OST
Eastern Switzerland
University of Applied Sciences

Distributed Systems Advanced & Blockchain

Repetition VSS Part 2

Thomas Bocek

19 September 2020

Lecture 6

Authentication

- Authentication
  - Single-factor authentication
    - E.g. password
  - Multi-factor authentication / 2FA
    - E.g. password **and** software token
- Password rules
  - Don’t use:
    - The name of a pet, child, family member, or significant other
    - Anniversary dates and birthdays
    - Birthplace
    - Name of a favorite holiday
    - Something related to a favorite sports team
    - The word "password“
  - Don’t’ reuse passwords, use password managers

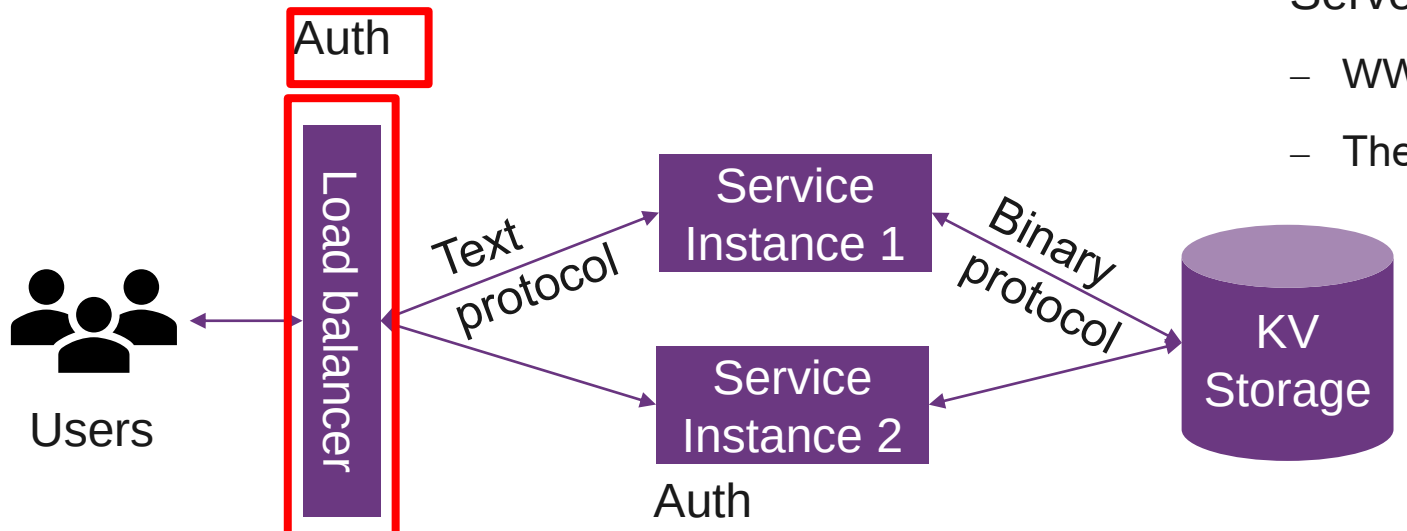
- Don’t enter passwords on unencrypted sites
- Password length: password cracking with 5000$ in 2018 with hashcat
  - Hashtype: WPA/WPA2: 1190.5 kH/s

Pw len	com	time
6	11m	0.3d
7	656m	21d
8	38b	3.4y
9	$7 * 10^{15}$	197 y
10	$4 * 10^{17}$	11k y
11	$2 * 10^{19}$	665k y
12	$1 * 10^{21}$	38m y

OST

Authentication

- Software token: TOTP (Time-based One-time Password)
  - Often used as 2nd factor, e.g., in andOTP
  - Based on keyed-hash message authentication code
  - ~ hash(key + message)
  - TOTP: message → T = (Current Unix time - T0) / X (X default is 30sec)



- Challenge Task
  - In service, e.g., your HTTP server
  - In load balance, e.g. traefik
  - Choices…
- Load balancer
  - Basic Auth, only with HTTPS, no logout!
  - Server will reply with header
    - WWW-Authenticate: Basic realm="restricted area"
    - The user will see the information "restricted area"

OST

Authentication

- Digest Auth
  - Also available in traefik
  - Hash + nonce, against replay attacks
  - Server sends
    - WWW-Authenticate: Digest realm="testrealm@host.com",

      …

      nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",

      …
  - Client sends in HTTP header
    - Authorization: Digest
      username="Alice",

      realm="testrealm@host.com",

      nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",

      uri="/dir/index.html",

      …

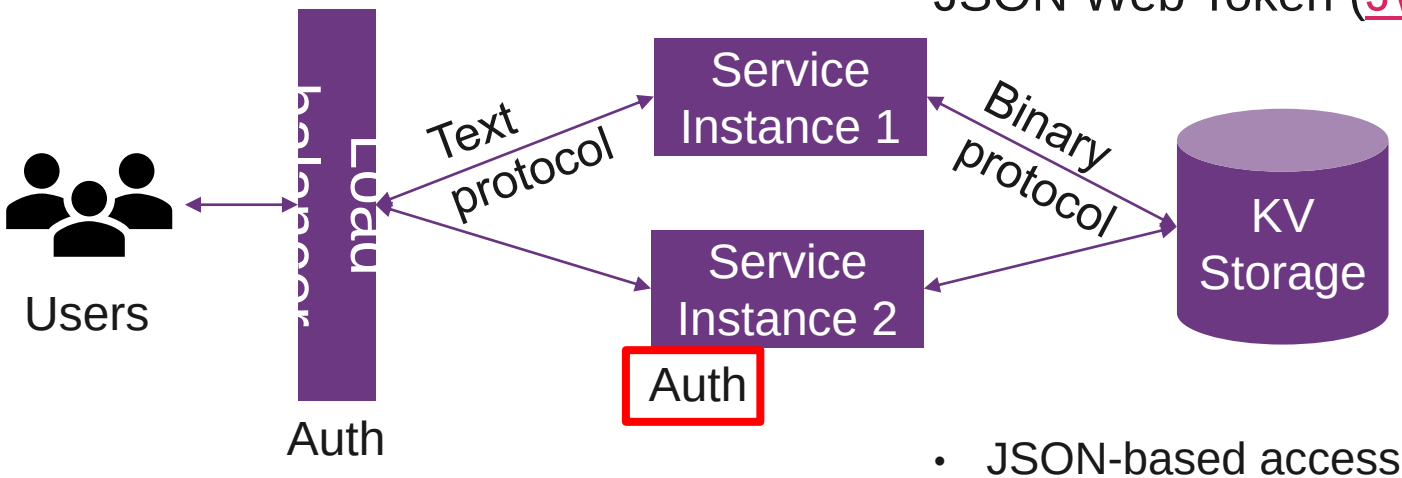
      response="6629fae49393a05397450978507c4ef1"
  - Advantages
    - PW not in clear text
    - Nonce for replay protection for client and server
  - Disadvantages
    - Browser L&F
    - Strong security is optional
    - Cannot use scrypt or bcrypt to store PWs
- Public/private key
  - Client-side certificate in nginx:

```
# This will return a 403 to all clients without a proper certificate
if ($ssl_client_verify != "SUCCESS") { return 403; }

# This tells Nginx what CA to verify against
ssl_client_certificate /tmp/ca.pem;

# This tells Nginx to verify clients
ssl_verify_client on;
```


OST

Authentication

- Session-based authentication (stateful)
  - org.apache.catalina.session.StandardManager based on ConcurrentHashMap
- JSON Web Token (JWT) (stateless)

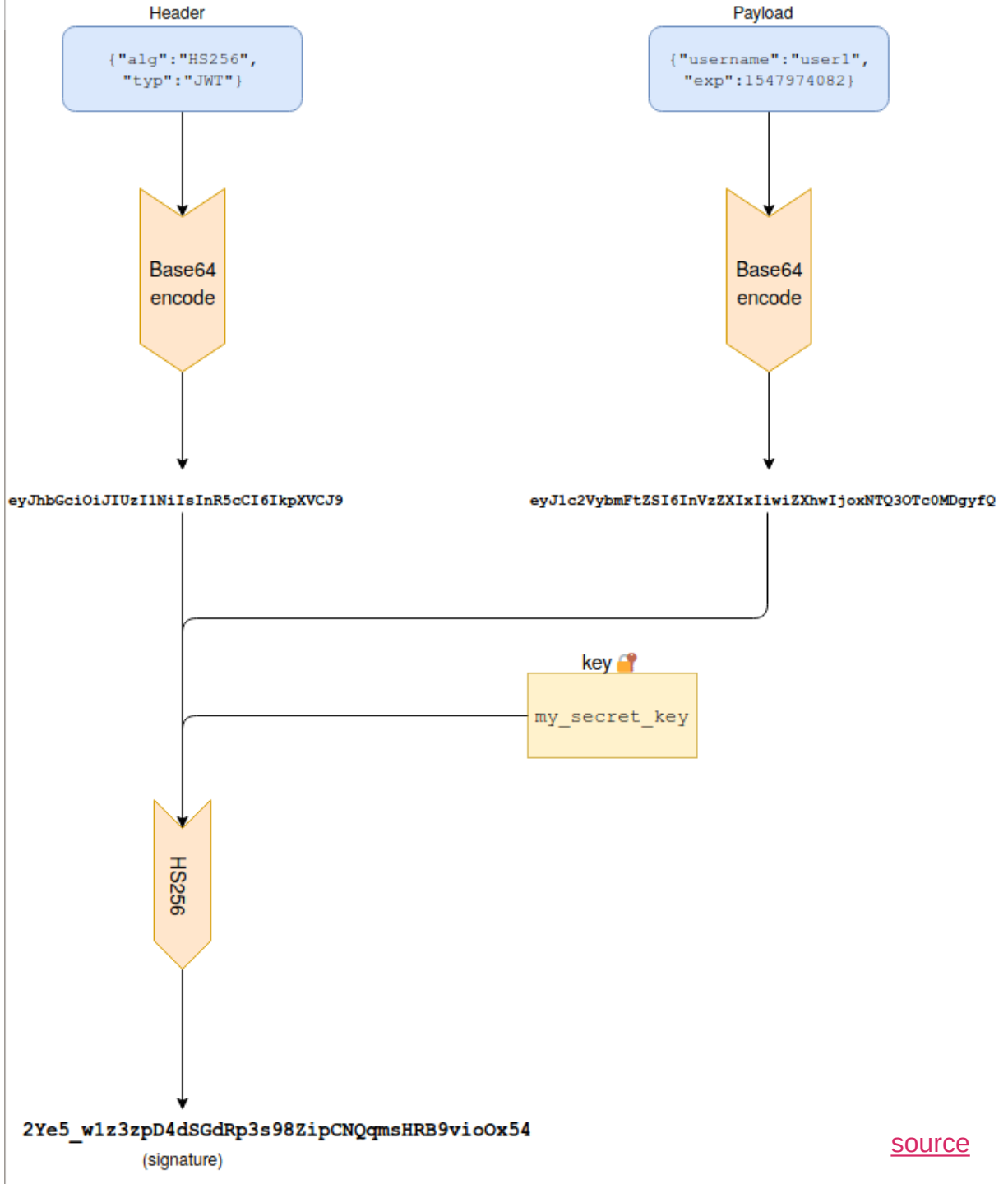


- E.g., spring-boot
  - Simple login app
  - JSESSIONID, Session information, that user was successfully authenticated: memory
- JSON-based access tokens
  - All server instances know a secret token
  - When user logs in, server responds with token:
  - Authorization: Bearer <token>
  - const user_token = base64urlEncoding(header) + '.' + base64urlEncoding(payload) + '.' + base64urlEncoding(signature)

Authentication

- JSON-based access tokens
  - Header: {"alg" : "HS256"}
  - Payload: {"user" : "tom", "role" : "admin", “exp" : 1422779638}
- Signature (simple): keyed-hash message
  - ~hash(base64(header)+base64(payload) + secret token)
- Client can store user_token in
  - localStorage.setItem(“token", userToken);
- Example in golang with JWT
  - Tutorial: here and here

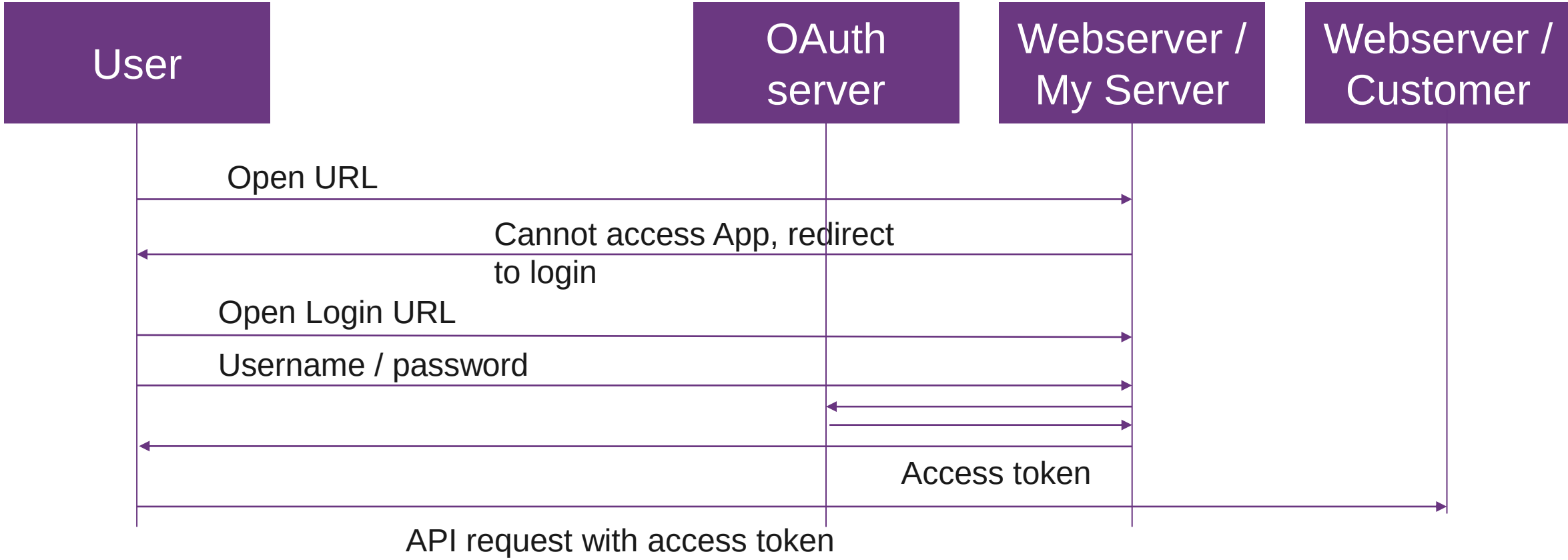
- OAuth - protocol for authorization 3rd party integration
  - Grant access on other websites without giving them the passwords



source

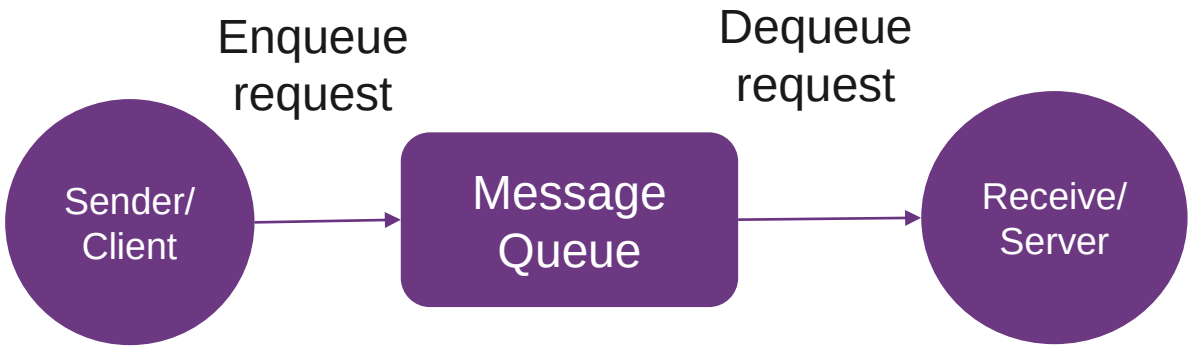
OAuth

- Example OAuth flow
  - Tokens have expiry date, tokens can be refreshed

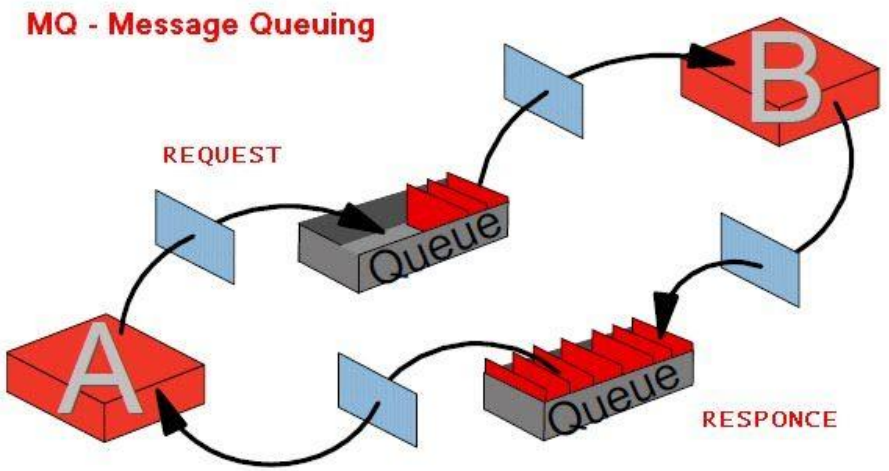


Message Queues

- Message Queues in Distributed Systems
  - Communicating without continuous connection
  - Decoupling: hardware is not reliable (lecture 1)
  - Systems can be heterogeneous (small, large machines)
- Messages placed on queue by one systems, taken by another system



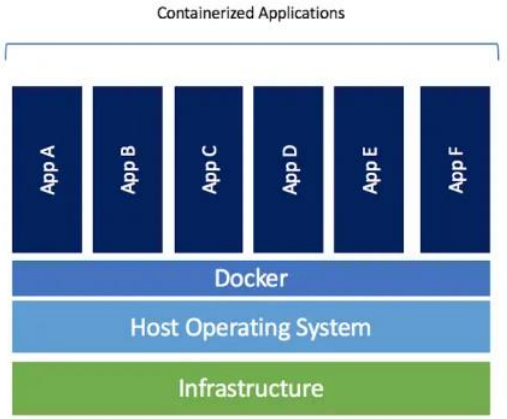
- Channels in Go (synchronization of goroutines)
- Message queues and mailboxes
  - Similar to Email
  - Can be used for IPC
  - Asynchronous communications protocol
    - Sender and receiver do not need to interact with message queue at the same time
  - Messages in queues may be persisted

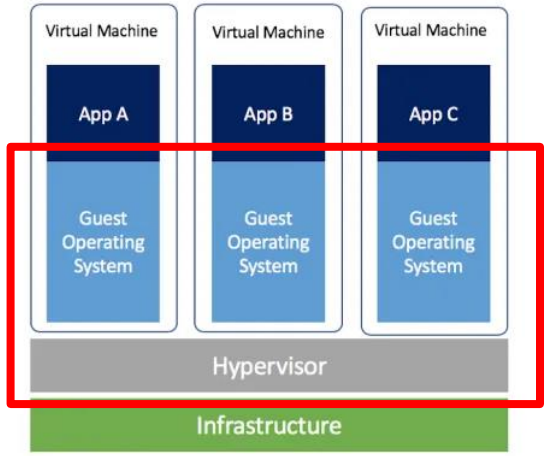


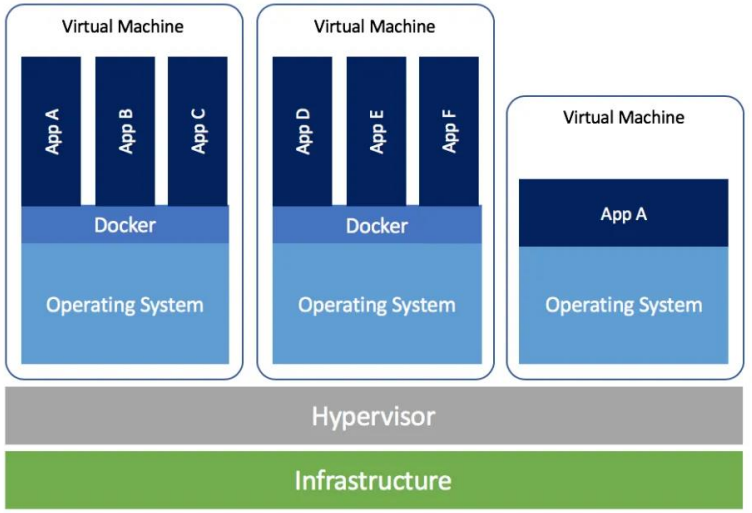
Lecture 7

Introduction

- Docker is a software delivery framework
  - Packages software into containers
  - Provides OS-level virtualization
  - Containers are isolated from each other
    - Communicate over well-defined channels
  - Docker, Inc is the company behind its tooling
  - Alternatives: Podman
- OS virtualization (Containers, e.g., Docker) vs virtual machine (VirtualBox)
  - Containers Are More Agile than VMs
    - “works on my hypervisor”
  - Containers Enable Hybrid and Multi-Cloud Adoption
  - Integrate Containers with Your Existing IT Processes
  - Containers Save on VM Licensing
    - Kernel-based Virtual Machine (KVM)







Comparison

Container

+ Reduced IT management resources

+ Reduced size of snapshots 2MB vs 45MB

+ Quicker spinning up apps

+ / - Available memory is shared

+ / - Process-based isolation (share same kernel)

+ Container app is infrastructure agnostic

Use case: complex application setup, with container less complex configuration

Providers: ECS, Kubernetes Engine, Docker on Azure (or Kubernetes)

Virtual Machine

+ App can access all OS resources

+ Live migrations

+ / - Pre allocates memory

+ / - Full isolation

Type 2 VM providers (Vmware, VirtualBox, Paralles, KVM) (Type2 is hosted – Type 1 is bare metal)

Use case: better hardware utilization / resource sharing

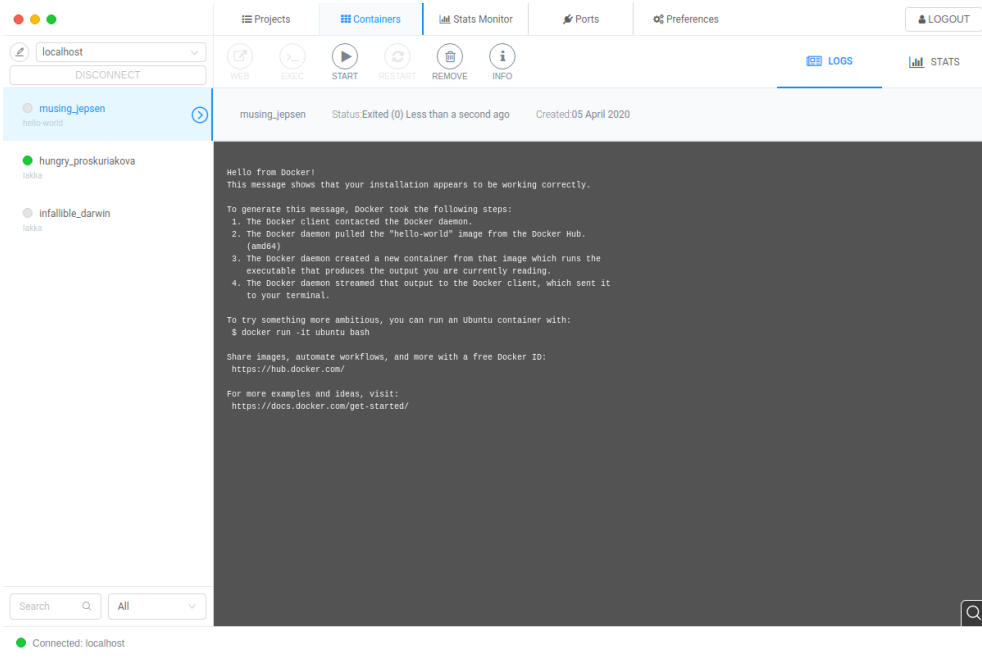
EC2, Virtual Machines, Compute Engine, Droplets

Market shares, market hares

OST

Docker Examples

- Install docker [ubuntu, Mac, Windows]
  - `docker run hello-world`
  - Fetches the hello world example from docker hub
  - No version provided – latest
  - Docker Hub: container image repository
    - Community / official
    - Alpine
  - `docker save hello-world –o test.tar`
  - `tar xf test.tar`
  - `tar xf cdccdf50922d90e847e097347de49119be0f17c18b4a2d98da9919fa5884479d/layer.tar`
  - `./hello`
  - For demonstration purposes
- See your installed images
  - `docker images / docker images –a`
  - `docker rmi hello-world / docker rmi fce289e99eb9`
  - `docker ps -a`
  - `docker rm 913edc5c90c4`
- GUI: e.g., DockStation, other



Docker Compose

- Docker Compose to deploy multiple containers
  - E.g, load balancer, services, DB
  - Configure your services

```
#docker-compose.yml
version: '3'
services:
  service:
    build: .
    ports:
      - "8080:8081"
  traefik:
    image: "traefik"
    ports:
      - "8888:80"
    volumes:
      - $PWD/traefik.toml:/etc/traefik/traefik.toml
      - $PWD/dynamic_load.toml:/etc/traefik/dynamic_load.toml
```

- Dockerfile
  - Build your binary with a Dockerfile
  - Text document contains commands to assemble an image
  - Up to now, we used existing images, now we create our own image

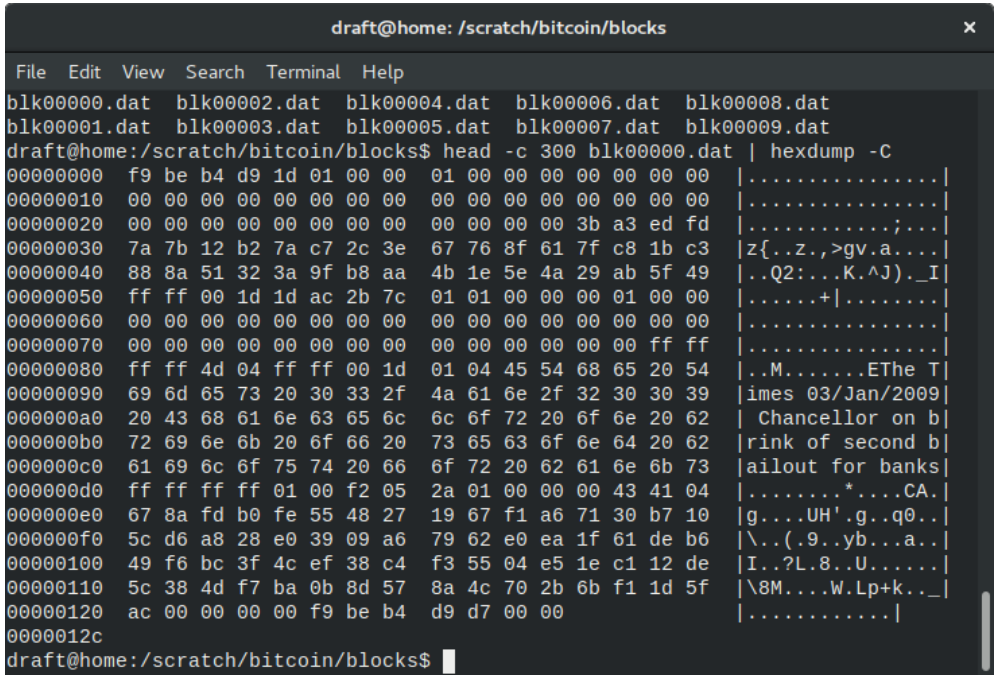
```
#Dockerfile
FROM golang:alpine AS builder
WORKDIR /build
COPY server-stateful.go .
RUN apk --no-cache add git
RUN go get -d -v
RUN go build server-stateful.go

FROM alpine:latest
RUN apk --no-cache add ca-certificates
WORKDIR /root/
COPY --from=builder /build/server-stateful .
CMD ["./server-stateful", "-p", "8081"]
```


OST

Lecture 9

Introduction

- Bitcoin is an experimental digital currency
  - Bitcoin is fully peer-2-peer (no central entity)
  - 1st Bitcoin issued on January 3, 2009
  - Smallest unit: 0.00000001 BTC (1 satoshi)
- Key characteristics
  - Maximum of ~21 million BTC
  - Every transaction broadcast to all peers
    - Every peers knows all transactions (~300 GByte as of today)
  - Validation by proof-of-work (partial hash collision)
    - Difficult to fake proof-of-work
    - No double-spending
- The initiator is unknown so far



OST

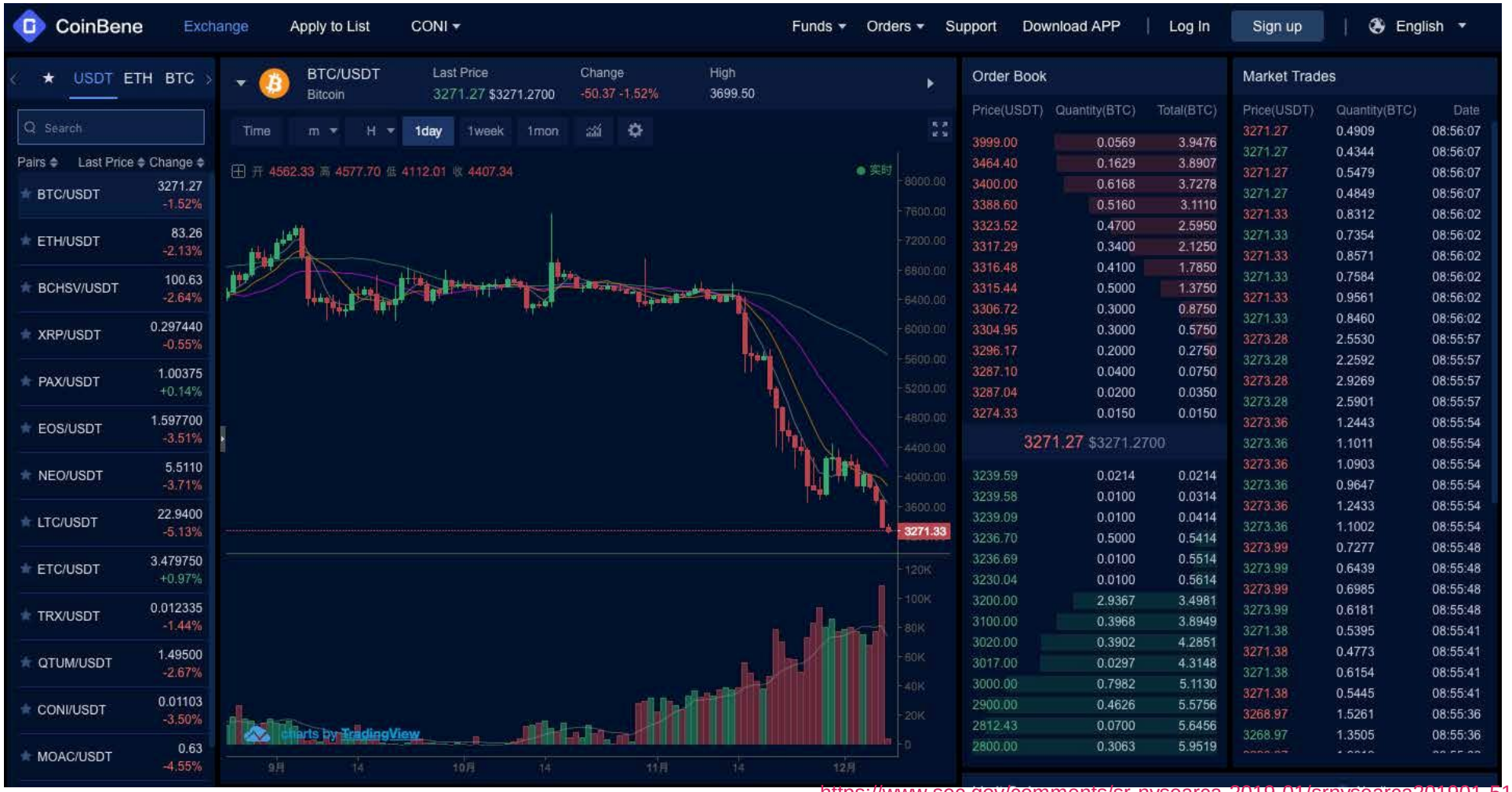
Who is Satoshi Nakamoto?

- The New Yorker believes that Satoshi Nakamoto was Michael Clear.
  - Analyzed texts from Nakamoto and searching for linguistic clues
  - 2nd possible candidate Vili Lehdonvirta
- Fast Company argues its either Neal King, Vladimir Oksman, or Charles Bry.
- Other names suggested: Martii Malmi (involved in Bitcoins since the beginning), Jed McCaleb (founder of Ripple), Donal O'Mahony, Michael Peirce, Hitesh Tewari (authors of Electronic Payment Systems for E-Commerce 2nd edition), Shinichi Mochizuki (Math Prof. Kyoto University), Hal Finney, Michael Weber, Wei Dai, Nick Szabo, Craig Wright (wired article),
- Dorian S Nakamoto (a guy with the same name)

- Satoshi is probably rich, first miner, may have ~1mio BTC
- Craig Wright, May 2016: «I'm Satoshi Nakamoto», fails to deliver proof

OST

CoinBene

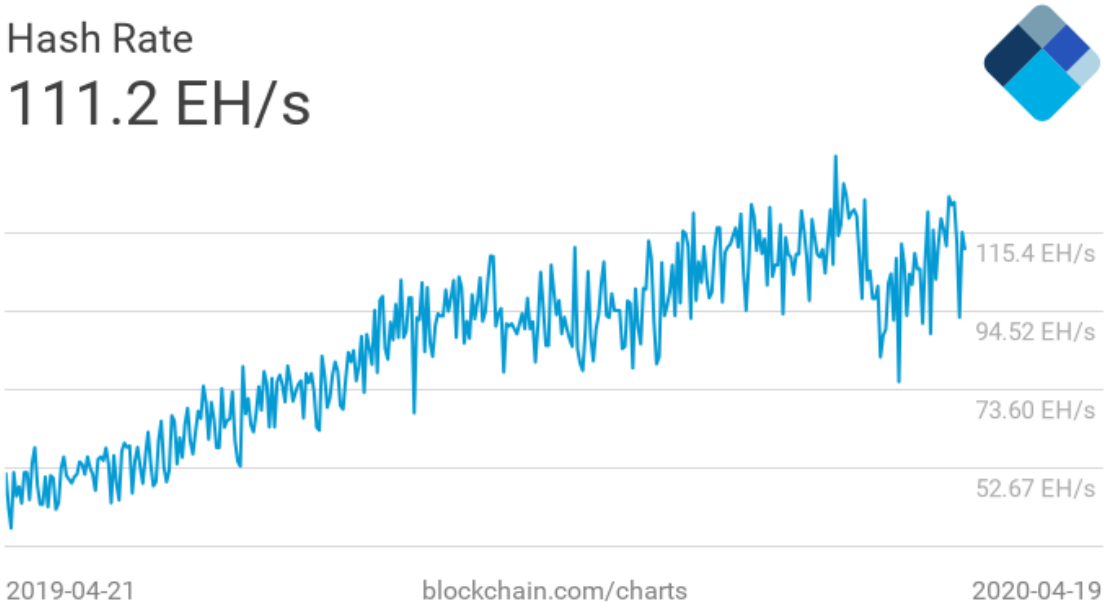


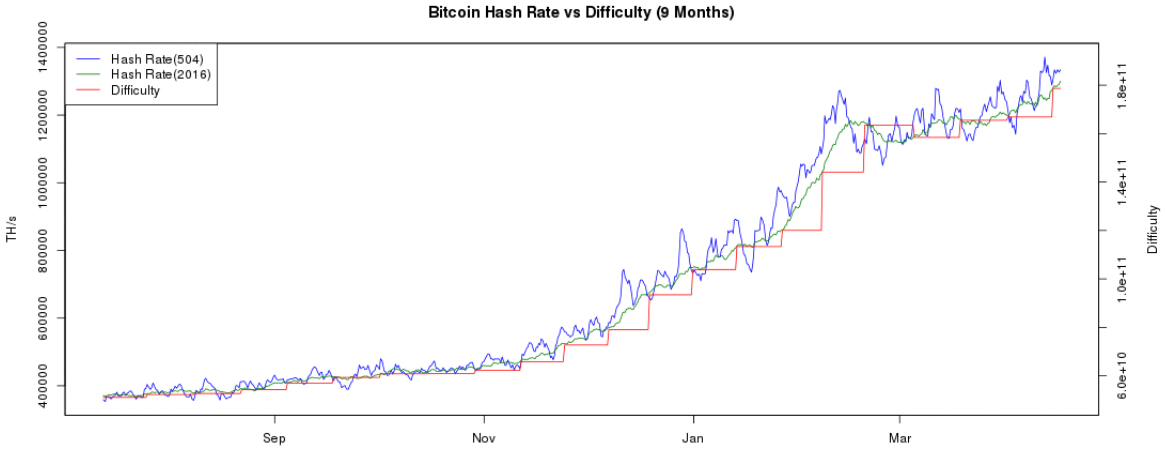
https://www.sec.gov/comments/sr-nysearca-2019-01/srnysearca201901-5164833-183434.pdf

OST

Bitcoin Numbers

- Network Hashrate
  - ~1.4 YottaFLOPS in 2020
  - ~635 ZettaFLOPS in 2019
  - ~4 ZettaFLOPS in 2015
  - ~714 ExaFLOPS in 2014
  - ~900 PetaFLOPS in 2013
  - ~155 PetaFLOPS in 2012
- Adjust time: ~14 days
- Fastest supercomputer (top500.org) Summit 148 PetaFLOPS (max), all 500 ~1.6 ExaFLOPS
- Frontier with 1.5 ExaFLOPS
  - exascale-level processing





Mechanism

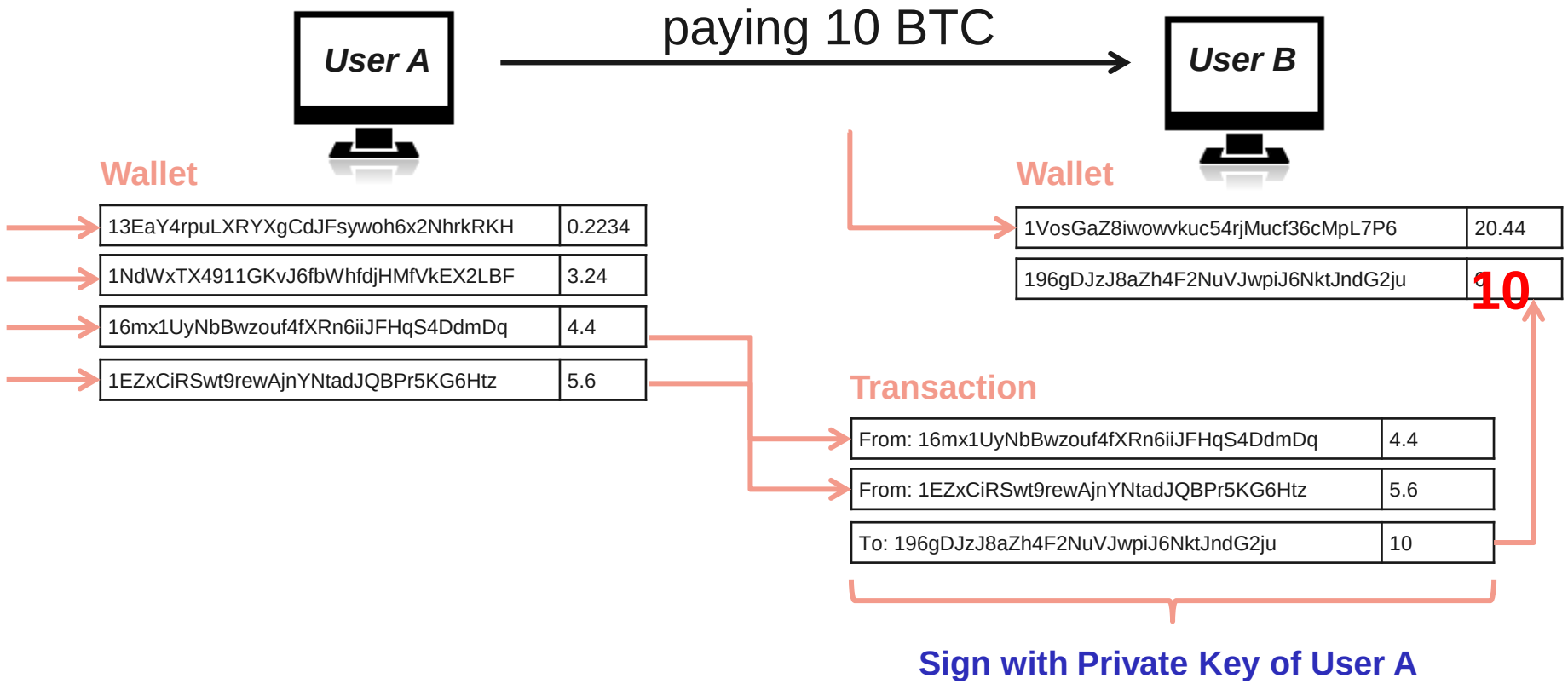
- A wallet has public-private keys (wallet.dat)
  - Public key, ECDSA 256 bit → Bitcoin address (can receive bitcoins)
  - Simple address ~ base58(RIPEM160(Sha256(ecdsa public key)))
    - E.g. 1GCeaKuhDYnNLNR6LGmBtKhPqEJD4KeEtF
  - Private key used for signing transactions

- Transaction
  - Peer A wants to send BTC to peer B → creates transaction message
  - Transaction contains input / output
    - where the BTC came from and where it goes
  - Peer A broadcasts the transaction to all the peers in the network
  - Transaction stored in blocks → block is created / verified ~10min

OST

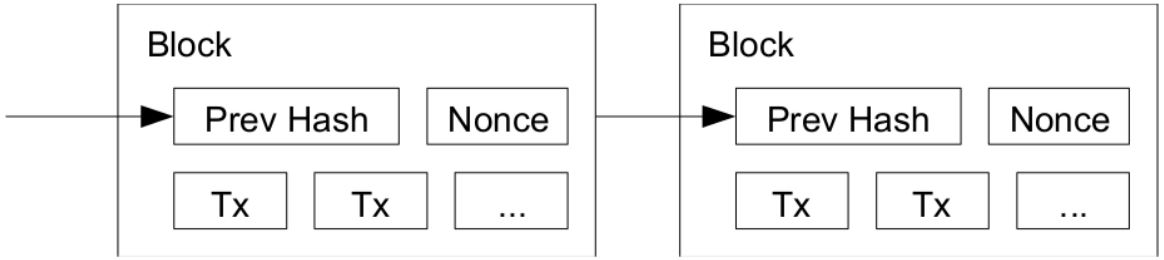
Key Bitcoin Operations (2)

- Private key authorizes the transaction (“access“)
  - If keys are stolen, thief may use “your” coins
  - If keys are lost, coins are lost
  - In UTXO (unspent transaction output) systems, complete output is spent



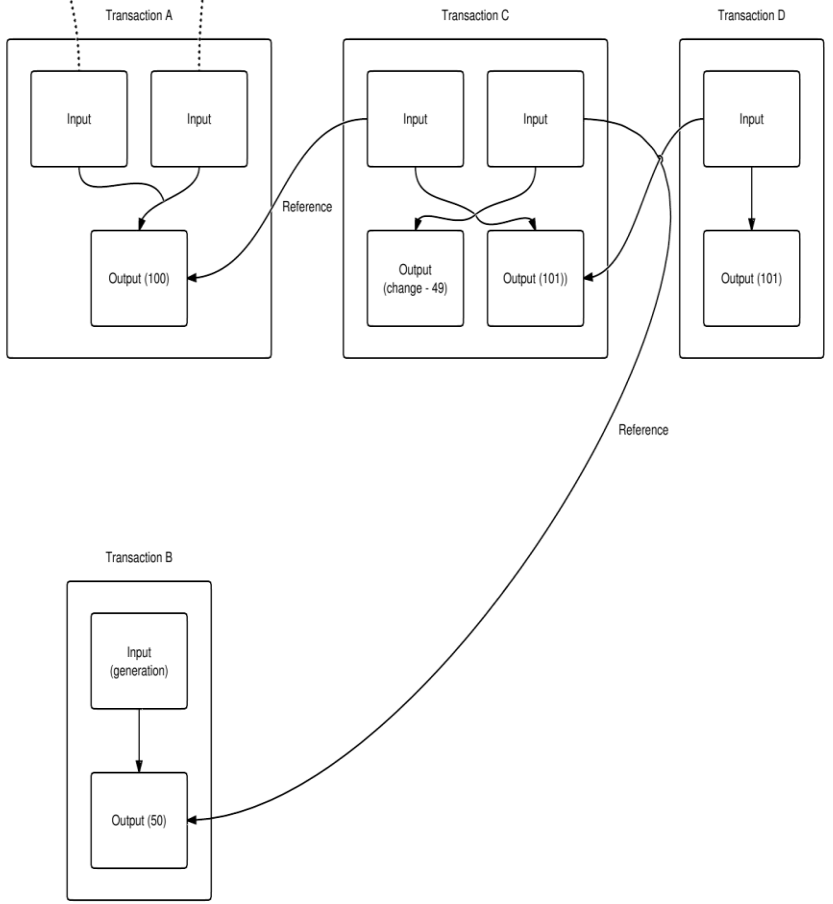
Mechanism

- Avoiding double spending
  - Transactions in blocks are confirmed.
  - guessing value that results in zero bits (0000000000001805ff174586 b6acf100f733aaf634e92f9580b4fac9272ed97)
  - Chained proofs of work



- Generation of coins
  - Mining / creating blocks → Miner get currently 12.5 BTC per creation
    - adjustable difficulty 6 blocks / h
    - Sometime in 2020 reward will be 6.25

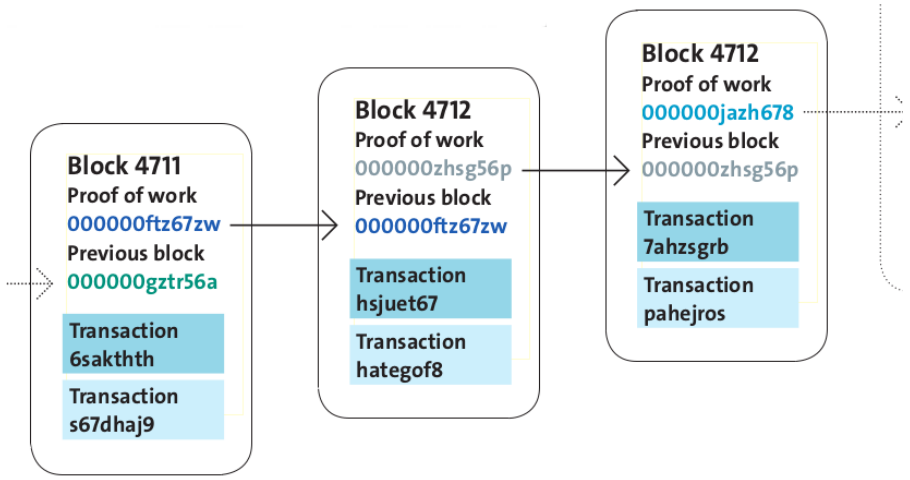
- Transactions have one or more inputs
  - A sends 100 BTC to C, C generates 50 BTC. C sends 101 BTC to D, and send himself some change. D sends the 101 BTC to someone



https://en.bitcoin.it/wiki/Transaction

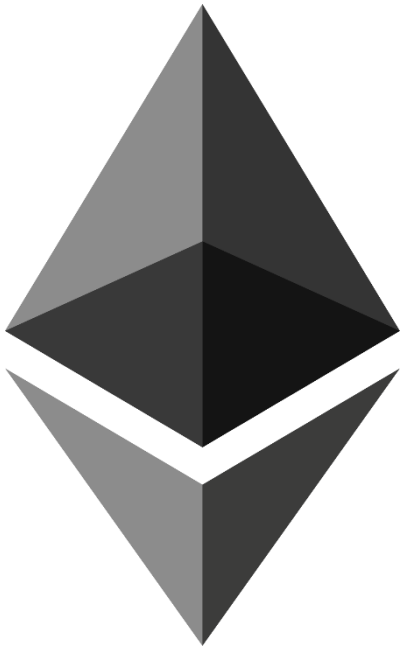
Blockchain (repetition)

- Transactions are collected in blocks
  - New block created approximately every 10 min
- Blocks contain solved crypto puzzles
  - In the form of partial hash collisions (SHA256)
- A block has a pointer to previous block → **Blockchain**
- Creation of blocks is called mining (reward)
  - Miners use highly specialized hardware!



Many Coins – Similar Mechanism

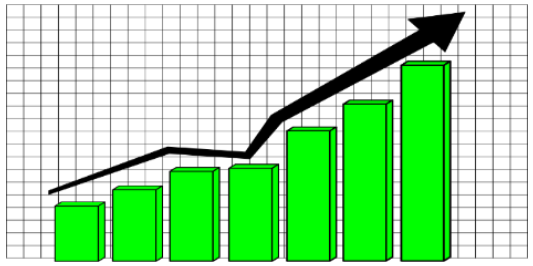
- All electronic backed by scarce resource - avoid: double spending
  - Bitcoin: SHA256 partial hash collision: time, CPU, electricity
  - Ethereum: variant of Dagger-Hashimoto, time, CPU, memory, electricity, miner store dataset: 1GB, verification only needs 16MB
  - Ethereum: Opcodes in Bitcoin, smart contracts in Ethereum
  - Litecoin: scrypt partial hash collision: time, CPU, memory, electricity
  - Ripple XRP: Unique node list (trusted validators, 1000): web of trust
  - PPCoin/Peercoin: proof of stake / (proof of work):
    - Holding/staking 1% will generate e.g., 1% of coins
    - Energy efficient / proof of stake
  - Cardano/EOS/…many more



Discussion (1)

Advantages

- Low (fixed) tx fees
  - ~26 satoshi per byte
- Scalable
  - Hardware/storage gets faster



- Anonymity
  - No privacy concerns/ datamining difficult

Disadvantages

- Power consumption
  - ~ 12 times AKW Beznau
- Not scalable
  - Bitcoin with 5 tps vs. VISA 57,000 tps (23.12)
    [tps: transactions per sec]

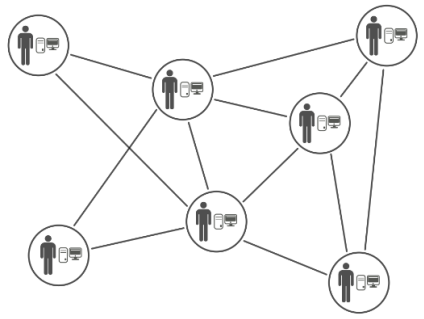
- Anonymity
  - Can be used for illegal activities

OST

Discussion (2)

Advantages

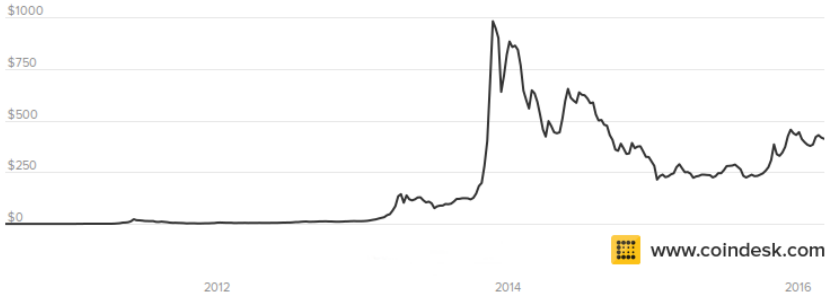
- No major “crashes”
  - Mt.Gox was exchange site!
- Decentralized
  - Open protocol
  - Forks



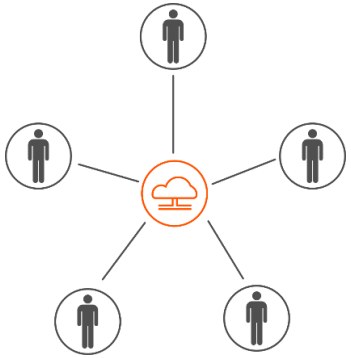
- Many other blockchain use cases
  - Smart contracts

Disadvantages

- Volatile exchange rate



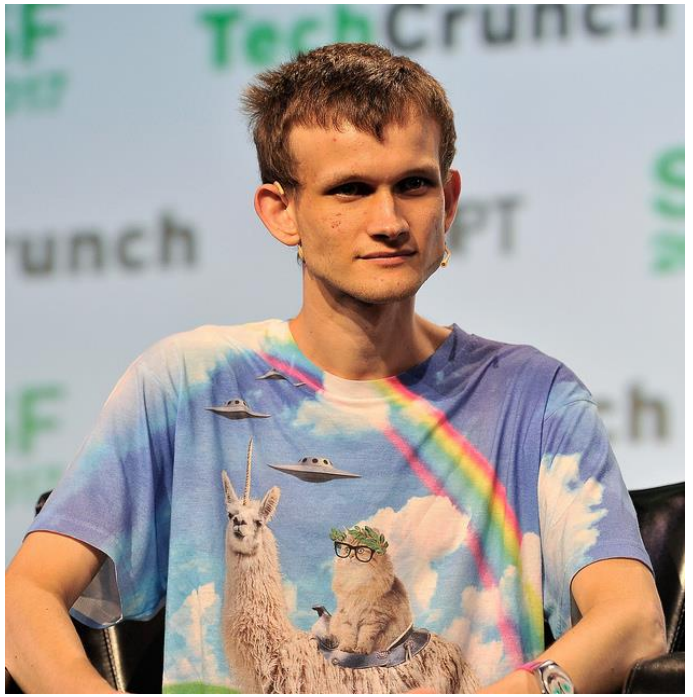
- Central elemements
  - Core developers



Lecture 10

Bitcoin / Ethereum

- “Issues” with Bitcoin
  - Bitcoin Script limited
    - But: many use-cases possible: Lightning network, escrow
  - Slow development, implementing new features difficult
    - E.g., when running into block limits: improve protocol vs. increase block size - SegWit
    - SegWit vs. BTU (segregated witness vs. increasing block size voting)
    - If you don’t agree: fork
- Ethereum (1 ETH ~470$)
  - Protocols designed from scratch (not like Bitcoin forks)
  - «General-purpose» blockchain (loops, arithemitcs, etc.)
  - White paper released in December 2013
  - Ethereum foundation located in Zug (initiators known) - non-profit foundation
  - Mining reward ~ block every ~14s – ~2 ETH (“always”, unlike Bitcoin’s ~21m BTC limit)



Vitaly Buterin

Ethereum History

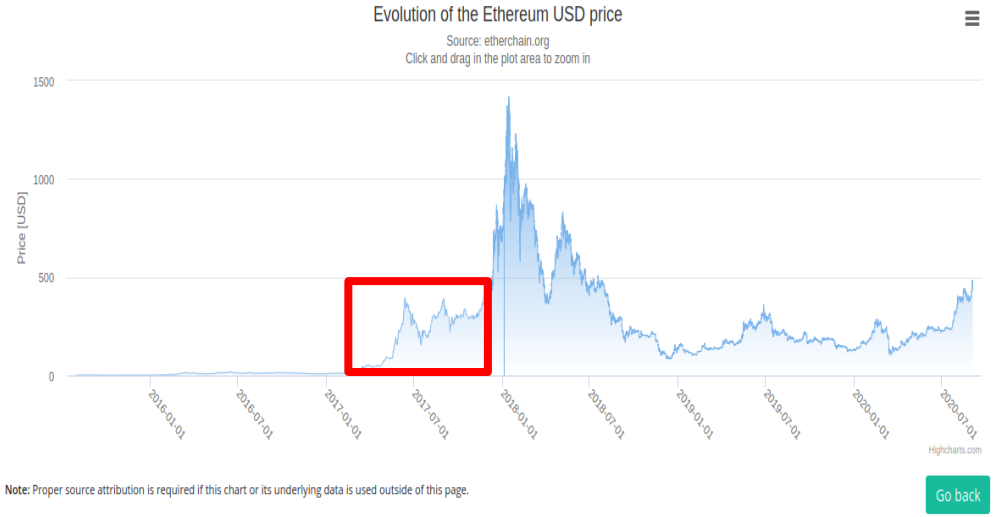
- Olympic (past) – released 09.05.2015
  - Last Ethereum Proof-of-Concept series
  - “Olympic will feature a total prize fund of up to 25’000 ether” (now 11.7m USD)
- Frontier (past) - released 30.07.2015
  - Main public network, “Beta”/use at your own risk
- Muir Glacier (now) - released 01.01.2020
  - Delay difficulty bomb
  - St. Petersburg: change from 3 ETH to 2 ETH reward
- Ethereum 2.0
  - Proof-of-stake (energy), sharding (storage space)

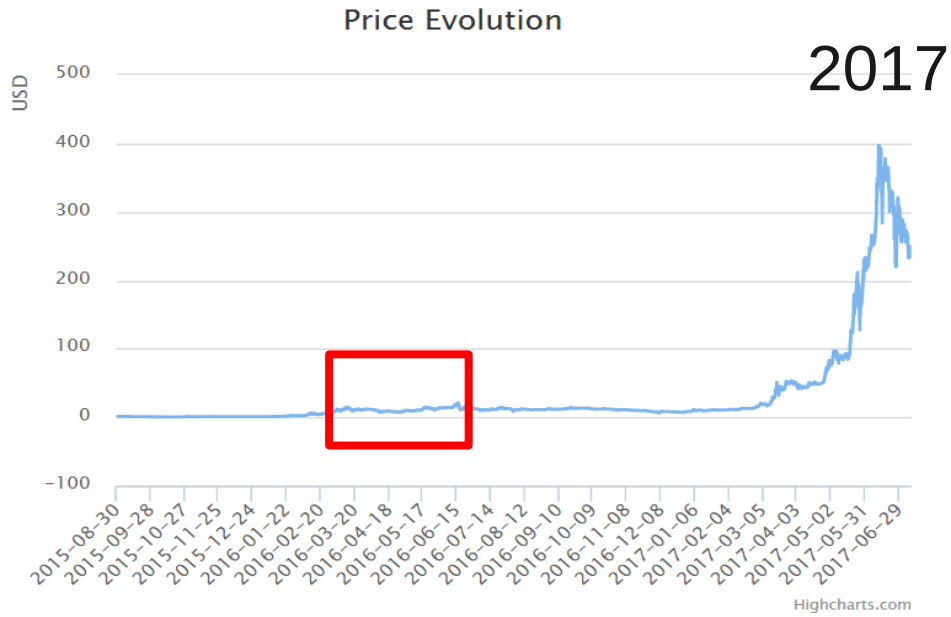
Frontier	2015-07-30	0
Ice Age	2015-09-08	200’000
Homestead	2016-03-15	1’150’000
DAO Fork	2016-07-20	1’920’000
Tangerine Whistle	2016-10-18	2’463’000
Spurious Dragon	2016-11-28	2’675’000
Byzantium	2017-10-16	4’370’000
Constantinople / St. Petersburg	2019-02-28	7’280’000
Istanbul	2019-12-08	9’069’000
Muir Glacier	2020-01-01	9’200’000
Berlin	?	?

OST

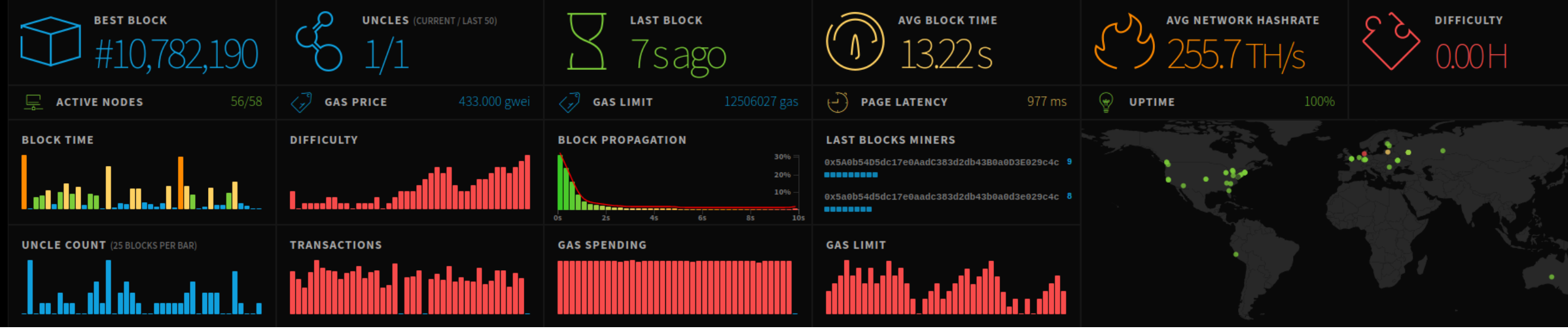
Ethereum Stats

- Basic Stats
  - 2nd in market cap ~ 50b USD
  - 112mio ETH supply (stats)
  - Transactions (highest ~15.5 TPS) now ~1mio per day
  - Node count (~9'000 nodes)
  - Blocksize ~40KB
  - Accounts (113mio)

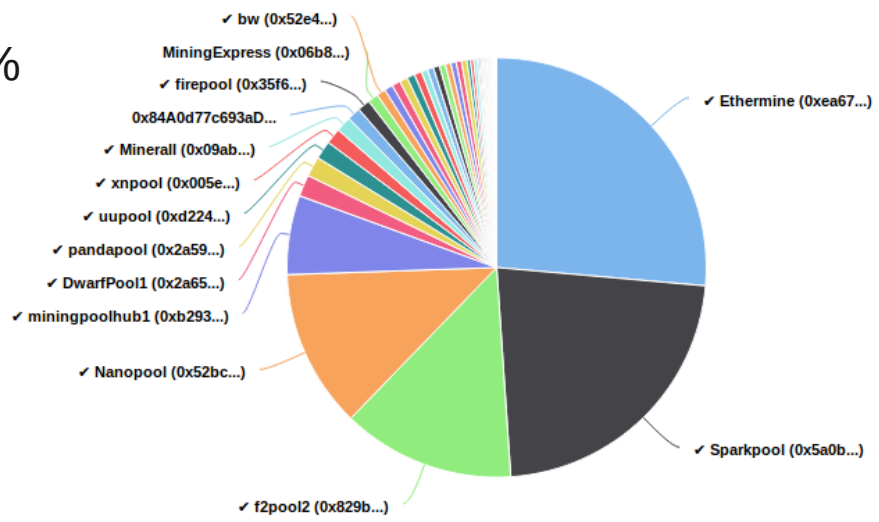


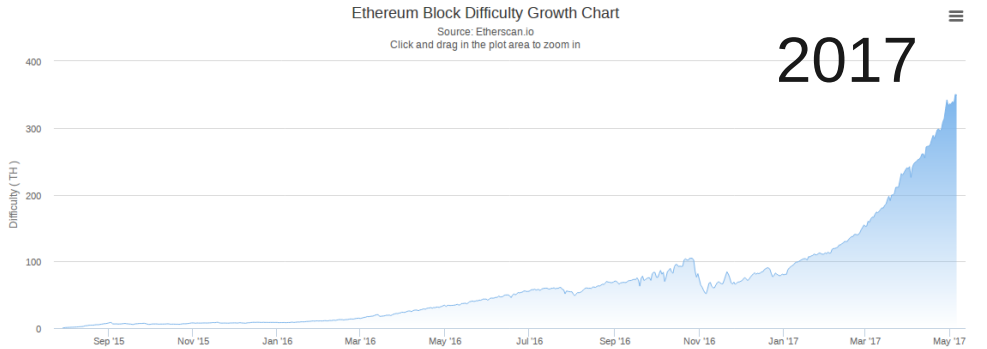


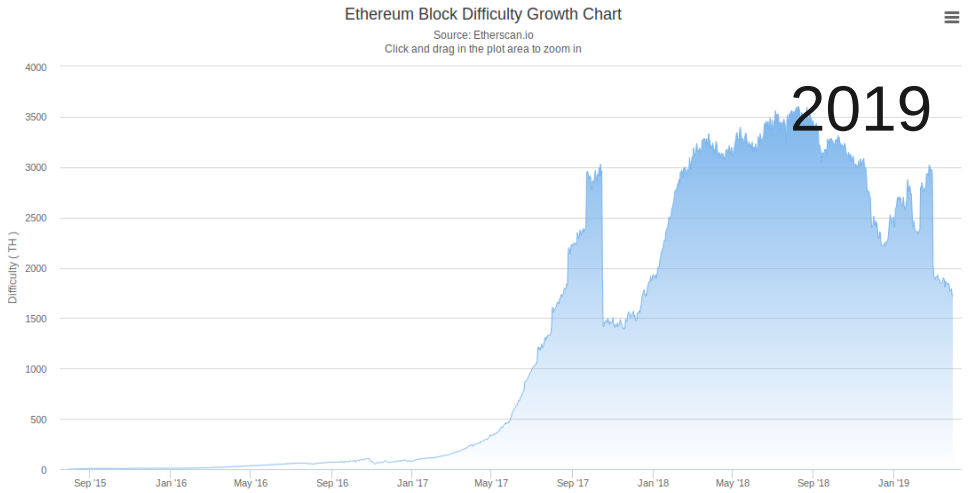
Mining Mechanisms



- Proof-of-work - Increasing difficulty
  - Mining in pools
  - Ethermine + Sparkpool 48.8%
  - Farms







51% Attack in Ethereum / Bitcoin

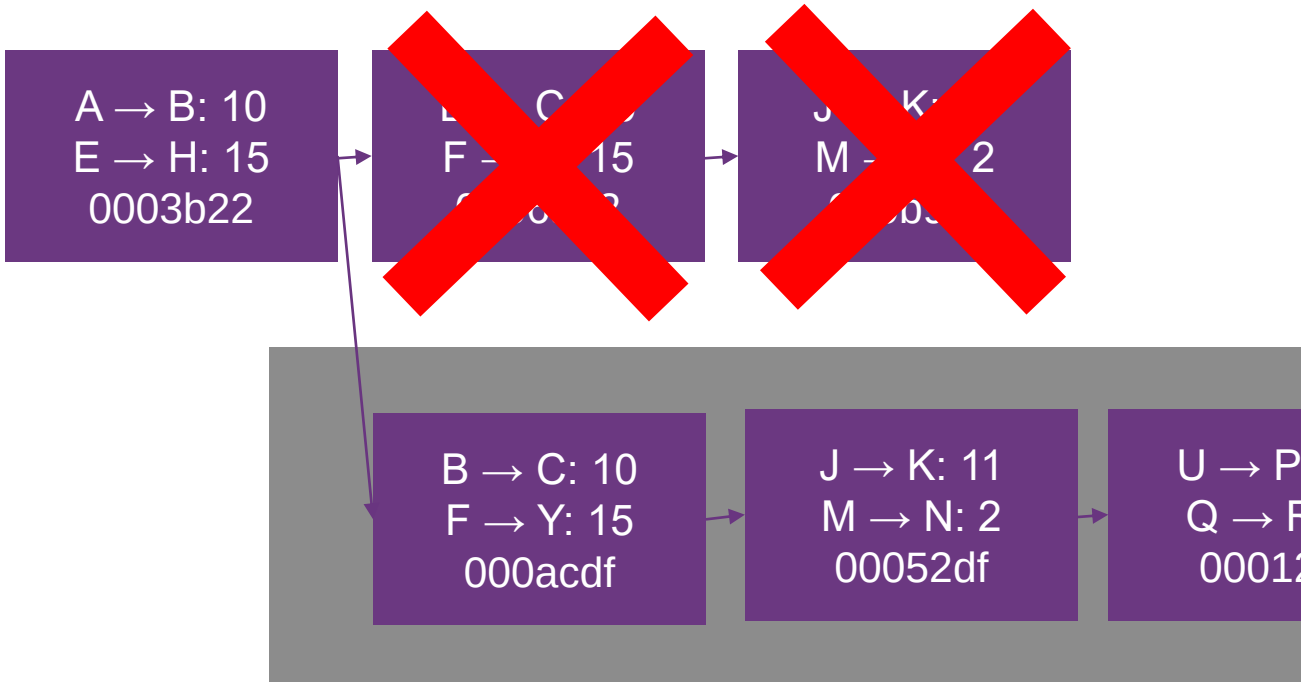
- “If a majority of CPU power is controlled by honest nodes, the honest chain will grow the fastest and outpace any competing chains.”
  - https://bitcoin.org/bitcoin.pdf
- How expensive is a 51% attack?
  - Buy an attack?
  - “Ethereum Classic Labs singled out crypto-mining marketplace NiceHash for allegedly facilitating multiple attacks against the network”

- 01.08.2020: Ethereum Classic suffered a 3,693-blockchain reorganization early Saturday morning, an event first thought to be a possible 51% attack, after a miner used old software after having been offline, according to Terry Culver, CEO of Ethereum Classic Labs.
- 06.08.2020: Ethereum Classic has suffered its second 51% attack in a week after more than 4,000 blocks were reorganized Thursday morning.
- 17.08.2020: OKEx has confirmed a loss of approximately $5.6 million in Ethereum Classic (ETC) from two recent 51% attacks and is considering removing ETC from its exchanges.
- 01.09.2020: Ethereum Classic (ETC) Suffers Yet Another 51% Attack, Major Changes Needed to Improve Blockchain Platform’s Security

OST

51% Attack in Ethereum / Bitcoin

- Longest chain survives (same as Bitcoin)
- Double spend: rollback transactions
  - X is an exchange address
  - Mine secretly, Y is attacker address
  - X arrived – exchange does payout USD (wait 2 block conf.)
  - You mine faster as you have 51%, broadcast secret chain
  - Tx F→X: 15 never happened, goes to Y, but attacker already got the USD
    - Attacker can spend the 15 coins again (double spend)



Transaction Time and Costs

Gas

- Recap: Ethereum is needs a lot of energy – proof-of-work – Miners are rewarded with 2 ETH – 900USD
  - Miner also gets TX fee
    - Bitcoin (simple TX)~ 3USD
    - Ethereum (simple TX) ~ 2USD
- How much do I need to pay?
  - Bitcoin easy, proportional to size: 300 bytes = 0.0003BTC, 600b = 0.0006BTC
  - Ethereum can have loops, so 300 bytes script may run forever
    - Solution: pay per instruction → 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 abstract operations that a transaction may effect.

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

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

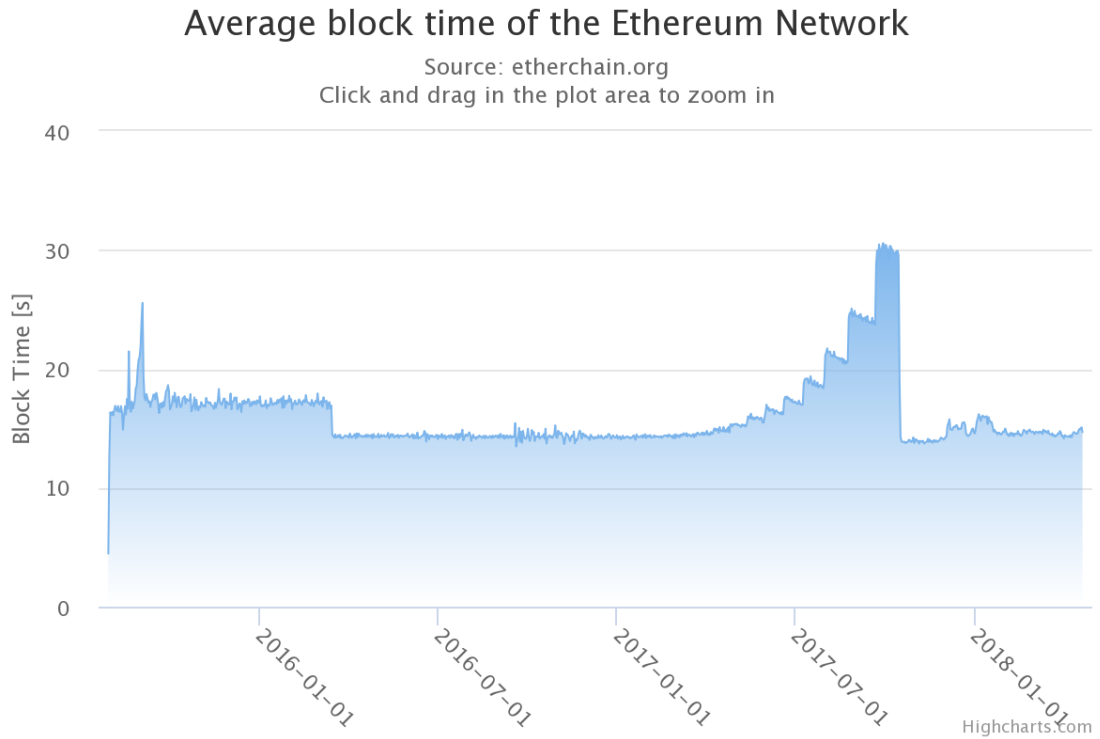
Blocktime and Gas

Gas

- Every instruction is paid for (example)
  - Total gas amount: dry-run contract
- Issue transaction: specify how much gas you want to use (max) and gas price
  - If you run out of gas, state is reverted, ETH gone
- Gas Price also set by Miner
  - Gas price ~375 gwei
  - Market for TX
    - Gas price too low, longer waiting time until TX will be included

Blocktime

- Block time: ~14-15s
  - But: Ice age (difficulty bomb)



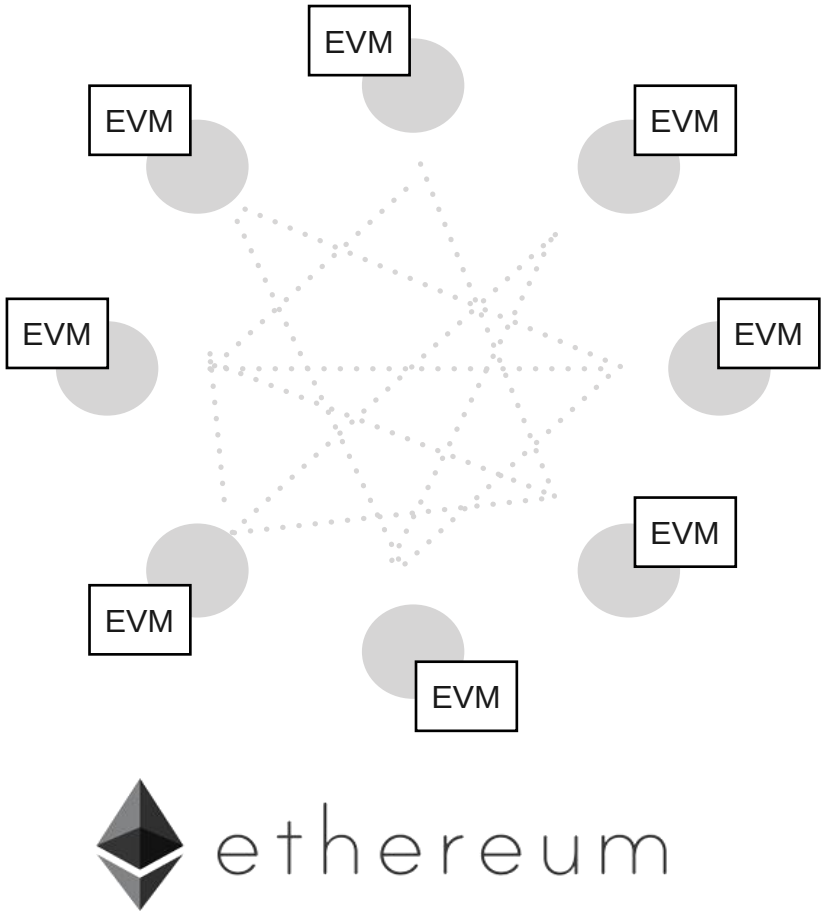
OST

Smart Contracts

OST

Ethereum smart contract

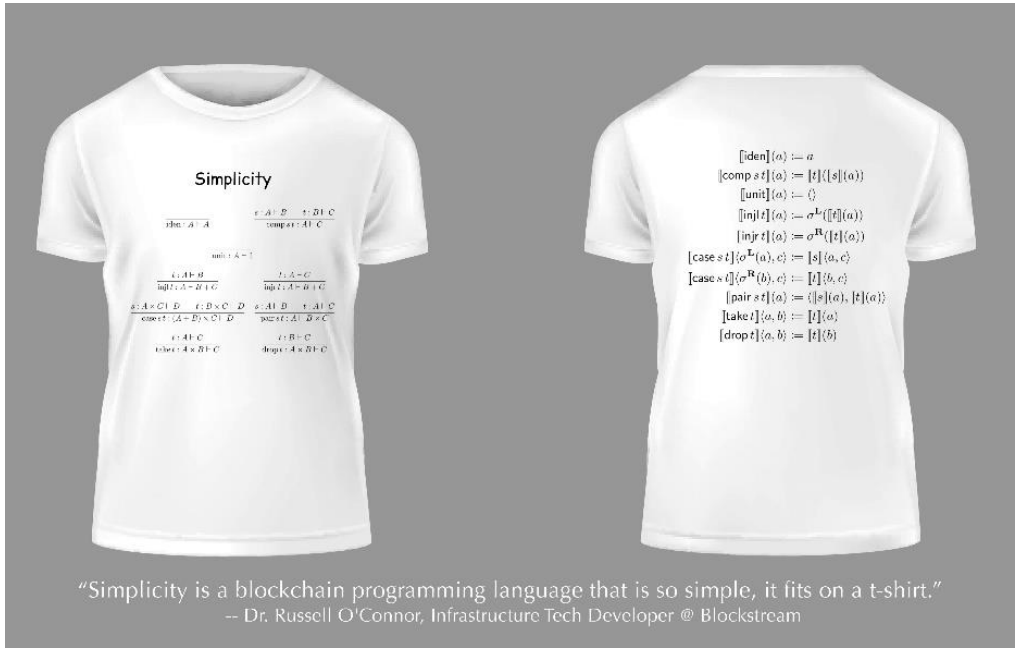
- Ethereum Virtual Machine runs instructions (slide 11)
- Computation on EVM is "very expensive“: every contract is run on every full Ethereum node
  - Global computer, always running, always correct
  - Result on every node is the same
- Coin flip?
  - Not easy - no random number generator in Ethereum



OST

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



Solidity Language https://solidity.readthedocs.io

- JavaScript-like language
  - Statically-typed that compiles to EVM instructions
- Comments

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

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

- Contract with State Variables (expensive!)

```
pragma solidity ^0.7.1;
//minimal contract
contract Example1 {
    uint256 counter;
}
```

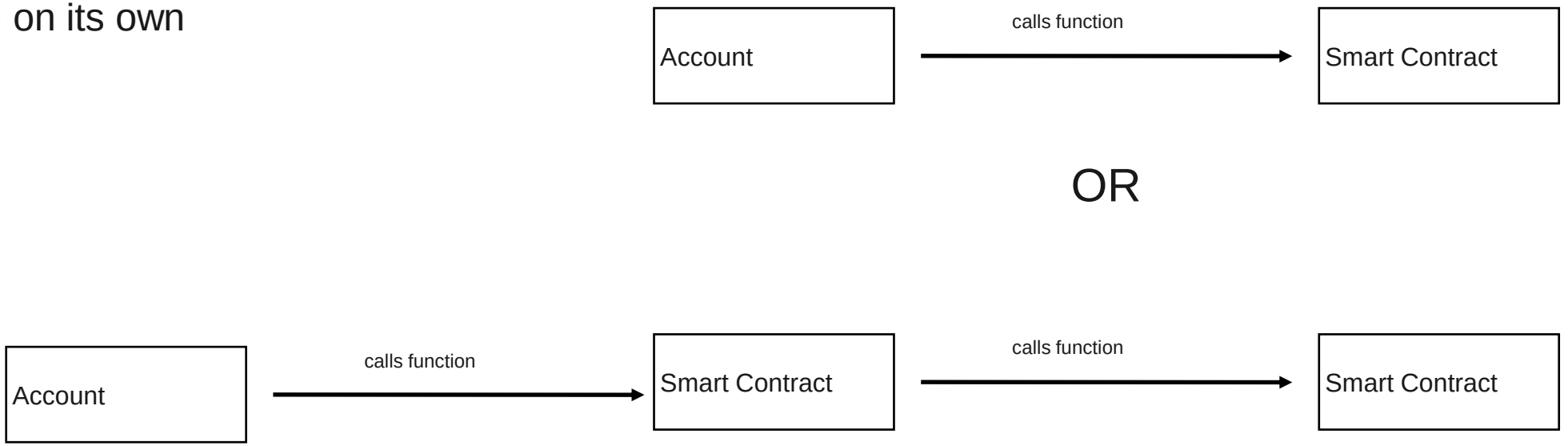
https://learnxinyminutes.com/docs/solidity/

https://solidity.readthedocs.io/en/v0.7.1/

OST

Solidity: Function Calls

- Smart contract function can call another smart contract function
- Initiator **always** needs to be an external account (smart contract can have account)
  - But smart contract can never initiate a transaction on its own



OST

Solidity: Version, Mapping

```
/* Specifies what versions of Solidity a
 * source file can work on
 */
pragma solidity ^0.7.1;

contract Notary {
    …
}
```

- 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

```
contract Notary {

  mapping (address => mapping (bytes32 => uint256)) stamps;
    …
}
```

Address	Hash	Timestamp
0xD6df5935cD03a768b7B9E92637A01b25e24cb709	ed52f3833d5d2c920fd43a0972add3555ad149f0201bbd134907498a851a5ec3	2009-06-24 12:39:54 +0900
0xD6df5935cD03a768b7B9E92637A01b25e24cb709	0c911c1d3bde4be991e3d39c2f7e97c621617feaf8f7070e2384c602430ee382	2015-04-14 10:34:24 +0800
0x1530df3e1C69501d4Ecb7E58eB045b90DE158873	0c911c1d3bde4be991e3d39c2f7e97c621617feaf8f7070e2384c602430ee382	2020-09-11 14:33:24 +0900
…	…	…

Example Solidity Contract

- Notary Contract
  - Simple Contract, 2 functions

```
pragma solidity ^0.7.1;

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];
    }
}
```


OST

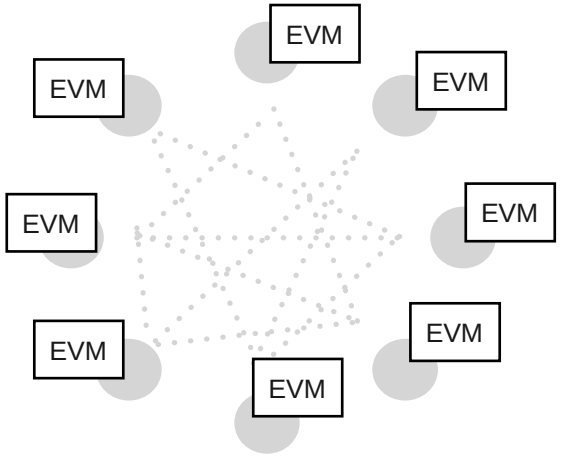
Architecture Overview

```
pragma solidity ^0.7.1;

contract Notary {
    …
}
```

METAMASK

Compiler

Remix IDE



ethereum

EVM

Local Ethereum Node

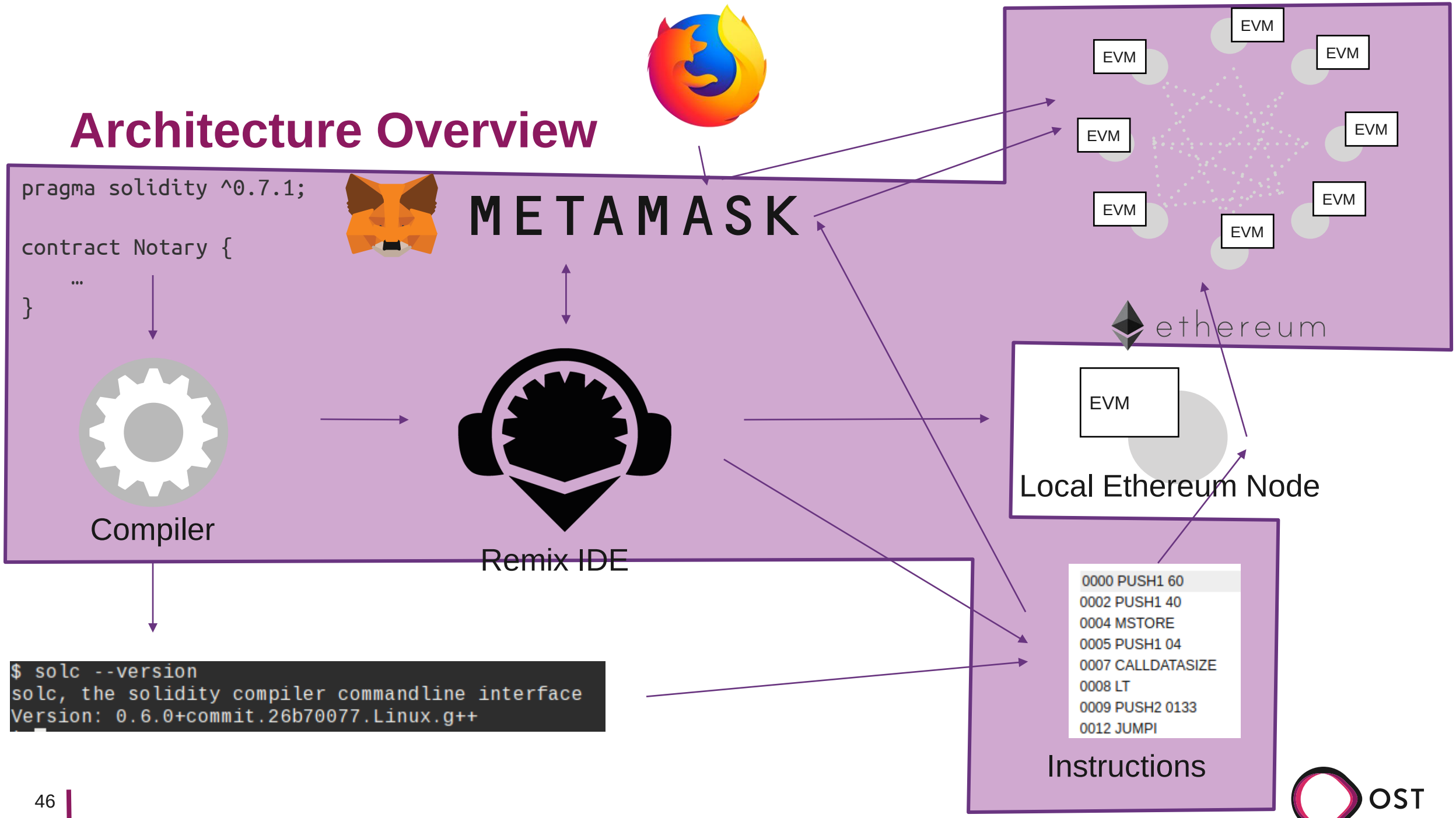
```
$ solc --version
solc, the solidity compiler commandline interface
Version: 0.6.0+commit.26b70077.Linux.g++
```

```
0000 PUSH1 60
0002 PUSH1 40
0004 MSTORE
0005 PUSH1 04
0007 CALLDATASIZE
0008 LT
0009 PUSH2 0133
0012 JUMPI
```

Instructions

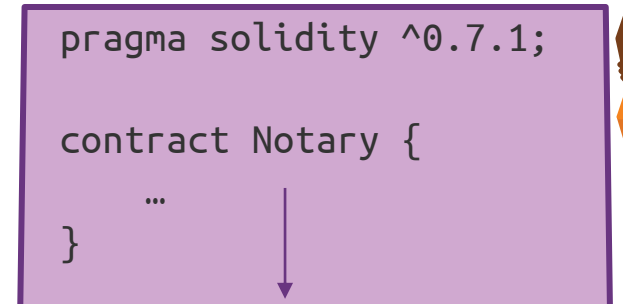
OST

Architecture Overview



OST

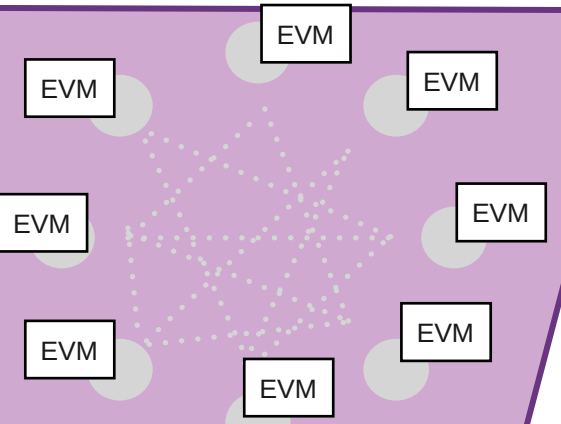
Architecture Overview



Compiler

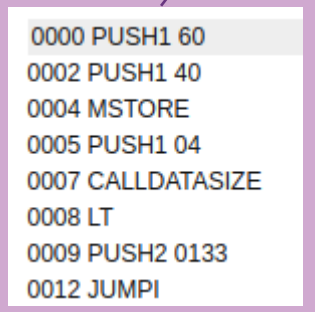
METAMASK

Remix IDE



Local Ethereum Node

```
$ solc --version
solc, the solidity compiler commandline interface
Version: 0.6.0+commit.26b70077.Linux.g++
```



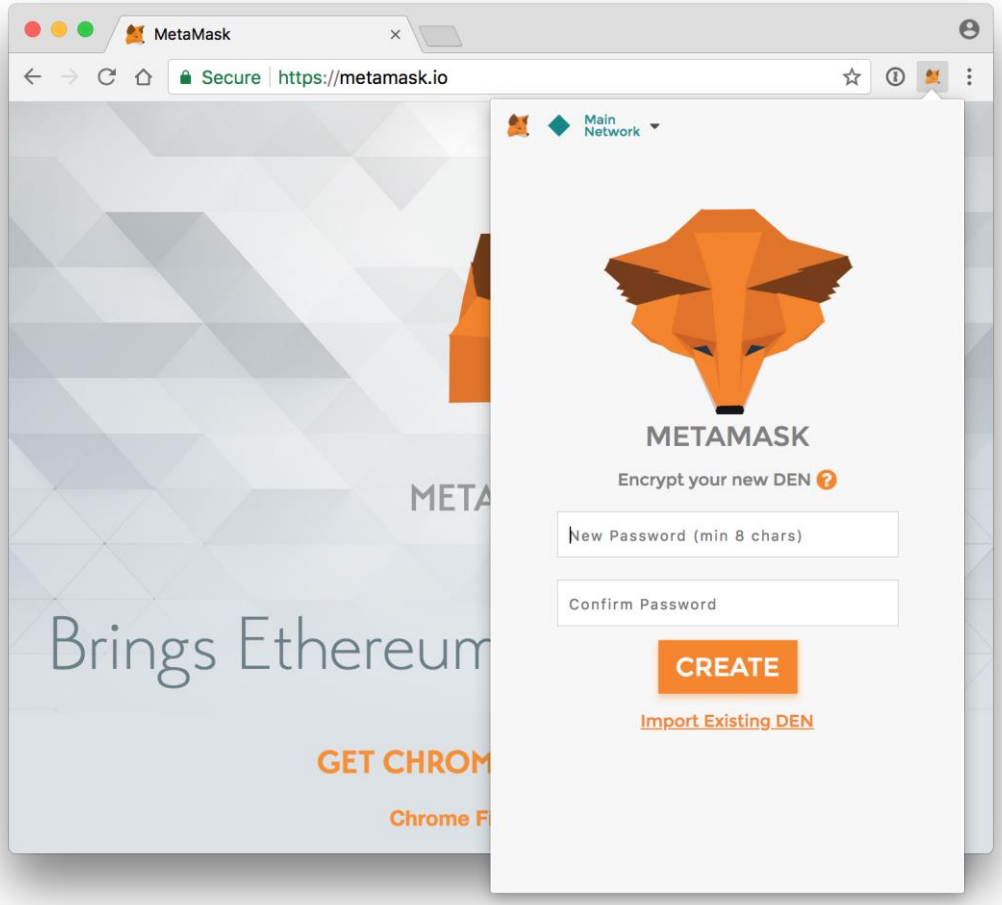
Instructions

OST

MetaMask

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

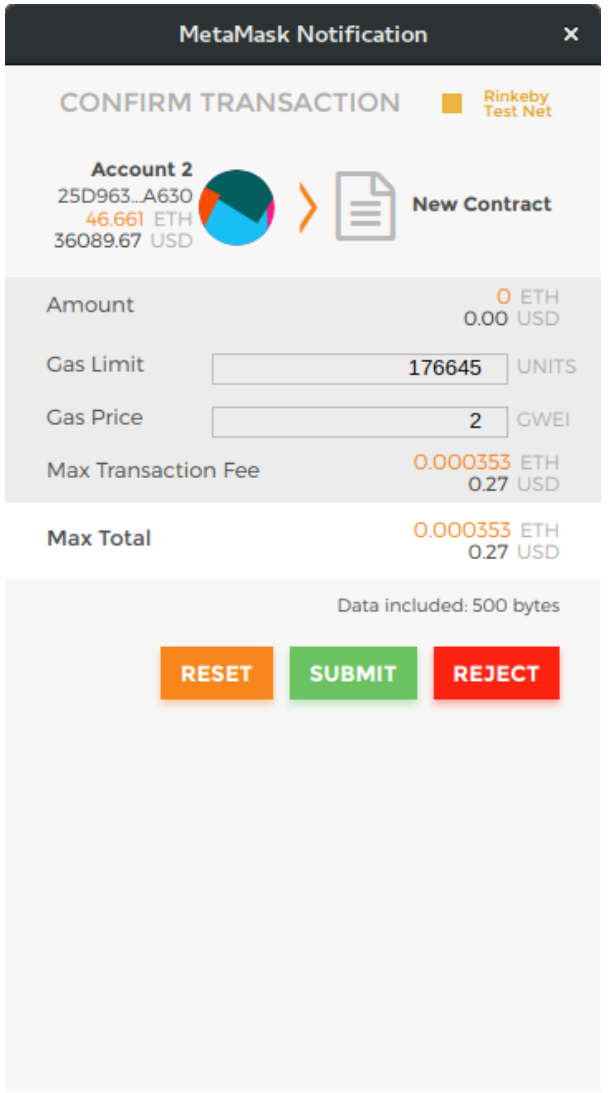
- Remix IDE: https://remix.ethereum.org
  - Use Notary.sol from https://github.com/tbocek/VSS-WEB3/blob/master/Notary.sol
  - Alternatively, use IntelliJ solidity plugin and deploy via geth, parity, or a local test blockchain

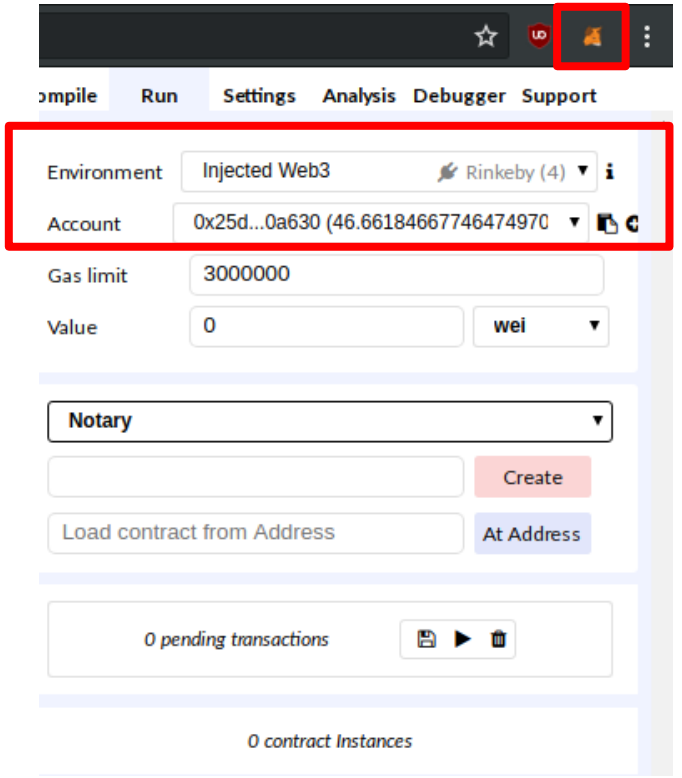


OST

Create Contract

- Connect to MetaMask
  - Injected Web3
  - If you see your accounts, good!
- Create contract!
  - Add address to the code (manually)
- Store value
  - 0x654cf88c**4cff3e62696f7cbb55fbba922b2a75897a010347e371e9398b658c4affa611aa**
  - Hash is: 0x4cff3e62696f7cbb55fbba922b2a75897a010347e371e9398b658c4affa611aa

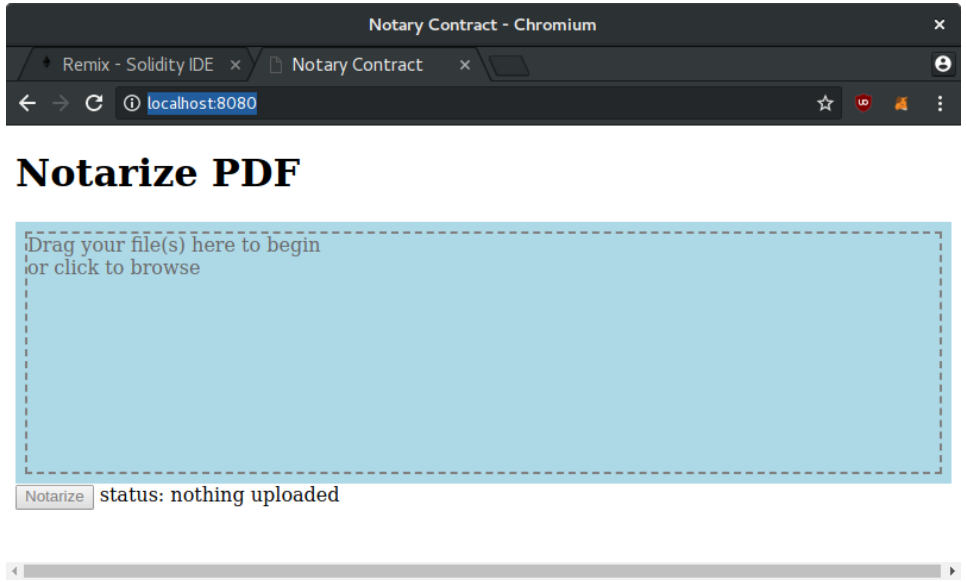




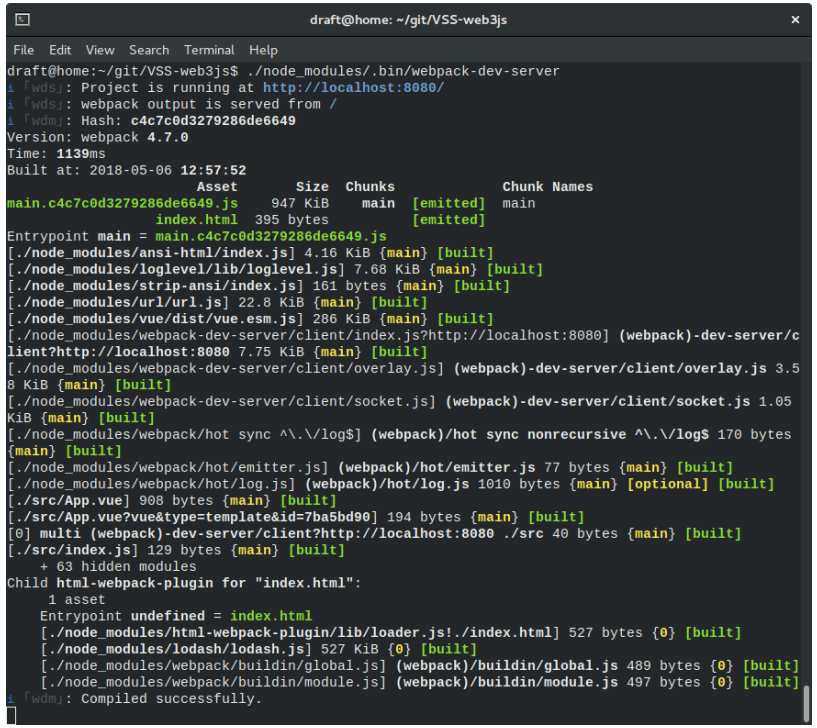
OST

Run it!

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



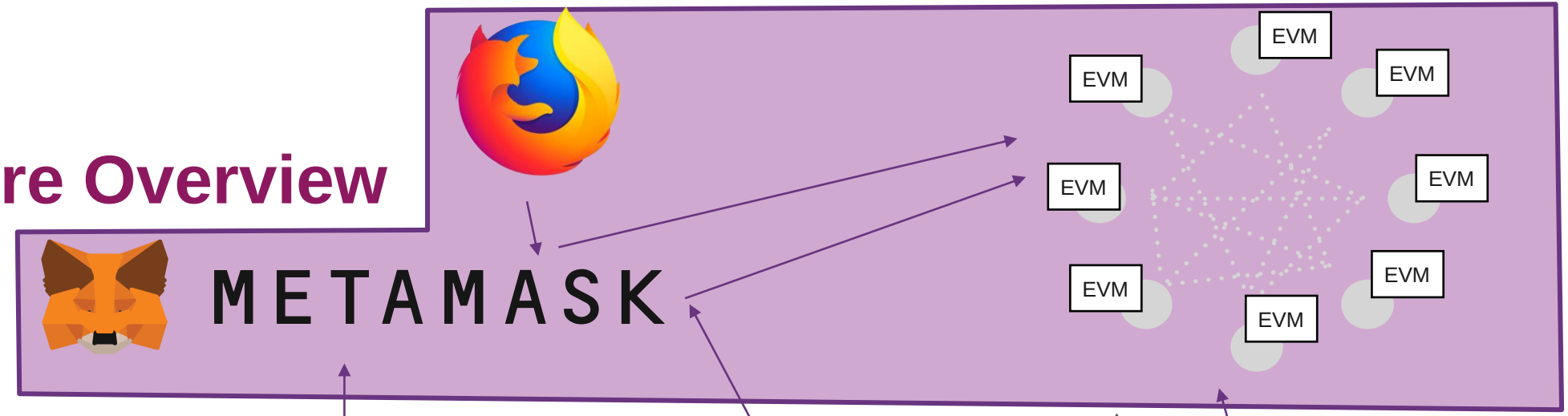
- Frontend
  - Vue.js, dependencies:
    - crypto-js, vue, web3
- App.vue, main file
  - Template
  - Create web3 instance
  - Events
    - fileChange()
    - store()



OST

Architecture Overview

```
pragma solidity ^0.7.1;

contract Notary {
    …
}
```



Compiler

Remix IDE

Local Ethereum Node

```
$ solc --version
solc, the solidity compiler commandline interface
Version: 0.6.0+commit.26b70077.Linux.g++
```

```
0000 PUSH1 60
0002 PUSH1 40
0004 MSTORE
0005 PUSH1 04
0007 CALLDATASIZE
0008 LT
0009 PUSH2 0133
0012 JUMPI
```

Instructions

OST

Lecture 11

Connecting everything

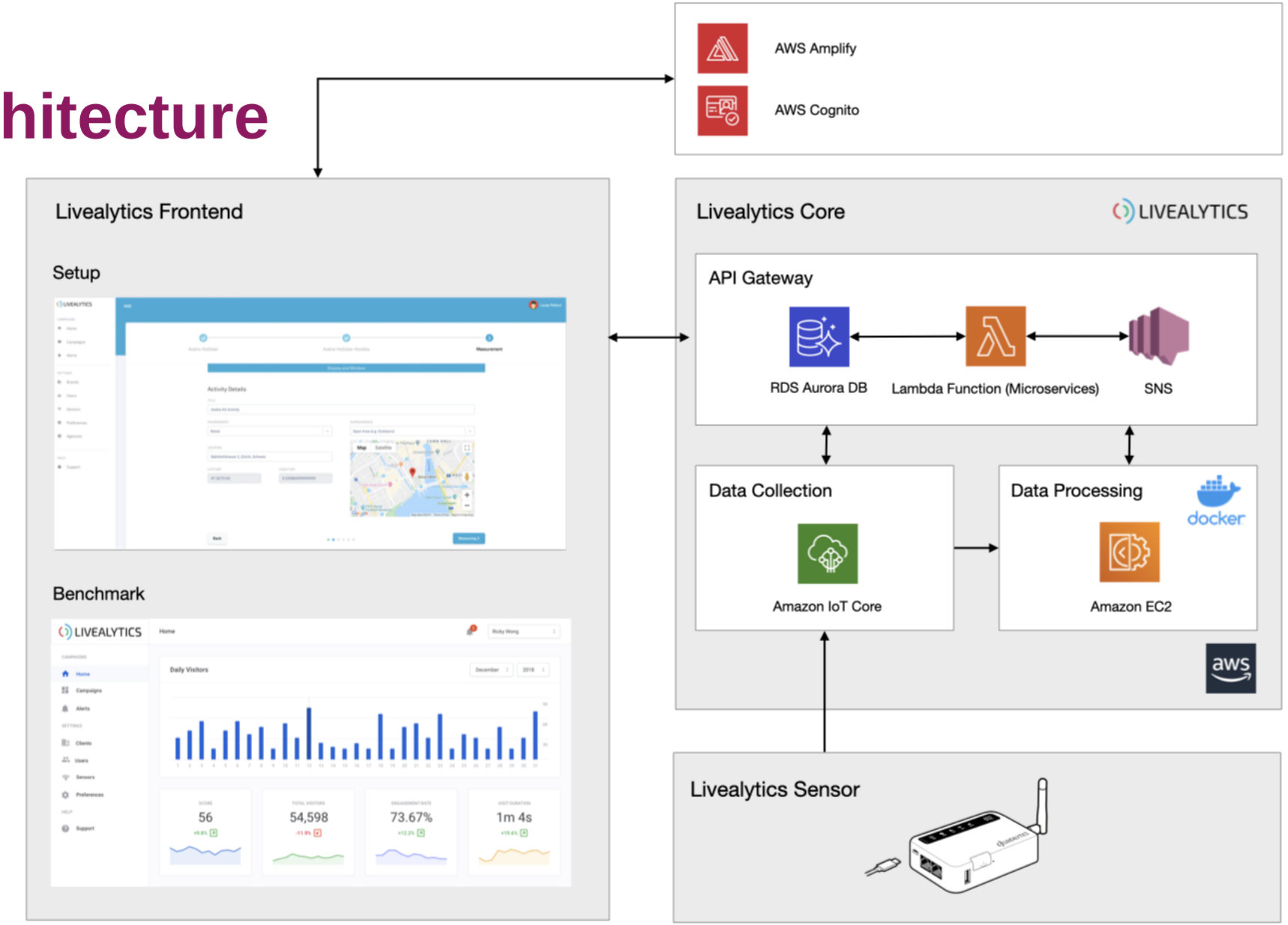
- docker-compose up --build --scale service=2
- Open browser: localhost:8080
- Development
  - Run single parts locally, don’t restart everything
- Deployment (use compose on single host)
  - Docker Swarm / docker-compose
    - Swarm mode for managing cluster of Docker Engines
    - docker deploy --compose-file docker-compose.yml / docker stack deploy …
  - Kubernetes
    - Kompose – convert docker-compose.yml to Kubernetes
    - Kubernetes (K8s) is an open-source system for automating deployment (k3s)

```
#docker-compose.yml
#run with: docker-compose up --build --scale service=2
version: '3'
services:
  traefik:
    image: "traefik"
    ports:
      - "8080:80"
    volumes:
      - $PWD/traefik/traefik.toml:/etc/traefik/traefik.toml
      - $PWD/traefik/dynamic_load.toml:/etc/traefik/dynamic_load.toml
  apiproxy:
    build: ./apiproxy
    ports:
      - "8081:8080"
  storage:
    build: ./storage
    ports:
      - "8082:8080"
  frontend:
    build: ./frontend
    ports:
      - "8083:8080"
  auth:
     build: ./auth
     ports:
      - "8084:8080"
  service:
    build: ./service
    ports:
      - "8085-8086:8080"
```


OST

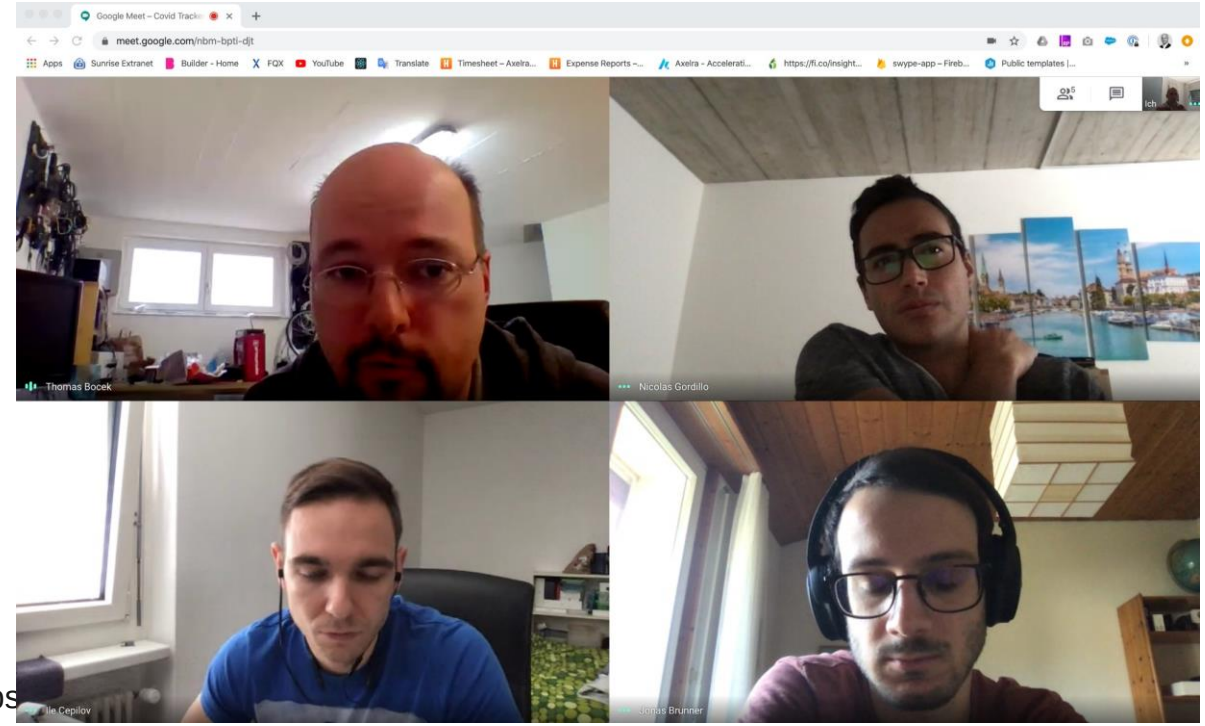
Lecture 12

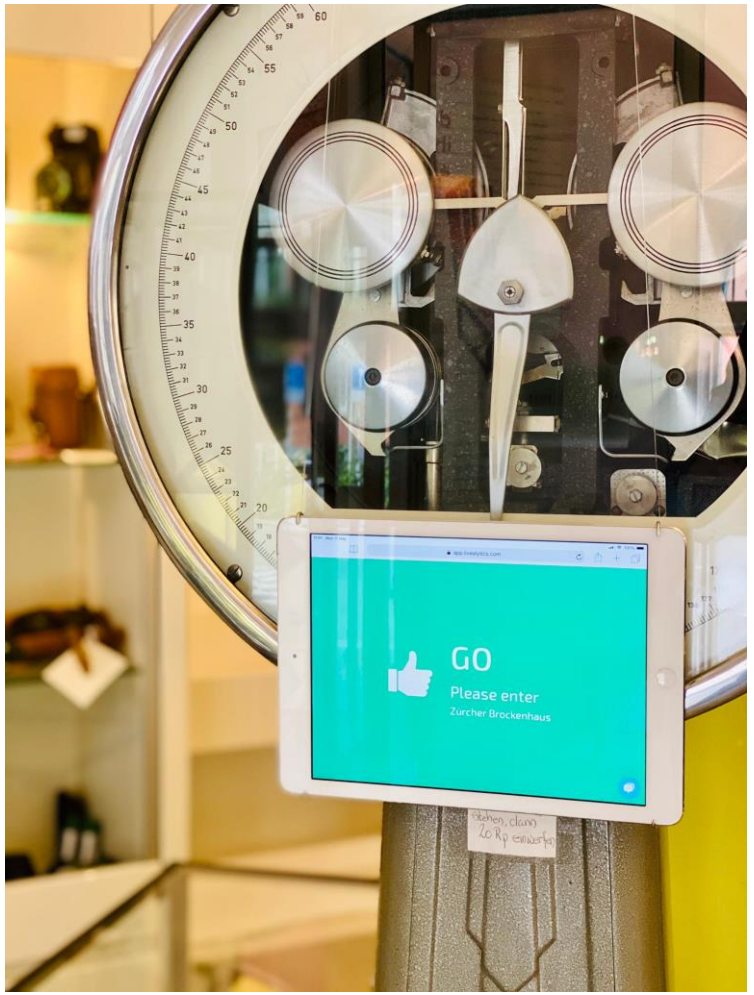
Architecture



Deployment

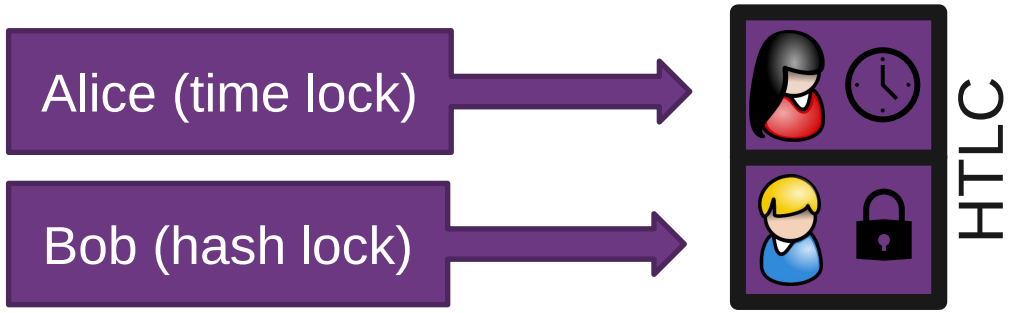
- Remote Team – Livealytics / Axerla
- Installation at Brockenhaus Zurich
  - Testvideo / Sensorvideo





Hashed Time-Locked Contracts

- Building Blocks of Cross-chain Atomic Swaps
  - Hash time lock
    - Store hashed secret – publicly stored in a smart contract
    - Unlock – if secret is provided (publicly) before timeout
    - Or
    - Unlock – after timeout



Hash time lock (HTLC)

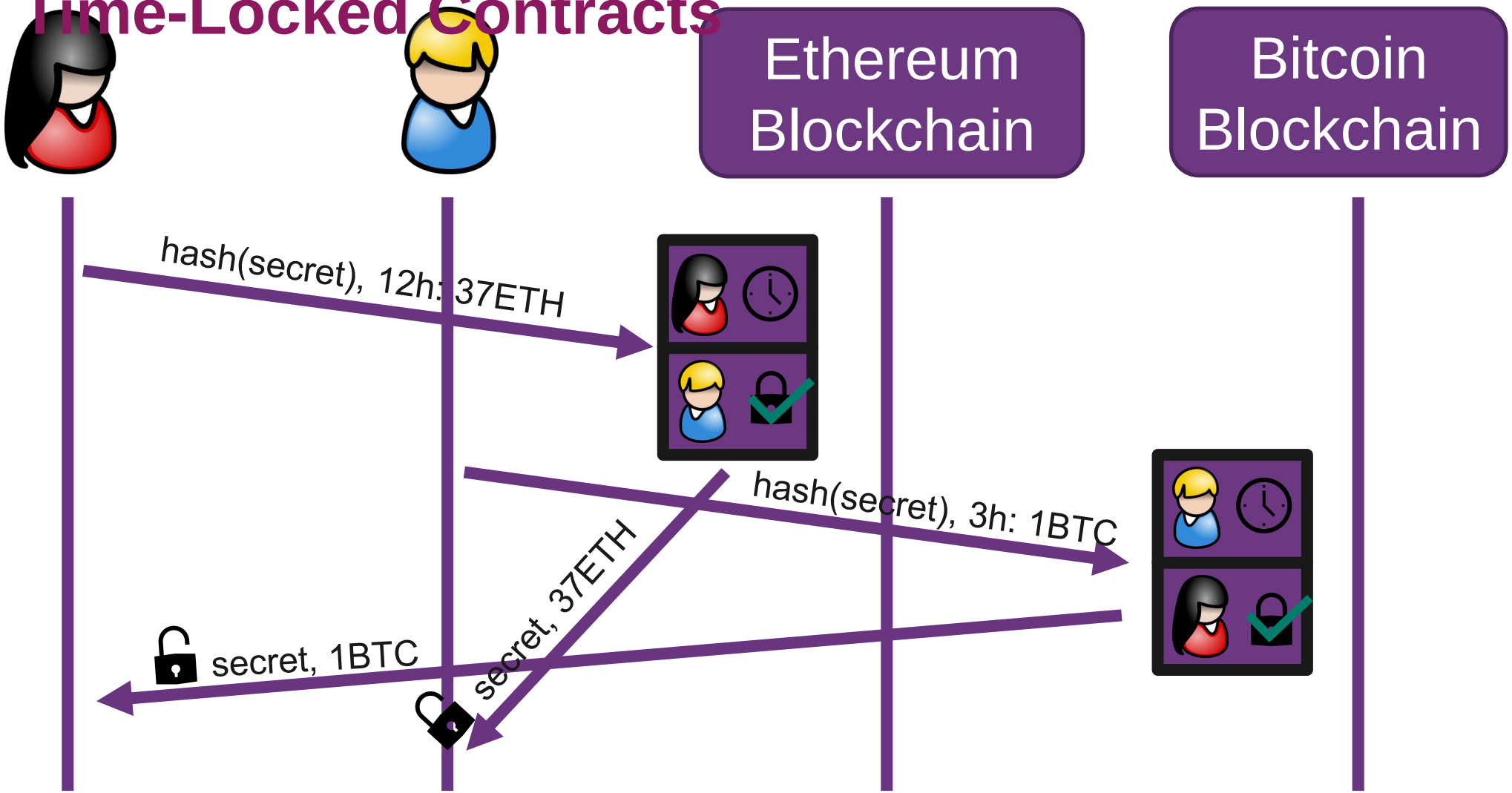
Time lock – to Alice

Hash lock – to Bob

```
OP_IF
  OP_SIZE
  AddInt64(secretSize)
  OP_EQUALVERIFY
  OP_SHA256
  AddData(secretHash)
  OP_EQUALVERIFY)
  OP_DUP
  OP_HASH160
  AddData(pkhThem[:])
OP_ELSE
  AddInt64(locktime)
  OP_CHECKLOCKTIMEVERIFY
  OP_DROP
  OP_DUP
  OP_HASH160
  AddData(pkhMe[:])
OP_ENDIF
OP_EQUALVERIFY
OP_CHECKSIG
```

OST

Hashed Time-Locked Contracts

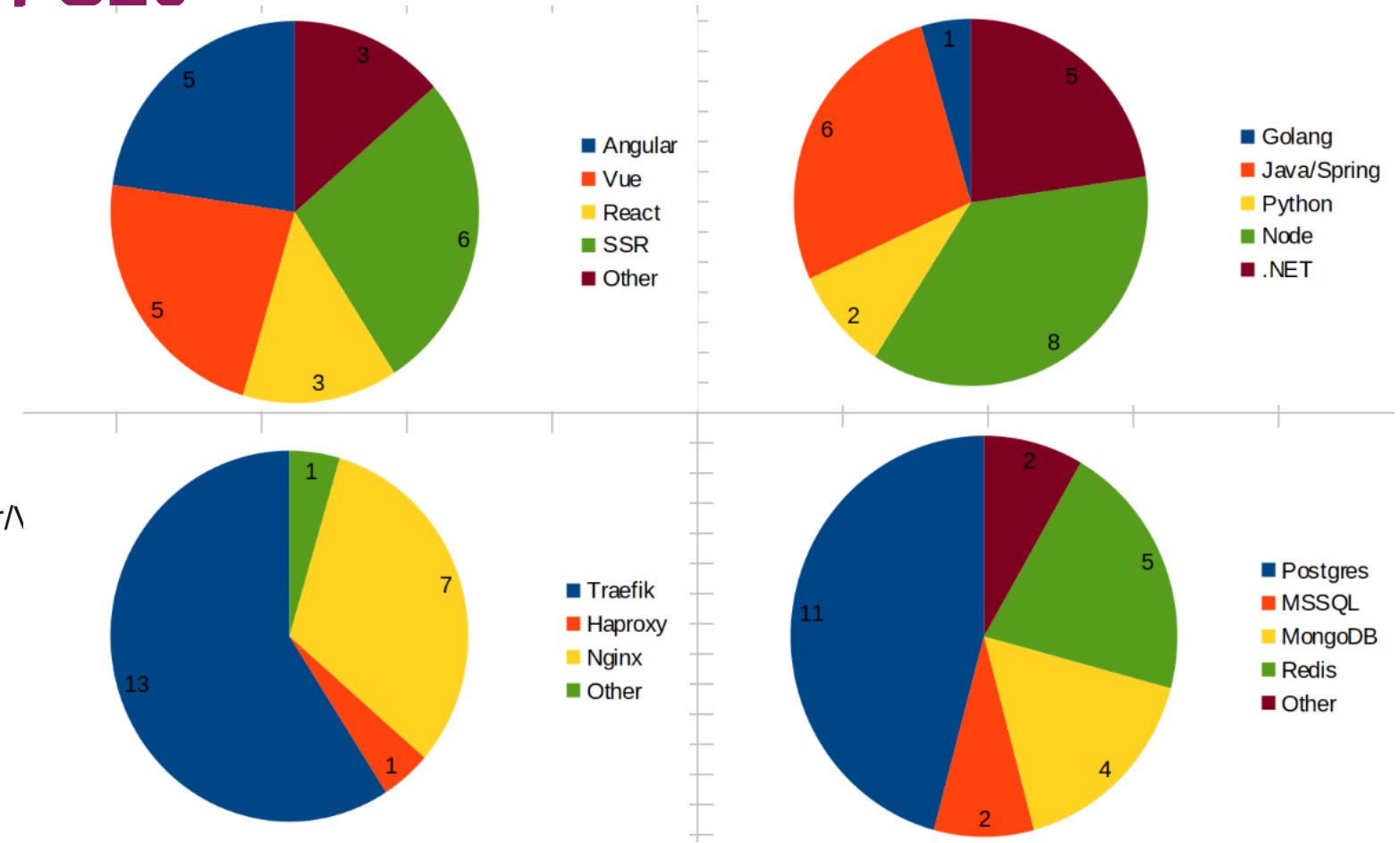


OST

Lecture 14

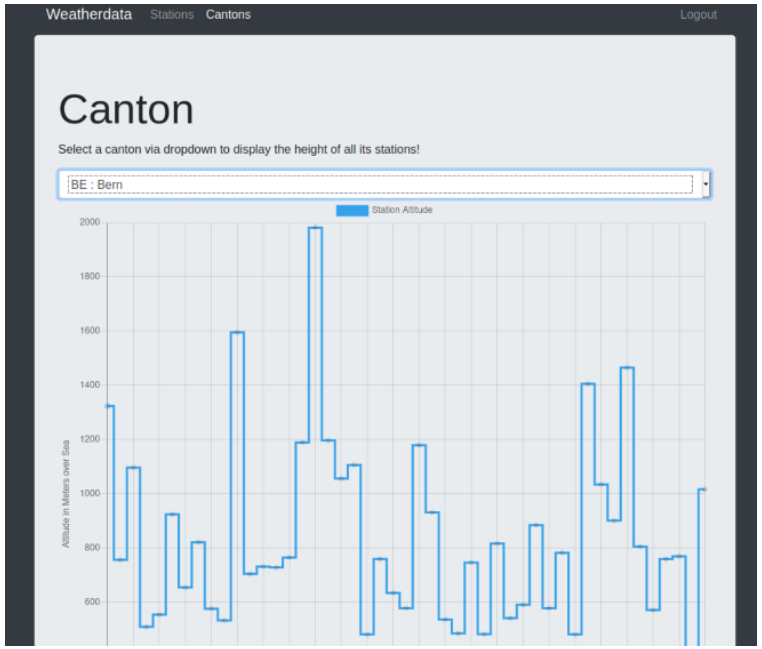
Challenge Task VSS FS20

- Only 2 Topics were similar
  - Shopping List
  - URL Shortener
  - 20 different projects
- Lots of freedom of technology
  - Most popular
    - Backend: node.js (Java/.NET)
    - Frontend: server-side rendering (Angular/\
    - Database: Postgres (Redis/MongDB)
    - Load balancer: Traefik (Nginx)
  - Least popular
    - Backend: golang
    - Frontend: React
    - Database: MSSQL
    - Load balancer: Haproxy



Challenge Task VSS FS20 Winner

- Weather Data Stack
  - Dominic Klinger, Lukas Leuenberger & Jan Ruch
  - Gather weather data from multiple locations
  - Used many concepts:
    - ActiveMQ
    - Eureka discovery services
    - Kibana
    - Docker compose – easy setup
    - Self-hosting of gitlab



OST