

Blockchains

**Prof. Dr. Burkhard Stiller¹, Bruno Rodrigues¹, Eder Scheid¹,
Prof. Dr. Thomas Bocek²**

*¹Communication Systems Group CSG, Department of Informatics IfI
University of Zürich UZH*

²Informatik, HSR Hochschule für Technik Rapperswil



**Universität
Zürich^{UZH}**

*[stiller|rodrigues|scheid]@ifi.uzh.ch
thomas.bocek@hsr.ch*



Outline Day 2 – Blockchain Platforms and Architectures

Part V: Ethereum

- 11. Basics
- 12. Mechanism and Protocol
- 13. Smart Contracts

Part VI: The Blockchain World

- 14. Other Blockchains
- 15. Blockchain Interoperability

Lunch Break

Part VII: Practical Exercises

- 16. Ethereum Wallet
- 17. Smart Contract

Part VIII: Blockchain Evaluations (Technical View, mainly)

- 18. To Blockchain or not to Blockchain
- 19. Comparisons and Assessment
- 20. Challenges and Risks
- 21. Expected Impacts and Conclusions

Day 2: Blockchain Platforms and Architectures

Part V: Ethereum

T. Bocek
thomas.bocek@hsr.ch

Ethereum Basics

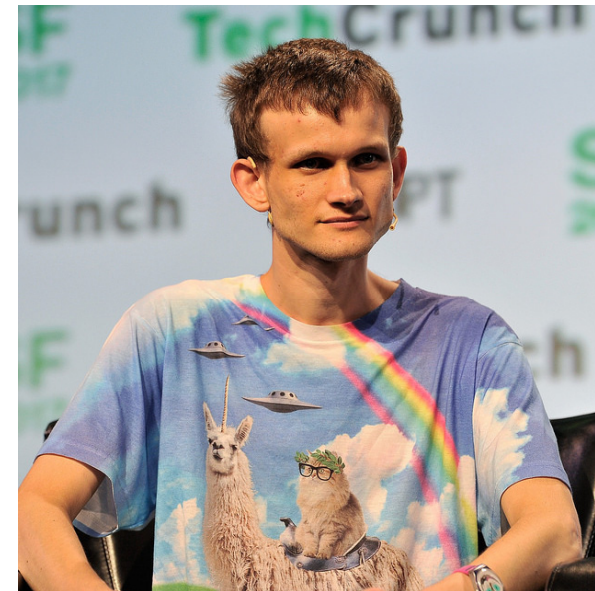
Bitcoin / Ethereum

■ “Issues” with Bitcoin

- Slow development, implementing new features difficult
 - [SegWit vs. BTU](#) (segregated witness vs. increasing block size voting) Cash vs. SV
- Bitcoin Script limited
 - But: Lightning network – [growing](#)

■ Ethereum ([1 ETH ~160\\$](#))

- Generalized blockchain (loops, arithemits, 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 – ~2 ETH (“always”, unlike Bitcoin’s [~21m BTC limit](#))



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 3.5m 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](#), past) - released 16.10.2017**
 - Initial supports for ZKP (private smart contracts), reducing mining reward to 3 ETH
- **Metropolis (Constantinople, now) – released 27.2.2019**
 - Bitshifting, mining reward 2 ETH (delay difficulty bomb), [other optimizations](#)
- **Serenity (future)**
 - Scalability, proof-of-stake, Sharding, ZKP



Ethereum Statistics

■ Basic Stats

- 2nd in [market cap](#) ~ 16b USD
- Transactions (highest ~15.5 TPS)
now ~[630'000 per day](#)
- [Node count](#) (~11'000 nodes)
- [Blocksize](#) ~20KB
- [Accounts](#) (58mio)

Evolution of the Ethereum USD price

Source: etherchain.org

Click and drag in the plot area to zoom in



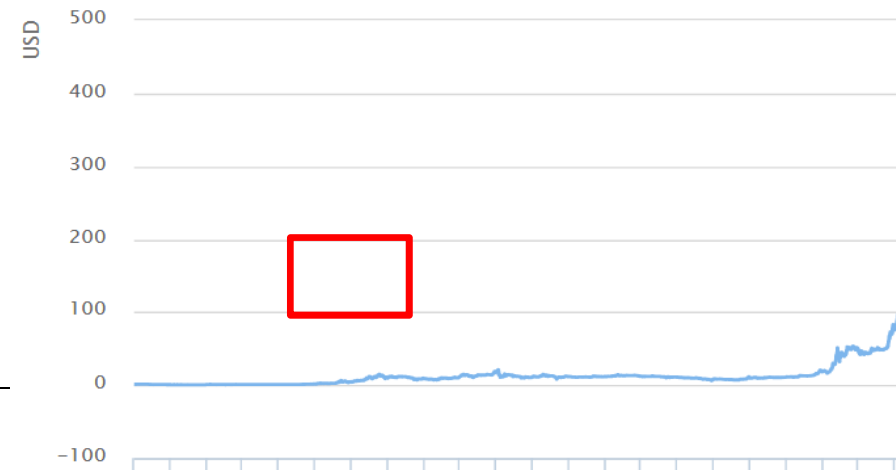
Price Evolution

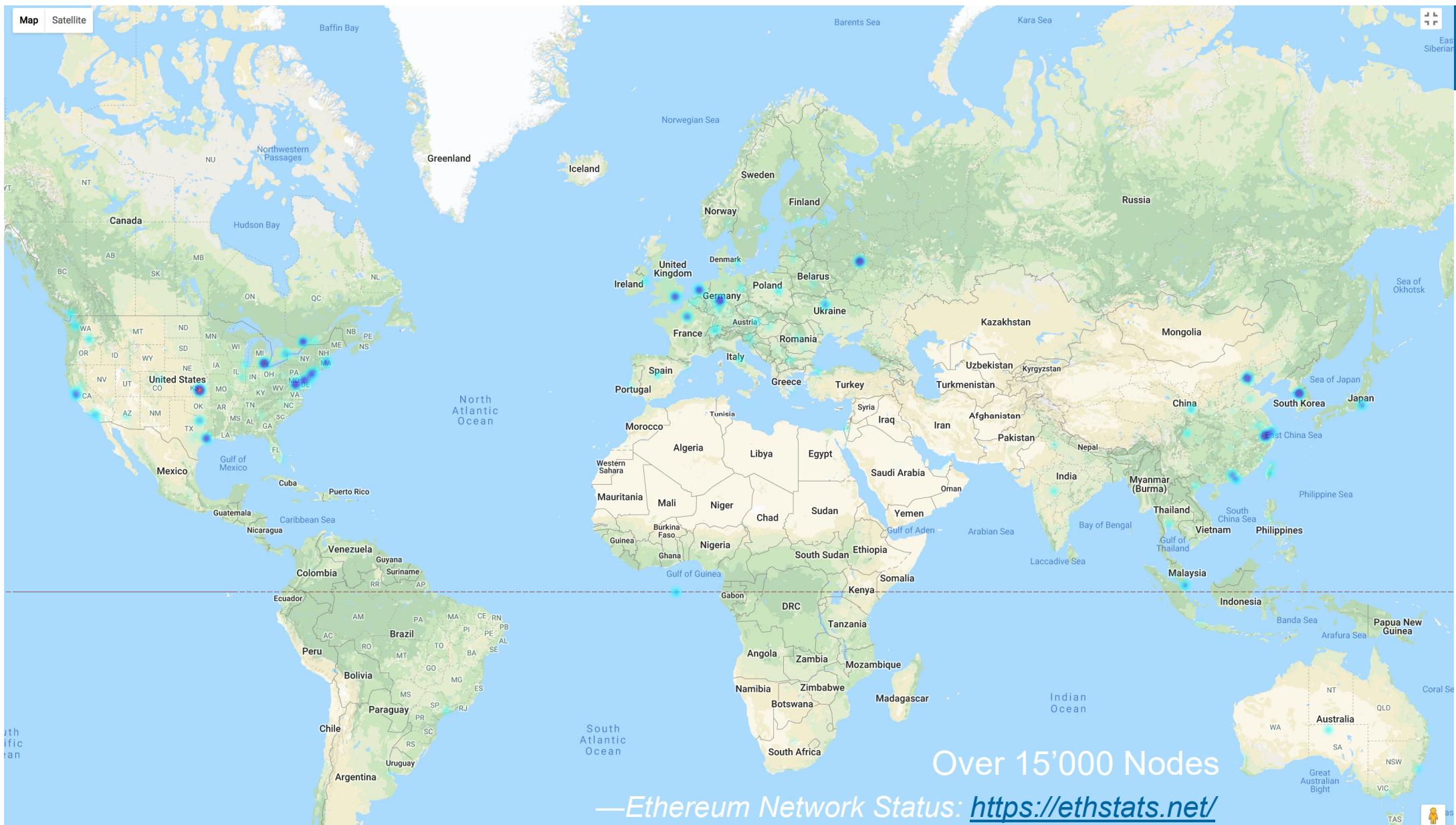
2016

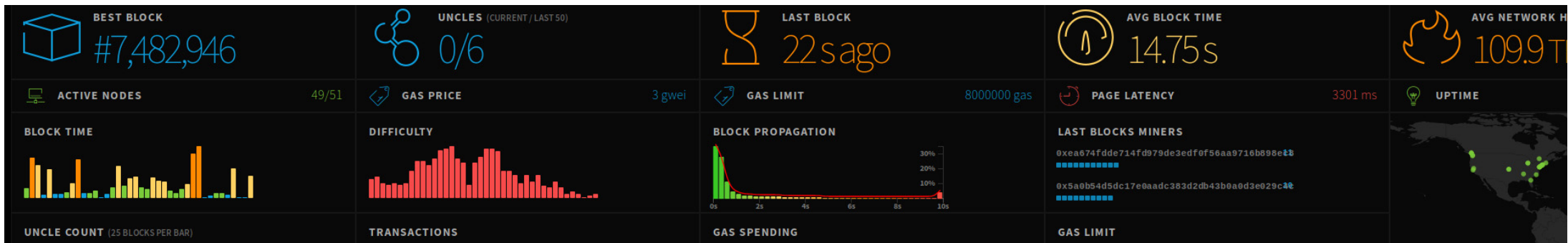


Price Evolution

2017



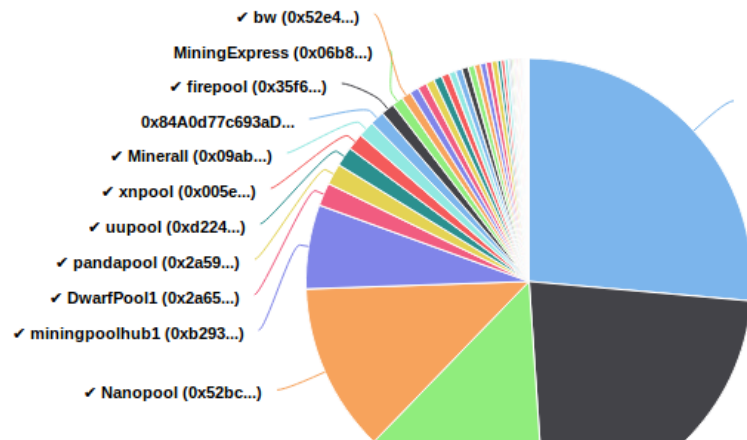




■ 105mio ETH supply ([stats](#))

■ Increasing difficulty

- [Mining in pools](#)
- Ethermine + Sparkpool 48.8%



Ethereum Block Difficulty Growth Chart
Source: Etherscan.io
Click and drag in the plot area to zoom in

2017



Ethereum Block Difficulty Growth Chart
Source: Etherscan.io
Click and drag in the plot area to zoom in

2019



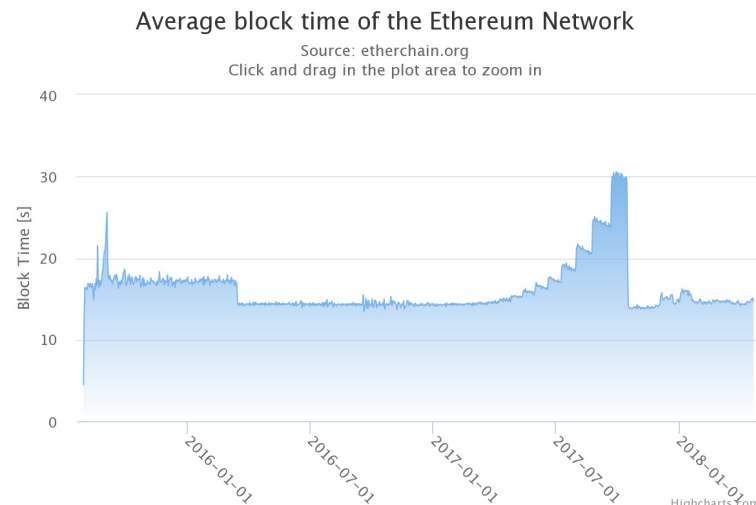
Blocktime and Gas

■ Block time: ~14-15s

- But: [Ice age](#)

■ Contracts are turing complete

- Every instruction needs to be paid for ([example](#))
- [In Gas](#)



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 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.
R_{sclear}	15000	Refund given (added into refund counter) when the storage value is set to 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.

$W_{zero} = \{\text{STOP, RETURN}\}$

$W_{base} = \{\text{ADDRESS, ORIGIN, CALLER, CALLVALUE, CALLDATASIZE, CODESIZE, GAS, TIMESTAMP, NUMBER, DIFFICULTY, GASLIMIT, POP, PC, MSIZE, GAS}\}$

$W_{verylow} = \{\text{ADD, SUB, NOT, LT, GT, SLT, SGT, EQ, ISZERO, AND, OR, XOR, BYTE, CALLCODE, MLOAD, MSTORE, MSTORE8, PUSH*, DUP*, SWAP*}\}$

$W_{low} = \{\text{MUL, DIV, SDIV, MOD, SMOD, SIGNEXTEND}\}$

$W_{mid} = \{\text{ADDMOD, MULMOD, JUMP}\}$

$W_{high} = \{\text{JUMPI}\}$

Blocktime and Gas

■ Gas Price / Gas Limit by TX

- If you run out of gas, state is reverted, ETH gone
- Gas price ~ 2 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:

1 ether =	
1000000000000000000	w
1000000000000000000	K
1000000000000000000	M
1000000000000000000	G
1000000000000000000	sz
1000	fir
1	et
0.001	K
0.000001	M

Ethereum Mechanisms and Protocols

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>

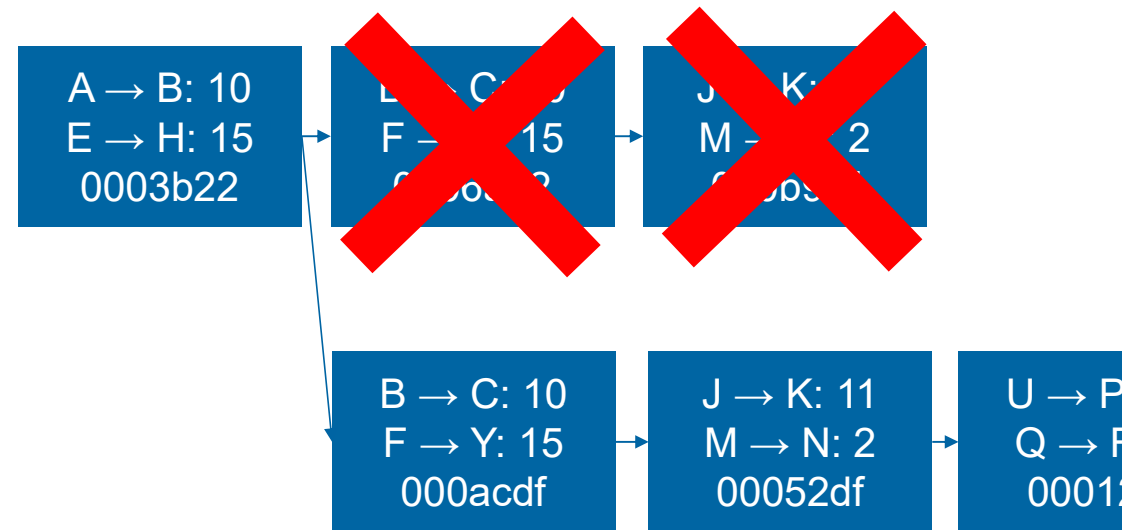
- PoW: majority of hashing power, PoS: majority of coins

- Double spend, or rollback transactions

- X is an exchange
- Mine secretly, Y is your address
- X arrived – payout (1 block conf.)
- You mine faster, broadcast secret chain
- Tx F→X: 15 never happened, goes to Y

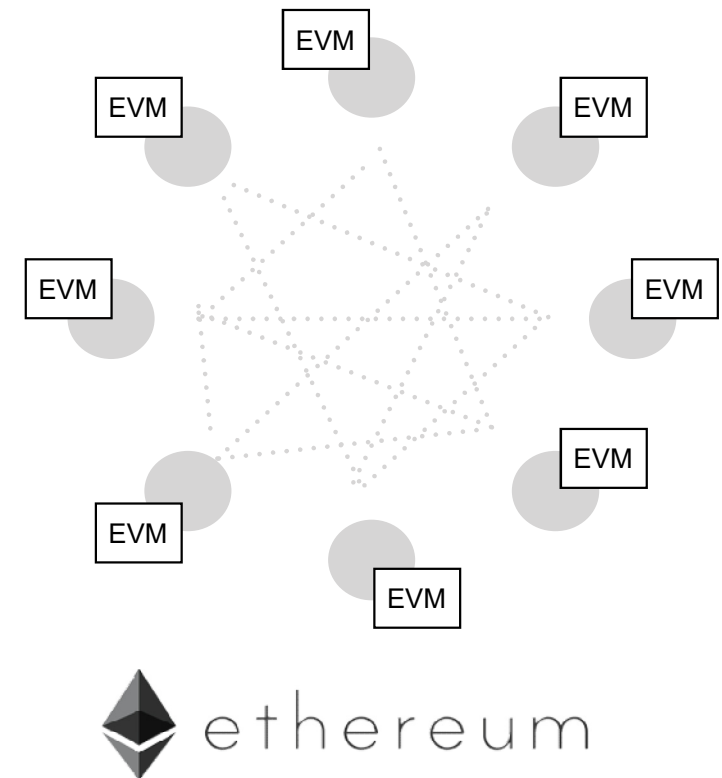
- How expensive is a 51% attack?

- Buy an attack?



Ethereum Smart Contract

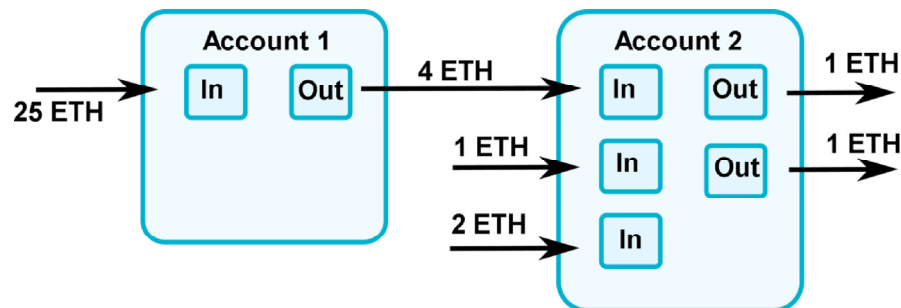
- Computation on EVM is "very expensive": every contract is run on every full Ethereum node
 - Result on every node is the same
 - Global computer, always running, always correct
- For now, proof of work (PoW)
 - But difficult with ASIC (memory-hard), starts at 1GB RAM for full node and miner (2GB RAM 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 – See slide 65,66, Chapter 5 “Blockchain Operations”

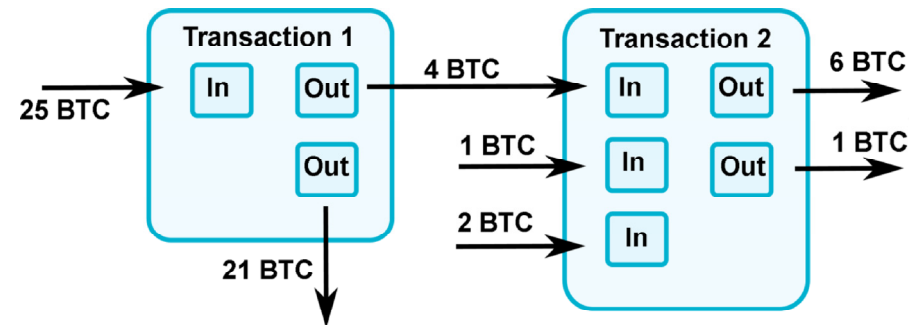
Account-based

- 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



The RLPx Transport Protocol

- **RLPx** is a cryptographic peer-to-peer network and protocol suite which provides a general-purpose transport and interface for applications to communicate via a p2p network.
 - All cryptographic operations are based on the secp256k1 elliptic curve (same as Bitcoin)
- **Elliptic Curve Integrated Encryption Scheme**
 - Shared key (derivation), KDF
 - Authentication, MAC
 - Symmetric encryption, AES
- **Messages: auth, auth-ack – + 1RTT (TLS + 2RTT)**

■ Framing

- multiplexing multiple protocols
- frame:
 - frame = frame-size || header-data || header-mac || frame-data || frame-mac

Cryptographic primitive	Hash	MAC	Digital signature
Integrity	Yes	Yes	Yes
Authentication	No	Yes	Yes
Non-repudiation	No	No	Yes
Kind of keys	none	symmetric	asymmetric

[source](#)

Serialization in Ethereum

■ Dynamic types

- `sum(bytes,bool,uint256[])`
- Location of the data (offset):
0x0000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000060
- Boolean:
0x0000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000001
- Location of data (offset):
0x0000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000a0
- At position 60, length of byte array:
0x0000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000004
- “dave” (left padded):
0x6461766500000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000

■ Number of items in array at position 0xa0

- 0x0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000003
- 3 times data item:
 - 0x0000000000000000000000000000000000000000000000000000000000000000
000000000000000000000001
 - 0x0000000000000000000000000000000000000000000000000000000000000000
000000000000000000000002
 - 0x0000000000000000000000000000000000000000000000000000000000000000
000000000000000000000003

■ In total 292 bytes, 16 bytes non-zero

- Efficiency? [Example](#)

Solidity

Scripts / Opcodes / Smart Contracts

■ Ivy - a non Turing complete higher-level language that compiles to Bitcoin Script

- Experimental

■ Rootstock

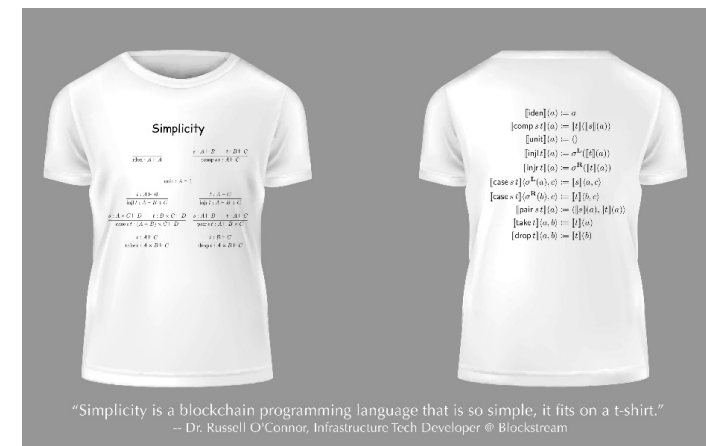
- Rootstock (RSK) is a smart-contract platform with a Turing Complete VM for Bitcoin.
- Bitcoin Sidechain, 2-way pegging
- Backward compatible with Ethereum VM

■ Simplicity - a typed, combinator-based, functional language without

- By Blockstream
- Definition fits on a t-shirt
- Building on UTXO model

■ Miniscript: language for writing Bitcoin scripts in a structured way

```
contract LockWithPublicKey(publicKey:
PublicKey, val: Value) {
  clause spend(sig: Signature) {
    verify checkSig(publicKey, sig)
    unlock val
  }
}
```



Solidity Language

■ 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 ^0.5.7;  
//minimal contract  
contract Example1 {  
    uint256 counter;  
}
```

<https://learnxinyminutes.com/docs/solidity/>
<https://solidity.readthedocs.io/en/v0.5.7/>
Details in Part VII

Solidity: Version, Mapping

```
/* Specifies what versions of Solidity a
 * source file can work on
 */
```

```
pragma solidity ^0.5.7;
```

```
contract Notary {
    ...
}
```

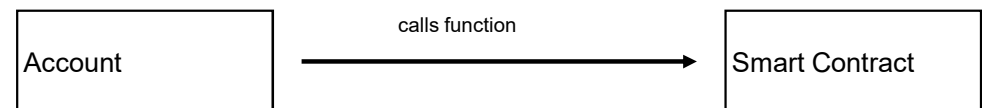
```
contract Notary {
    mapping (address => mapping (bytes32 => uint256)) stamps;
    ...
}
```

- Ensures that the source code is only meant for compiler versions that are not older than 0.5.0 and will also not work on a compiler starting from version 0.6.0.
- Mapping of address, of mapping of hash, and timestamp

Address	Hash	Timestamp
0xD6df5935cD03a768b7B9E92637A01b25e24cb709	ed52f3833d5d2c920fd43a0972add3555ad149f0201bbd134907498a851a5ec3	2009-06-24 12:39:54 +0900
0xD6df5935cD03a768b7B9E92637A01b25e24cb709	0c911c1d3bde4be991e3d39c2f7e97c621617feaf8f7070e2384c602430ee382	2015-04-14 10:34:24 +0800
0x1530df3e1C69501d4Ecb7E58eB045b90DE158873	0c911c1d3bde4be991e3d39c2f7e97c621617feaf8f7070e2384c602430ee382	2018-07-21 14:33:24 +0900
...	...	...

Solidity: Constructor / Function Calls

- The constructor is run only once, i.e., when the contract is created!
- Smart contract function can call another smart contract function
- Initiator always needs to be an external account
 - Externally owned account vs. contract account



OR



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;
    }

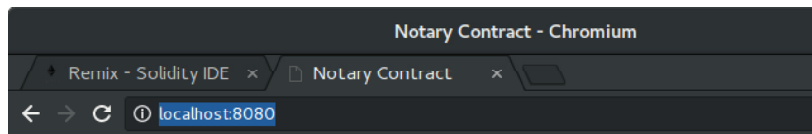
    function verify(address recipient, bytes32 hash)
        public view returns (uint256) {
        return stamps[recipient][hash];
    }
}
```

Run it locally

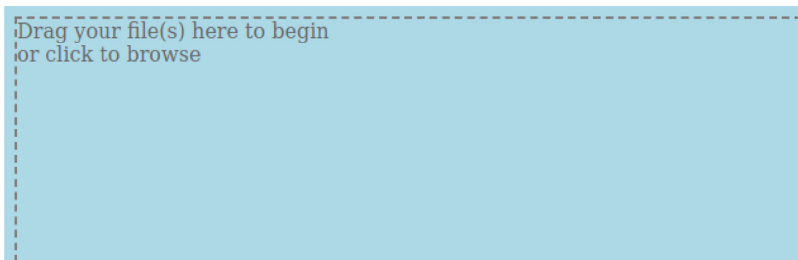
■ Installation

- yarn install
- ./node_modules/.bin/webpack-dev-server

■ Open Browser: <http://localhost:8080/>



Notarize PDF



```
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 [wds]: 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       0  [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 8 KiB {main} [built]
[./node_modules/webpack-dev-server/client/socket.js] (webpack)-dev-server/client/socket.js 8 KiB {main} [built]
[./node_modules/webpack/hot sync ^\\.\\.\\/log$] (webpack)/hot sync nonrecursive ^\\.\\.\\/log$ {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} [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]
+ 63 hidden modules
```

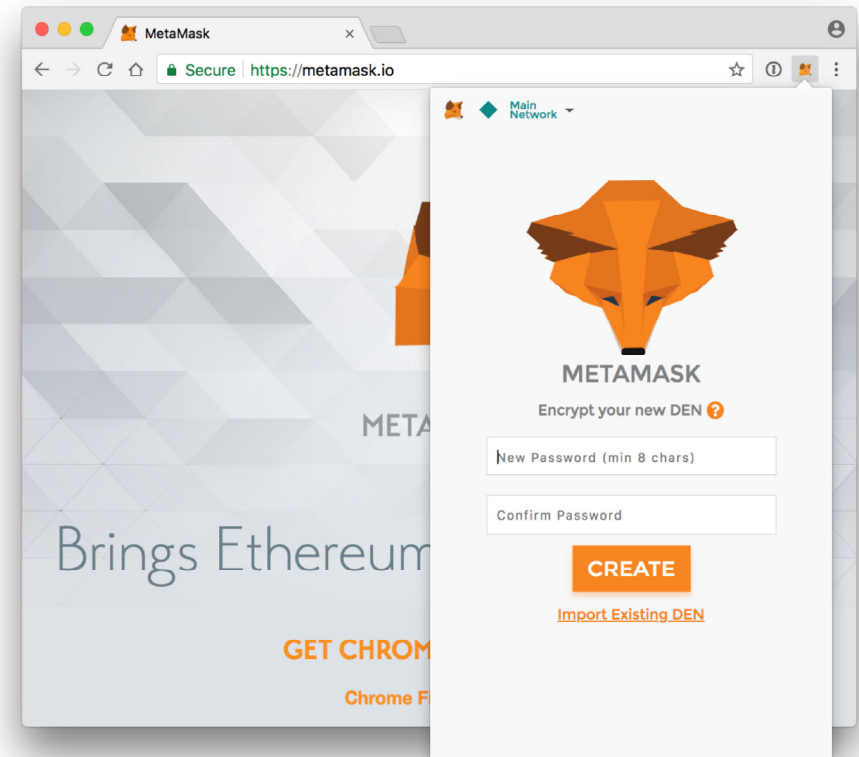
MetaMask

■ 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](#)

■ Open Remix IDE: <https://remix.ethereum.org>

- Use Notary.sol from <https://github.com/tbocek/VSS-FS18/blob/master/VSS-web3js/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

- **0x654cf88c4cff3e62696f7cbb55fbba922b2a75897a010347e371e9398b658c4affa611aa**
- Hash is:
0x4cff3e62696f7cbb55fbba922b2a75897a010347e371e9398b658c4affa611aa
- What is 0x654cf88c?

■ First four bytes of the Keccak (SHA-3) hash of the signature of the function.

- [Keccak](#)(store(bytes32))

MetaMask Notification

CONFIRM TRANSACTION

Account 2
25D963...A630
46.661 ETH
36089.67 USD

Amount 0.0003

Gas Limit 176645

Gas Price 2

Max Transaction Fee 0.0003

Max Total 0.0003

Data included: 5

RESET SUBMIT RE

Environment Injected Web3 Rinkeby

Account 0x25d...0a630 (46.66184667746474970)

Gas limit 3000000

Value 0 wei

Notary

Create

Load contract from Address At Add

ERC20

- **ERC20 is a technical standard for smart contracts on the Ethereum blockchain for implementing tokens**
 - proposed on November 19, 2015
 - January 2018: more than [21000](#) [ERC20 token](#) contracts: EOS, Tronix, VeChain, OmiseGO, MOD Token
- **ERC20 Token (Simplified)**
 - No allowance / approval / transferFrom, also no Approval event
 - Using SafeMath (always use it)
 - Creator gets 1000 coins

```
contract SimpleERC20 {  
    function totalSupply() public view returns (uint256);  
    function balanceOf(address who) public view returns  
(uint256);  
    function transfer(address to, uint256 value) public  
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;  
  
using SafeMath for uint256;  
mapping(address => uint256) balances;  
uint256 totalSupply_;  
  
constructor() public {  
    totalSupply_ = 1000 * (10**18);  
    balances[msg.sender] = totalSupply_;  
}
```

...

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;
}
```

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);
    }
}
```

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 ($N = \text{prime1} \times \text{prime2}$).
 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

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 {
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;
}
```

Random Numbers

■ Random Numbers

- There is no random number in Ethereum
- Every EVM needs to come to the same result

■ Random Numbers from Oracle (External source)

- Or: commitment schemes

■ Or: use future blockhash

- Only up to 256 blockhashes from the past can be accessed.
- Deduct / add tokens/ETH from the past address
- Miner can influence the random value in a sense
 - Value needs to be low (miner gets 2 ETH)

```
function transfer(address _to, uint256 _value) public returns (bool) {
    ...
    //since this is a lucky coin, the value transferred is not what you
    expect
    luckyTransfer();
    previousTransferBlockNr = block.number;
    previousTransferAddress = msg.sender;
    ...
    uint256 val = _value.sub(potIncrease);
    pot = pot.add(potIncrease);
}

function luckyTransfer() private {
    if(block.number != previousTransferBlockNr && (block.number -
previousTransferBlockNr) < 256 ) {
        uint256 rnd =
uint256(keccak256(block.blockhash(previousTransferBlockNr)));
        if(rnd % 200 == 0) { //0.5%
            balances[previousTransferAddress] =
balances[previousTransferAddress].add(pot);
            Transfer(this, previousTransferAddress, pot);
            //tokens are from pot, thus no tokens are created from thin air
            pot = 0;
            previousTransferBlockNr = 0;
            previousTransferAddress = 0;
        }
    }
}
```

Part VI: The Blockchain World

14. Other Blockchains

Blockchains, Cryptocurrencies, Tokens

- App. 2500 different ones are listed at <https://coinmarketcap.com/>

- A classification wrt **application domain** had been presented already earlier

- A mapping on a set of **selected BCs to these eras** here is used to dive into further details:

– Done and to come

Era 1.0	Era 2.0	Era 3.0	Era 4.0
BinanceCoin	Algorand	Tetha	Aelf
Bitcoin	Bazo		Polkadot
Bitcoin Cash	Chainlink		
Bitcoin Gold	Corda		
Dasch	Elements		
Dodgecoin	EOS		
Libra	Ethereum		
Litecoin	Hyperledger		
Monero	IOTA		
Paxos	MultiChain		
Ripple	NEO		
Stellar	OpenChain		
Tether	Tezos		
ZCash			

Libra

- ❑ URL: <https://libra.org>
- ❑ BC type: Permissioned
 - Node types: Participant and Validators
- ❑ Token type: Currency
- ❑ Account model: Account-based
- ❑ Consensus mechanism: LibraBFT
 - Variation of the HotStuff protocol
 - About 100 founding members as Validators



Libra Characteristics

❑ Major characteristics

- Mining scheme used: Dedicated protocol based on trusted Validators
- Block size: There is no concept of a block of transactions
- Transaction size: 5 KByte
- Transaction rate: 1,000 tx/s
- Security features: Proof of authority, cryptographically authenticated database, memory-safe using Rust
- Smart Contract
 - Language: Move Programming Language
 - Key details: Implemented using modules containing code values and resources containing data values.

Libra Economics

- ❑ Cryptocurrency: Yes, Libra Coins
 - Details: New coins are minted and excess burnt
 - Value of the coin will be tightly bound to the global economy
 - Exchange rates to fiat: Stable (planned for)
 - Transaction fees: Low fees, variable according to gas price
- ❑ Wallet
 - Not available yet
- ❑ Listed on crypto exchange(s)
 - Not listed yet
- ❑ Market capitalization (as of July 31, 2019)
 - The stablecoin will be released only in 2020

Libra Organization

- ❑ Developer community: Libra Association
 - Country of origin: Switzerland
 - SDKs: Not available yet
 - Blockchain explorer: Yes, <http://libra-explorer.net> (testnet)
 - Full or light client: Testnet only, <https://developers.libra.org>
- ❑ Known vulnerabilities and attacks: Double spending attack, avoided by using the LibraBFT
- ❑ Major application domain(s)
 - Financial services, payments, e-commerce

Libra Overall Evaluation

- ❑ Scalability
 - Initial set-up at 1,000 tx/s for a set of 100 validator nodes
 - Plans to become permissionless as technology matures
- ❑ Centralization degree
 - 50%, partial decentralization
- ❑ Energy efficiency
 - Projected to use energy more like current data centers
- ❑ Security level
 - Designed to deal with most-common attacks (e.g., double spending) and tolerate faults within the Validator network,
 - Not storing any personal data of users (plan)

Monero

- ❑ URL: <https://getmonero.org>
- ❑ BC type: Public
- ❑ Token type: Currency
- ❑ Account model: UTXO
- ❑ Consensus mechanism: Proof-of-Work (PoW)
 - Block size changes based on the transaction volume
- ❑ Additional mechanisms:
 - Pruning, ASIC-resistance PoW



Monero Characteristics

❑ Major characteristics

- Mining scheme used: Proof-of-Work
- Block size: Variable
 - Defined according the previous 100 blocks
 - Storage efficiency: 1,883,942 (block height)
 - ~ 72 GB (full node)
- Transaction size: ~ 14 KByte
- Transaction rate: ~ 7 tx/s (1,700 tx/s estimated)
- Security features: ring signatures, ring confidential transactions (RingCT), and stealth addresses.
 - All transactions on the network are private by mandate
 - Pseudo-anonymous

Monero Economics

- ❑ Cryptocurrency: Yes, XMR
 - Exchange rates to fiat: Unstable
 - Transaction fees: Variable
 - Based on transaction priority, \$0.021 USD (average)
- ❑ Wallet
 - Reference implementation in Javascript, <https://mymonero.com>
 - API: JSON RPC Interface
- ❑ Listed on crypto exchange(s)
 - <https://localmonero.co>, <https://anycoindirect.eu>
- ❑ Market capitalization (as of July 31, 2019)
 - \$1,388,479,350 USD

Monero Organization

- ❑ Developer community: Monero Team
 - Country of origin: N/A
 - SDKs: Daemon RPC, Wallet RPC
 - Blockchain explorer: <https://moneroblocks.info>
 - Full or light client: Both, <https://getmonero.org/downloads>
- ❑ Known vulnerabilities and attacks:
 - Zero-amount miner TX, DDoS risk, Traceable transactions
 - Incident Response Team and community of developers work on reported bugs/vulnerabilities
- ❑ Major application domain(s)
 - Cryptocurrency with a focus on private and censorship-resistant transactions

Monero Overall Evaluation

- ❑ Scalability
 - Large size of the blockchain, poorly scalable.
- ❑ Centralization degree
 - 100%, fully decentralized
- ❑ Energy efficiency
 - 645.62 GWh of electricity (higher than a country like Haiti)
- ❑ Security level
 - 100% privacy-driven project

EOS

- ❑ URL: <https://eos.io/>
- ❑ BC type: dApp Ecosystem
 - Node types: block producers, backup block producers, api nodes, seed nodes, validators
- ❑ Token type: Utility
- ❑ Account model: Account (name in a human readable format)
- ❑ Consensus mechanism: BFT-dPoS
 - 21 allowed block producers → centralization
 - EOS holders vote for block producers, EOS tokens for votes
- ❑ Additional mechanisms:
 - Free of charge for users
 - Contracts mutable, freezing accounts possible



EOS Coin

❑ Major characteristics

- Mining scheme used: BFT-dPoS
- Block size: limited by network capacity
 - Storage efficiency: 4 TB after 8 months of operation
- Transaction size: N/A, max 30 ms for execution
- Transaction rate: max 3,996 tx/s
- Security features: SHA-256 is used to generate the digest and ECDSA to sign the block
- Smart Contract
 - Supported binaries Web Assembly (WASM/ABI files), Turing-complete
 - Supported Languages: C++
 - Other languages that produce WASM code: RUST, Python, Solidity

EOS Economics

- ❑ Cryptocurrency: No
 - Utility token to acquire EOS CPU/RAM to run dApps
 - Exchange rates to fiat: floating, “rather” stable
 - Transaction fees: free of charge for users, accounts with dApps need to stake EOS tokens against EOS CPU and trade EOS tokens against EOS RAM
- ❑ Wallet
 - Reference implementation C++, Cleos, Nodeos, Keosd
<https://github.com/EOSIO/eos> and API in C++
- ❑ Listed on crypto exchange(s)
 - Bitfinex, Binance, Kraken, HitBTC, Liqui, Gate.io, EtherDelta, Kucoin, Mercatox, Exrates, Livecoin, IDEX, COSS, TIDEX
- ❑ Market capitalization (as of August 15, 2019)

EOS Organization

- ❑ Developer community: 188 developers (GitHub)
 - Country of origin: Cayman Islands
 - Deployment needs: depending on the role, *i.e.*, block producers, backup block producers, api nodes, seed nodes
 - SDK: Jeos (Java), Eospy (Python)
 - Blockchain explorer: Yes, <https://bloks.io>
 - Full or light client: Full <https://github.com/EOSIO/eos>
- ❑ Known vulnerabilities and attacks:
 - Against dApps: Numerical Overflows, Authorization Checks, Apply Checks, Transfer Error Prompt, Random Number Practice, Rollback Attack
- ❑ Major application domain(s)
 - SC-based distributed Applications (dApps)

EOS Overall Evaluation

- ❑ Scalability
 - High: both horizontal and vertical scaling
- ❑ Centralization degree
 - Centralized among block producers
- ❑ Energy efficiency
 - High
- ❑ Security level
 - Pseudo anonymous, Smart Contract mutability, account freezing, account code change

HyperLedger



HYPERLEDGER

- ❑ No currency
- ❑ Private distributed ledger
- ❑ Umbrella Project
 - **Fabric**: Platform for building blockchain-based products
 - **Composer**: tools to build, test, and operate a blockchain
 - **Cello**: allows Blockchain-as-a-Service (BaaS)
 - **Explorer**: dashboard for blockchain monitoring
 - **Burrow**: Ethereum Smart Contracts execution
 - **Sawtooth**: modular enterprise-level blockchain
 - **Caliper**: blockchain benchmarking tool

Active Hyperledger Projects

- ❑ Open source effort created to advance cross-industry Distributed Ledger Technologies (DLT) hosted by “The Linux Foundation”
 - <https://www.hyperledger.org>
- ❑ Active HL projects include
 - HL Fabric a general-purpose, permissioned DL
 - HL Indy a special-purpose, permissioned DL
 - Focusing on decentralized identity management
 - HL Iroha is a permissioned DL
 - Tailored for digital asset management
 - HL Sawtooth a general-purpose, permissioned or permissionless DL

Hyperledger Fabric

- ❑ Hyperledger Fabric (HLF) initialized by IBM
- ❑ HLF implemented in Golang
 - Smart contracts in Go, JavaScript, or Java
 - SDKs exist in Node.js, Java, Go, REST and Python
- ❑ Network roles and components of HLF
 - **Channels** for complete data isolation b/w a set of participants
 - **Private data collections** enable transactions between participants on the same ledger, but keep data private to a subset of transactors
 - Private data is shared peer-to-peer, with only hashes stored on the shared ledger as
 - **Identity Mixer** for anonymity of transaction submitters



Hyperledger Indy

- ❑ Hyperledger Indy focused on identity management
 - DIDs (Decentralized Identifiers) without requiring central resolution authorities.
 - Verifiable claims are an interoperable format for the exchange of digital identity attributes and relationships
 - Currently in the W3C standardization pipeline
 - Zero Knowledge Proofs enable that some or all of the data in a set of claims is true without leaking any additional information, including the identity of the prover
- ❑ HL Indy is independent, but commonly associated with “The Sovrin Foundation”, which offers a public utility for identity, built on top of Indy’s codebase



Hyperledger Iroha

- ❑ Hyperledger Iroha is written in C++, initially developed by a group of Japanese developers, who built their own BC for mobile use cases
- ❑ BFT Consensus algorithm called YetAnotherConsensus (YAC)
 - Iroha offers ready-to-use set of commands and queries, as well as multi-signature transactions
- ❑ Digital asset management capabilities
 - *E.g.*, create digital assets, register accounts, and transfer assets between accounts.
 - Robust permission system, allowing permissions to be set for all commands, queries, and joining of the network



Hyperledger Sawtooth

- ❑ Hyperledger Sawtooth initialized by Intel
- ❑ Permissioned and permissionless deployment possible
- ❑ Proof-of-Elapsed-Time (PoET) consensus algorithm
 - Dependent on Intel's Software Guard Extensions (SGX) to elect leaders to cut blocks based on random wait times.
 - Peers have access to all transaction data, unlike in Fabric, where “private” channels can be established
- ❑ Smart contract capabilities
 - SDK for Python, Javascript, Go, and Rust
 - Ethereum contract compatibility is provided with Seth, enabling the deployment of Sawtooth contracts to Ethereum





- ❑ URL: <https://neo.org>
- ❑ BC type: Public
 - Node types: Participants, Validators, Speakers
- ❑ Token type: Currency, Utility
- ❑ Account model: UTXO
- ❑ Consensus mechanism: dBFT 2.0
 - Decision on consensus nodes for next round by voting
 - Users only need to wait for one confirmation (15 s)
- ❑ Additional mechanisms:
 - Digital Identity based on KYC (Know Your Customer)
 - “Roll back the chain” with consensus

NEO Characteristics

❑ Major characteristics

- Mining scheme used: dBFT 2.0
- Block size: ~ 5 KB (average)
 - Storage efficiency: 4,038,763 (block index)
- Transaction size: ~ 200 Byte
- Transaction rate: 1,000 tx/s (10,000 tx/s theoretically)
- Security features: Anti-quantum cryptography mechanism (NeoQS), digital certificate
- Smart Contract
 - Language: .Net, Java, C, C#, Go, Python, Kotlin, Javascript
 - Key details: Turing-complete, support concurrent operation, sharding, and unlimited scalability, when running in NeoVM
 - All operations have GAS costs, but, the first 10 GAS are for free

NEO Economics

- ❑ Cryptocurrency: Yes
 - Details: 100 millions NEO created in the Genesis Block
 - Exchange rates to fiat: Unstable
 - Transaction fees: Variable based on operations (GAS)
- ❑ Wallet
 - Reference implementation in Javascript
<https://docs.neo.org/docs/en-us/node/gui/wallet.html>
 - API: RPC, Smart Contract Framework
- ❑ Listed on crypto exchange(s)
 - <https://coinall.com>, <https://etoro.com>
- ❑ Market capitalization (as of July 31, 2019)
 - \$807,485,181 USD

NEO Organization

- ❑ Developer community: NEO Contributors
 - Country of origin: China
 - SDKs: NEO SDK (current version: 2.9.0)
 - Blockchain explorer: <https://neotracker.io>
 - Full or light client: both, <https://neo.org/client>
- ❑ Known vulnerabilities and attacks:
 - NEO Vulnerability Bounty Program paid up to \$10,000 USD per vulnerability reported
- ❑ Major application domains:
 - Smart economy, digital assets, digital identity

NEO Overall Evaluation

- ❑ Scalability
 - Theoretically, NEO achieves infinite scalability with NEOVM
- ❑ Centralization degree
 - 50%, centralization of control
- ❑ Energy efficiency
 - Lower electricity costs and attendant carbon footprints
- ❑ Security level
 - Uses digital identity to link accounts to real life identity, traceable, assets protection
- ❑ Other dimensions of importance
 - Peer-to-peer networking, cross-chain interoperability

THETA

- ❑ URL: <https://www.thetatoken.org/>
- ❑ BC type: Public
 - Node types: Validators, Guardian, Normal
- ❑ Token type: Currency and Utility
- ❑ Account model: Account/Balance
- ❑ Consensus mechanism: PoS/BFT
 - 10-20 validators
 - Guardian nodes finalize blocks
- ❑ Additional mechanisms
 - Blockchain pruning, off-chain payments



THETA Characteristics

❑ Major characteristics

- Mining scheme used: Multi-Level BFT
- Block size: up to 8,192 tx per block
 - Storage efficiency: N/A
- Transaction size: N/A
- Transaction rate: > 1,000 tx/s
- Security features: Pseudo-anonymous addresses
- Smart Contract
 - Language: Solidity
 - Key details: Ethereum-based Smart Contracts, Turing-complete, Ethereum Virtual Machine

THETA Economics

- ❑ Cryptocurrency: Yes
 - Details: THETA and TFuel (related to gas in Ethereum)
 - Exchange rates to fiat: unstable
 - Transaction fees: variable on transaction size (gas)
- ❑ Wallet
 - Reference implementation in Go,
<https://github.com/thetatoken/theta-protocol-ledger>
 - API: Web, Command-line only
- ❑ Listed on crypto exchange(s)
 - Binance, <https://www.binance.com>
- ❑ Market capitalization (as of July 24, 2019)
 - 101,154,392 USD

THETA Organization

- ❑ Developer community: 10 (Theta Developer Team)
 - Country of origin: United States of America (U.S.A.)
 - Deployment needs: Theta Ledger Node
 - SDKs: Android SDK, REMIX
 - Blockchain explorer: <https://explorer.thetatoken.org/>
 - Full or light client: Full <https://github.com/thetatoken/theta-protocol-ledger>
- ❑ Known vulnerabilities and attacks:
 - 51% Attacks (Expensive, but doable)
- ❑ Major application domain(s)
 - Video Streaming, Live TV

THETA Overall Evaluation

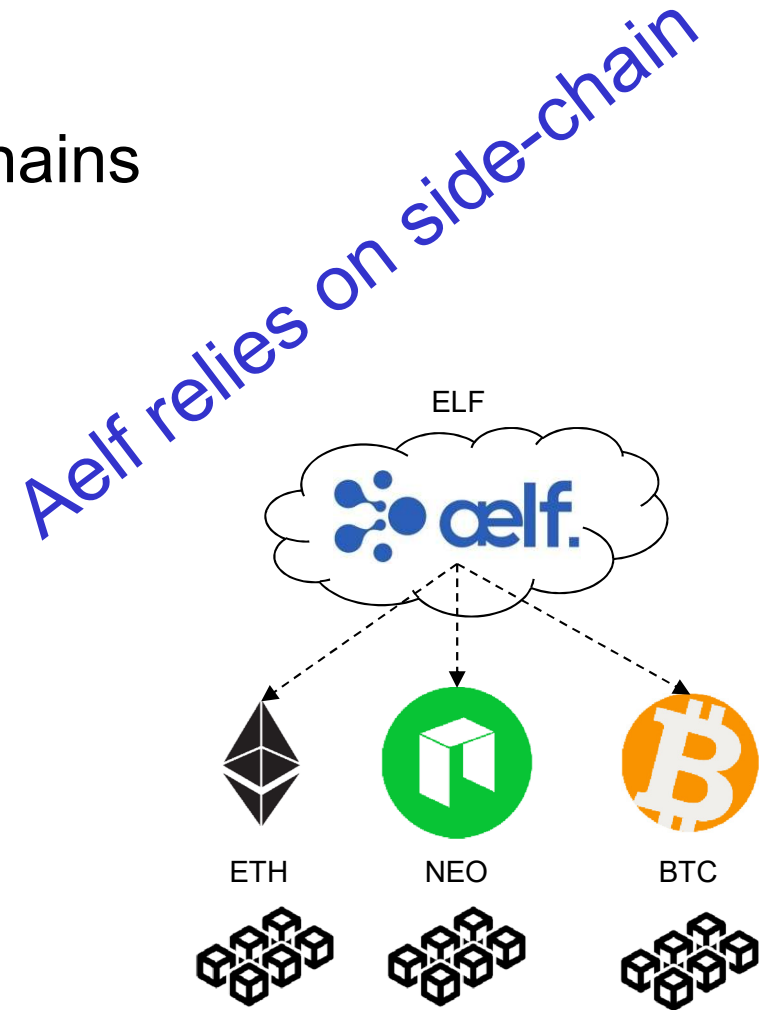
- ❑ Scalability
 - High
- ❑ Centralization degree
 - 10-20 Validators (small number)
 - Limited number of Guardian nodes
- ❑ Energy efficiency
 - Energy-efficient (BFT-based)
- ❑ Security level
 - Standard BC-security
- ❑ Other dimensions of importance
 - Plans to reduce dependency from Content Delivery Networks

- ❑ URL: <https://aelf.io/>
- ❑ BC type: multi-purpose chain
 - Node types: Participant and Miner (stake)
- ❑ Token type: Currency
- ❑ Account model: Account-based
- ❑ Consensus mechanism:
 - Delegated Proof-of-Stake (DPoS)
 - Token holders cast votes to elect nodes that produce blocks
- ❑ Additional mechanisms:
 - Cross-chain communication



Aelf Characteristics

- ❑ Major characteristics
 - Mining scheme used: DPoS
 - Block size: making use of side-chains
 - Storage efficiency: N/A
 - Transaction size: variable
 - Transaction rate: 14,968 tx/s*
 - Security features: N/A
 - Smart Contract
 - Language: N/A



*Data from Aelf test scenario

Aelf Economics

- ❑ Cryptocurrency: Yes (ELF)
 - Details: multi-chain (and multi-purpose) blockchain
 - Exchange rates to fiat: unstable
 - Transaction fees: variable based on the market perception
- ❑ Wallet
 - Reference implementation: [Coinomi](#)
 - Also available on Android, iOS, Windows, Linux and MacOS
- ❑ Listed on crypto exchange(s)
 - [Binance](#), [BitMex](#), [Poloniex](#)
- ❑ Market capitalization (as of July 31, 2019)
 - \$60,977,204 USD

Aelf Organization

- ❑ Developer community: 34 contributors on GitHub
 - Country of origin: China
 - Deployment needs: Wallet
 - SDKs: Docker, Java, C#, JavaScript, Lua
 - Blockchain explorer: <https://explorer.bitcoin.com/bch>
 - Full or light client: full
 - Known vulnerabilities and attacks:
 - Heavy centralization due to DPoS
 - 51% Attack, i.e., single organization holds 51% of the mining capacity and may censor transactions
- ❑ Major application domain(s)
 - General purpose applications

Aelf Overall Evaluation

- ❑ Scalability
 - High
- ❑ Centralization degree
 - Partially Decentralized
- ❑ Energy efficiency
 - High
- ❑ Security level
 - Immutable, transparent
- ❑ Other dimensions of importance
 - Enable cross-chain communications

Polkadot

Polkadot.

- ❑ URL: <https://polkadot.network>
- ❑ BC type: interoperability platform
 - Network roles: Validators, Collators and Nominators
 - Network structure: Parachains, Relay chains and Bridge chains
- ❑ Token type: Utility (DOT) tokens
- ❑ Consensus mechanism: Hybrid GRANDPA/BABE
 - **G**HOST-based **R**ecursive **A**ncestor **D**eriving **P**refix **A**greement/**B**lind **A**ssignment for **B**lockchain **E**xtension
 - Hybrid and variant of Proof-of-Stake consensus

Polkadot Characteristics

❑ Major characteristics

- Heterogeneous multichain with scalable security and an interoperability protocol
- Central chain of Polkadot termed **relay-chain**
 - On which all validators are staked on with DOT tokens.
- Polkadot network based on **parachains**
 - Parallelizable blockchains comprising the Polkadot network
 - Each parachain can have a unique architecture
 - All parachains are connected and secured by the relay chain
- **Hybrid consensus mechanism**
 - Splits up the finality gadget (GRANDPA) from the block production (BABE) mechanism of the relay-chain
 - BABE runs b/w validator nodes and determines authors of new blocks

Polkadot Economics

- ❑ Cryptocurrency: Yes
 - Details: Native DOT tokens are used for 4 purposes
 - Used for governance of the network
 - Staked for operation of the network
 - Bonded to connect a chain to Polkadot as a parachain
 - Enabling interoperability between blockchains
- ❑ Multiple wallets in active development
- ❑ Not yet listed on crypto exchanges
 - First ICO raked in \$144 million
 - Second ICO may be launched
- ❑ Market valuation
 - ~1\$ billion USD (as of August 15, 2019)

Polkadot Organization

- ❑ Polkadot is a project of the Web3 Foundation
 - Sponsoring community development and software services
 - Web3 Foundation is based in Switzerland, Zug
 - SDKs: Substrate framework
 - Blockchain explorer available: <https://telemetry.polkadot.io/>

- ❑ Major application domain
 - Easy deployment of public or private parachains
 - Enabling interoperability between those parachains
 - Using “Bridges” to interact with other Blockchains
 - Such as Ethereum or Bitcoin

Bazo

- ❑ Bazo is a **permissionless blockchain** for research
 - Its **not an ICO**, not looking for funding, but for ideas
 - Testing of new ideas, new mechanisms, new algorithms
- ❑ Blockchain **from scratch at CSG and now HSR**
 - Started with simple PoW, a solid, well-tested mechanism
 - Gradually extended with new ideas and concepts
 - Now with unique feature proposals/ideas
 - Self-contained proofs
 - Sharding
 - Pruning
 - Smart Contracts with only big-ints
 - <https://github.com/bazo-blockchain>



Bazo Functionality (as of March 2019)

- PoW consensus mechanism initially
- PoS consensus mechanism later + cryptographic proof
- Progressive Web App (PWA)-based Mobile Wallet for Bazo
- Blockchain Explorer for Bazo
- Mobile Light Client for Bazo
- Pruning in Bazo and self-contained proofs
- Bazo Smart Contracts
- Sharding (in progress)

□ Application

- Oysy (Aduno) BC-based loyalty system

Credits go to T. Bocek (now HSR), S. Rafati, B. Stiller, and MA/BA students with CSG: L. Sgier, J. van der Assen, L. Boillat, S. Bachmann, M.-A. Chételat, A. Kürsat and HSR students R. Blum, S. Fontana, E. Meier, M. Steiner

Bazo's PoS

- ❑ Miner (“**validator**”) of a block chosen depending on his wealth, also defined as “stake”

- Validators restricted to 1 hash per second (H/s)

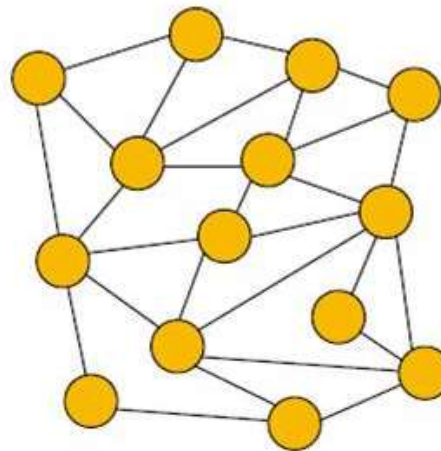
- ❑ **Requirements**

- Balance greater or equal minimum staking fee
 - Default: 1000
 - Published and mined Stake Transactions
 - Minimum waiting time elapsed (default: 0)

Public
PubKey: 4dc4...b904
Hashed Seed: 3ae7...32cd

Private
PrivKey: 8063...aeac
Seed: bd75...3428

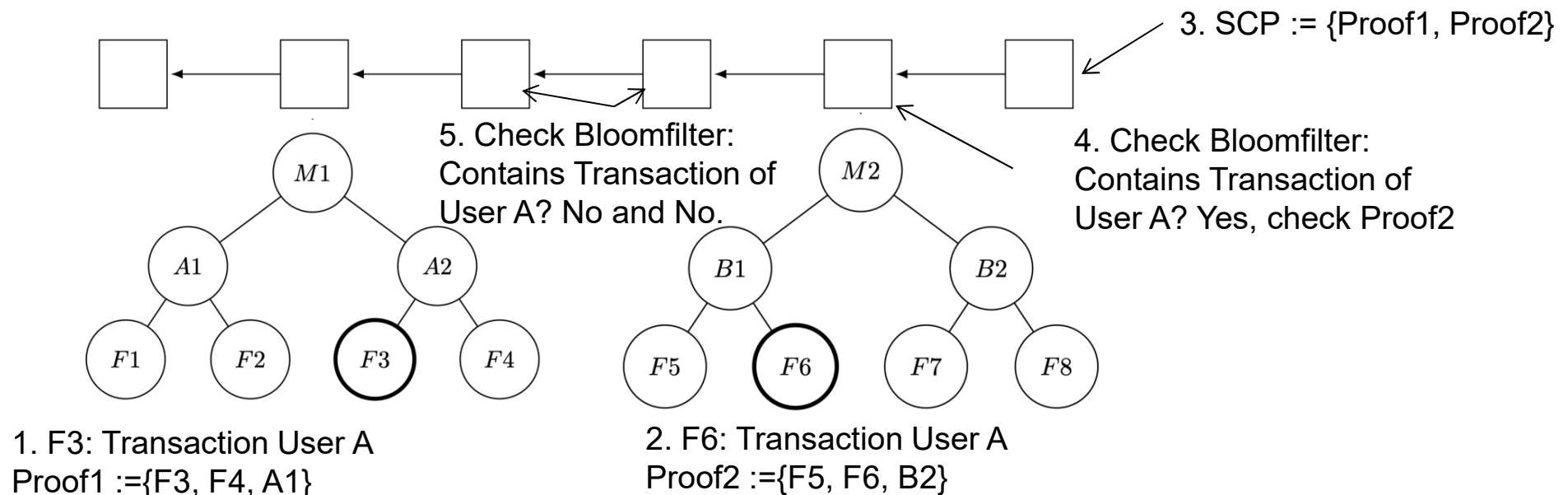
Publish Stake Transaction to the network



StakeTx #4'301	
Fee	2
Is Staking	True
Account	4dc4...b904
Timestamp	2018-04-27 23:11:54
Hashed Seed	3ae7...32cd
Signature	ffa1...43e1

Bazo's Self-contained Proofs (SCP)

- ❑ Traditionally, miners require the full blockchain to validate a transaction of a user
- ❑ With SCPs a user has to provide all required proofs in the transaction in order for validators to verify a transaction independent of the blockchain



Bazo's Sharding

- ❑ Inspired by the applicability in database research
- ❑ Current approach
 - Partition the Blockchain state into shards (communities)
 - Enables parallel validation of transactions
 - Build up several shard-chains concurrently
 - Each shard has own set of transactions
- ❑ Next steps (possibly)
 - Entity management in shards according to the SCP concept
 - Save state in the newest block of a shardchain
 - Merkle Patricia Trees

15. Blockchain Interoperability

Need for Interoperability

- ❑ Each blockchain (2100+) has its **pros and cons**
 - Transparency vs. privacy
 - Transaction fees, speed of transaction confirmation
- ❑ Interoperable chains enable several **opportunities**
 - Move assets between different platforms
 - Access information from one chain inside other chain
 - Different payment schemes
 - Allowing to choose the best alternative BC possible
- ❑ **V. Buterin:** “In the longer term, **building interoperable** applications is something that **should be done on a basis of responding to actual need**; while the underlying technology can certainly be developed in the abstract, it would likely to be a waste of resources to try to build applications that connect chains without a clear understanding on the reasons driving the use case.”

Types of Interoperable Chains

❑ Notary schemes

- One or more trusted parties validate events on another chain
- Ease of implementation: medium

❑ Sidechains/relays

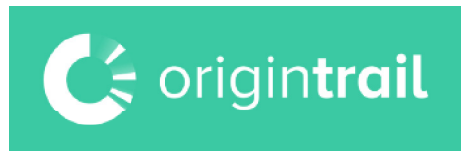
- The chain validate events on external chains as part of validating local events
- Ease of implementation: hard

❑ Hash-locking

- Events on different chains are unlocked using a secret value
- Ease of implementation: easy

State-of-the-art

- ❑ Projects that **use** or **provide** a platform for interoperable chains include:



...

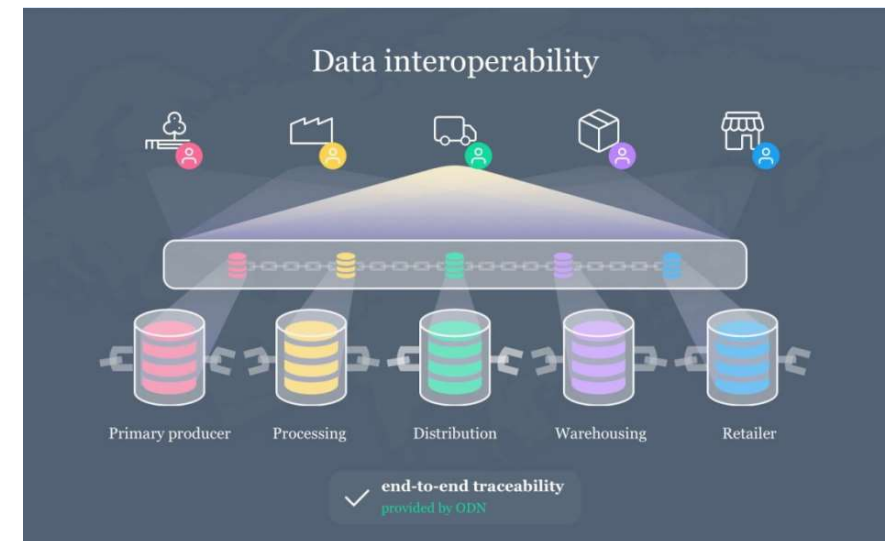
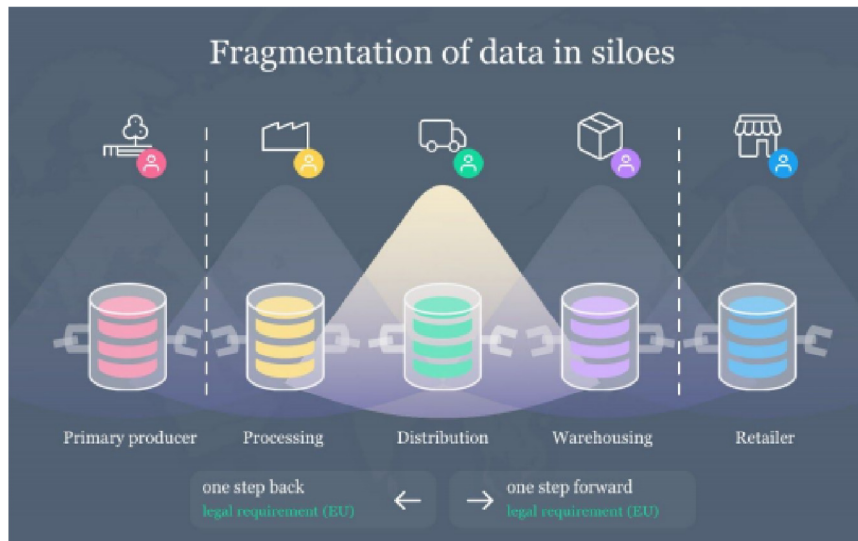
Origintrail.io

❑ Origintrail.io

- Type: Sidechain/Relay
- Description:

- Blockchain-agnostic protocol for asset tracking in supply-chain
- Provide a cost-effective solution to store sensitive data in a secure manner through a blockchain-agnostic approach

In order to provide the optimal solution we implement the OriginTrail protocol that runs on an off-chain decentralized peer to peer network, called the OriginTrail Decentralised Network (ODN). It enables peers on the network to negotiate services, transfer, process and retrieve data, verify it's integrity and availability and reimburse the provider nodes. This solution minimizes the amount of data stored on the blockchain in order to reduce cost and inefficiency.



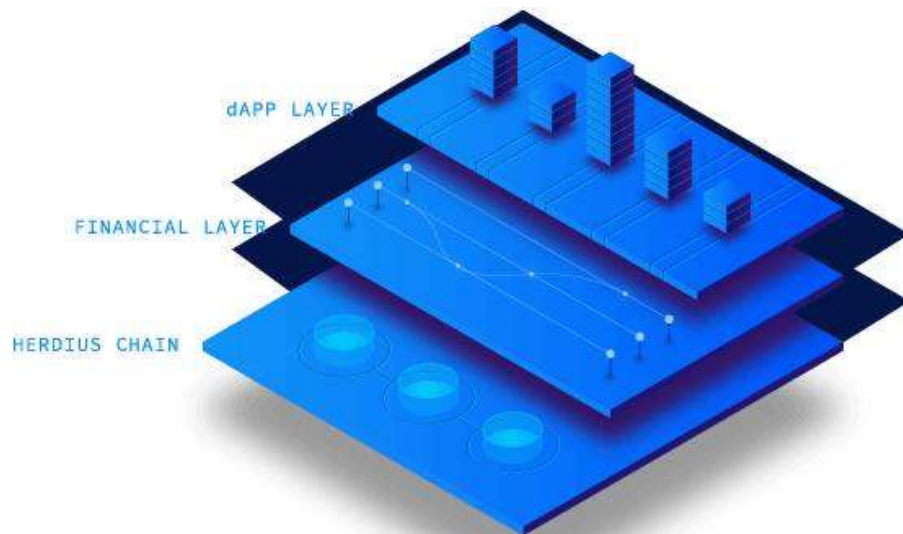
Herdus

□ Herdus

- Type: **Notary-scheme**
- Description:

- Build a highly performant decentralized financial platform enabling users to transact assets between different chains

verification: *Supervisors* will have two minutes to inspect child blocks for traces of misconduct from validators and, if necessary, to invalidate those transactions. If validators create invalid transactions, supervisors are responsible for reporting it.



User Layer
Transaction Layer
Rootchain Layer



Aion



The Third Generation Blockchain Network

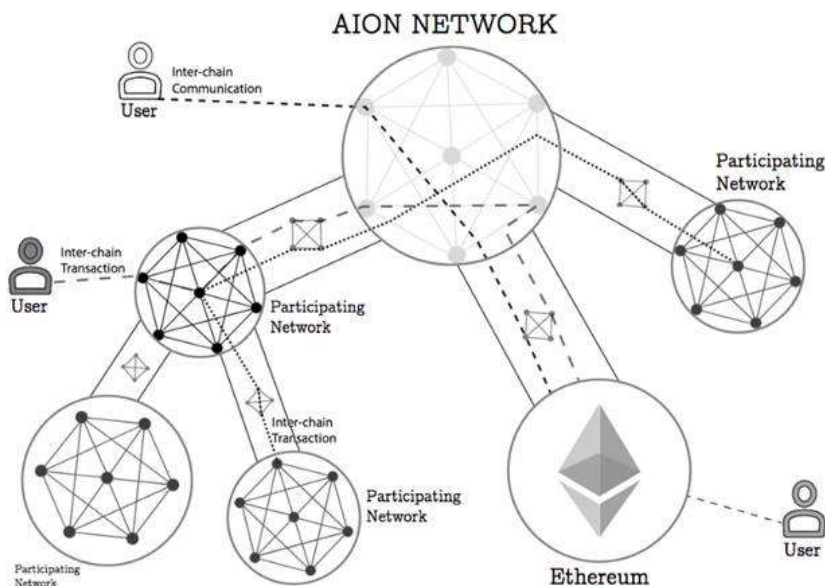
□ Aion

– Type: **Hash-locking**

– Description:

Interchain transactions are initially **created on a source blockchain and then processed and forwarded by bridges** and connecting networks before finally reaching the target blockchain. As stated previously, the creator of an interchain transaction must pay a transaction fee for the communication cost using AION tokens, thereby incentivizing all the participants in each junction of the route.

- A multi-tier blockchain system designed to address unsolved questions of scalability, privacy, and interoperability in blockchain networks.



- **Connecting networks**: facilitate interchain communication and interchain transactions between multiple private or public blockchain networks
- **Inter-chain transaction**: is a trust-free message between blockchains, which allows any connected blockchain to exchange information
 - Defined format, routing, states
- **Bridges**: communication protocol or API responsible for registering connected chains
- **Participating networks**: any AION-compliant network, can be purpose-specific blockchains, private networks, or consortium blockchains representing collections of entities

Crowdmachine

❑ Crowdmachine

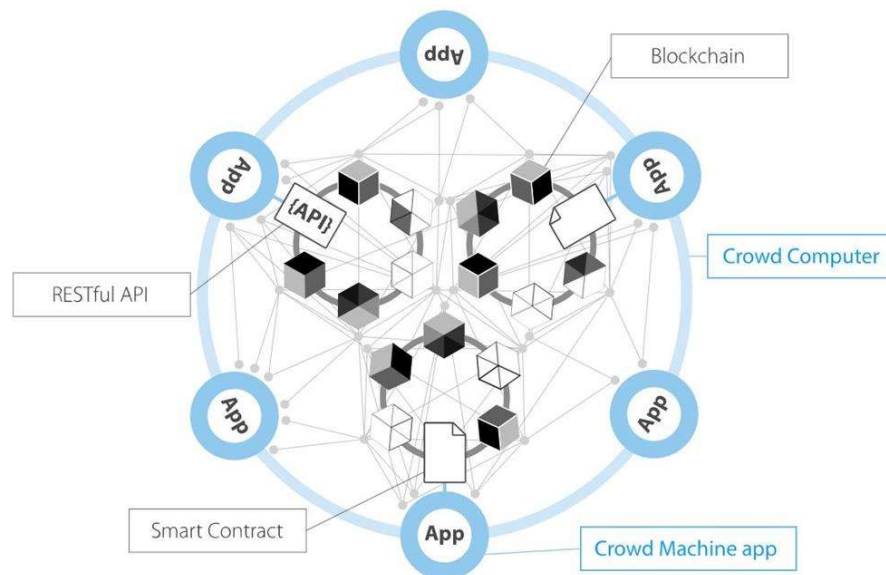
– Type: Sidechain/Relay

– Goals:

1. Enable any mobile device or computer owner to allow the spare capacity of their device to function as a member of a global network to run apps, and be rewarded for the use of their device; and
2. Allow developers and non-developers to create apps, without writing a line of code, that run on that global network and be rewarded every time the app is used.



Confusing description! Distributed Cloud Computing?



- **Crowd Machine:** consists of the Crowd Computer, Crowd App Studio and Crowd Share, being designed to execute blockchain smart contracts and decentralized apps that meet any requirement.
- **Crowd Computer:** consists of a peer-to-peer network of Crowd Virtual Machines ("CVM") that run on the peer devices. Device owners are compensated for the use of their surplus processing power to run the CVM.
- **Crowd App Studio:** allows App feature and function to be drawn as logic diagrams, making it faster and easier to create apps than current development approaches.
- **Crowd Share:** a GitHub-like source repository enabling the commercialization of source code by the developer community

Blockstack



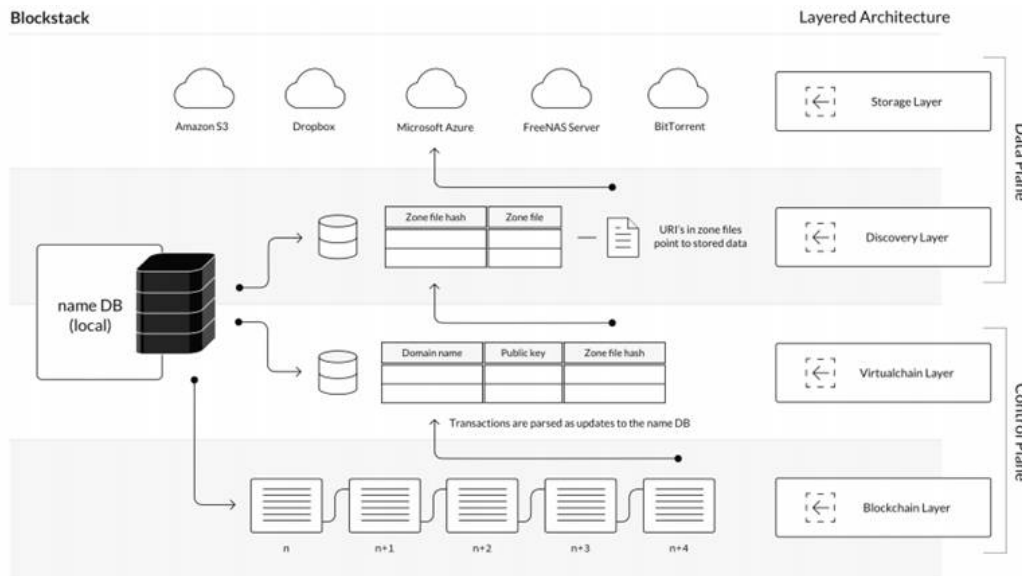
blockstack

❑ Blockstack

– Type: Side-chain/Relay

– Description:

- Digital Identity service. Instead of relying on remote, third party servers for access to data and the Internet, Blockstack aims to remove those trust points and give them back to the users via blockchains.



- **Storage layer:** includes the actual data values and represents a part of the data plane.
- **Discovery layer:** Blockstack uses zone files to store routing information. The format of zone files resembles that of DNS zone files.
- **Virtualchain layer:** designates new operations without having to modify the underlying blockchain, which are not aware of the virtual layer
- **Blockchain layer:** provides the medium for storage of operations and it also provides consensus regarding the order of writing of operations

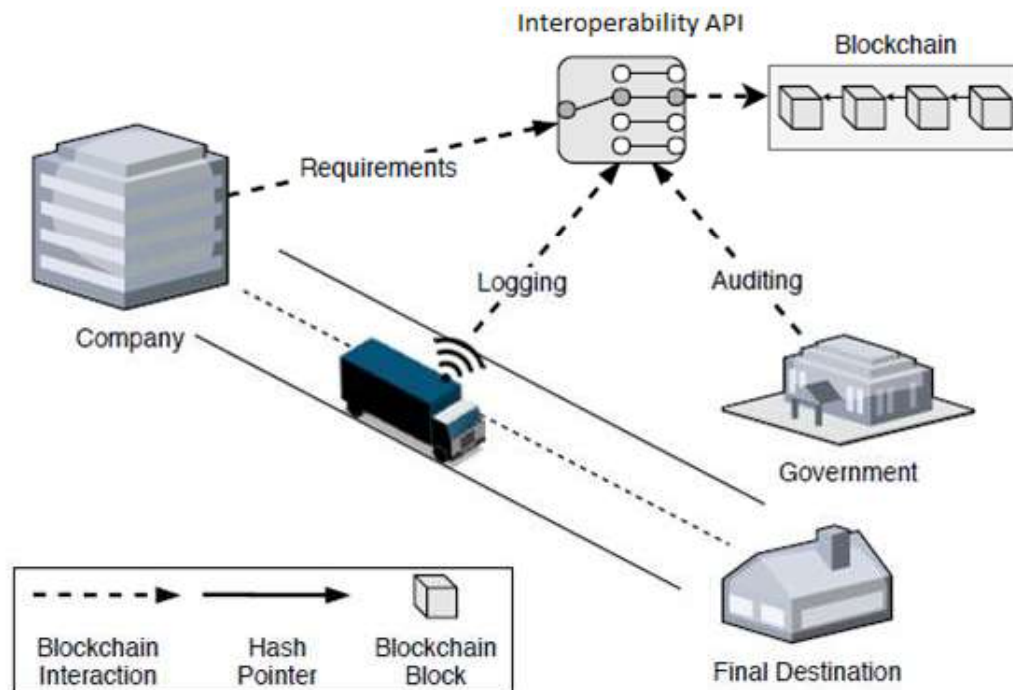
BC4CC Interoperability API

- ❑ Enable the **interaction** with **multiple blockchains**
 - Supporting different logistic chains
- ❑ Provide a blockchain **interoperability**
 - Relying in a Notary Scheme
 - Central solution managing transactions
- ❑ Allow **transparent** blockchain interaction
 - Technical details hidden from users
- ❑ Requirement: storage of **arbitrary** data
 - *E.g.*, character strings



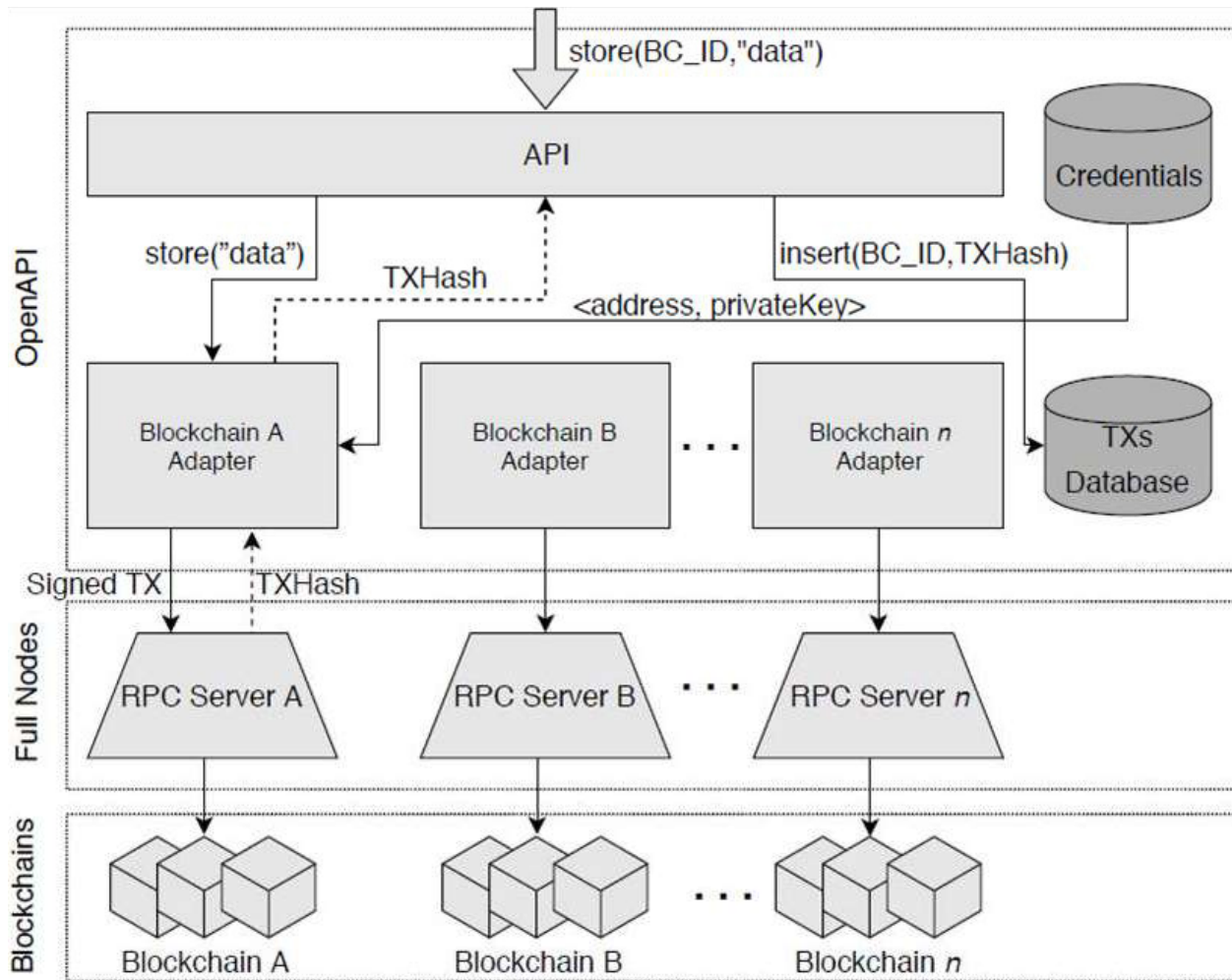
HYPERLEDGER

Applicability Scenario – Coldchain



- ❑ Medical drug companies
 - Standards compliance
 - *Store*: drug quality
- ❑ Transportation service
 - Temperature monitoring
 - *Store*: sensor readings
- ❑ Final destination and authorities
 - Standards verification
 - Data auditing
 - *Retrieve*: quality and temperature readings

Interoperability API – Workflow

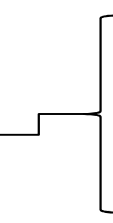


- ❑ **API Entry Point**
 - No technical details
 - Implementation hidden
- ❑ **Credentials DB**
 - Pub/Private key
- ❑ **Transactions DB**
 - Hash
 - Timestamp
 - Blockchain ID
- ❑ **Specialized Adapters**
 - BC-specific functions

API Implementation

- ❑ User **Entry-Point** Example
- ❑ Interfaces implemented in the API
 - Store data interface

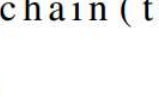
```
def store(text, blockchain):  
    adapter = Adapter[blockchain]  
    tx_hash = adapter.store(text)  
    return tx_hash
```



```
tx = create_transaction("data")  
signed_tx = sign_transaction(tx)  
transaction_hash = send_raw_transaction(signed_tx)  
add_transaction_to_database(tx_hash)  
return tx_hash
```

- Retrieve data interface

```
def retrieve(transaction_hash):  
    blockchain = database.find_blockchain(tx_hash)  
    adapter = Adapter[blockchain]  
    text = adapter.retrieve(tx_hash)  
    return text
```



```
tx = get_transaction(tx_hash)  
data = extract_data(tx)  
return to_text(data)
```

- ❑ Functions implemented in each blockchain adapter

Impact of Blockchain Heterogeneity

- ❑ UTXO and Account-based blockchains
 - Bitcoin Transactions != Ethereum Transactions
 - Each BTC transactions must have an input transaction to verify if the address has funds or not (unspent)
 - Whereas, Ethereum is Account-based, *i.e.*, an account holds the funds, and is updated on sending/receiving of coins
- ❑ Many other technical differences
 - Address generation mechanisms, fees, data structure...
 - RPC IP, port number, communication protocol, authentication...
- ❑ Need for specialized blockchain adapters!

Blockchain Adapters

- ❑ Specialized adapters **must** implement
 - Transaction **creation function**
 - Transaction **signing function**
 - **Communication** with RPC function
- ❑ Currently implemented

Blockchain	Python Library Name
Bitcoin	python-bitcoinrpc
Ethereum	web3.py
Stellar	py-stellar-base
IOTA	PyOTA
EOS	eosjs_python
Hyperledger	sawtooth_sdk
Multichain	python-bitcoinrpc

Specialized Adapter Implementation (1)

□ Bitcoin Adapter – Transaction Creation [1]

```
def create_transaction(cls, text):
    input_transaction_hash = database.find_latest_transaction(Blockchain.BITCOIN) → Input transaction
    inputs = [{'txid': input_transaction_hash, 'vout': 0}] → Unspent input
    data_hex = cls.to_hex(text) → Data encoded in hexa
    output = cls.create_transaction_output(data_hex, input_transaction_hash) → Fee estimation and change
    # Necessary so that the address is the first output of the TX
    output = collections.OrderedDict(sorted(output.items()))
    transaction_hex = cls.client.createrawtransaction(inputs, output)
    return transaction_hex
```

□ Transaction Signing

```
def sign_transaction(cls, transaction_hex):
    parent_outputs = []
    signed = cls.client.signrawtransactionwithkey(
        transaction_hex, → Recently created transaction
        [cls.key] → Private key from address
    )
    assert signed['complete'] → Verify if signing was completed
    return signed['hex']
```

<https://gitlab.ifl.uzh.ch/scheid/bc4cc/>

Specialized Adapter Implementation (2)

❑ Ethereum Adapter – Transaction Creation [1]

```
def create_transaction(cls, text):  
    transaction = {  
        'from': cls.address,   
        'to': cls.address,   
        'gasPrice': cls.client.gasPrice,  
        'value': 0,   
        'data': bytes(text, 'utf-8'),   
        'nonce': cls.get_transaction_count()  
    }  
    transaction['gas'] = cls.estimate_gas(transaction)  
    return transaction
```

Diagram annotations for `create_transaction`:

- A bracket groups `'from': cls.address,` and `'to': cls.address,` with an arrow pointing to **Equal**.
- An arrow points from `'value': 0,` to **No funds transfer**.
- An arrow points from `'data': bytes(text, 'utf-8'),` to **Data encoded in bytes**.
- An arrow points from `transaction['gas'] = cls.estimate_gas(transaction)` to **Transaction fee estimation**.

❑ Transaction Signing

```
def sign_transaction(cls, transaction):  
    signed = cls.client.account.signTransaction(transaction, cls.key)  
    return signed.rawTransaction
```

Diagram annotations for `sign_transaction`:

- A bracket groups `transaction,` and `cls.key` with an arrow pointing to **Private key from address**.
- A bracket is placed under `transaction` with an arrow pointing to **Recently created transaction**.

<https://gitlab.ifi.uzh.ch/scheid/bc4cc/>

Transaction Confirmation

- ❑ Transactions (tx) **might not be processed**
 - Refused by the network due to low fees
 - May remain in the tx pool for unknown duration
- ❑ Tx **finality**
 - Number of blocks that a tx is considered immutable
 - I.e., the probability of a 51% attack is close to zero
- ❑ Solution
 - **Wait** for X amount of time before confirming tx
 - Value of X depends on the blockchain in use
 - E.g., Bitcoin requires 6 blocks to ensure finality (3,600 s)
 - **Replace** tx with increased tx fees
 - Tx re-broadcast to increase the system's robustness

[1] I. Weber *et al.*, "On Availability for Blockchain-Based Systems," *IEEE 36th Symposium on Reliable Distributed Systems (SRDS 2017)*, September 2017, Hong Kong, China, pp. 64-73.

Data Size

- ❑ Blockchains not designed as a database!
 - Data size is limited (*i.e.*, non-transaction related data)
 - Possible solution: rely on off-chain storage, *e.g.*, IPFS

Blockchain	Maximum String Size
Bitcoin	80 Byte
Ethereum	46 kByte
Stellar	28 Byte
EOS	256 Byte
IOTA	1300 Byte
HyperLedger	20 Byte
Multichain	80 Byte



Cannot store a
SHA256 hash
(32 Byte)

Private Keys

- ❑ Blockchains based on public/private key cryptography
- ❑ Private key required to sign a transaction
 - Different address for each BC → several private keys
- ❑ Private key never leaves the server
 - RPC node receives a signed raw transaction
- ❑ However,
 - If private key stored as plain text in the DB → no encryption
 - Incentivization of invasions
 - *E.g.*, steal/ransom address funds

Basic Comparison

Blockchain	Area	Interoperability Type
Origintrail.io	Supply-chain	Sidechain/Relay
Ælf (ELF)	General purpose	Sidechain/Relay
Wanchain	Finance	Hash-locking
Herdius	Finance	Notary-scheme
Cosmos	General purpose	Sidechain/Relay
Aion	General purpose	Hash-locking
Polkadot	General purpose	Sidechain/Relay
Ark.io	General purpose	Sidechain/Relay
Crowdmachine	Cloud-computing	Sidechain/Relay
Blockstack	Identity	Sidechain/Relay
BC4CC	Supply-chain	Notary-scheme

Discussed
above

Part VII: Practical Exercise

16. Ethereum Wallet

Exercise Goals

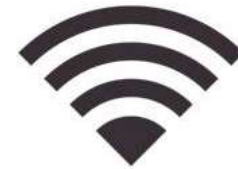
1. Differentiate between blockchain nodes
2. Create an address and send a transaction
3. Send funds and verify account balance
4. Verify a transaction sent using a blockchain explorer



Set-up Overview

- ❑ Focus on **Ethereum blockchain**
 - Ethereum Basics (cf. above) and Geth Commands here
 - Connecting to your Node
 - Connecting to CAS Blockchain

- ❑ **Connect to the CAS Wi-Fi**
 - Wi-Fi: **cas2019**
 - Password: **nopassword**



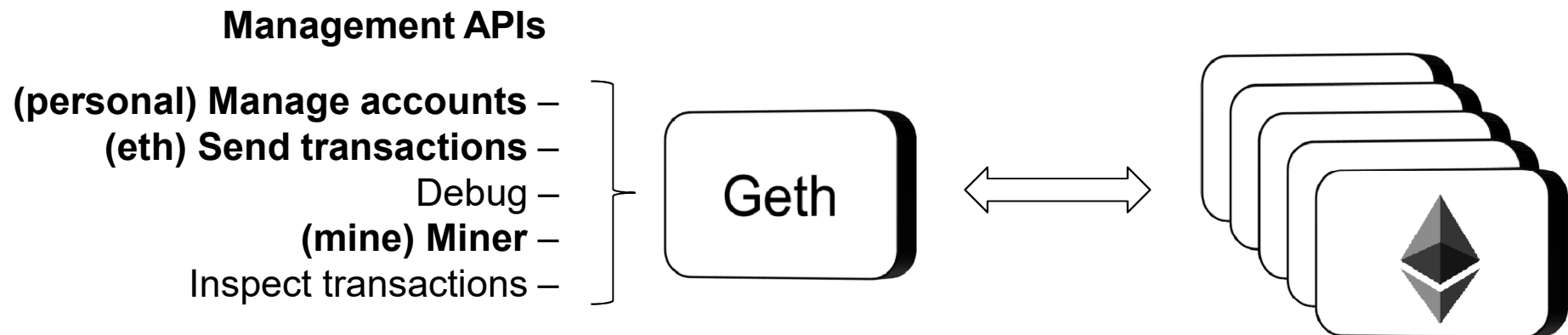
Ethereum Basics – Recalled

- ❑ Ethereum is a **general purpose** blockchain
 - Support the execution of complex Smart Contracts (SC)
 - Empowered the blockchain beyond financial applications

- ❑ An Ethereum Virtual Machine (EVM) processes and executes smart contracts.
 - Abstraction layer: the EVM connects requests (transactions) to and from the network
 - SC operations requires Gas
 - Mathematical operations, write data
 - [Table](#) of EVM OP costs

Geth Client

- ❑ Go implementation of the Ethereum Protocol
 - CLI (Command Line Interface) client
 - Entry point to the main, test and private networks



Management APIs

❑ Personal API:

- `personal.newAccount()` // create a new acc.

❑ Eth API:

- `eth.accounts[0]` //returns the first created acc.
- `eth.sendTransaction({from:acc, to:acc, value:ethers})` //send transaction to acc.

❑ Miner API:

- `miner.setEtherBase(account)` // define which account to receive ethers

Geth Commands

```
$ geth account <command> [options...] [arguments...]
```

Description	Command
Print summary of existing accounts	<i>\$ list</i>
Create a new account	<i>\$ new</i>
Update an existing account	<i>\$ update</i>
Import a private key into a new account	<i>\$ import</i>
Info about subcommands	<i>\$ [command] --help</i>

Geth Options (1)

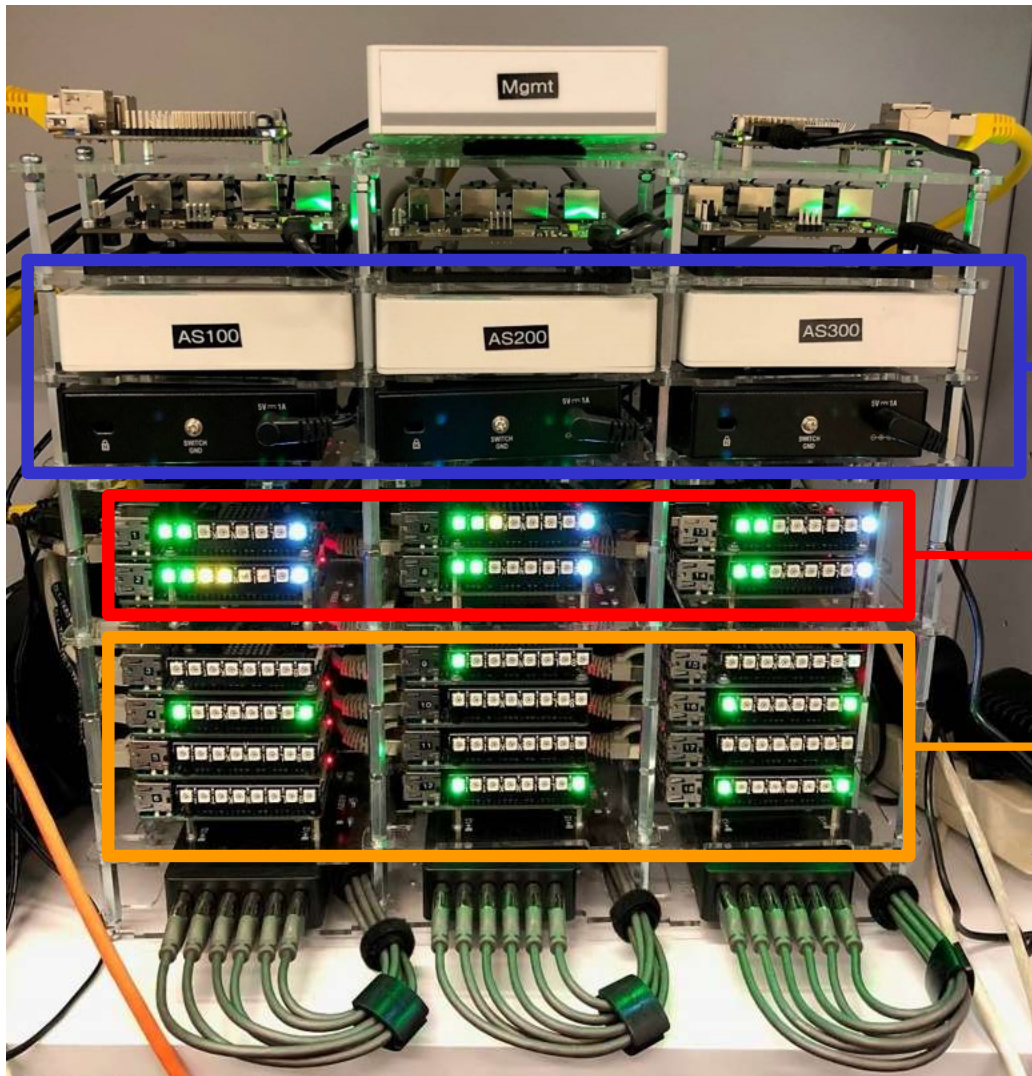
```
$ geth [options] command [command options] [arguments...]
```

Description	Command
Data directory for the databases and key-store	\$ <i>--datadir</i> "path"
Directory for the keystore (default = inside the datadir)	\$ <i>--keystore</i>
Network identifier (integer, 1=Frontier, 2=Morden (disused), 3=Ropsten, 4=Rinkeby) (default: 1)	\$ <i>--networkid</i> value
Ropsten network: pre-configured proof-of-work test network	\$ <i>--testnet</i>
Rinkeby network: pre-configured proof-of-authority test network	\$ <i>--rinkeby</i>

Geth Options (2)

blockchain sync mode ("fast", "full", or "light")	<i>\$ --syncmode "fast"</i>
Enable the HTTP-RPC server	<i>\$ --rpc</i>
Change, respectively, the address and port of the HTTP server endpoint	<i>\$ --rpcaddr value</i> <i>\$ --rpcport value</i>
Ropsten network: pre-configured proof-of-work test network	<i>\$ --testnet</i>
Rinkeby network: pre-configured proof-of-authority test network	<i>\$ --rinkeby</i>
blockchain sync mode ("fast", "full", or "light")	<i>\$ --syncmode "fast"</i>

Blockchain Testbed Overview



Network Components

Blockchain Miners Nodes

Blockchain Client Nodes
(You will be accessing these)



Connecting to Your Node

- ❑ Nodes host a **web-based** shell:

XX = Written in provided card

- <https://172.10.15.XX:4200>

- ❑ User: **student**

- ❑ Password: **123456**



Your connection is not private

Attackers might be trying to steal your information from **172.10.15.41** (for example, passwords, messages, or credit cards). [Learn more](#)

NET::ERR_CERT_AUTHORITY_INVALID

☐ Help improve Safe Browsing by sending some [system information and page content](#) to Google.
[Privacy policy](#)

Hide advanced

Back to safety

This server could not prove that it is **172.10.15.41**; its security certificate is not trusted by your computer's operating system. This may be caused by a misconfiguration or an attacker intercepting your connection.

[Proceed to 172.10.15.41 \(unsafe\)](#)

Blockchain Initialization

□ Type: `sh init-blockchain.sh`

```
student@node401:~$ sh init-blockchain.sh
rm: cannot remove 'history': No such file or directory
INFO [05-04|15:29:12] Maximum peer count           ETH=25 LES=0 total=25
INFO [05-04|15:29:12] Allocated cache and file handles database=/home/student/geth/chaindata
INFO [05-04|15:29:12] Writing custom genesis block
INFO [05-04|15:29:12] Persisted trie from memory database nodes=356 size=65.35kB time=10.523177ms
INFO [05-04|15:29:12] Successfully wrote genesis state database=chaindata
INFO [05-04|15:29:12] Allocated cache and file handles database=/home/student/geth/lightchaindata
INFO [05-04|15:29:12] Writing custom genesis block
INFO [05-04|15:29:12] Persisted trie from memory database nodes=356 size=65.35kB time=6.780313ms
INFO [05-04|15:29:12] Successfully wrote genesis state database=lightchaindata
```

□ Type: `sh start-blockchain.sh`

```
student@node401:~$ sh start-blockchain.sh
INFO [05-04|15:30:02] Maximum peer count           ETH=25 LES=0 total=25
INFO [05-04|15:30:03] Starting peer-to-peer node   instance=Geth/v1.8.2-stable-b8b9f7f4/linux-arm/go1.9.4
INFO [05-04|15:30:03] Allocated cache and file handles database=/home/student/geth/chaindata cache=768 handles=512
WARN [05-04|15:30:04] Upgrading database to use lookup entries
INFO [05-04|15:30:04] Initialised chain configuration config="{ChainID: 2019 Homestead: 1 DAO: <nil> DAOSupport: false EIP150: 2 EIP155: 3 EIP158: 1}"
INFO [05-04|15:30:04] Initialising Ethereum protocol versions="[63 62]" network=2019
INFO [05-04|15:30:04] Loaded most recent local header number=0 hash=7ae695...90bfb4 td=1
INFO [05-04|15:30:04] Loaded most recent local full block number=0 hash=7ae695...90bfb4 td=1
INFO [05-04|15:30:04] Loaded most recent local fast block number=0 hash=7ae695...90bfb4 td=1
INFO [05-04|15:30:04] Database deduplication successful deduped=0
INFO [05-04|15:30:04] Regenerated local transaction journal transactions=0 accounts=0
INFO [05-04|15:30:04] Starting P2P networking
INFO [05-04|15:30:06] UDP listener up              self=enode://26cae4bd6a1dd523861b0f3cf821f40c212744eb1eca54e3963824689f2cd14a84f2f9ce0@192.168.1.1:30303
0:30303
INFO [05-04|15:30:06] RLPx listener up             self=enode://26cae4bd6a1dd523861b0f3cf821f40c212744eb1eca54e3963824689f2cd14a84f2f9ce0@192.168.1.1:30303
0:30303
INFO [05-04|15:30:06] IPC endpoint opened           url=/home/student/geth/ipc
```

Blockchain Syncing

- ❑ If the blockchain is syncing you will see this type of information:

```
> INFO [05-04|15:30:15] Block synchronisation started
INFO [05-04|15:30:16] Imported new chain segment      blocks=137 txs=0  mgas=0.000 elapsed=383.255ms  mgasps=0.000 number=137 hash=7a970e
INFO [05-04|15:30:18] Imported new chain segment      blocks=823 txs=8  mgas=2.107 elapsed=2.307s    mgasps=0.913 number=960 hash=9ef2d3
WARN [05-04|15:30:20] Synchronisation failed, dropping peer  peer=c50eee4954f2902f err="retrieved hash chain is invalid"
INFO [05-04|15:30:24] Imported new chain segment      blocks=1  txs=0  mgas=0.000 elapsed=8.955ms   mgasps=0.000 number=1379 hash=7d733
INFO [05-04|15:30:29] Imported new chain segment      blocks=1  txs=0  mgas=0.000 elapsed=7.700ms   mgasps=0.000 number=1380 hash=7ae35
INFO [05-04|15:30:34] Imported new chain segment      blocks=1  txs=0  mgas=0.000 elapsed=7.680ms   mgasps=0.000 number=1381 hash=3698c
INFO [05-04|15:30:39] Imported new chain segment      blocks=1  txs=0  mgas=0.000 elapsed=15.529ms  mgasps=0.000 number=1382 hash=ca556
INFO [05-04|15:30:44] Imported new chain segment      blocks=1  txs=0  mgas=0.000 elapsed=10.861ms  mgasps=0.000 number=1383 hash=43356
WARN [05-04|15:30:45] Synchronisation failed, dropping peer  peer=c50eee4954f2902f err="retrieved hash chain is invalid"
INFO [05-04|15:30:49] Imported new chain segment      blocks=1  txs=0  mgas=0.000 elapsed=12.175ms  mgasps=0.000 number=1384 hash=262bb
INFO [05-04|15:30:54] Imported new chain segment      blocks=1  txs=0  mgas=0.000 elapsed=11.500ms  mgasps=0.000 number=1385 hash=c859c
INFO [05-04|15:30:59] Imported new chain segment      blocks=1  txs=0  mgas=0.000 elapsed=13.292ms  mgasps=0.000 number=1386 hash=b96c7
INFO [05-04|15:31:04] Imported new chain segment      blocks=1  txs=0  mgas=0.000 elapsed=11.004ms  mgasps=0.000 number=1387 hash=03eaa
INFO [05-04|15:31:09] Imported new chain segment      blocks=1  txs=0  mgas=0.000 elapsed=11.047ms  mgasps=0.000 number=1388 hash=19cf8
INFO [05-04|15:31:14] Imported new chain segment      blocks=1  txs=0  mgas=0.000 elapsed=11.896ms  mgasps=0.000 number=1389 hash=fdeee
WARN [05-04|15:31:16] Synchronisation failed, dropping peer  peer=c50eee4954f2902f err="retrieved hash chain is invalid"
INFO [05-04|15:31:24] Imported new chain segment      blocks=1  txs=0  mgas=0.000 elapsed=8.481ms   mgasps=0.000 number=1391 hash=fa510
INFO [05-04|15:31:29] Imported new chain segment      blocks=1  txs=0  mgas=0.000 elapsed=7.796ms   mgasps=0.000 number=1392 hash=ef345
```

- ❑ You can also see the blocks being appended
- ❑ Blockchain Explorer: <http://172.10.15.22:8000>

Practical Steps

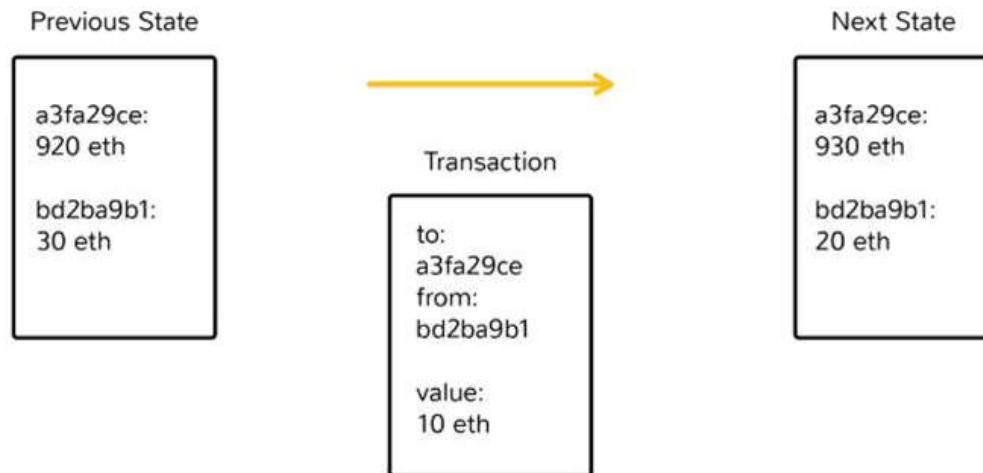
1. Create account, check balance, receive Ethers
2. Send/transfer Ethers
3. Interaction with a Smart Contract (SC)



1. First Step

- ❑ Create an account, check your balance, and receive Ethers

Ethereum



<https://www.coindesk.com/information/how-ethereum-works>

Wei	1000000000000000000
Kwei, Ada, Femtoether	1000000000000000
Mwei, Babbage, Picoether	1000000000000
Gwei, Shannon, Nanoether, Nano	1000000000
Szabo, Microether, Micro	1000000
Finney, Milliether, Milli	1000
Ether	1
Kether, Grand, Einstein	0.001
Mether	0.000001
Gether	0.000000001
Tether	0.0000000000001

<https://etherconverter.online/>

Interacting with the Blockchain (1)

- ❑ Open a new console to your node in a new tab
- ❑ Type: **geth attach http://localhost:8545**
- ❑ Now you can interact with the blockchain node

A screenshot of a terminal window. The top bar shows navigation icons and a warning 'Not secure' next to the URL 'https://172.10.15.61:4200'. The terminal text shows a user at 'student@node601:~\$' typing 'geth attach http://localhost:8545'. The output says 'Welcome to the Geth JavaScript console!' followed by system information: 'instance: Geth/v1.8.2-stable-b8b9f7f4/linux-arm/go1.9.4' and 'modules: eth:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0 web3:1.0'. The prompt '>' is followed by a cursor.

```
student@node601:~$ geth attach http://localhost:8545
Welcome to the Geth JavaScript console!

instance: Geth/v1.8.2-stable-b8b9f7f4/linux-arm/go1.9.4
modules: eth:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0 web3:1.0

> █
```

Interacting with the Blockchain (2)

- ❑ Create a new account (do not forget your password)
- ❑ Type: `personal.newAccount()`

```
> personal.newAccount()  
Passphrase:  
Repeat passphrase:  
"0x01b93652d702a967f9eb88de3cafa770858ece34"  
>
```

- ❑ Set the new account as your main geth account
- ❑ Type: `miner.setEtherbase(eth.accounts[0])`

```
> eth.accounts[0]  
"0x01b93652d702a967f9eb88de3cafa770858ece34"  
> miner.setEtherbase(eth.accounts[0])  
true  
>
```

Interacting with the Blockchain (3)

- ❑ Unlock your account to send Ethers
- ❑ Type: `personal.unlockAccount(eth.accounts[0], "password", 9999999)`

```
> personal.unlockAccount(eth.accounts[0], "typeYourPassword", 9999999)
```

- ❑ If you want to see your account address
- ❑ Type: `eth.accounts[0]`

```
> eth.accounts[0]  
"0x688fcf36ce33356a37f129e1c1c486d7dcbba28e"
```

Check your Balance

- ❑ Set the new account as your geth main account:
- ❑ Type:
 - `eth.getBalance(eth.accounts[0])`
 - Should return 0
 - Value returned in “wei”, which is the smallest unit in Ethereum
- ❑ If you want to see your balance in Ether:
- ❑ Type:
 - `balance = eth.getBalance(eth.accounts[0])`
 - `web3.fromWei(balance, "ether")`

Receive Ethers

- ❑ Access the CAS Etherpad in your browser
 - <http://172.10.15.13:9001/p/cas>
- ❑ Inform your account to receive Ethers

```
30 Write down your name and account below:
31
32 -----
33 NAME - ACCOUNT
34
35 Eder - 0x444076e9eeafe57aa8fb63e88f942ddd1986e171
36 Bruno - 0x9d29cf9313cea1f801f2725a0e2bc82100f71616
37
38
```

- ❑ Check your balance again

2. Second Step

- ❑ Transfer Ethers to another account
- ❑ On geth console type:
 - `sender = eth.accounts[0]`
 - `receiver = "<account from colleague>"`
 - `valueEth = web3.toWei(0.5, "ether")`
 - `eth.sendTransaction({from: sender , to: receiver, value: valueEth, gas:21000})`
- ❑ Check your balance
- ❑ Verify the transaction hash in the Blockchain Explorer
- ❑ You should have less 0.5 ethers in your account

17. Smart Contracts

Exercise Goals

1. Register your account in the Smart Contract
2. Retrieve an account from the Smart Contract
3. Send funds to the account retrieved



3. Third Step

□ Interacting with a Smart Contract (SC)

```
1  pragma solidity ^0.5.1;
2
3  //CAS BC 2019 Smart Contract Example
4  contract Accountbook {
5
6      address owner;
7      mapping (uint => string) accounts;
8      uint accountCount = 0;
9
10     constructor() public {
11         owner = msg.sender;
12     }
13
14     function register(string memory end) public returns (uint) {
15         accounts[accountCount] = end;
16         accountCount++;
17         return accountCount-1;
18     }
19     function getAccountCount() public view returns (uint) {
20         return accountCount;
21     }
22
23     function getAccount(uint n) public view returns (string memory) {
24         return accounts[n];
25     }
26 }
```

Each account has an unique ID

First ID is 0

New registered account receives an ID

Returns the account associated with the ID

SC Storage

- What data does the SC stores?
 - List of IDs and Account (Example)

ID	Account
0	0xC0A565abfa50349c59a4D0647834Bb6763b8fFC9
1	0xe3c01b34bBcfeDf5D5Cf090226D50767c8DBC029
2	0xd41aA1F353CdcC3d3bd22D2EcD2e4fF490f2847
3	0x5e694ceCb56d4C24a79547468B80c5ffD186B06F
4	0xAF418BAe762962dEdb6aC027A0706972260F910d
5	0xeB23D0a558482D6D706C51234F4210aef0797F5F
6	0xAB38C7C0e9FA8f30bFF71c25781e1207E6BCfb06

- Number of Accounts: 7
- New Account will have ID **7**

3.1 Step: Interact with the SC

- ❑ Create the functions to interact with the SC
- ❑ Type: `loadScript('initContract.js')`
- ❑ Retrieve the number of accounts stored in the SC
- ❑ Type: `example.getAccountCount()`
- ❑ Get the account with ID `0`
- ❑ Type: `example.getAccount(0)`
- ❑ Send 0.1 Ethers to the returned account address
- ❑ Check your balance

3.2 Step: Register your Account

- ❑ Register your account in the SC
- ❑ Type:

```
example.register(eth.accounts[0], {from: eth.accounts[0], gas: 100000})
```

The account to register

Sending from
your account

Gas paid to miners
to execute your
transaction (tx fee)

- ❑ Retrieve the number of accounts stored in the SC
- ❑ Type: `example.getAccountCount()`
- ❑ Was there any changes in the number of stored accounts?
- ❑ Find your account ID and from colleagues 😊

Part VIII: Blockchain Evaluations (Technical View)

18. To Blockchain or not to Blockchain?

Choosing a Blockchain

Posted November 22, 2015 by [Gideon Greenspan](#) in [Private blockchains](#).

Avoiding the pointless blockchain project

How to determine if you've found a real blockchain use case

Do you need a Blockchain?

The Use of Blockchains: Application-Driven Analysis of Applicability

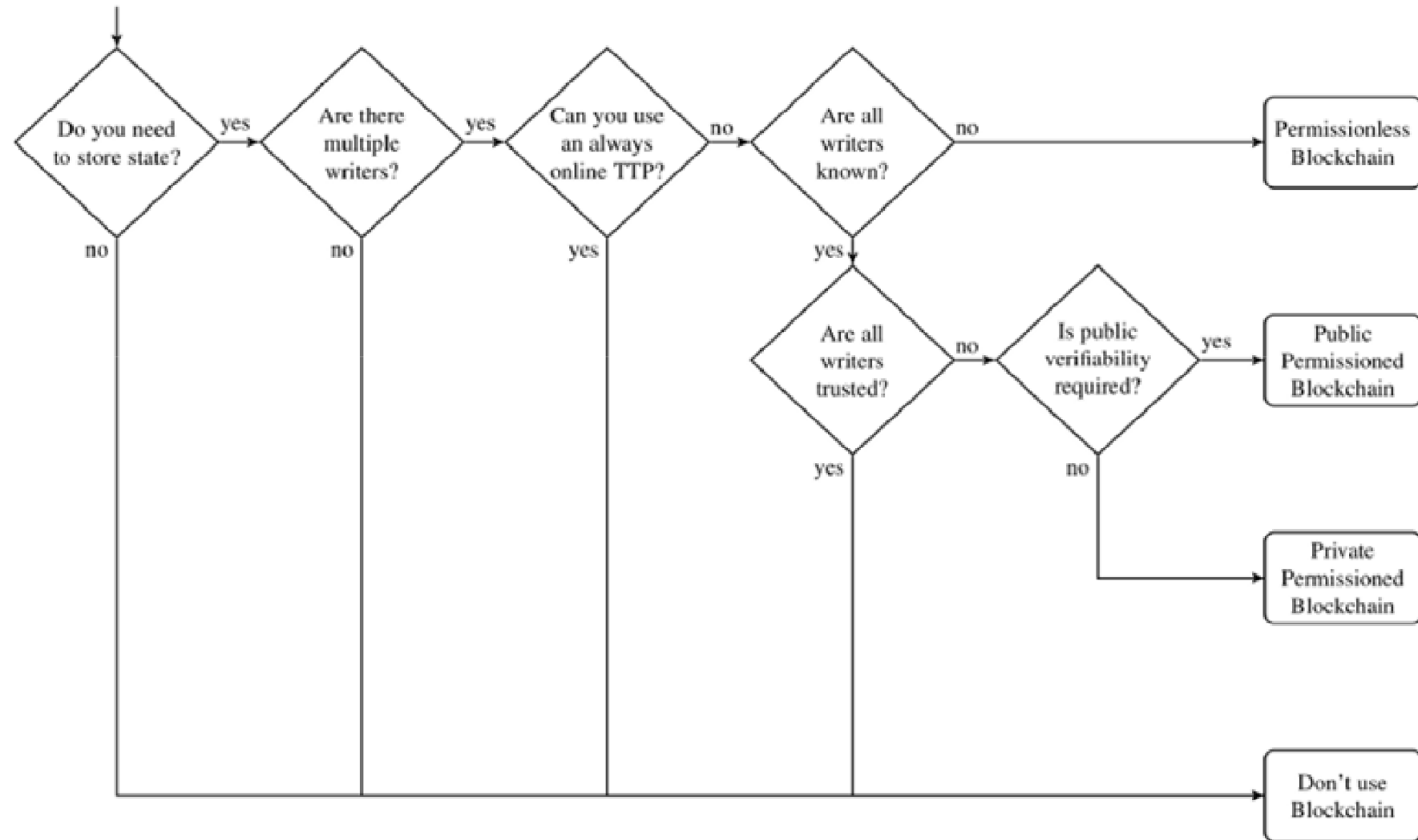
Bruno Rodrigues, Thomas Bocek, Burkhard Stiller

Communication Systems Group (CSG), Department of Informatics (IfI), Universität Zürich (UZH),
Zürich, Switzerland

Karl Wüst
Department of Computer Science
ETH Zurich
karl.wuest@inf.ethz.ch

Arthur Gervais
Department of Computing
Imperial College London
a.gervais@imperial.ac.uk

Blockchain Demand “Checker”



K. Wüst, A. Gervais, 2017

Impact Analysis

- Performance and scalability requirements impact alternative BC solutions and data bases in comparison

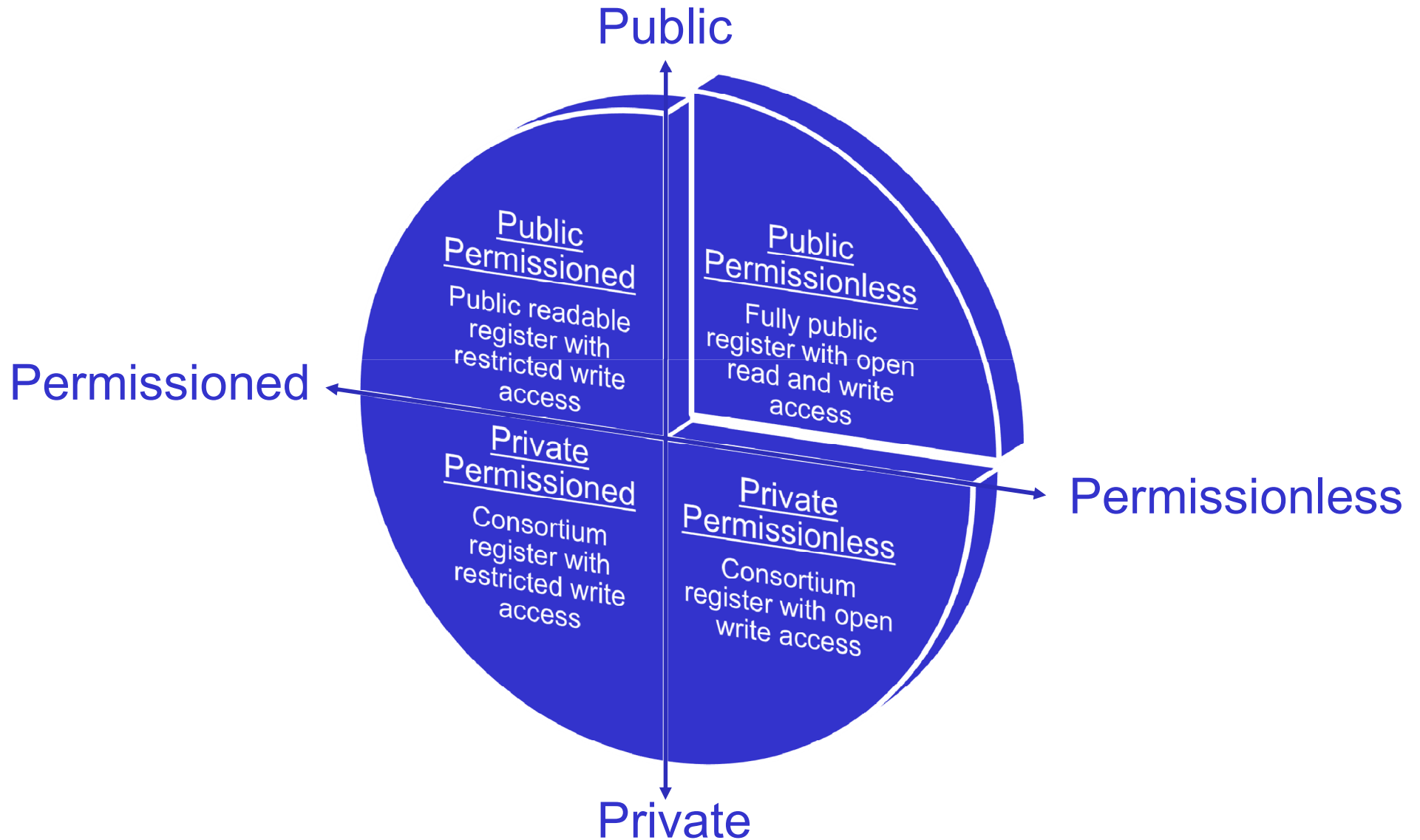
	Permissionless Blockchain	Permissioned Blockchain	Central Database
Throughput	Low	High	Very High
Latency	Slow	Medium	Fast
Number of readers	High	High	High
Number of writers	High	Low	High
Number of untrusted writers	High	Low	0
Consensus mechanism	Mainly PoW, some PoS	BFT protocols (e.g. PBFT [6])	None
Centrally managed	No	Yes	Yes

19. Comparisons and Assessment

Key BC Characteristics

- ❑ **Permissions** (write/read)
 - **Fully**: everyone in the world can participate (W/R)
 - **Partially**: selected public can participate (W/R)
- ❑ **Transparency**
 - **Fully**: all members access the history of transactions
 - **Partially**: some members access the history of transactions
 - Better a traditional DB.
- ❑ **Permanent storage, *i.e.*, immutability**
 - Transactions cannot be removed
- ❑ **Level of decentralization**
 - **Fully**: consensus with elected leader
 - **Partially**: consensus with selected leader(s)

Deployment Models



Mapping Tradeoffs to Blockchain Types

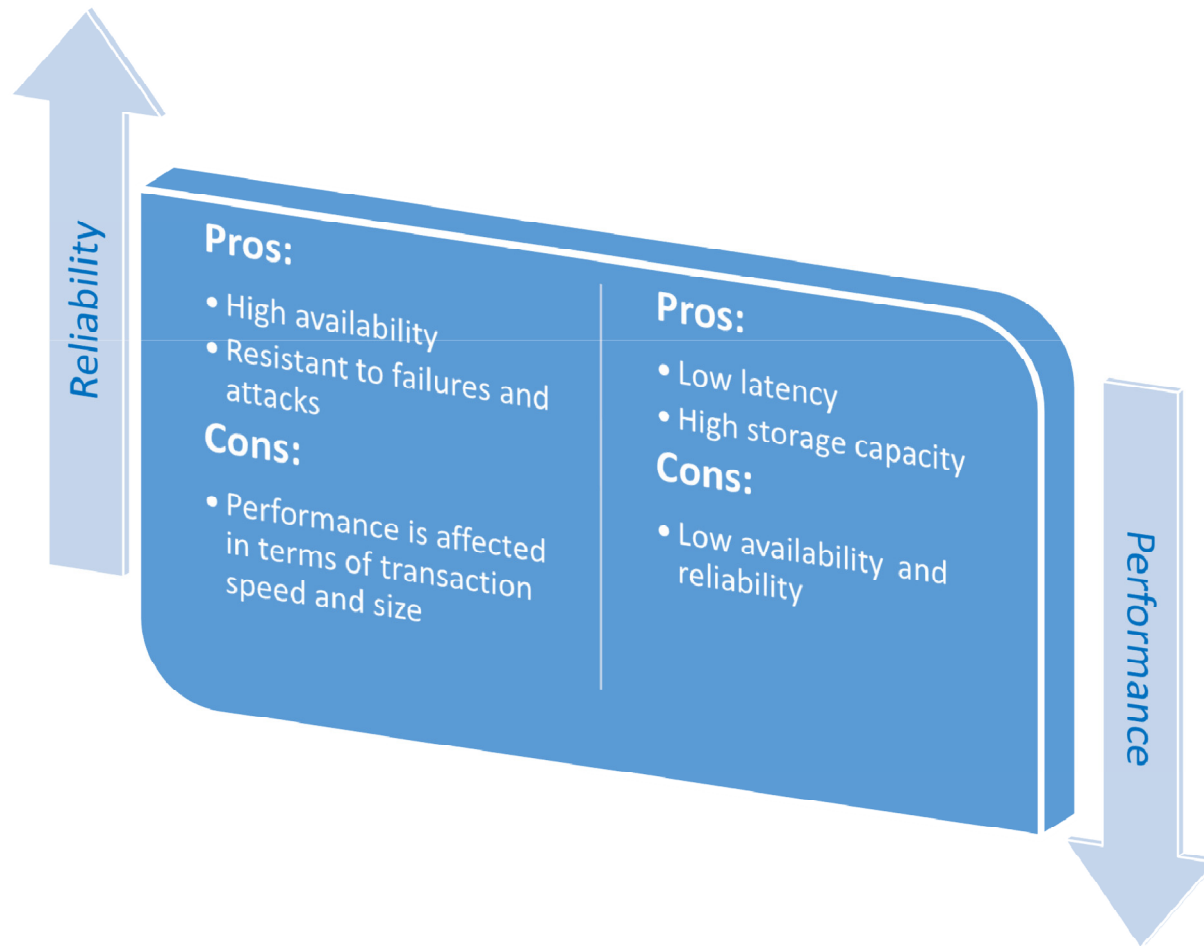
	Public Permissionless	Public Permissioned	Private Permissionless	Private Permissioned
Transparency	World visibility	World visibility	Community visibility	Role-based visibility
Control	Distributed due to the election process	Distributed but validators are defined in a selection process	Distributed but validators are defined in a selection process	Centralized based on trusted nodes
Reliability	Full replication (light nodes always rely on full nodes)	Full or partial replication (possible to define super nodes)	Full or partial replication (possible to define super nodes)	Full or partial replication (master- slave)
Performance	Slow due the consensus and replication models	Intermediate depending on consensus and replication models	Intermediate depending on consensus and replication models	Fast because its mostly centrally managed

Blockchain Comparison (Selected Examples)

Blockchain	Type	Consensus	Blocktime (s)
Bitcoin	Public	PoW	600
Ethereum	Public	PoW	15
Stellar	Public	SCP	5
EOS	Public	dPoS	0.5
IOTA	Public	IOTA	60
HyperLedger	Private	PoET	20
Multichain	Private	PoA	15
Bazo	Public	PoS	N/A

Reliability vs. Performance

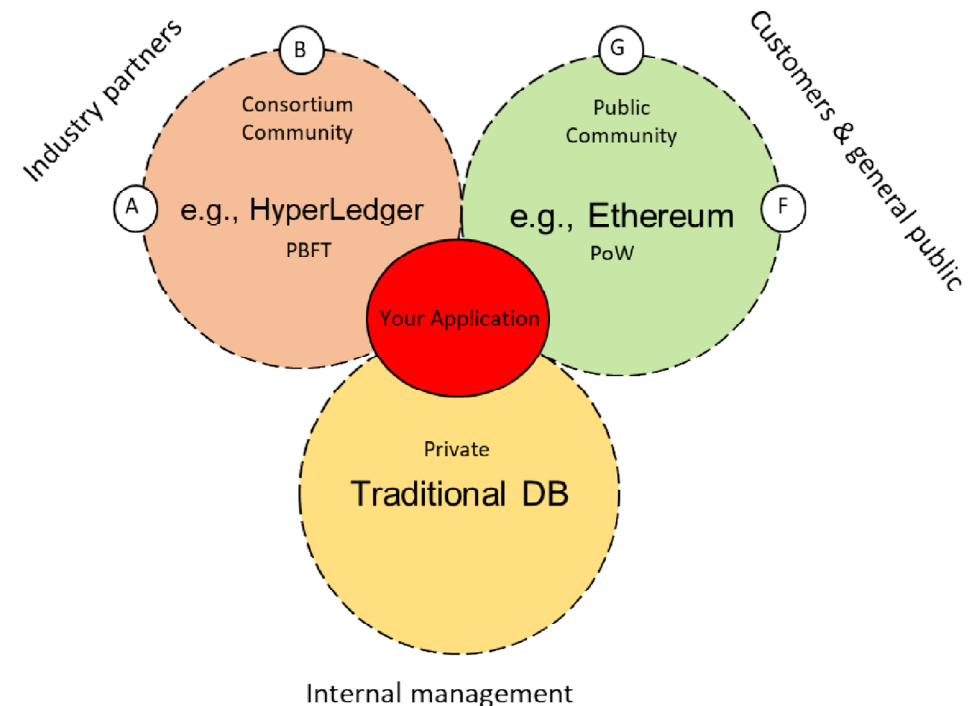
□ Analysis case-by-case



Distributed vs. Centralized Control

- ❑ Distributed control based on elected leader (e.g., PoW)
 - ❑ Partially based on selected leaders (e.g., PoA, PBFT)
 - ❑ Centralized Control based on trust (e.g., traditional databases)
-
- ❑ Multiple possibilities
 - At the same time...

Company	Model	Control
		Distributed
		Partial
		Centralized



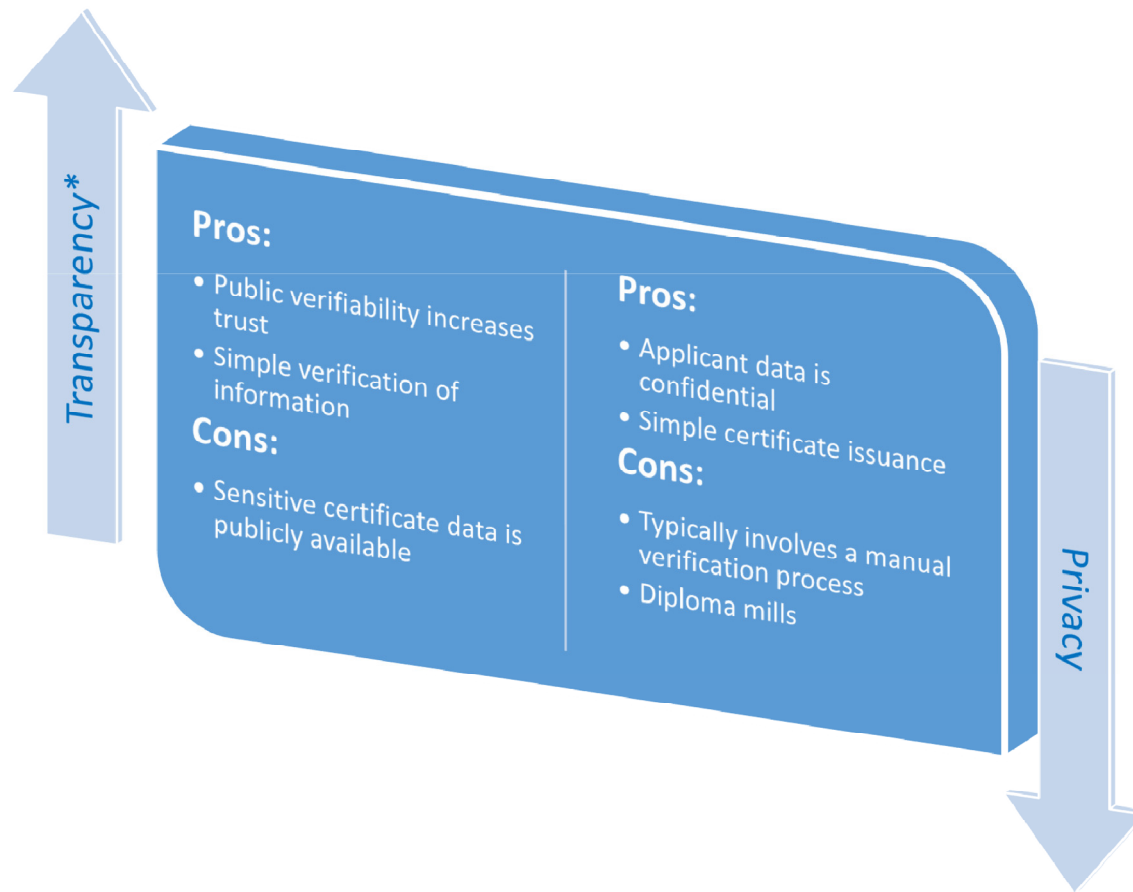
Comparison of Consensus Mechanisms

Incomplete!

Mecha-nism	Security Level	Depending on	Scalability	Remarks
BFT	“Reasonable” Leader pre-elected 51% failure	-	Medium	Trust in pre-election
dBFT	“Reasonable” Set of leaders pre-elected	-	Medium	Trust in set of leaders
PoW	High 51% attack	Hashes	Controversial	Energy consumption high, needed to ensure high security level (by design)
PoS	Unknown “Nothing-at-Stake”	PoW-based “stake”	Under discussion	“Costless” forking, thus, measurable assets needed
PoA	Identity-based	PoS, PoW	Under discussion	Authorities required
Shards	Unknown	Any PoX	Unknown	Communities, interoperability

Transparency vs. Privacy

- Analysis case-by-case
 - Possible to store only checksums of information (integrity)

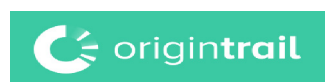


Blockchain to Database Comparison

	Blockchains (BC)	Databases (DB)
Operations	Operations	Insert, read
	Replication	Insert, read, delete, update
	Consensus	<i>E.g.</i> , master-slave model
	Invariants	Distributed transactions
Characteristics	Disintermediation	DB manager in charge of validation
	Performance	Central management (logical view), while physical distribution possible
	Storage size	All scales reachable
	Reliability	1 database, optional replicas
	Integrity	Based on failover and redundancy
	Confidentiality and Privacy	Typically based on ACID principle
	Trust	Dependent on access control regime and storage regulations of DB
	History	Database (operator) trusted
		Only partially, DB archiving function

Blockchain Interoperability

- ❑ Projects **using** or **providing** a platform for interoperable chains



The Third Generation Blockchain Network



blockstack

Blockchain	Area	Interoperability Type
Origintrail.io	Supply-chain	Sidechain/Relay
Ælf (ELF)	General purpose	Sidechain/Relay
Wanchain	Finance	Hash-locking
Herdius	Finance	Notary-scheme
Cosmos	General purpose	Sidechain/Relay
Aion	General purpose	Hash-locking
Polkadot	General purpose	Sidechain/Relay
Ark.io	General purpose	Sidechain/Relay
Crowdmachine	Cloud-computing	Sidechain/Relay
Blockstack	Identity	Sidechain/Relay

Cryptocurrency Comparison

Blockchain ≠ Bitcoin

Bitcoin	Ethereum
A maximum of 21 million bitcoins supply, halving newly generated supply every 4 years	Unlimited ethers supply
10 minutes block creation time	14 seconds block creating time
Crypto puzzle via partial SHA256 hash collision, requiring CPU time and minimal RAM; dedicated hardware, application-specific integrated circuit (ASIC) used	Crypto puzzle is variation of Dagger-Hashimoto which requires besides CPU time also RAM; GPU cards currently used
Limited scripting language, not Turing complete	General-purpose scripting language, Turing complete
Started in 2009, creator unknown (pseudonym used: Satoshi Nakamoto)	Started in 2015, initiator Vitalik Buterin
Balance based on unspent transaction outputs	Balance is account-based
Transaction costs driven by transaction size	Transaction costs driven by operations in the smart contract
Max throughput: 3-7 transactions per second	Max throughput: 23 - 25 transaction per second
Transactions created by users	Transactions created by users or smart contracts
Market capitalization: 9.9 billion USD	Market capitalization: 1 billion USD
Bitcoin created: 15.8 million BTC (2016)	Ethers created: 83 million ETH

However, Bitcoin's and Ethereum's BC can be used and applied elsewhere.

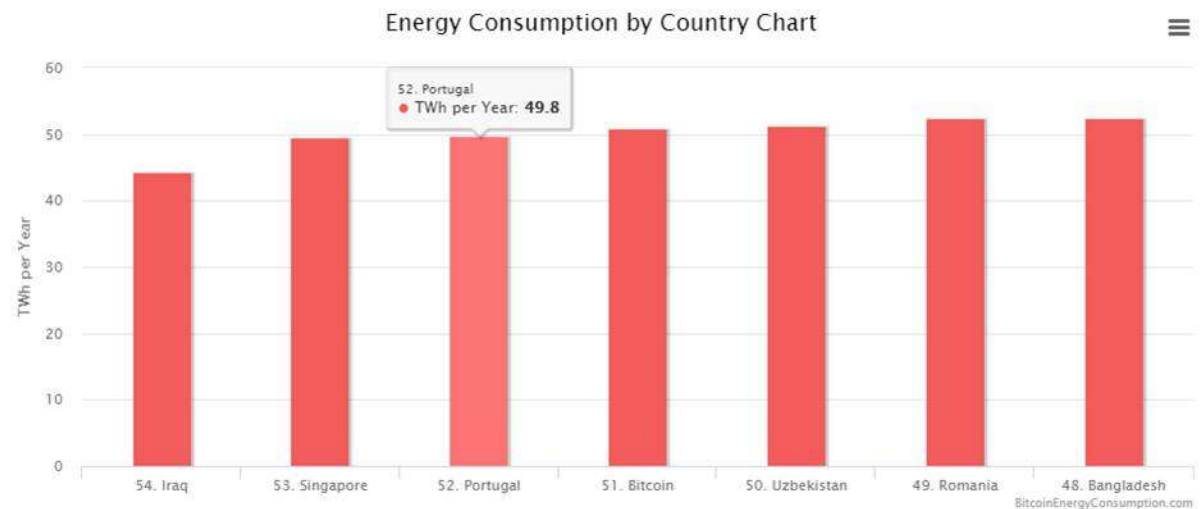
Assessment of BC Operations

- ❑ **Trust** (depends on consensus mechanism, cryptography)
 - “No” power to change or delete previously persisted block
 - Auditability, traceability for data of a transaction
- ❑ **Decentralization** (full autonomy)
 - No-one “owns”, no single instance controls the BC
 - Immutability, no single-point-of-failure
- ❑ **Integrity**
 - State of a transaction cryptographically secured (signed)
 - Privacy depends on handling of the blocks/transactions content
- ❑ **Sustainability**
 - Depending on the consensus mechanism

Example: Sustainability

- ❑ **Sustainability:** Energy efficiency of consensus mechanisms
 - Energy consumption for Bitcoin BC alone in 2019 $\approx$ Portugal production
- ❑ Proof-of-Work (PoW) does not produce socially “beneficial” output

- ❑ **Possible Solution**
 - Different Consensus Mechanisms
 - Proof-of-Useful-Work



<https://digiconomist.net/bitcoin-energy-consumption>

Assessment of Transaction Durations

❑ Bitcoin Cash:	615 s
❑ Bitcoin:	504 s
❑ Litecoin:	135 s
❑ Ethereum:	15 s
❑ Ripple (XRP):	4 s
❑ EOS:	1.5 s

Tx Durations termed “Block Time”, too

**Sample effect of BCs
very visible by the user!**

Bitcoin report, March 2018, <http://www.bitcoin.report.de>

*EOS “provides accounts, authentication, databases, asynchronous communication, and the scheduling of applications across many of CPU cores or clusters. The resulting technology is a blockchain architecture that **may ultimately scale** to millions of transactions per second, eliminates user fees, and allows for quick and easy deployment and maintenance of decentralized applications, in the context of a **governed blockchain**”, thus a private BC.*

<https://globalcoinreport.com/eos-strides-towards-recovering-its-record-against-bitcoin/>

Blockchain Benefits (1)

- ❑ Trust derived from network's protocol transparency and the assurance of immutable blocks protected by cryptographic mechanisms
 - Trust included into every single transaction
 - All parties have access to a “single” shared copy
- ❑ Personal, persistent, digital, ID ownership-enabled, which supports licensing of owned “information”
 - Higher security, better cost efficiency, optimized reconciliation process
- ❑ Cost reductions expected due to SCs automated execution on the BC

Blockchain Benefits (2)

- ❑ Application domains “not” limited as such
 - But, digital assets used determine a major application advantage
- ❑ FinTech
 - Reduction of transaction costs for financial transactions, such as within e-payment systems
 - Potential participation of many participants in financial economy
- ❑ Rights Protection
 - Immutable blocks (world-wide distributed) complicates fraud on digital data, such as stealing or misuse of land titles, Intellectual Property
- ❑ Supply Chain
 - Persisting of production, transport, and delivery characteristics

Assessment of Data Storage

- ❑ Data within a transaction is **limited**

Blokchain	Maximum String Size in tx
Bitcoin	80 Byte
Ethereum	46 kByte
Stellar	28 Byte
EOS	256 Byte
IOTA	1300 Byte
HyperLedger	20 Byte
Multichain	80 Byte

- ❑ Possible Solution

- Off-chain storage, e.g., IPFS (Inter-planetary File System)
- Hashing of data, e.g., SHA-256
- Smart Contracts (still limited and costly)

SC's Assessment

- ❑ **Trust** (depends on consensus mechanism, cryptography)
 - “No” power to change or delete previously persisted SC
 - Auditability, traceability for the processing of data of a transaction
- ❑ **Decentralization** (full autonomy)
 - No single instance controls the processing of a SC
 - Immutability, no single-point-of-failure
- ❑ **Integrity**
 - State of SC and processing results cryptographically secured
- ❑ **Costs**
 - Depend on the SC and the exchange value of tokens in use
- ❑ **Legal relevance** of “coded”, more general contracts?

Mapping Challenges (1)

	Public Permissionless	Public Permissioned	Private Permissionless	Private Permissioned
Scalability	Public usage → size growth hard to be controlled	Only selected nodes create blocks → more control over size	Blockchain designed for specific use case → controlled size	Blockchain designed for specific use case → controlled size
Data Storage	Not designed as DB → High fees, size is limited	Know writers → No fees, no size limit	Know participants → Low fees, partial size limit	Know writers → No fees, no size limit
Sustainability	PoW → computational power has no “social benefit”	PoA → Sustainable, no significant computations	PoS → Sustainable, no significant computations	PoA → Sustainable, no significant computations
Identity Management	Pseudo-anonymity, data visible → Hard to link to physical user, data encryption	Data is supposed to be visible → Ensuring integrity	Know participants → Trusted environment	Know participants → Trusted environment

Mapping Challenges (2)

	Public Permissionless	Public Permissioned	Private Permissionless	Private Permissioned
Standardization	No standard → Complex Interoperability	No standard → Complex Interoperability	No standard → Complex Interoperability	No standard → Complex Interoperability
Trust	Data in the BC → Trusted Input data → Untrusted	Know writers → Trusted	Know participants → Trusted Environment	Know participants → Trusted Environment
Economics and Regulations	No clear regulations → “Gray area”	No clear regulations → “Gray area”	Regulated by participants → Defined rules	Regulated by participants → Defined rules

(Direct) Impact Factors

Influencing Factor	Network	Distributed System	Remarks
Access: Public BC	In principle unaffected	Very many nodes possible	The <i>real</i> BC case
Access: Private BC	Unaffected	Typically “centralized”	“No” BC
Cryptography	-	Compute load affected	Mechanisms’ break?
BC size	Larger throughput	-	-
Consensus mechanisms	Availability essential	PoW: high compute load	Problem of energy efficiency unsolved
Incentive/reward mechanisms	Availability necessary	Number of nodes in BC network affected	-
Creation of blocks	Load affected	Compute load affected	-
Block size	Load affected	Compute load affected	-
Smart Contracts	-	Compute load affected	-
Governance	Affected	Affected	In multiple facets

Based on an incomplete survey, but originating from an investigation of those applications developed ourselves.

BC Advantages and Drawbacks

Selection only!

Characteristics	Advantages	Drawbacks	Remarks
Distributed	No central control, no “master” needed	No central control, no “master” exists	Censorship vs. conflict resolution!
Unknown stakeholders	Everyone can participate	Lacking control of participants’ “writes”	Application-specific needs
Open, transparent	No hiding possible	Stakeholders’ activities publicly viewable	Application-specific needs
Immutable	Once persisted, persisted forever	Wrongly deployed SC(s) not retractable	Realistic for “useful” SCs? GDPR?
Append-only	No deletions	Growing in size	GDPR compliance?
Traceable	Proof of actions	No hiding of actions	Error handling?
Technical aspect	Effective	Efficiency, energy dem.	Sustainability?
Economic aspect	Cryptocurrency (fully elect., decen.)	Impacts on economic stability, currencies, ...	Survivability of too many “tokens” or “coins”?
Legal aspect	Contracts without intermediary	“Unknown” conflict resolution instance(s)	No jurisdictional borders, enforceability?

20. Challenges and Risks

Public Blockchain Challenges!

Selection only!
Solutions seem possible

- How to handle reliably **tangible (non-digital) assets** in BC?
 - A token is represented in **bits** vs. property/real estate as **physical items**
- **Sustainability**: **Energy efficiency** of consensus mechanisms?
 - Energy consumption Bitcoin BC only in 2018 $\approx$ The Netherlands' production
- **Scalability**: BC throughput as a number of **transactions per second**, **volume of data** persisted in Mega (?) bytes, **costs**?
 - *E.g.*, BC sizes grow faster than the density of HDDs/SSDs
 - BC (always) better than a (distributed) data base? Exorbitant costs?
- **Identity management** (users, objects) and **anonymity**
- **Standardized APIs** for switching BCs for BC-based dApps
 - *E.g.*, in contrast, databases from different vendors offer “similar” APIs
- Many **economic effects** of BC-based cryptocurrencies unknown
 - Role of **national “e”-currency**, **interrelationships** of about 2100+ cryptocurr.
- **Legal/regulative** compliance, **societal/governmental** acceptance

Public Blockchain Risks!

Selection only!
Fundamental concerns

- ❑ BCs' "true semantics" depend on the input received!
- ❑ BCs' security, privacy, and reliability
 - Unknown attack vectors (& 51% attack), Programming errors in SCs
 - Alternative consensus mechanisms beyond PoW? Security at stake?
 - The breaking of currently used security algorithms
 - Long-term storage? Quantum Computing impacts?
 - Privacy: persisted data at stake? GDPR?
 - The right to forget vs. immutability
 - Transparency (public knowledge of BC) vs. privacy (private data)
- ❑ Networking infrastructure's reliability (critical infrastructures)
 - Lacking Internet connectivity for a "longer" period of time?
- ❑ Economic/legal risks (cryptocurrency/tokens/coins, BC)
 - Fraudulent profitability projections, volatility, dispute resolutions, no central entity or simply a central Customer Relationship Management

GDPR: General Data Protection Regulation

Private Blockchain Risks? Challenges?

Selection only!
Real needs?

- How to handle reliably **tangible (non-digital) assets** in BC?
 - A token is represented in **bits** vs. property/real estate as **physical items**
- **Sustainability**: Energy efficiency of consensus mechanisms!
 - Known stakeholders' tx persisting based on round-robin schemes
- **Scalability**: BC throughput as a number of transactions per second, volume of data persisted in “any” volume
 - *E.g.*, BC sizes grow faster than the density of HDDs/SSDs
 - BC (always) better than a (distributed) data base?
- **Service** offered: **trusted third party time stamping service**
 - Cryptographically sealing electronic data with a tamperproof digital signature and BC timing information (well known from the past)
- BCs' **security, privacy, and reliability**
 - Depending on known stakeholders' agreements (like within today's IT)
 - With a **very different trust model** as compared to public BCs

Broader Challenges? Risks?

- ❑ Discontinuation of “cash” and replacement with a cryptocurrency – loss of any anonymity?
- ❑ Cryptocurrencies only – failure of energy supply?
Sustainability? How are payments made?
 - On-the-spot payments? Input channels are “delicates”
- ❑ Decentralization of token handling – full control of the individual? No central “CRM” for sure. CRM: Customer Relation Management
 - Full decentralization today impossible, thus, “central” players
- ❑ Reliability of IT systems – here of “billions” of handhelds part of the game?
 - Different OSes? Different HW? Hundreds of applications? IF?

21. Expected Impacts and Conclusions

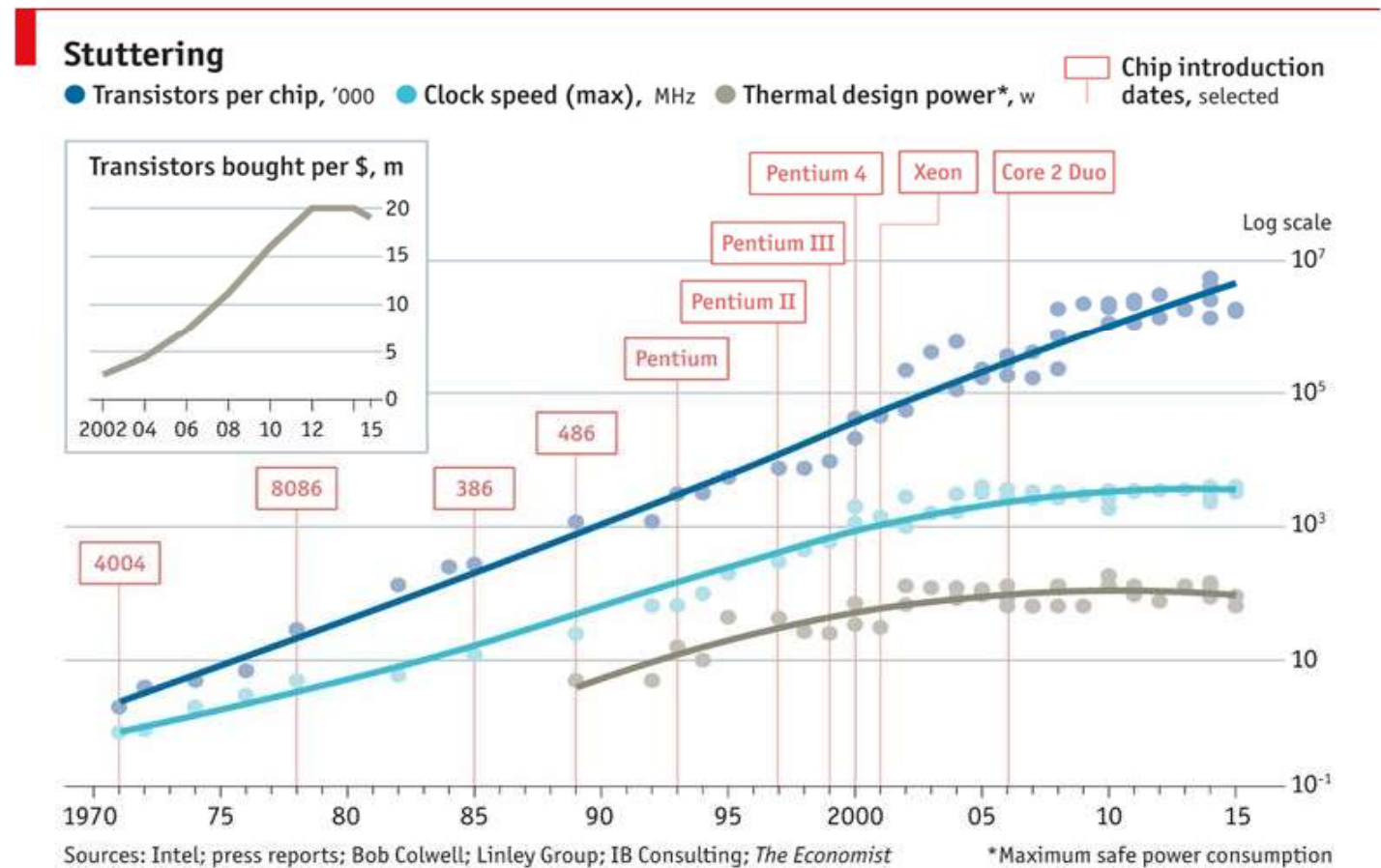
Application Trade-offs

- ❑ Trade-offs based on technical Blockchain characteristics
 - Distributed vs. centralized control
 - No central authority (PoW) or trusted nodes (PBFT)
 - Performance vs. reliability
 - BC offers slow throughput but more robustness than traditional DBs
 - Privacy vs. transparency
 - More transparency (trust) and less confidentiality
- ❑ Limited storage
- ❑ “Unknown” regulations
 - Different countries, different regulations
- ❑ Lack of standards
 - Blockchain 4.0 target

B. Rodrigues, T. Bocek, B. Stiller: The Use of Blockchains: Application-Driven Analysis of Applicability; in: P. Raj, G. Deka (Edt.), "Blockchain Technology: Platforms, Tools and Use Cases, Volume 111 (Advances in Computers)", Springer, Waltham, MA, U.S.A, No. 111, September 2018, pp 163–198

General Technology Trends (1)

- ❑ **Moore's Law:** The number of transistors doubles every 18 month.
- ❑ Hard to maintain, likely to end in 2021
- ❑ Processing, hashing limited?

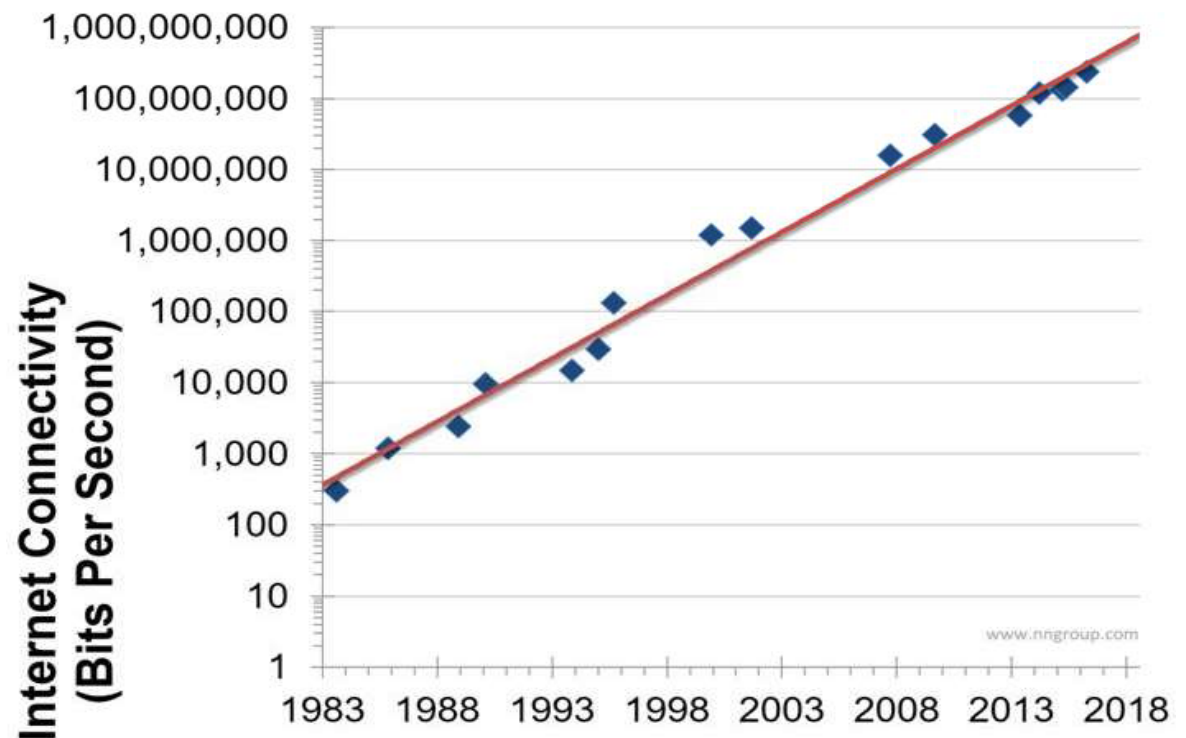


<https://www.techrepublic.com/article/moores-law-dead-in-2021-heres-what-the-next-revolution-will-mean/>

General Technology Trends (2)

- ❑ **Nielsen's Law:** A high-end user's connection speed grows by 50% per year.
- ❑ However, bandwidth grows slower than computer power
 - Zmap completes scan of IPv4 address space in under 5 min

<https://zmap.io/>



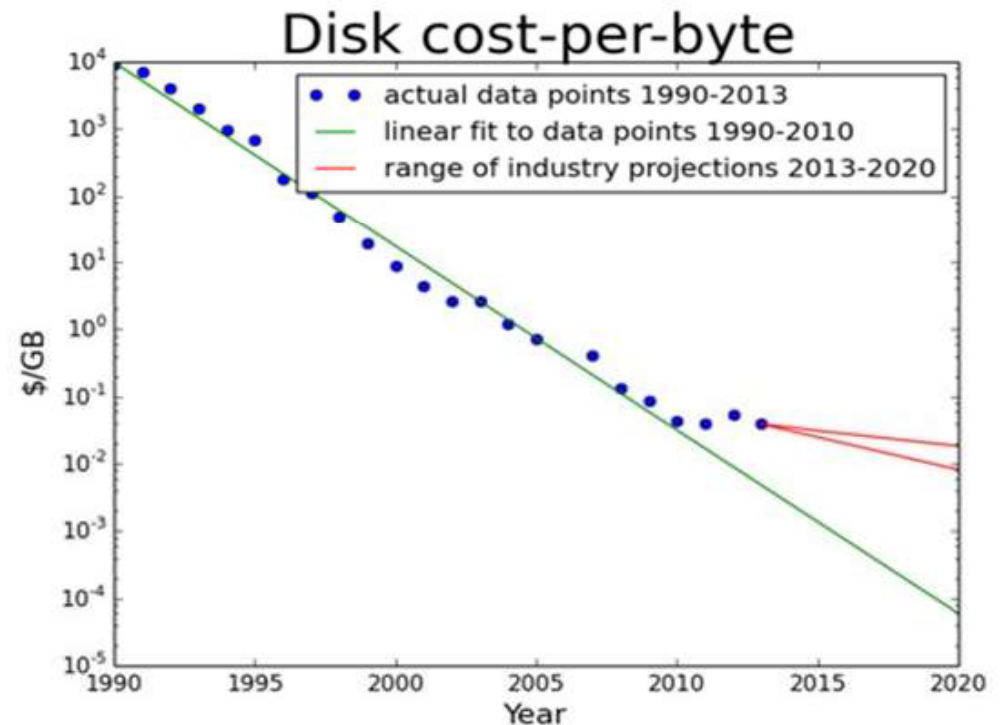
<https://www.nngroup.com/articles/law-of-bandwidth/>

General Technology Trends (3)

- ❑ **Kryder's Law**: Disk density doubles every 13 month.
- ❑ «Soon hard drives will migrate into phones, still cameras, PDAs, cars and everyday appliances»

<https://www.scientificamerican.com/article/kryders-law/>, August 2005

- ❑ User behavior changes
 - SSD, speed is important
 - Clouds: e.g., Dropbox
 - Streaming: e.g., Spotify



<http://blog.dshr.org/2016/05/the-future-of-storage.html>

The Renaissance of the DAOs

- ❑ Decentralized Autonomous Organizations (DAO)
- ❑ 2016: The ICO “The DAO” was hacked and 15% of all available Ether were locked up in SCs
<https://medium.com/swlh/the-story-of-the-dao-its-history-and-consequences-71e6a8a551ee>
 - Minimal developments in the area of DAOs since then
 - But, clear focus on scalability, security, and privacy
- ❑ 2019: DAO renaissance and On-Chain Governance – welcome back?!?!
 - **MolochDAO**: Minimum Governance Functionality On-Chain, enables flexible entry and exit, a Simple Voting Mechanism and a Guild Bank to fund proposals (e.g., to advance the Ethereum ecosystem)

<https://medium.com/@simondlr/the-moloch-dao-collapsing-the-firm-2a800b3aa2e7>

Zero-Knowledge Proof (ZKP) Systems

- ❑ Optimizations in ZKP systems enable
 - Novel scalability and privacy solutions
- ❑ **Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge (zk-SNARKs)**
 - Require a “trusted” setup phase, which, if compromised, enables fake proof generation
 - Privacy: Enables private and shielded payments
- ❑ **Zero-Knowledge Succinct Transparent Arguments of Knowledge (zk-STARKs)**
 - Do not require “trusted” setups and are post-quantum resistant
 - Scalability: Enables provable and verifiable computation off-chain

<https://z.cash/technology/zksnarks/>

<https://medium.com/coinmonks/zk-starks-create-verifiable-trust-even-against-quantum-computers-dd9c6a2bb13d>

Development Trends (2019)

❑ Libraries and developers

- Redundant development efforts led the push toward libraries
 - [Libp2p](#): Modular P2P networking stack (used by IPFS and others)
 - [Matrix.org](#): Open standard for secure, interoperable, decentralized real-time E2E encrypted communications
 - [OpenZeppelin](#): Establishing Ethereum SC security best-practices via open-source, modular, security-focused Ethereum SCs and SDKs

❑ Programming languages

- Shift from [Go](#) to [Rust](#): Open-source Systems Programming Language initialized by Mozilla
 - Designed toward reliability and efficiency (memory usage, ownership)
- [WebAssembly \(Wasm\)](#) as compilation target (W3C)
 - Low-level binary instruction format for stack-based Virtual Machines

Observations

- ❑ Distributed apps show a **larger coordination effort**
- ❑ Blockchain's **cost/benefit perspective (incentives)**
 - Costs: known in advance (e.g., HW, VM, network, setup, or fees)
 - Benefits: less central infrastructure plus “unknown” soft factors
- ❑ Once risks are mitigated and **legal/regulatory constraints** are known, applications may benefit from blockchains
 - Addressing value destruction problems
 - ICO vs. traditional emission rules
 - Country-specific rules and handling will become obstacles
- ❑ Dedicated **applications maintaining inherently digital assets** do show a much larger (disruption) potential
 - Solving the “media break” may increase this potential



Conclusions (1)

1. Blockchains **do** show a logical evolution of linked lists, however, public BCs “exaggerate” processing demands
 - By design: Proof-of-Work (PoW), but this ensures immutability
2. Blockchains **do not** have a relevant impact on general networking, however, “unreliable” networks do have an impact on the BC
 - Especially in case of longer outages
3. Blockchains **do not** have a relevant impact on Distributed Systems, however, the full decentralization is (very) costly (PoW) or still not secure (other Po“X”)
 - Especially (in case of PoW) BC-related energy demands

Modulo “measurable” effects, but at no impact

Conclusions (2)

4. Blockchains **do not** solve the media break, thus, leading to a lack of trust for tangible assets, key: trust in “sensor”
 - Major trust model differences for public vs. private blockchains
5. The technical future of blockchains is still based on **security ingredients** of today’s technology, however, long-term security management is key
 - Unknown impact of quantum computing (certainly on all IT!)
6. Blockchains show **no revolution**, but a typical Computer Science (Abstract Data Type) **evolution** of linked lists
 - But “distribution” of consensus **does not always** make sense
 - Any system as of the past has **not** been replaced fully by a BC

Selected References (1)

- ❑ A. Antonopoulos: *Mastering Bitcoin – Unlocking Digital Cryptocurrencies*; O'Reilly, Sebastopol, California, U.S.A., 2015
- ❑ T. Bocek, B. Rodrigues, T. Strasser, B. Stiller: *Blockchains Everywhere - A Use-case of Blockchains in the Pharma Supply Chain*; IFIP/IEEE International Symposium on Integrated Network Management (IM 2017), Lisbon, Portugal, May 2017
- ❑ T. Bocek, B. Stiller: *Smart Contracts - Blockchains in the Wings*; in: Claudia Linnhoff-Popien, Ralf Schneider, Michael Zaddach (Edts.) “Digital Markets Unleashed”, Springer, Germany, 2016
- ❑ C. Chinchilla (Edts.): *A Next-Generation Smart Contract and Decentralized Application Platform*; Ethereum White Paper, June 2019, <https://github.com/ethereum/wiki/wiki/White-Paper>
- ❑ M. Herlihy: *Blockchains from a Distributed Computing Perspective*; ACM CCR, Vol. 62, No. 2, February 2019
- ❑ S. Nakamoto: *Bitcoin: A Peer-to-Peer Electronic Cash System*; <https://bitcoin.org/bitcoin.pdf>, 2009
- ❑ A. Narayanan, J. Bonneau, E.W. Felten, A. Miller, S. Goldfeder, J. Clark: *Bitcoin and Cryptocurrency Technologies*; Princeton University Press, July 19, 2016, <http://bitcoinbook.cs.princeton.edu/>
- ❑ M.E. Peck: *Do You Need a Blockchain?* IEEE Spectrum, September 29, 2017, <https://spectrum.ieee.org/computing/networks/do-you-need-a-blockchain>

Selected References (3)

- ❑ S. Rafati, L. Pelloni, S. Wullschleger, A. Schaufelbühl, T. Bocek, B. Stiller, L. Rajendran: *A Blockchain-based Scientific Publishing Platform*; IEEE International Conference on Blockchains and Cryptocurrencies (ICBC 2019), Seoul, South Korea, May 2019
- ❑ B. Rodrigues, B. Stiller: *Cooperative Signaling of DDoS Attacks in a Blockchain-based Network*; ACM SIGCOMM 2019, Demo, Beijing, China, August 2019
- ❑ E. Scheid, T. Hegnauer, B. Rodrigues, B. Stiller: *Bifröst: a Modular Blockchain Interoperability API*; 44th Conference on Local Computer Networks (LCN 2019), Osnabrück, Germany, October 2019
- ❑ N. Szabo: *Smart Contracts: Building Blocks for Digital Markets*; http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html, 1996
- ❑ D. Tapscott, A. Tapscott: *Blockchain Revolution*; Portfolio/Penguin, 2016
- ❑ B. Rodrigues, T. Bocek, B. Stiller: *The Use of Blockchains: Application-Driven Analysis of Applicability*; in: Pethuru Raj, Ganesh Deka (Edts.), “Blockchain Technology: Platforms, Tools and Use Cases”, Volume 111 (Advances in Computers), Springer, Waltham, MA, U.S.A, No. 111, September 2018
- ❑ T. Spies: *Blockchain Mechanics*; <http://slideplayer.com/slide/11374043/>
- ❑ A. Welfare: *Commercializing Blockchains – Strategic Applications in the Real World*, John Wiley & Sons Ltd., Chichester, West Sussex, U.K., 2019
- ❑ K. Wüst, A. Gervais: *Do you need a Blockchain?* Crypto Valley Conference on Blockchain Technology (CVCBT 2018), Zug, Switzerland, June 2018

Thank you for your attention!

**End of Day 2:
Blockchain Platforms and Architectures**