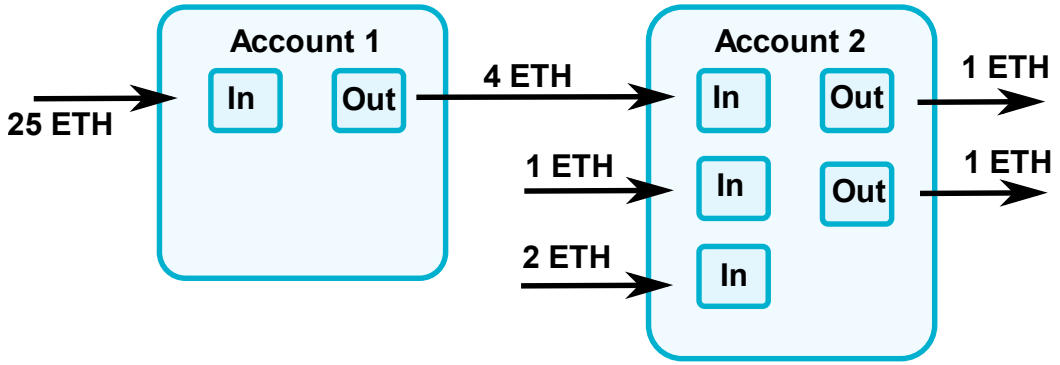


Vitalik Buterin

VSS W13: Account vs UTXO - Introduction

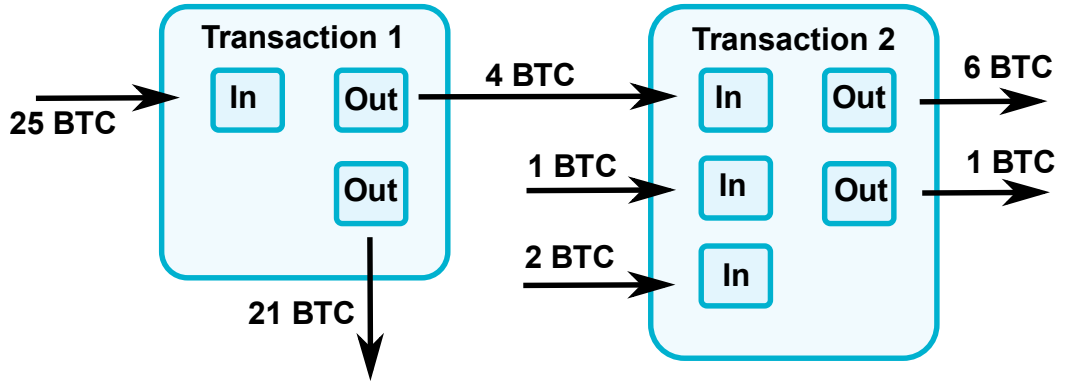
Account-based

- 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



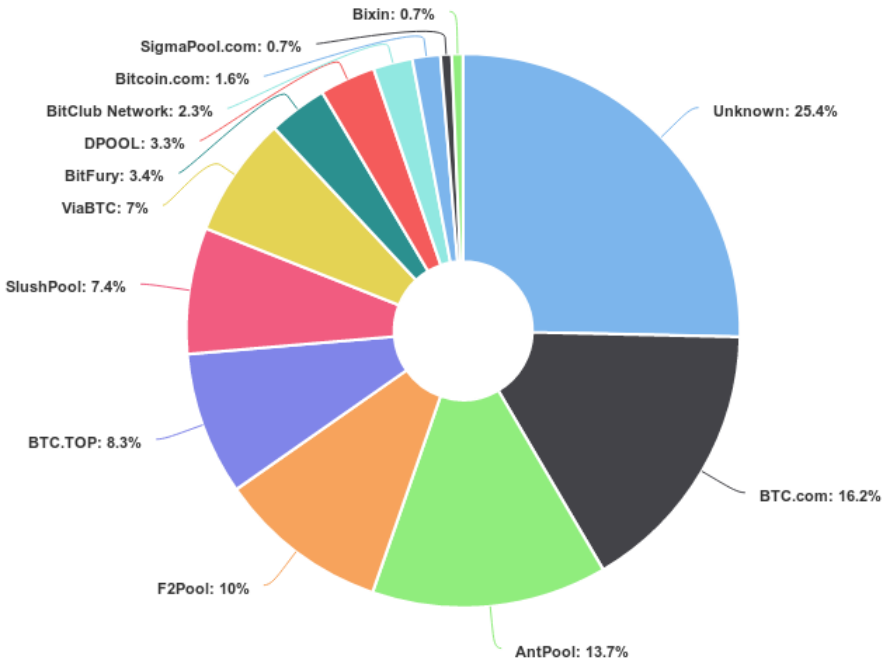
UTXO-based

- 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



51% Attack

- **Control over 50% of the scarce resources**

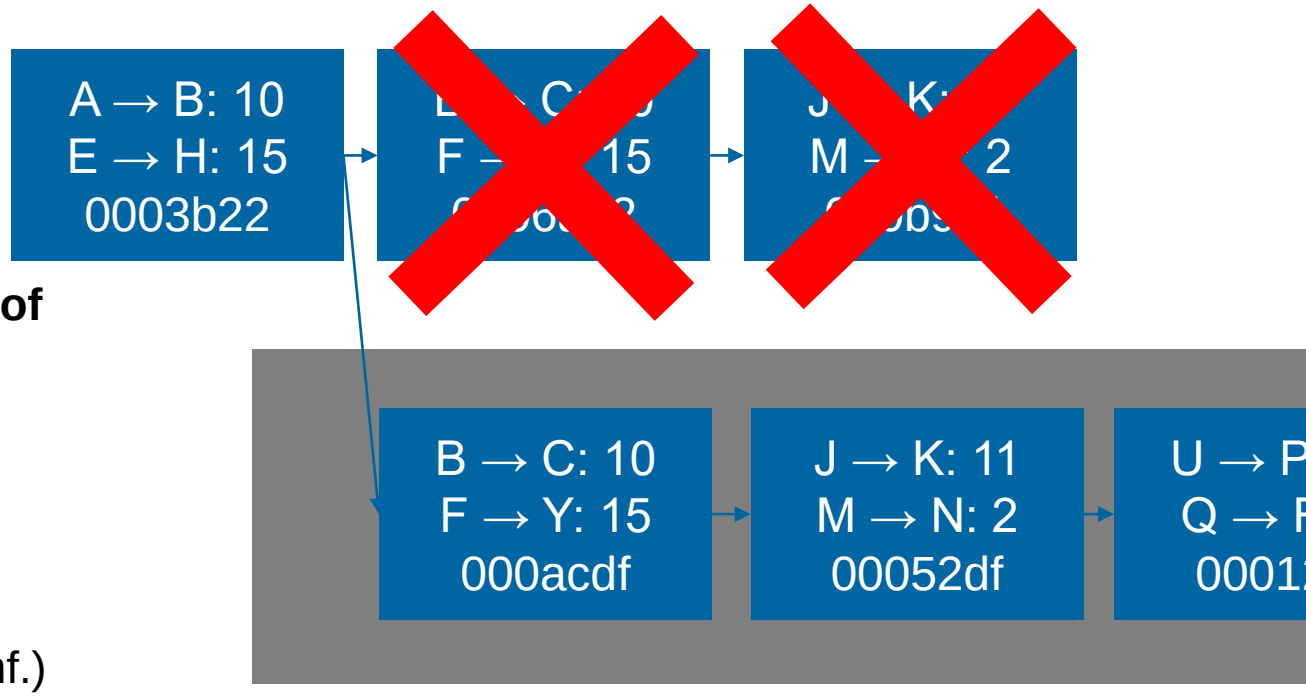


http://blockchain.info/pools

- **03.01.2019:** [BTC.TOP, Mining Pool, Controls Over 50% Of The BCH Hashrate]
  - Bitcoin Cash
- **25.06.2018:** [Bitmain's Mining Pools Now Control Nearly 51 Percent of the Bitcoin Hashrate]
  - Was at ~42%, now ~30%
- **07.01.2019:** [Deep Chain Reorganization Detected on Ethereum Classic (ETC)]
  - "The total value of the double spends that we have observed thus far is 219,500 ETC (~$1.1M)."

51% Attack in Ethereum / Bitcoin

- **“If a majority of CPU power is controlled by honest nodes, the honest chain will grow the fastest and outpace any competing chains.”**
  - https://bitcoin.org/bitcoin.pdf
- **PoW: majority of hashing power, PoS: majority of coins**
- **Double spend, or rollback transactions**
  - X is an exchange
  - Mine secretly, Y is your address
  - X arrived → miner does payout USD (2 block conf.)
  - You mine faster, broadcast secret chain
  - Tx F→X: 15 never happened, goes to Y
- **How expensive is a 51% attack? (broken!)**
  - Buy an attack?

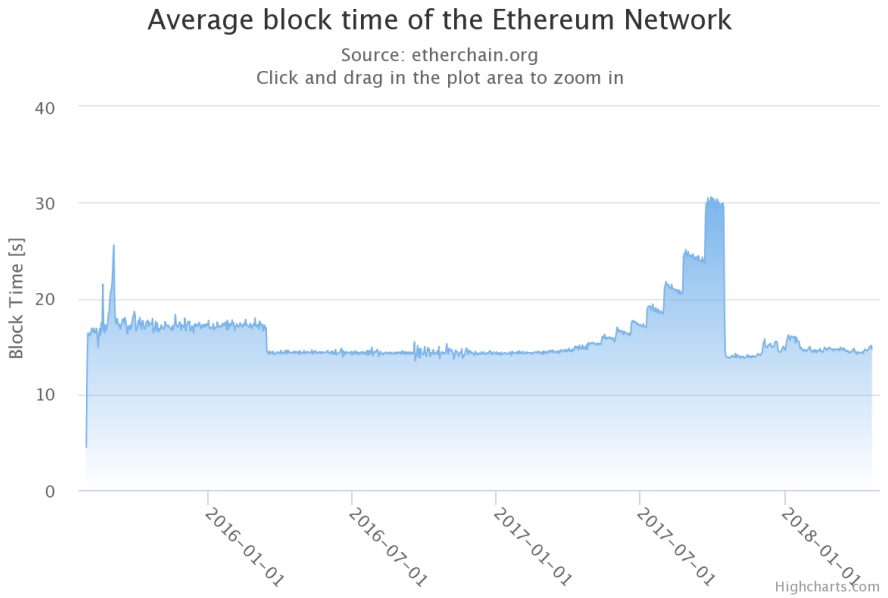


URL: b.socrative.com/student/

Room: HSR

Blocktime and Gas

- **Block time: ~14-15s**
  - But: Ice age
- **Contracts are turing complete**
  - Every instruction needs to be paid for (example)
  - In Gas, gas Limit by TX
  - If you run out of gas, state is reverted, ETH gone



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.

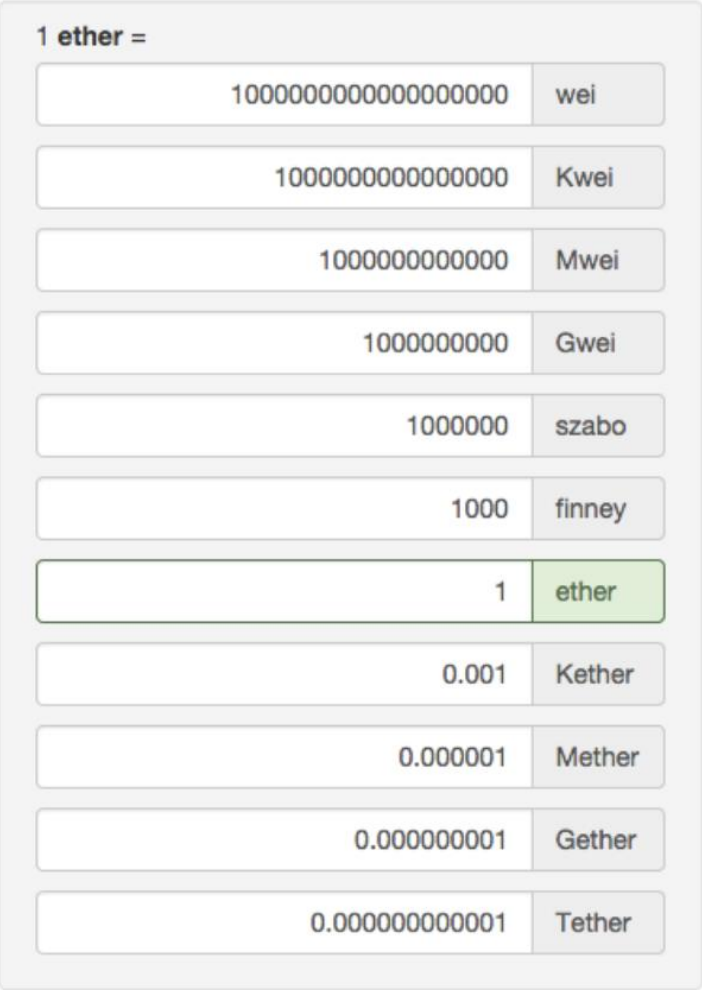
Name	Value	Description*
G_{zero}	0	Nothing paid for operations of the set W_{zero}.
G_{base}	2	Amount of gas to pay for operations of the set W_{base}.
$G_{verylow}$	3	Amount of gas to pay for operations of the set $W_{verylow}$.
G_{low}	5	Amount of gas to pay for operations of the set W_{low}.
G_{mid}	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.

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

Blocktime and Gas

- **Gas Price set by Miner**
  - Gas price ~23 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**

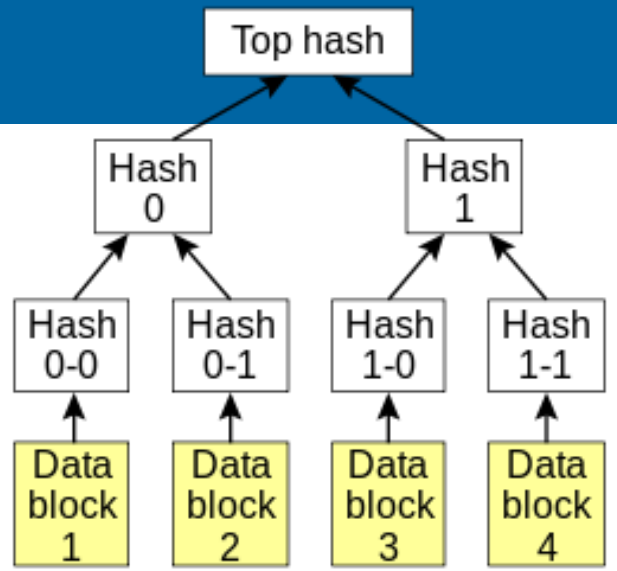
- **Units:**



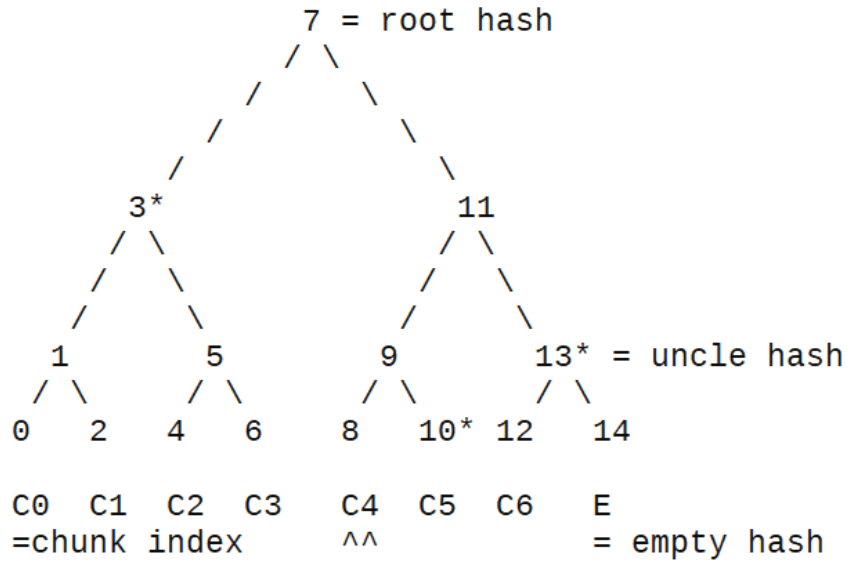
VSS W13: Merkle hash

- **Hash Tree**
  - Tree of hashes (|| → concatenation)
  - `hash 0 = hash( hash 0-0 || hash 0-1 )`
  - `hash 1 = hash( hash 1-0 || hash 1-1 )`
  - `Top hash = hash( hash 0 || hash 1 )`
- **Verification (BitTorrent)**
  - Peer A has top hash (root hash)
  - Peer downloads C4 from peer B
  - create hash 8
  - Need hash 10, 13, 3 (uncle hash)
  - Can be from peer B
  - With 8,10,13,3 can create root hash

  → verify this root hash with Merkle proof
- **Merkle tree / hash tree also used in [Bitcoin / Ethereum]**



http://en.wikipedia.org/wiki/Hash_tree



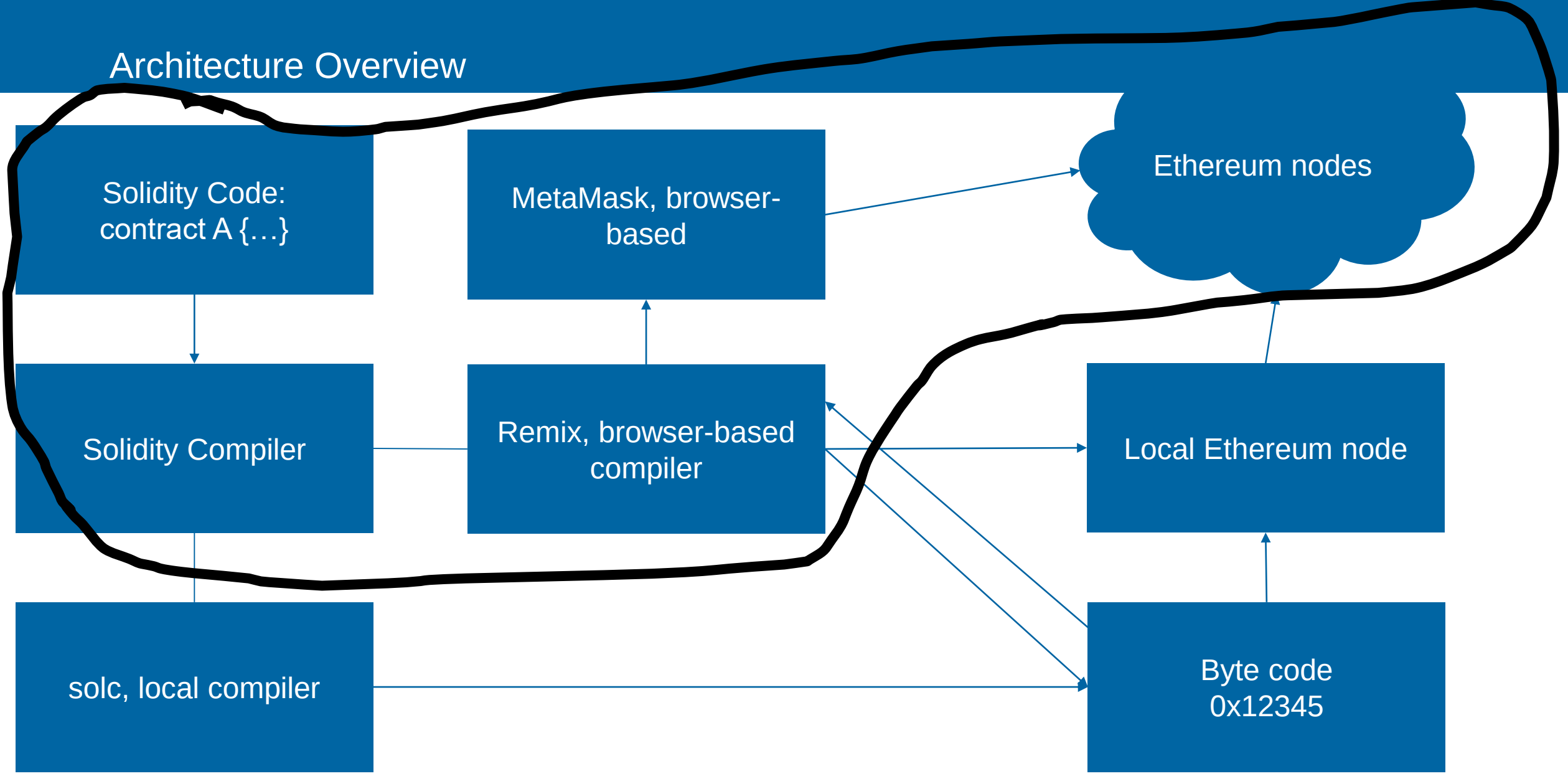
The Merkle hash tree of an interval of width W=8

http://datatracker.ietf.org/doc/draft-ietf-ppsp-peer-protocol/ Section 5.2

URL: b.socrative.com/student/

Room: HSR

Architecture Overview



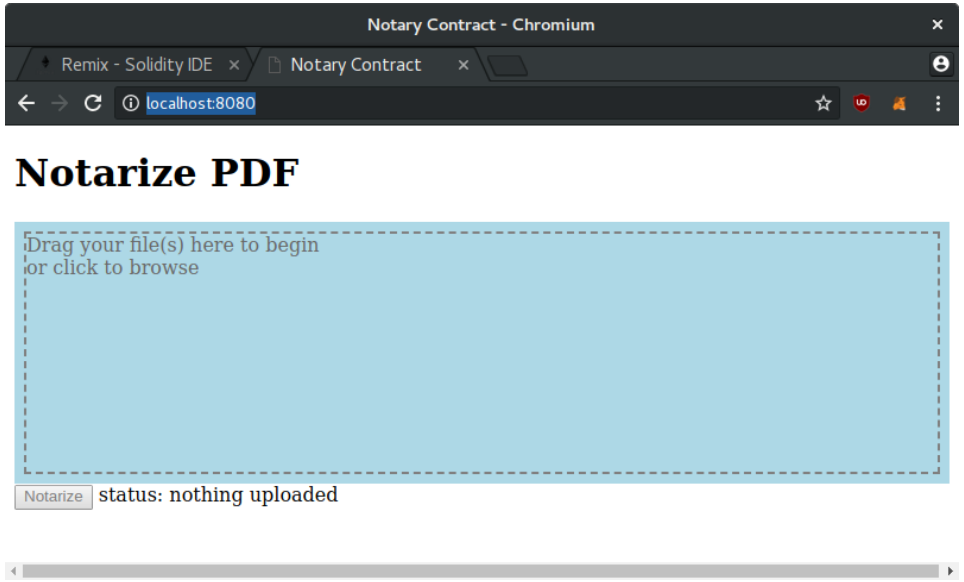
Example Solidity Contract

- **Notary Contract**
  - Simple Contract
  - No constructor
- **Frontend**
  - Vue.js, dependencies:
    - crypto-js
    - vue
    - web3
- **App.vue, main file**
  - Template
  - Create web3 instance
  - Events
    - fileChange()
    - store()

```
pragma solidity ^0.5.7;

contract Notary {

    mapping (address => mapping (bytes32 => uint256)) stamps;

    function store(bytes32 hash) public {
        stamps[msg.sender][hash] = block.timestamp;
    }

    function verify(address recipient, bytes32 hash)
                            public view returns (uint256) {
        return stamps[recipient][hash];
    }
}
```

Run it locally

- **Installation**
  - yarn install
  - ./node_modules/.bin/webpack-dev-server
- **Open Browser: http://localhost:8080/**

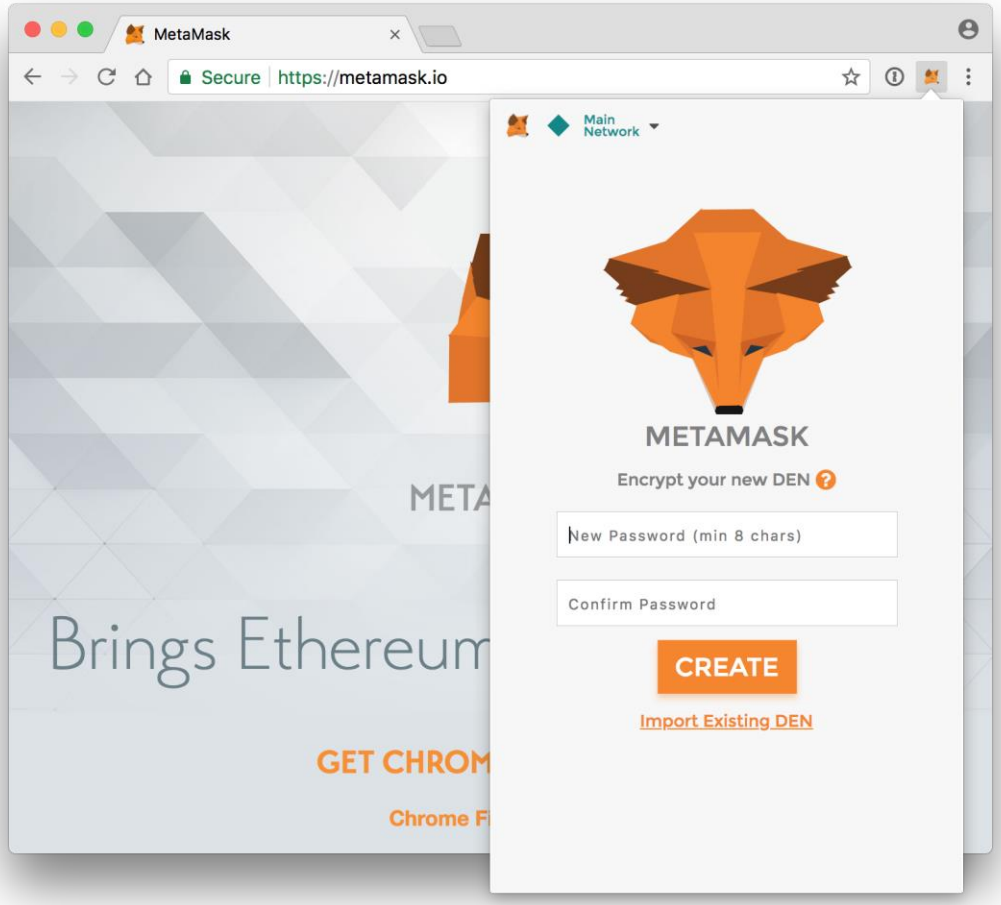



```
draft@home: ~/git/VSS-web3js
File  Edit  View  Search  Terminal  Help
draft@home:~/git/VSS-web3js$ ./node_modules/.bin/webpack-dev-server
i ｢wds｣: Project is running at http://localhost:8080/
i ｢wds｣: webpack output is served from /
i ｢wdm｣: Hash: c4c7c0d3279286de6649
Version: webpack 4.7.0
Time: 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/c
lient?http://localhost:8080 7.75 KiB {main} [built]
[./node_modules/webpack-dev-server/client/overlay.js] (webpack)-dev-server/client/overlay.js 3.5
8 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.
```

MetaMask

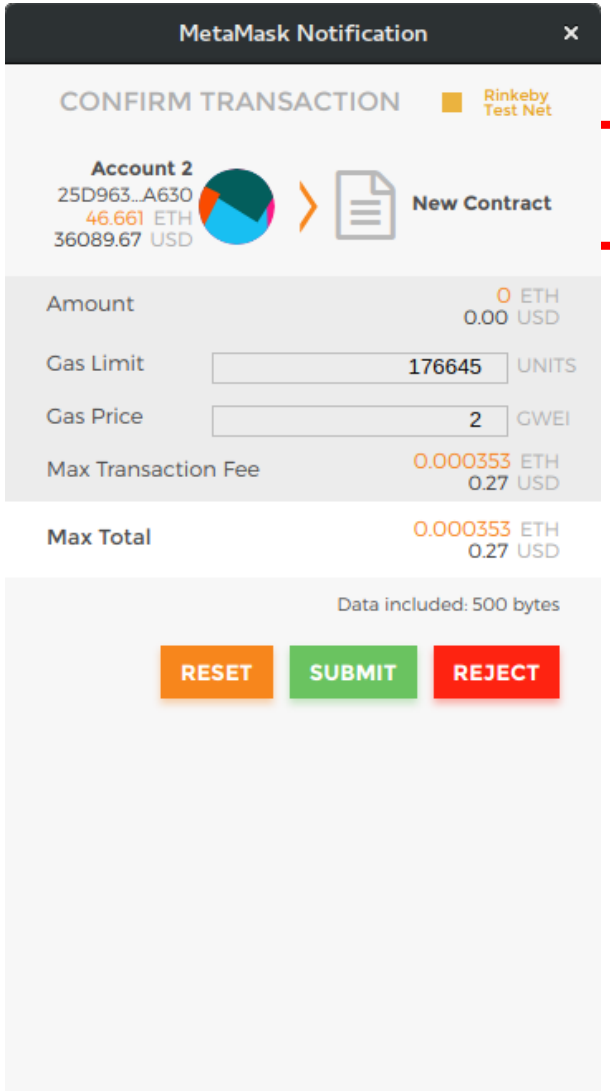
- **MetaMask**
  - Browser plugin to make Ethereum transactions in browsers
  - Manage your key pairs and sign blockchain transactions
  - MetaMask injects javascript library - web3.js
  - Uses infura

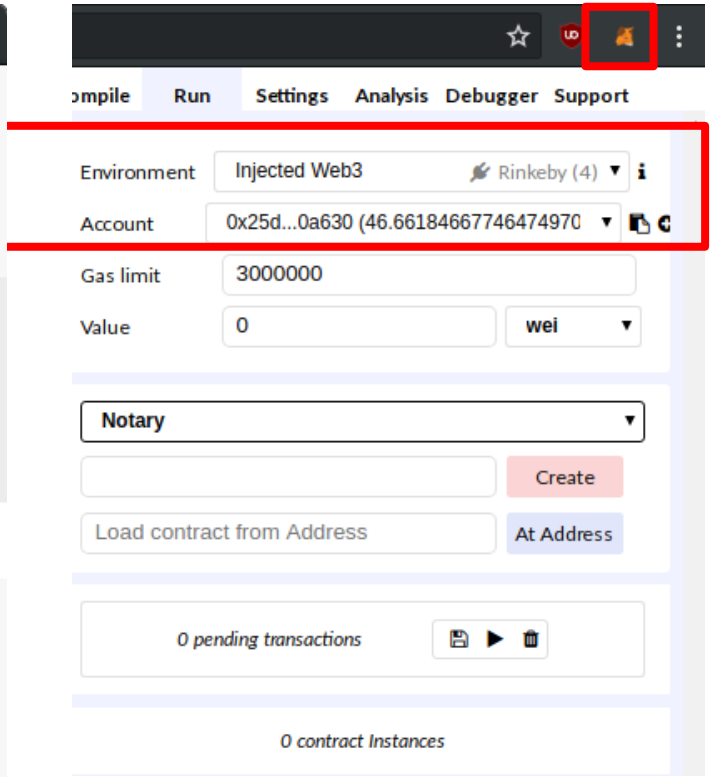
- **Remix IDE: https://remix.ethereum.org**
  - Use Notary.sol from https://github.com/tbocek/VSS-WEB3/blob/master/Notary.sol
  - Alternatively, use IntelliJ solidity plugin and deploy via geth, parity, or a local test blockchain



Create Contract

- **Connect to MetaMask**
  - Injected Web3
  - If you see your accounts, good!
- **Create contract!**
  - Add address to the code (manually)
- **Store value**
  - 0x654cf88c**4cff3e62696f7cbb55fbba922b2a75897a010347e371e9398b658c4affa611aa**
  - Hash is: 0x4cff3e62696f7cbb55fbba922b2a75897a010347e371e9398b658c4affa611aa
  - What is 0x654cf88c?
- **First four bytes of the Keccak (SHA-3) hash of the signature of the function.**
  - Keccak(store(bytes32))





Questions?

Dr. Thomas Bocek thomas.bocek@hsr.ch