FS18

BLOCKCHAIN, BITCOIN, ETHEREUM III

Ethereum Smart Contracts I

T. Bocek

thomas.bocek@hsr.ch

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Distributed Systems - In the *News*

- **11.05.2018: Korea's Biggest Crypto Exchange Raided Over Suspected Fraud**
  - Raided the largest cryptocurrency exchange in the country UPbit
  - “allegedly selling cryptocurrency to customers that it does not actually hold”
- **12.05.2018: ERC-20 Tokens, Explained**
  - Will be discussed in the next lecture
- **07.05.2018 (email): qr-pirate**
  - Scan search engines (google, bing, baidu) for specific keywords
  - Use QR-scanner to look for private keys, get balance
- **27.3.2018: The Dark Web’s Favorite Currency Is Less Untraceable Than It Seems**
  - Mixing of inputs – enforced now, but “In any mix of one real coin and a set of fake coins bundled up in a transaction, the real one is very likely to have been the most recent coin to have moved prior to that transaction.”

- **21.05.2017 Ausfall (Pfingsten)**
  - 1 Übungsgruppe am Mo fällt weg (Pfingsten)
  - 3 Übungsgruppen ohne Vorlesung am Mittwoch, Catch-up on Part II Vorlesung

- **Exercises this week**
  - Monday 13-15: Thomas Bocek
  - Wednesday 10-12: Thomas Bocek
  - Wednesday 13-15: Roman Blum
  - Wednesday 15-17: Thomas Bocek

URL: https://qfeedback.hsr.ch/q-feedback/login

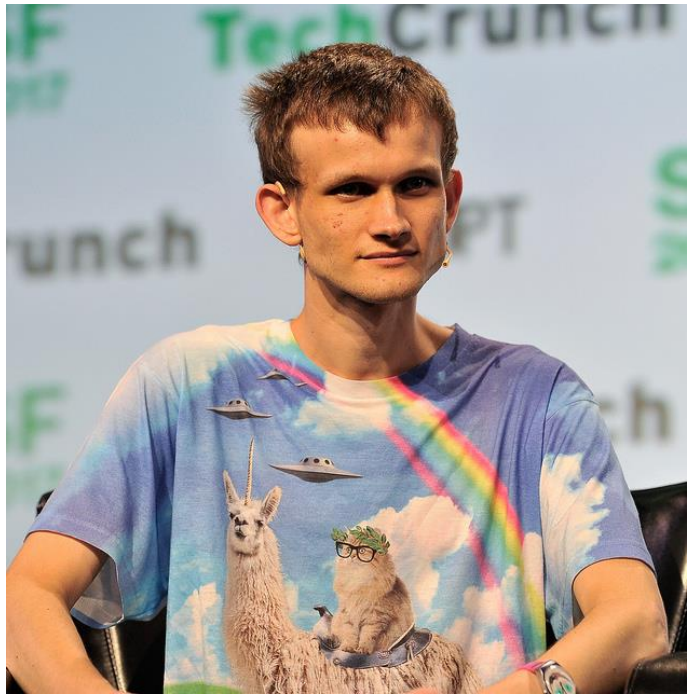
Please submit feedback, already at 22%

Agenda

- **Ethereum**
  - Introduction
  - Statistics
- **Solidity**
  - Examples

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 - beta
- **Ethereum (1 ETH ~700$)**
  - Generalized blockchain (loops, arithemitcs, etc.)
  - White paper released in December 2013
  - Protocols designed from scratch (not like LiteCoin, PeerCoin)
  - Ethereum foundation located in Zug (initiator known) - non-profit foundation
  - Mining reward ~ block every ~14s – ~3 ETH (always, unlike Bitcoin)



Vitalik Buterin

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 15.6m USD)
- **Frontier (past) - released 30.07.2015**
  - Main public network, “Beta”/use at your own risk
- **Homestead (past) - released 03.14.2016**
  - Public network considered “stable”, Integrate critical protocol changes
- **Metropolis (Byzantium, now) - released 16.10.2017**
  - Initial supports for ZKP (private smart contracts), reducing mining reward to 3 ETH
- **Metropolis (Constantinople, future)**
  - Foundations for proof-of-stake, account abstraction (define external accounts with smart contract)
- **Serenity (future)**
  - Scalability, proof-of-stake, Sharding, ZKP

URL: b.socrative.com/student/

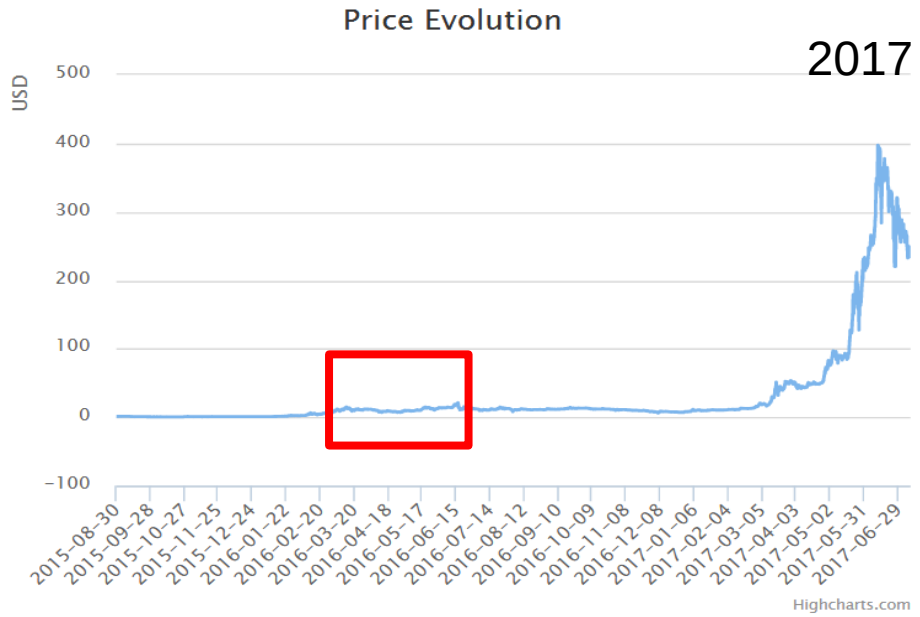
Room: HSR

Ethereum Stats

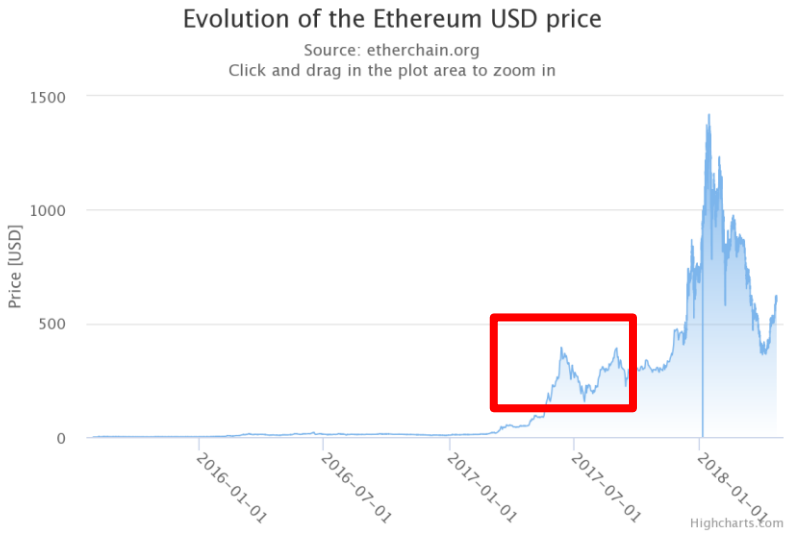
- **Basic Stats**
  - 2nd in market cap ~ 61b USD
  - Transactions (highest ~15 TPS) 700’000 per day
  - Node count (~14’000 nodes)
  - Blocksize ~25KB
  - Accounts (34mio)

2016

2017

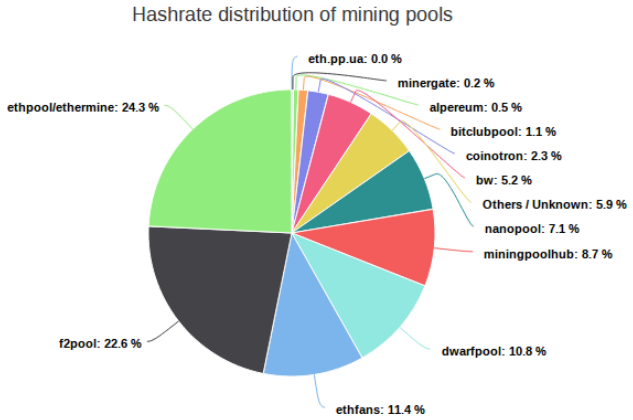


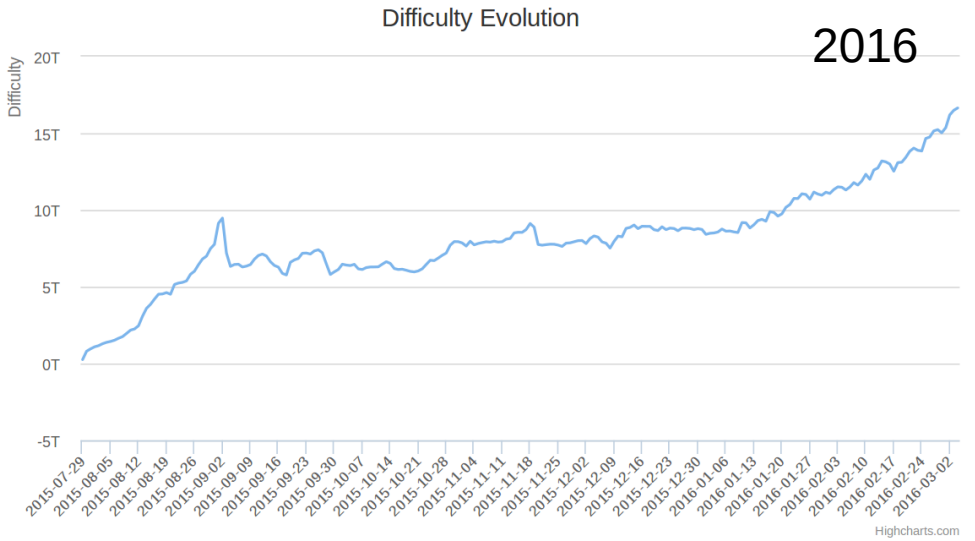
2018

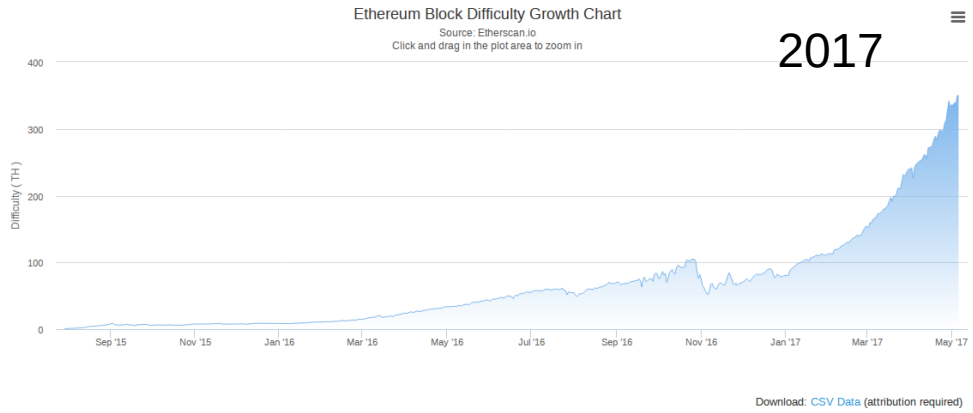


Ethereum Stats

- **98mio ETH supply (stats)**
- **Increasing difficulty**
  - Mining in pools
  - «ethpool» ~25%

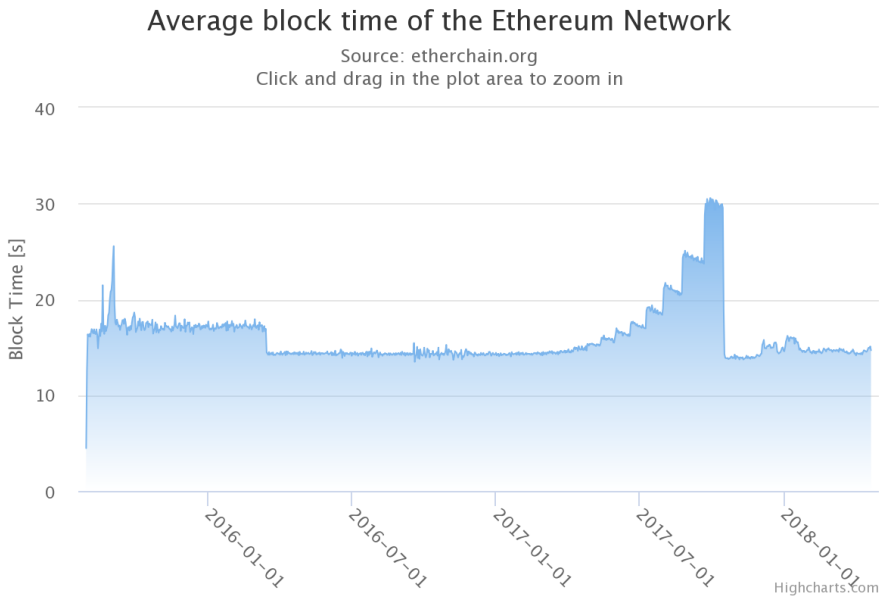






Blocktime and Gas

- **Block time: ~14-15s**
  - Ice age
- **Contracts are turing complete**
  - Every instruction needs to be paid for (example)
  - Gas price ~16 gwei
- **Gas Price / 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.

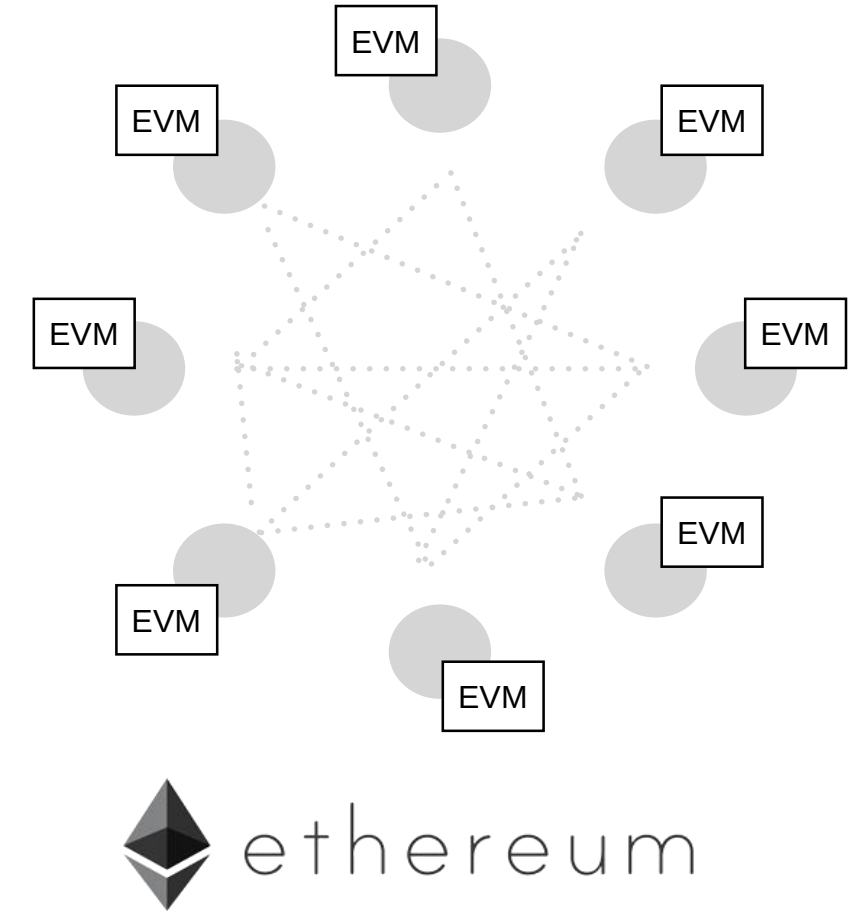
$W_{zero} = \{\text{STOP, RETURN}\}$
$W_{base} = \{\text{ADDRESS, ORIGIN, CALLER, CALLVALUE, CALLDATASIZE, CODESIZE, GASPRICE, COINBASE, TIMESTAMP, NUMBER, DIFFICULTY, GASLIMIT, POP, PC, MSIZE, GAS}\}$
$W_{verylow} = \{\text{ADD, SUB, NOT, LT, GT, SLT, SGT, EQ, ISZERO, AND, OR, XOR, BYTE, CALLDATALOAD,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

URL: b.socrative.com/student/

Room: HSR

Ethereum smart contract

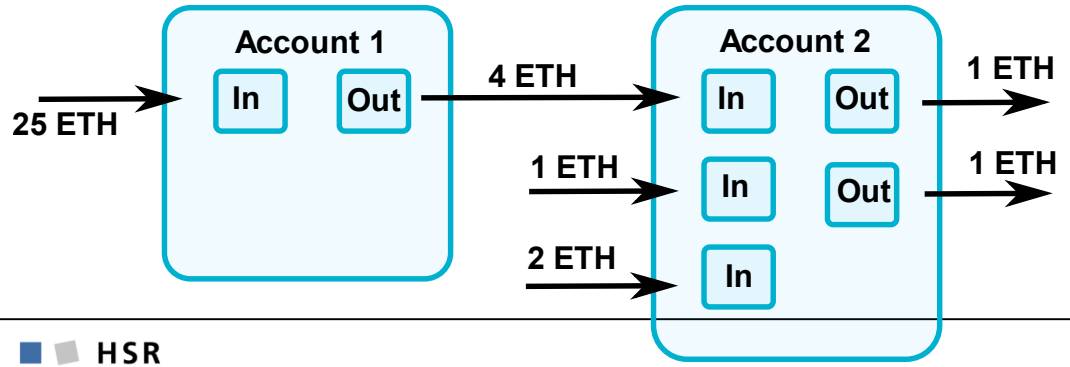
- **Computation on EVM is "very expensive“: every contract is run on every full Ethereum node**
  - Result on every node is the same
  - Global computer, always running, always correct
- **For now, proof of work (PoW)**
  - But difficult with ASIC (memory-hard), starts at 1GB RAM for full node and miner (2GB RAM should be enough)
- **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



Account vs UTXO - Introduction

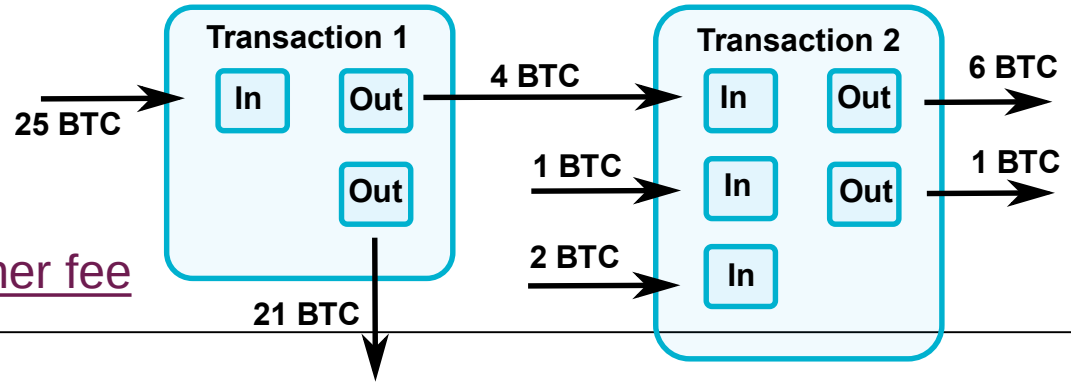
Account-based

- 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



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)

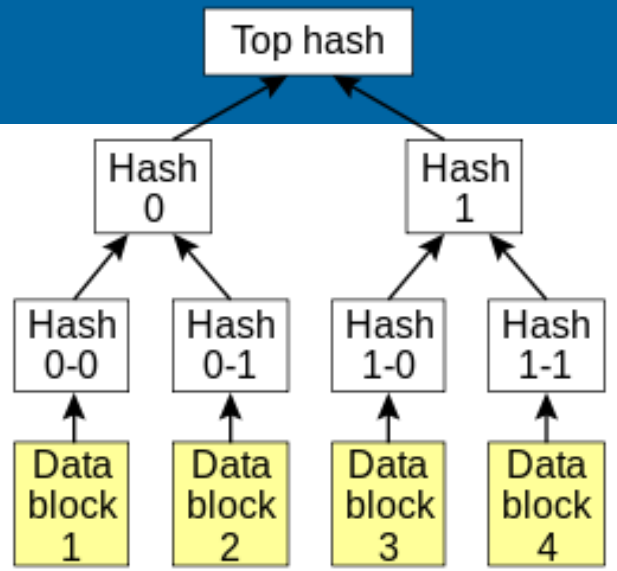
UTXO-based

- 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

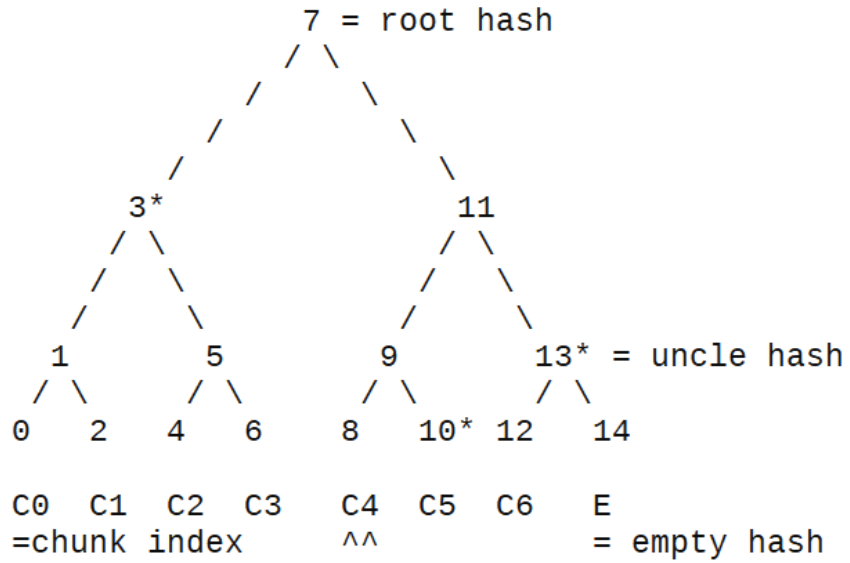
Merkle hash

- **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
- **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

Solidity

Solidity -

- **Version Pragma - reject compiled with specific compiler versions**

```
pragma solidity ^0.4.23; //not before 0.4.23, not after 0.5.0
```

- **Comments**

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

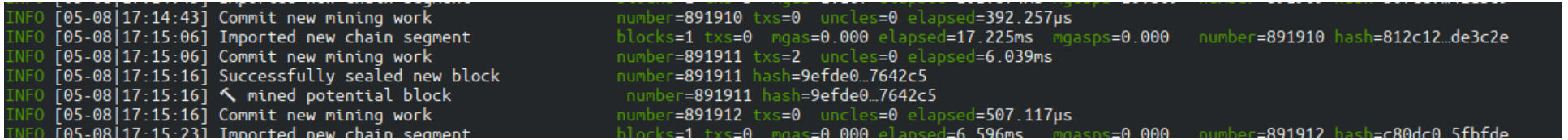
- **Contract with State Variables (expensive!)**

```
pragma solidity ^0.4.23;
//minimal contract
contract Example1 {
    uint counter;
}
```

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

Solidity Deployment (with a full client)

- **Create address (geth) – or parity, or MEW, or…**
  - Ethereum address = create random private key (256 bits) → derive public key (2x256bits) → hash it, take last 20bytesgeth --rinkeby account new
- **Important sites**
  - https://rinkeby.etherscan.io/ (blockchain explorer)



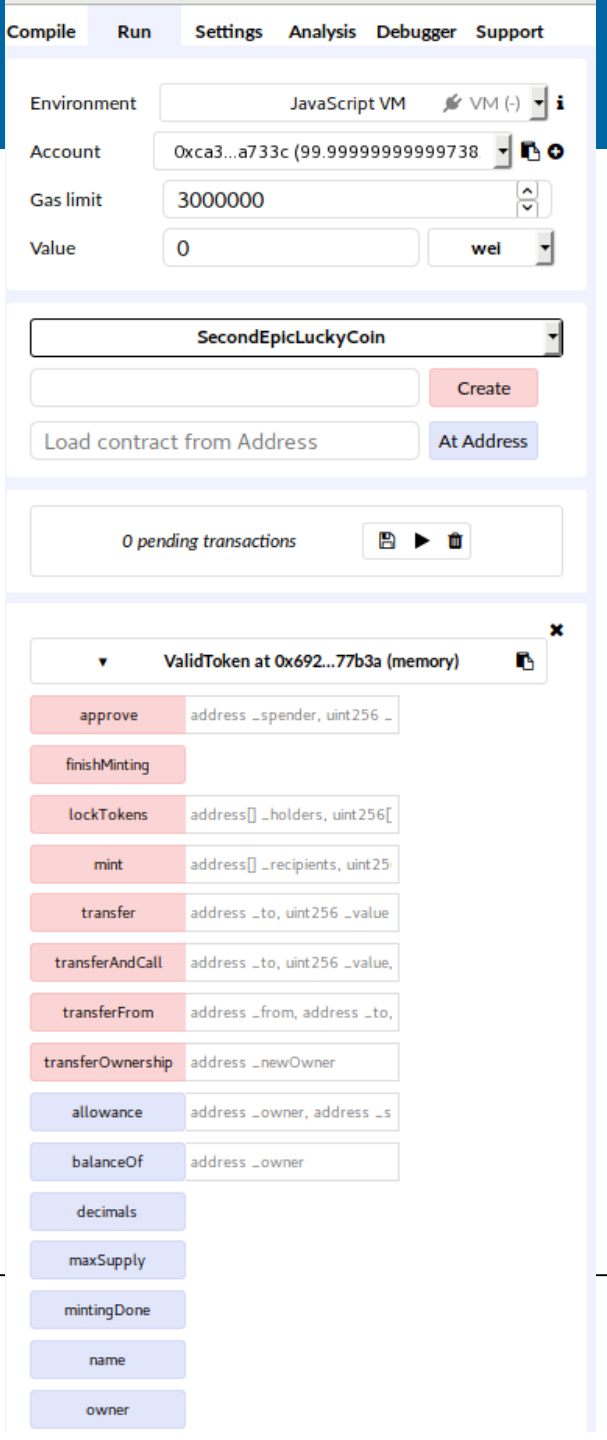
  - No mining (use github with https://www.rinkeby.io)
- **Start geth**
  - geth --rinkeby --rpc --rpccorsdomain '*' --rpcapi "eth,net,web3,personal" --unlock 0 --password <(echo 123456)
  - geth --rinkeby attach http://127.0.0.1:8545 or
  - Connect Remix Solidity IDE to geth

Solidity IDE

- **Create your first contract**

```
pragma solidity ^0.4.23;
//minimal contract
contract Example1 {
    uint counter;
}
```

- **Remix – Solidity IDE – Environment: Web3 Provider**
  - http://localhost:8545
- **Push «Create» (red button)**
- **Mixed content - workaround: use http**
  - **https://remix.ethereum.org**
- **Select Web3 Provider (geth is your Web3 Provider)**
  - Alternatively: use scripts

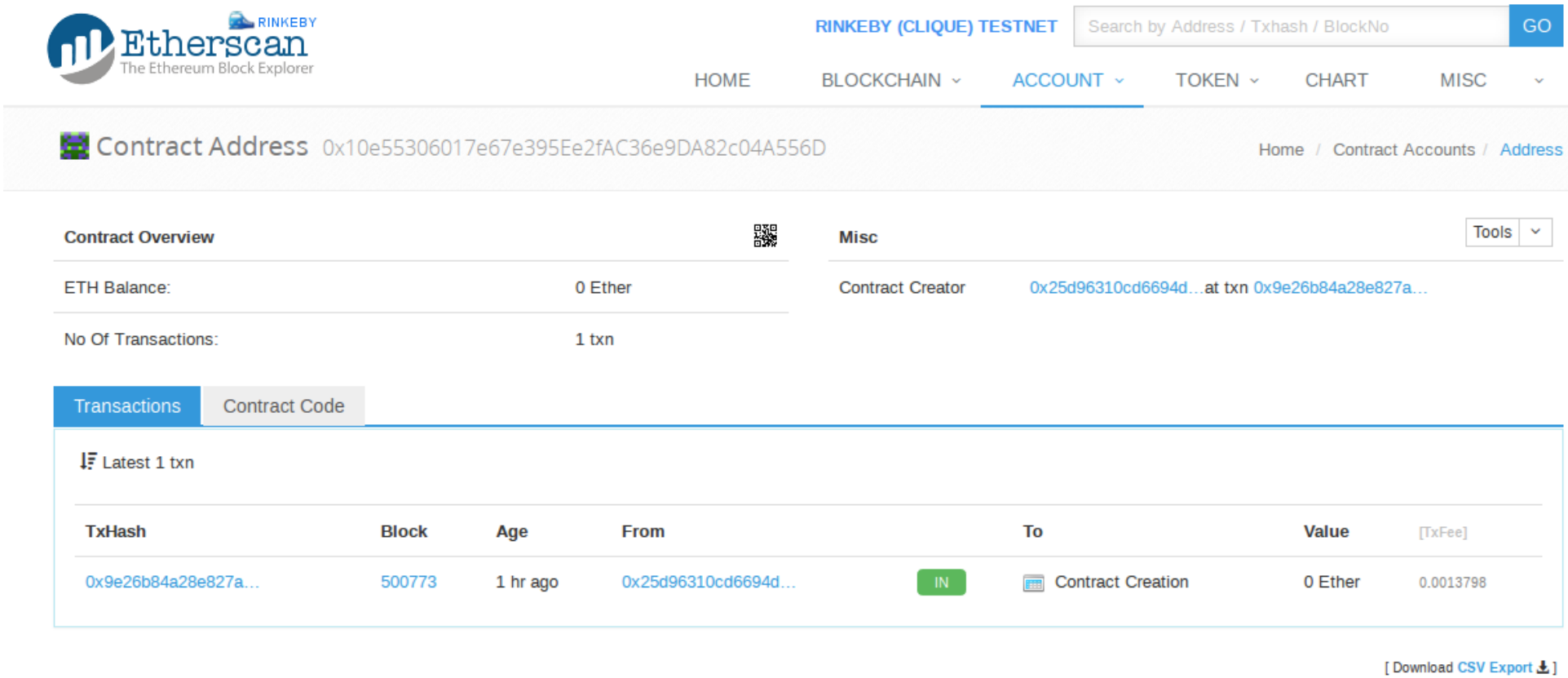


URL: b.socrative.com/student/

Room: HSR

Check Deployment

- **Etherscan.io Contract mined!**



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 addr;
  uint amount;
}
```

- **Mapping**

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

Solidity – Basics

- **Arrays**

```
bytes32[] names
uint newLength = names.push("John");
```

- **Another contract**

```
pragma solidity ^0.4.22;
contract Example2 {
    struct Account {
            string addr;
            uint amount;
    }
    uint public counter;
    mapping (uint => Account) accounts;
}
```

- **Create getter/setter**

```
function get(uint nr) public constant returns (string) {
    return accounts[nr].addr;
}
function set(uint nr, string addr) public {
    require(owner == msg.sender);
    accounts[counter++] = Account(addr, nr);
}
```

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

Solidity – ownership

- **Preconditions**
  - Special Variables and Functions

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

- **Set owner – old syntax**

```
pragma solidity ^0.4.10;
contract Example3 {
    address owner;
    function Example3() {
        owner = msg.sender;
    }
}
```

- **Constructor – new syntax**

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

- **Events – notifications**

```
pragma solidity ^0.4.22;
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)});
```

URL: b.socrative.com/student/

Room: HSR

Questions?

Prof. Dr. Thomas Bocek
thomas.bocek@hsr.ch

Upvote this lecture ☺

1MGDdtoJHQkDEmPwWtwEQxhtRvJfMMAz83