HS19

DISTRIBUTED SYSTEMS ADVANCED & BLOCKCHAIN

Ethereum Security, HTLC, ICOs, NEO

T. Bocek

thomas.bocek@hsr.ch

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Distributed Systems - In the News

- **21.10.2019: Libra Could Drop ‘Basket’ and Issue Individual Fiat Stablecoins**
  - issue individual coins pegged to national fiat currencies such as the dollar, pound and euro
  - “kickback from global regulators”
  - “threat to financial stability”
  - Libra is still sticking to its launch timeline of mid 2020
- **21.10.2019 Researchers Uncover Bitcoin ‘Attack’ That Could Slow or Stop Lightning Payments**
  - “It is extremely easy to execute. It takes opening a few lightning channels to key points, promising zero fees, and then not relaying any payments,” [paper]
- **17.10.2019 The Distributed Bloom Filter**

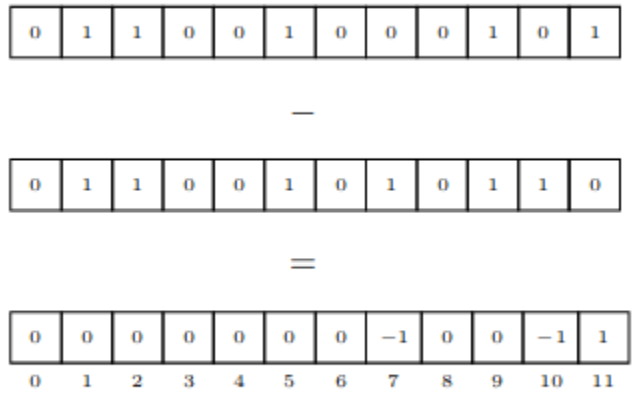


Figure 3: Reducing a bloom filter from another bloom filter

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 ~ 15s
  9. Short Address Attack – not yet exploited
  10. Unknown Unknowns

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

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

Christoph Jentzsch

URL: b.socrative.com/student/

Room: HSR

ERC20 - Considerations

- **ERC865: Pay transfers in tokens instead of gas, in one transaction**
  - Delegate transfer of tokens to a third party
  - UX: pay service in tokens – user needs to have ETH **and** tokens
    - Create account at exchange, KYC, Bank transfer – and wait
- **Client could be offchain, third party not**
  - Make sure the signature is unique
- **ERC677**
  - Add transferAndCallPerSigned()

```
 function transferAndCallPreSigned(bytes memory _signature, address _to, uint256 _value,
uint256 _fee, uint256 _nonce,
        bytes4 _methodName, bytes memory _args) public returns (bytes memory) {

        require(contractWhitelist[_to]);
        require(!signatures[_signature]);
        bytes32 hashedTx = Utils.transferAndCallPreSignedHashing(address(this), _to,
_value, _fee, _nonce, _methodName, _args);
        address from = Utils.recover(hashedTx, _signature);

        require(from != address(0));
        require(!nonces[from][_nonce]);

        _transfer(from, _to, _value, _fee, msg.sender);
        signatures[_signature] = true;
        nonces[from][_nonce] = true;

        emit Transfer(from, msg.sender, _fee);
        emit TransferAndCallPreSigned(from, _to, msg.sender, _value, _fee, _methodName,
_args);

        // call receiver
        require(Utils.isContract(_to));

        //call on behalf of from and not msg.sender
        (bool success, bytes memory data) =
_to.call(abi.encodePacked(abi.encodeWithSelector(_methodName, from, _value), _args));
        require(success);
        return data;
    }
```

EVM Considerations - 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
- **Commitment Scheme**
  - Real world: flip coin
  - Ethereum: flip coin
    - Offering a Bet
    - Taking a Bet
    - Revealing the Flip
- **Commitment Scheme: random number**
  - Set expiration, e.g., 1 day, max players: 2
  - Commit phase:
    - Player 1: random value, store its hash, pay 1 ETH
    - Player 2: random value, store its hash, pay 1 ETH
  - If expiration passed: if commit not complete, each player gets back its ETH
  - Reveal, new phase, new expiration
    - Player 1: reveal random number on chain
    - Player 2: reveal random number on chain
      - If player 2 decides not to reveal, due to unfavorable result, lose 1 ETH after expiration
  - Now, take XOR of both random values, use this as random value

EVM Considerations - Random Numbers

- **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 low (miner gets 2 ETH)

1. Pot has e.g. 10 ETH
2. Player 1: pay 1 ETH in pot
   1. Bet: future blockhash % 2 == 0
3. Contract stores who send ETH and in which block (blockNr)
4. Player 2: pay 1 ETH in pot
5. Contract uses current blockhash from blockNr to test blockhash % 2 == 0
6. If true, 2 ETH goes to player 1 he loses

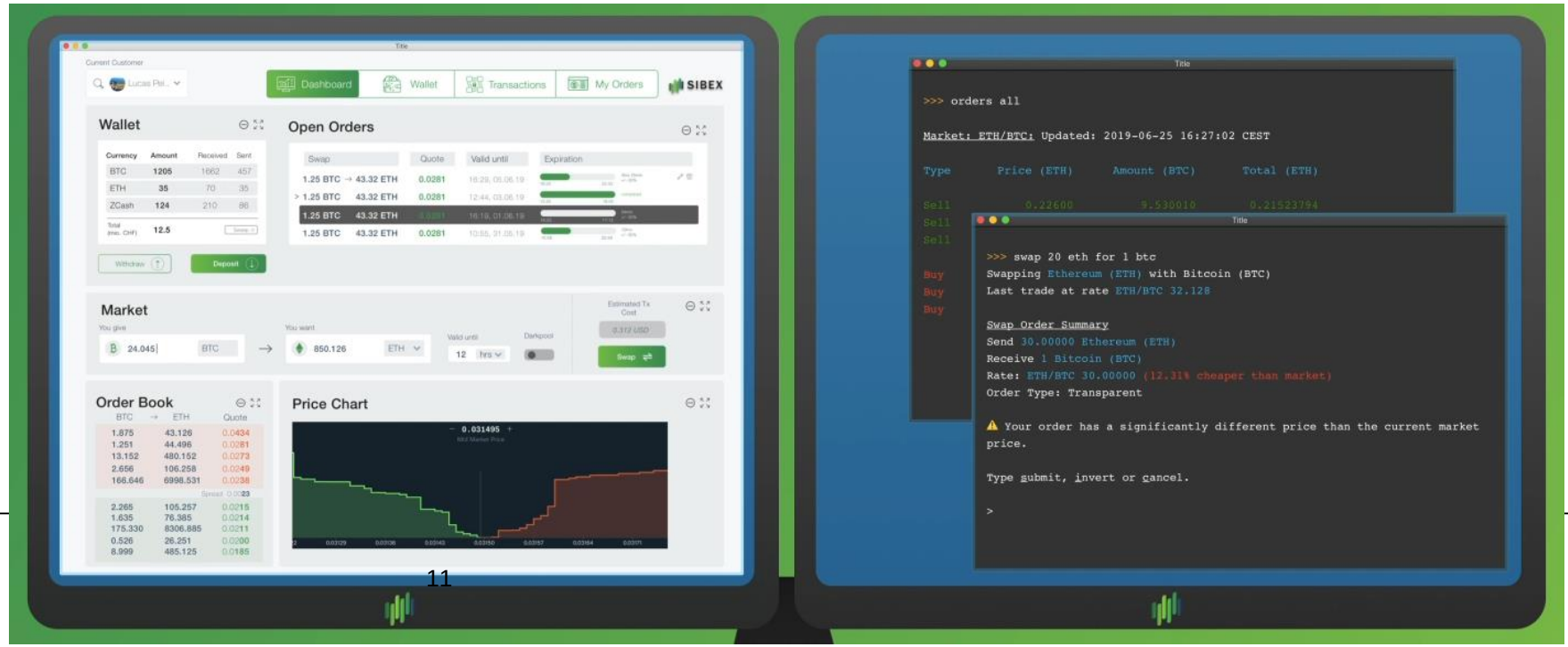
```
function transfer(address _to, uint256 _value) public returns (bool) {
     …
        //since this is a lucky coin, the value transferred is not what you
expect
        luckyTransfer();
        previousTransferBlockNr = block.number;
        previousTransferAddress = msg.sender;
        …
      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

URL: b.socrative.com/student/

Room: HSR

Hashed Timelock Contracts (HTLC)

- **Sibex CLI and GUI for cross chain trading**
  - Fully P2P, no need for central financial intermediary.
- **Uses Atomic Swaps**
  - Based on Hashed Timelock Contracts (HTLC)
  - Conditional payment
  - Either know the input of a hash, or time expired
- **First described by Tier Nolan back in 2013**



Atomic Swaps

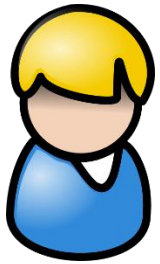
```
function newHTLC(address payable receiver, uint192 lockTime, bytes32
hashedSecret) external payable returns (bytes32 contractId) {

        require(msg.value > 0, "msg.value must be > 0");
        require(lockTime > now, "timelock time must be in the future");

        //encodePacked -> unpadded encoding
        contractId = sha256(abi.encodePacked(msg.sender, receiver, msg.value,
                            lockTime, hashedSecret));

        require(contracts[contractId].sender == address(0), "contract cannot
                exist");

        contracts[contractId] = LockContract(msg.sender, receiver, msg.value,
                                lockTime, hashedSecret, 0x0);
        emit NewHTLC(contractId, msg.sender, receiver, msg.value, lockTime,
             hashedSecret);

        return contractId;
    }
```

```
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
```

Atomic Swaps

- **Alice want to exchange 200 ETH for 10BTC with Bob, Bob agrees**
  - First, negotiate price, exchange public keys/addresses for ETH/BTC
  - Alice (initiator) creates “secret”, address BTC_A, ETH_A, shares with Bob, hash(secret), BTC_A,ETH_A
  - Bob now knows hash(secrect) and shares BTC_B, ETH_B

Alice



Bob

Blockchain BTC

Blockchain ETH

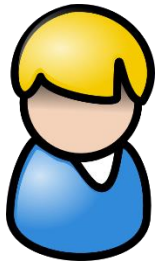
Blockchain BTC

Blockchain ETH

Atomic Swaps

- **Alice want to exchange 200 ETH for 10BTC with Bob, Bob agrees**
  - Alice sends 200 ETH to lock contract with condition
    - After timeout: Alice gets back 200 ETH
    - Within time: Bob gets 200ETH if he provides the secret

Alice



Bob

Blockchain BTC

Blockchain ETH

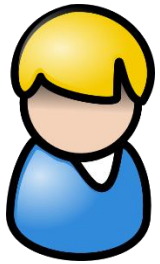
Blockchain BTC

Blockchain ETH

Atomic Swaps

- **Alice want to exchange 200 ETH for 10BTC with Bob, Bob agrees**
  - Bob sees the 200 ETH with the hashed secret, pays in 10 BTC with the same condition
  - If Alice wants the 10 BTC, Alice needs to reveal the secret onchain
  - Bob sees that and can redeem the 200 ETH

Alice



Bob

Blockchain BTC

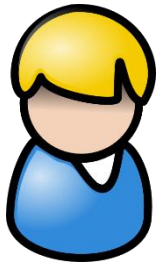
Blockchain ETH

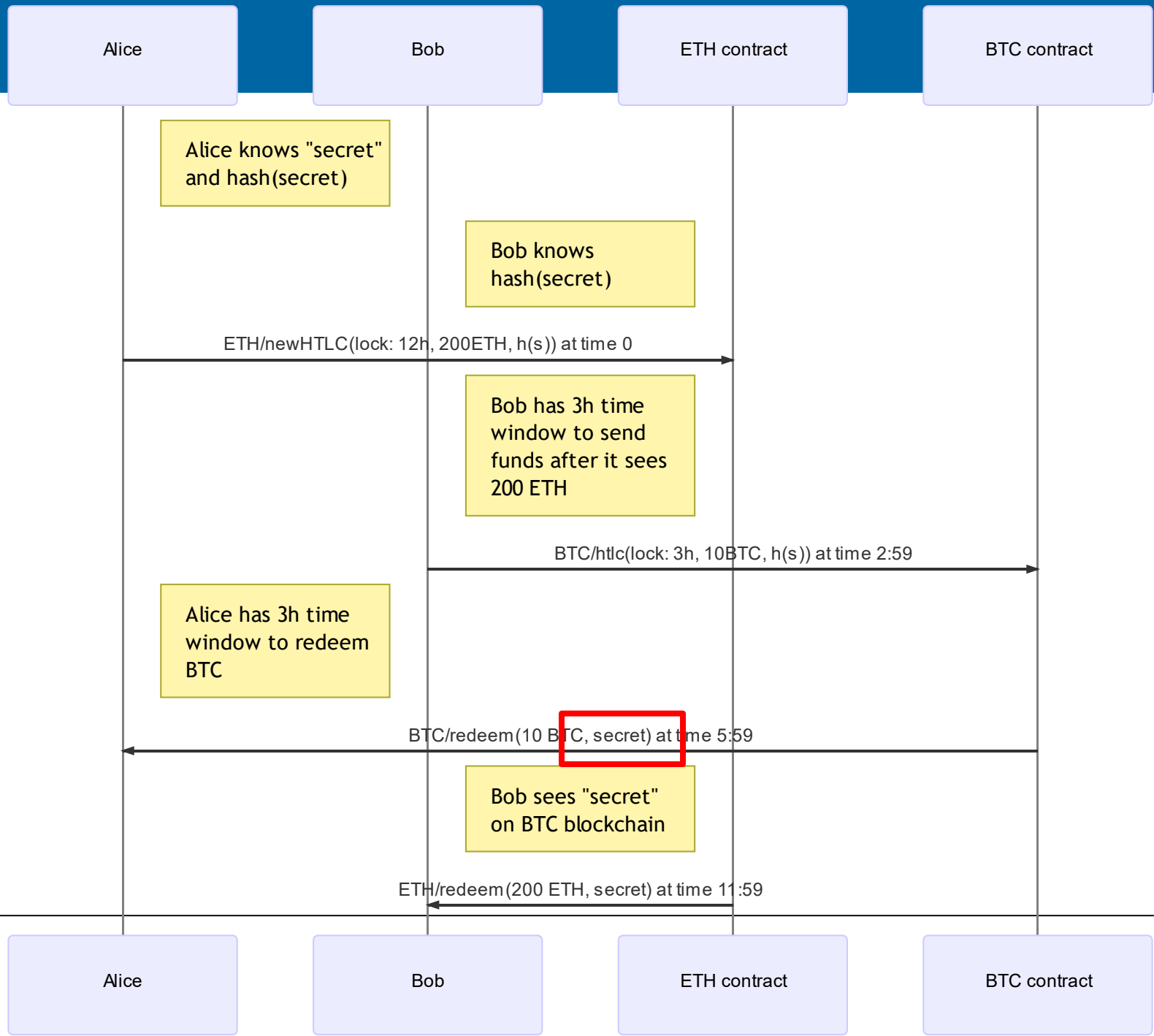
Blockchain BTC

Blockchain ETH

Atomic Swaps

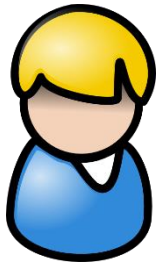
- **Goodpath, only delay**
- **Alice needs to reveal secret in order to redeem 10 BTC**
- **If Bob sees secret, he can redeem 200 ETH**

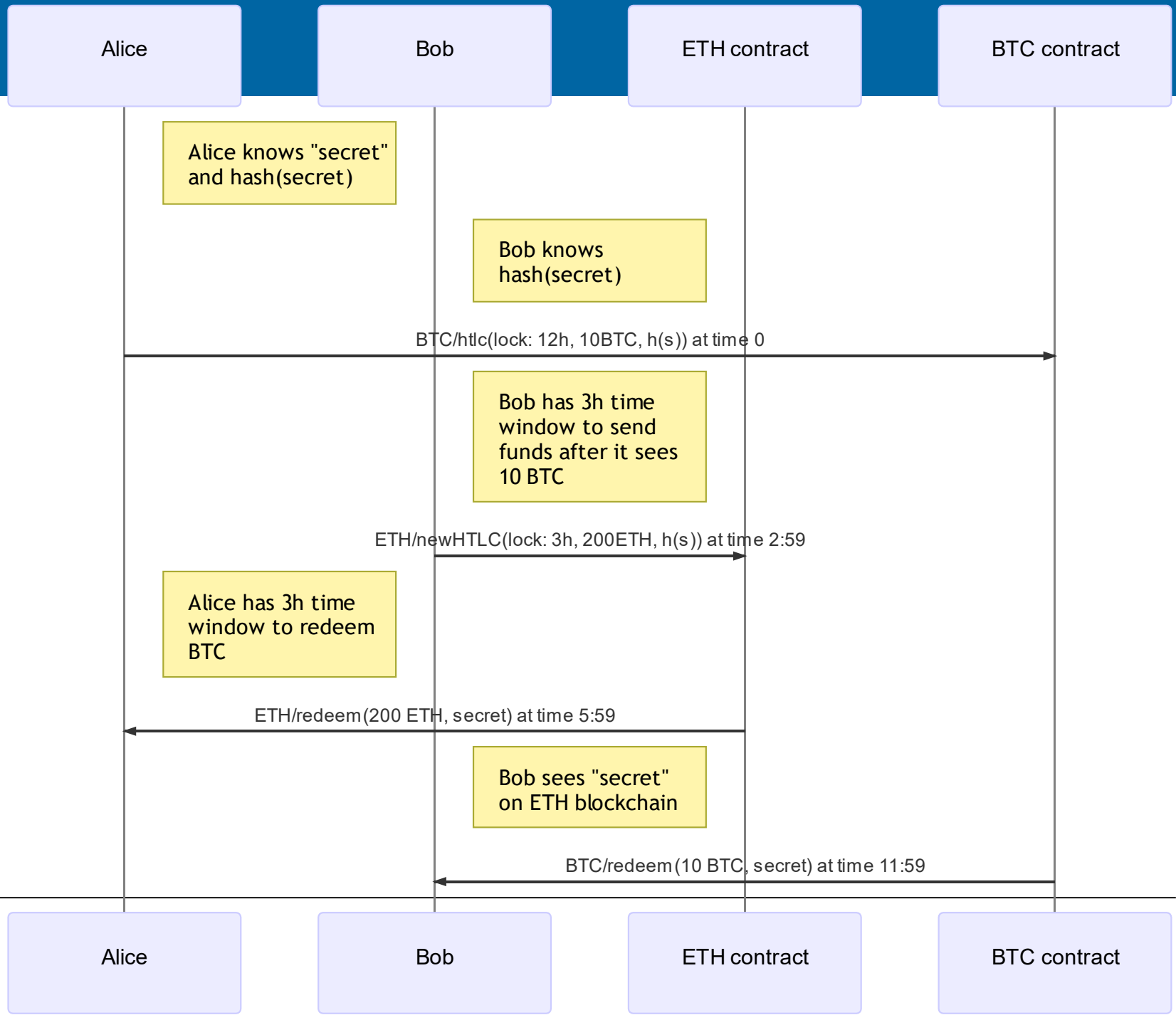




Atomic Swaps

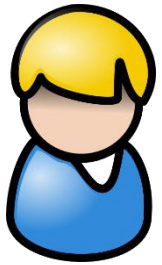
- **Goodpath, only delay**
- **Alice needs to reveal secret in order to redeem 200 ETH**
- **If Bob sees secret he can redeem 10 BTC**

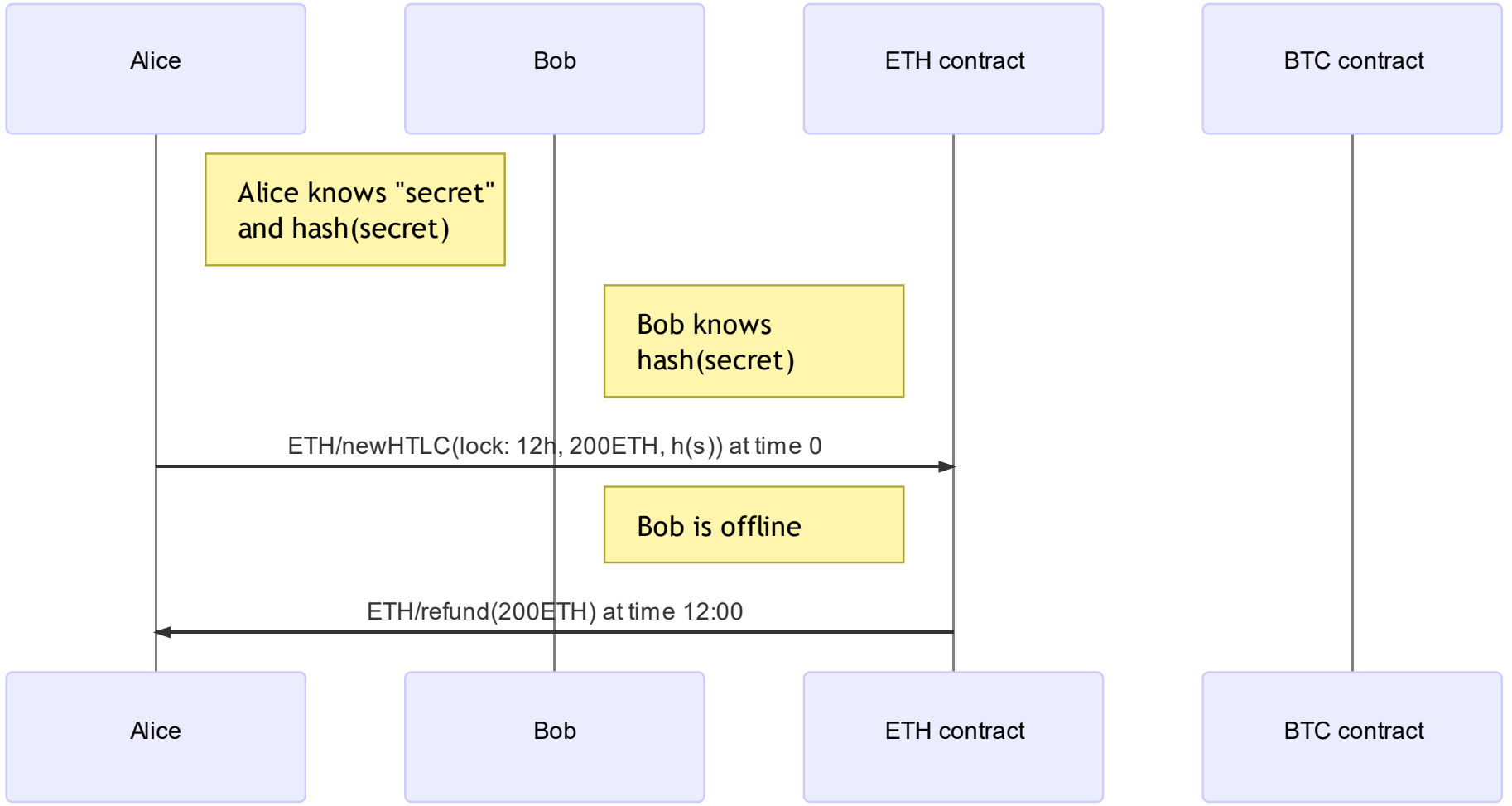




Atomic Swaps

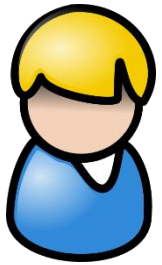
- **Bob is offline**

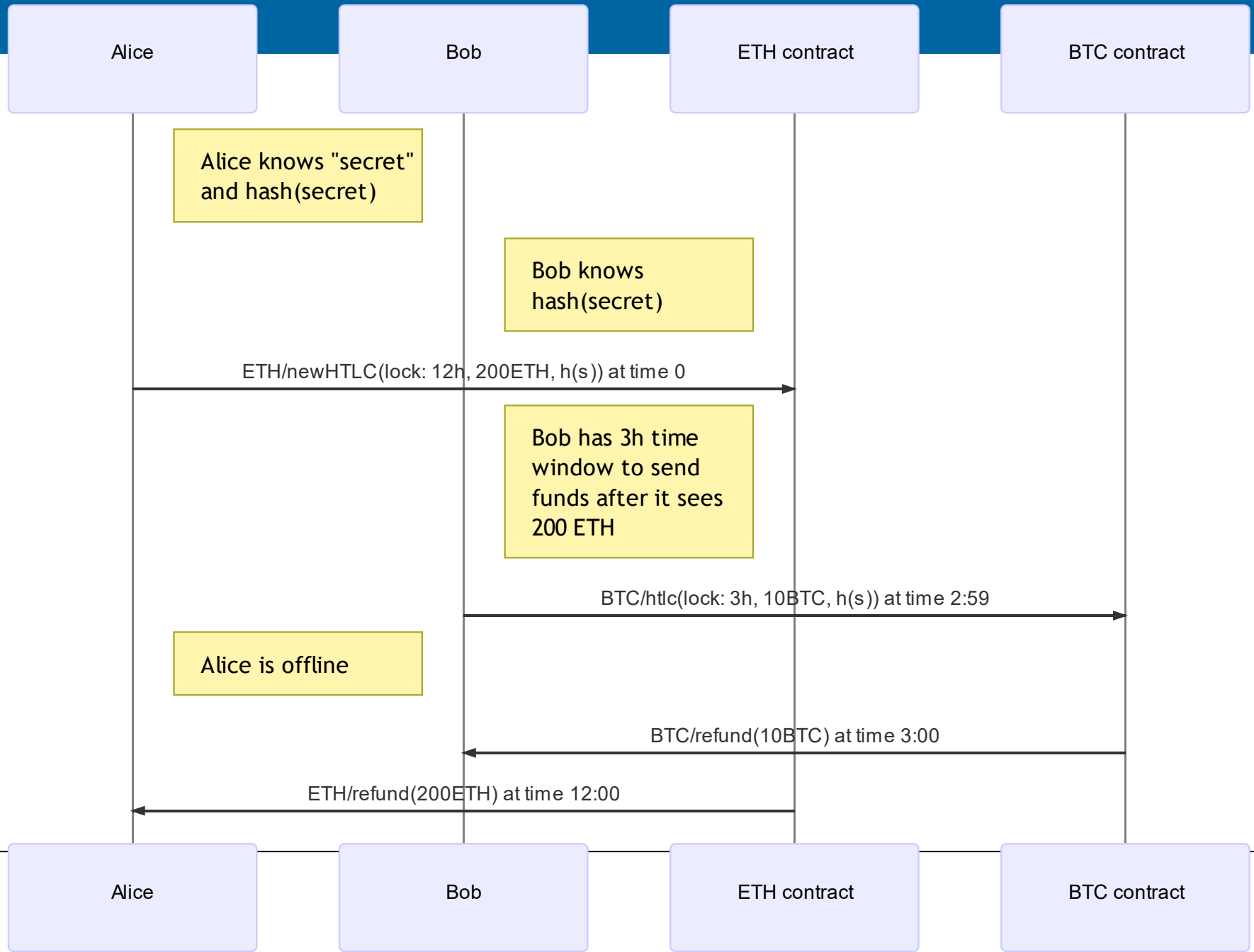




Atomic Swaps

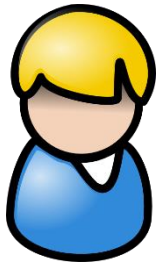
- **Alice is offline**

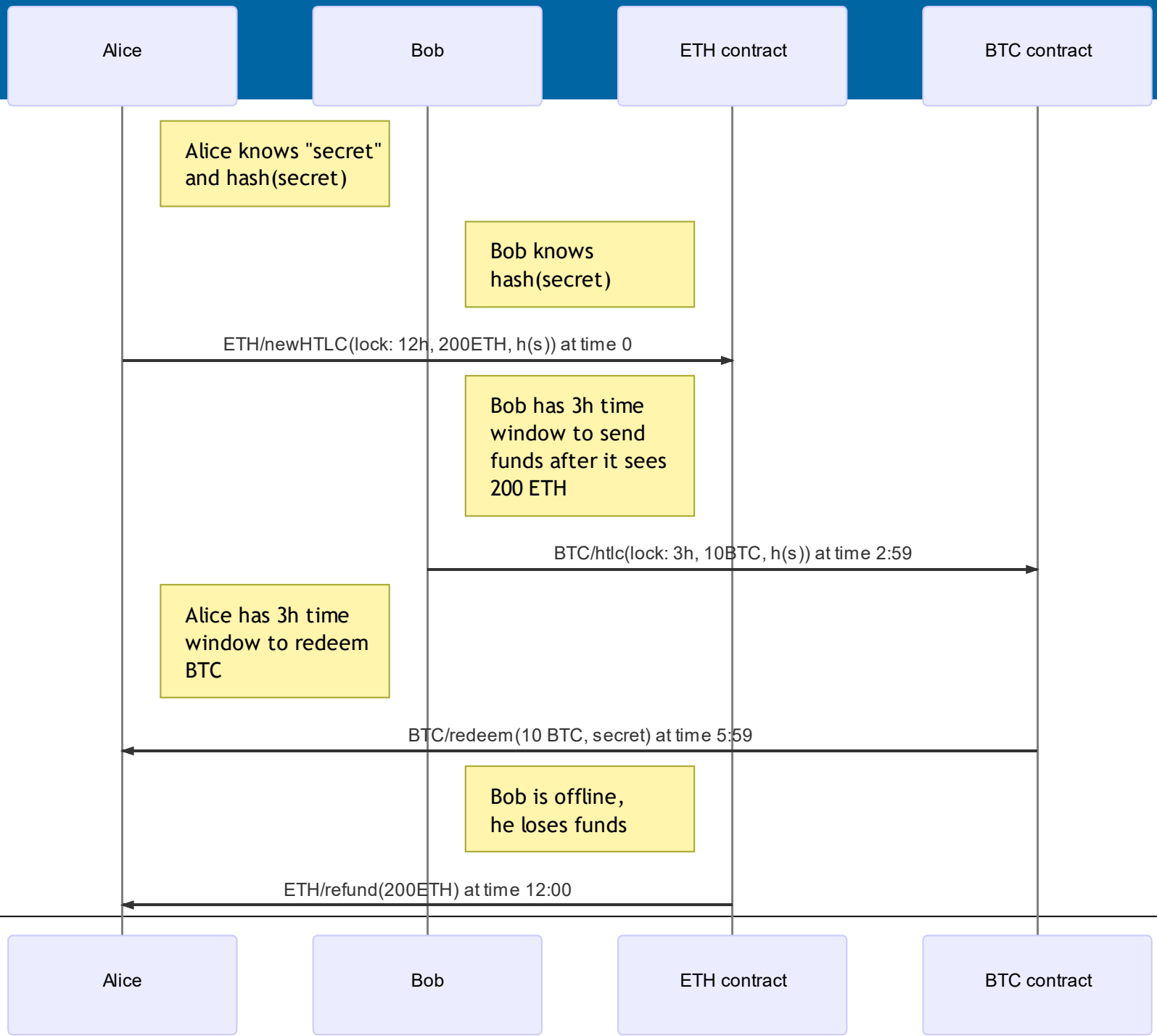




Atomic Swaps

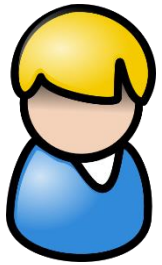
- **Bob loses funds**
  - Worst case

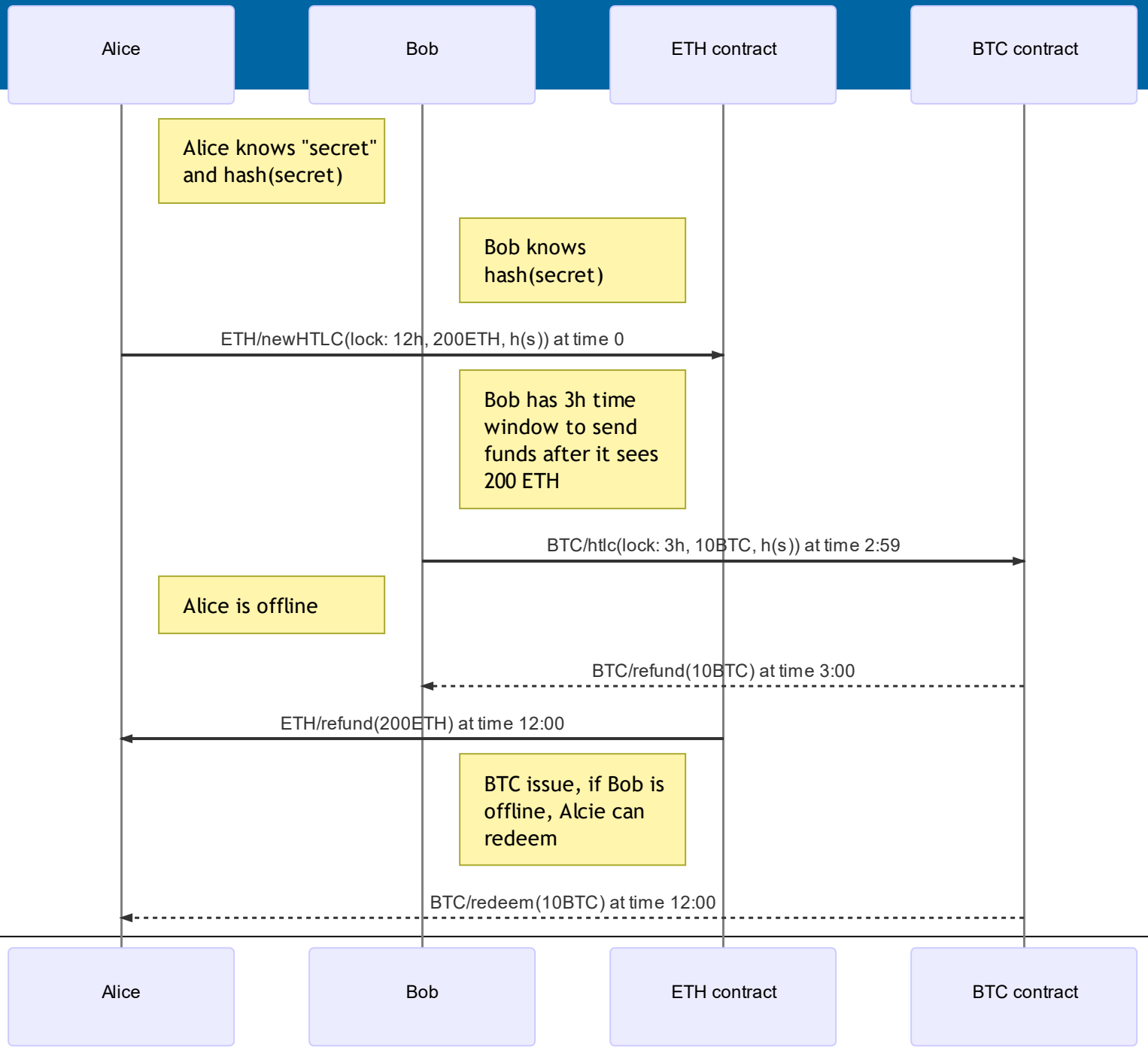




Atomic Swaps

- **Bob loses funds**
  - BTC issue



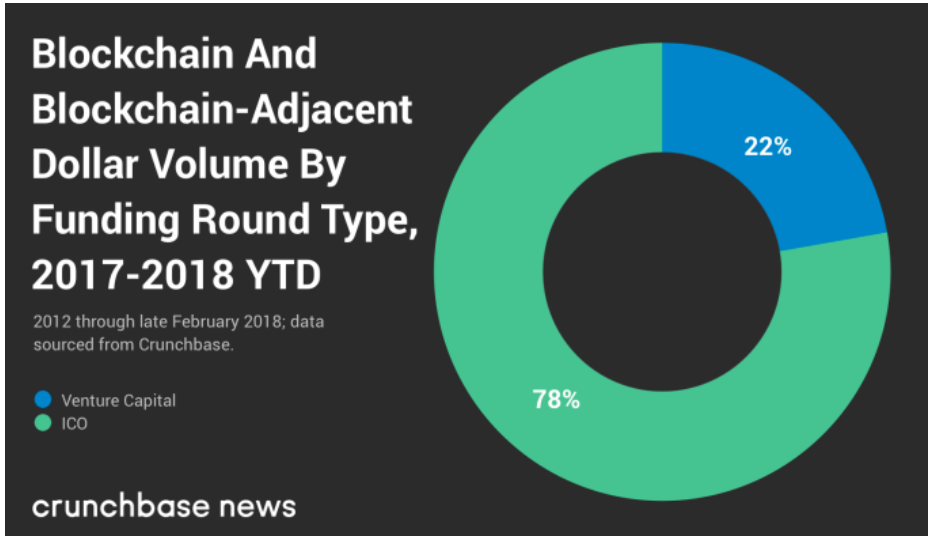


URL: b.socrative.com/student/

Room: HSR

What is an ICO

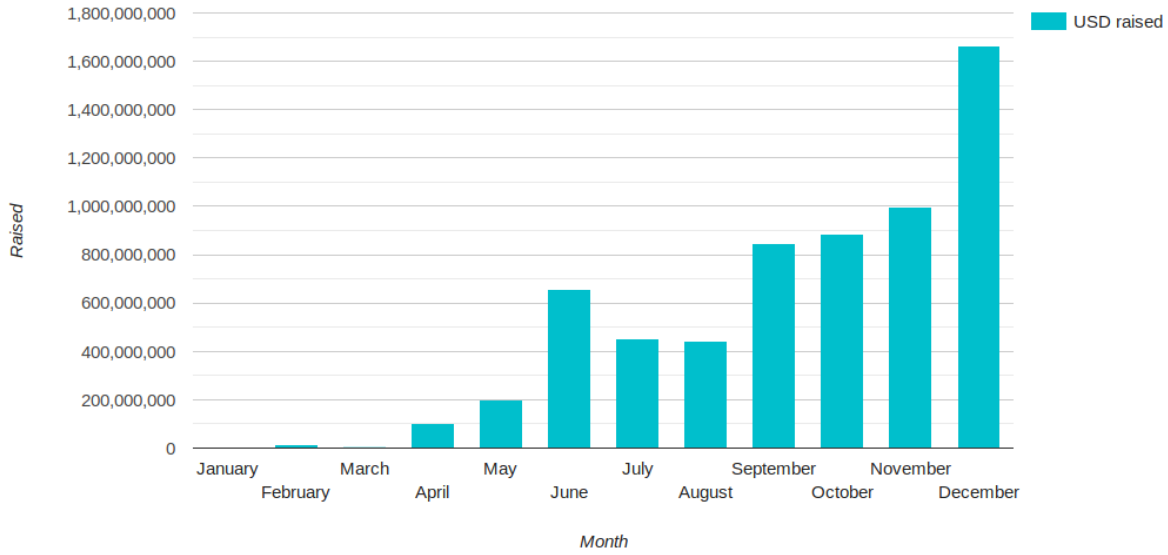
- **1st Blockchain Killer App**
  - Lots of tokens in Solidity / Ethereum, ERC20
- **An initial coin offering (ICO) is a means of crowdfunding**
  - Tokensale - release of a new cryptocurrency
  - 1490 listed cryptocurrencies with a marketcap, 4942 with coinlib.io, over 200K ERC20 contracts on mainnet
- **ICOs vs IPOs**
  - ICO little regulation, IPO are heavily regulated
  - ICO short in duration, IPO much longer
  - IPO are exclusive, ICO are open to anyone
- **TechCrunch: Over 1000 Crypto Projects Are Considered 'Dead' Now**
- **ICOs delivered at least 3.5x more capital to blockchain startups than VC since 2017**



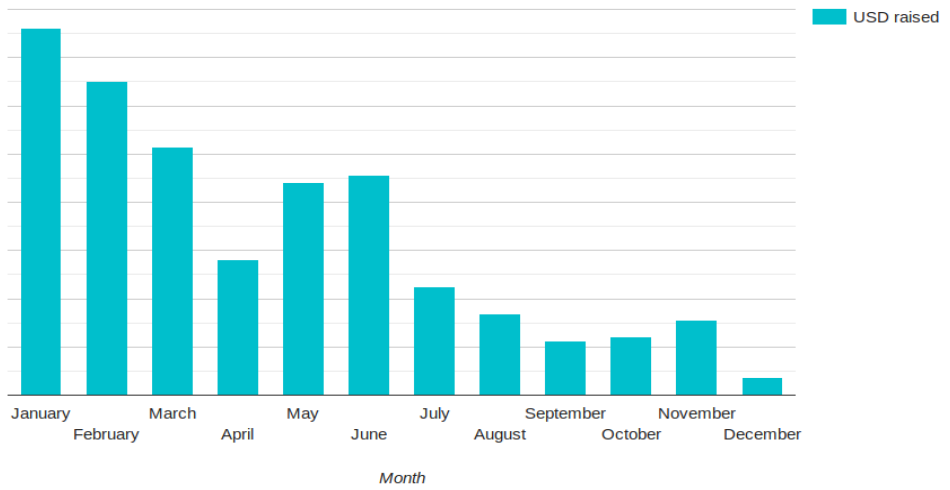
- **Wall Street Journal: 20 percent of all ICOs are fraudulent**
- **SATIS Group Report: '78% of ICOs are Scams'**

Total amount raised by all ICOs in total ($)

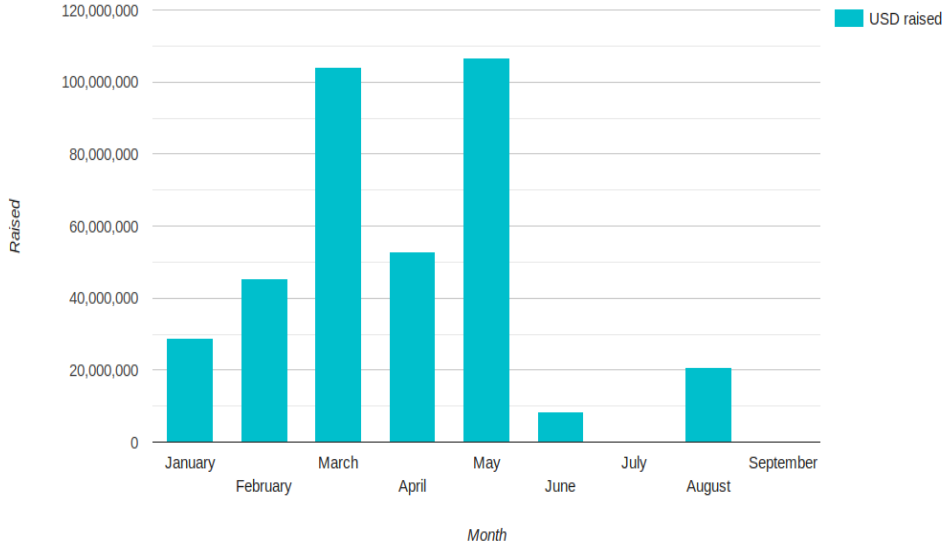
2017



2018



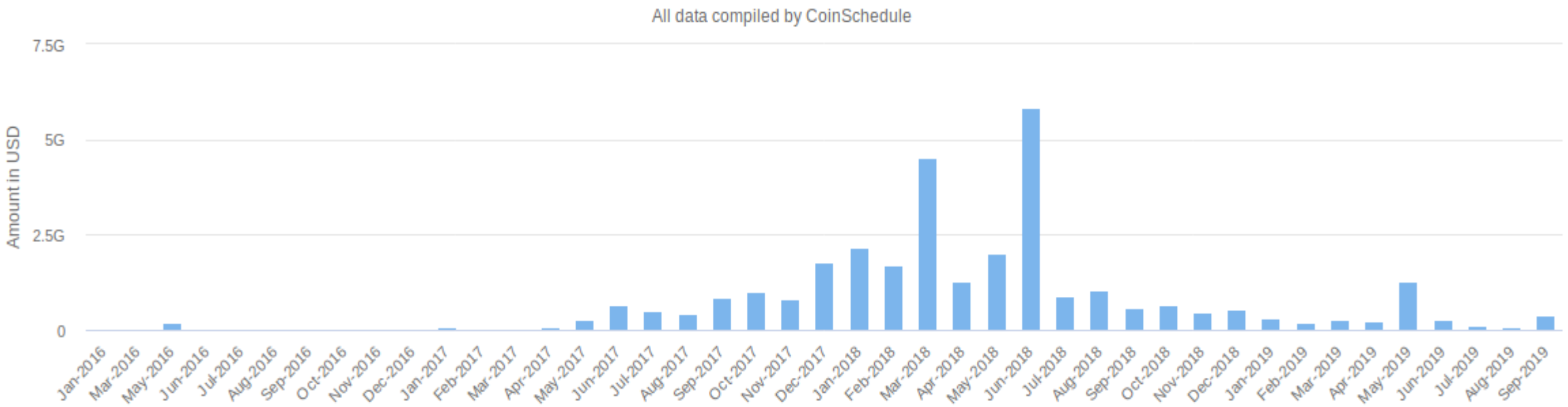
2019



Source: icodata.io

Total amount raised by all ICOs in total ($mln)

Source: Coinschedule



ICO countries and categories until 2018

Number of projects

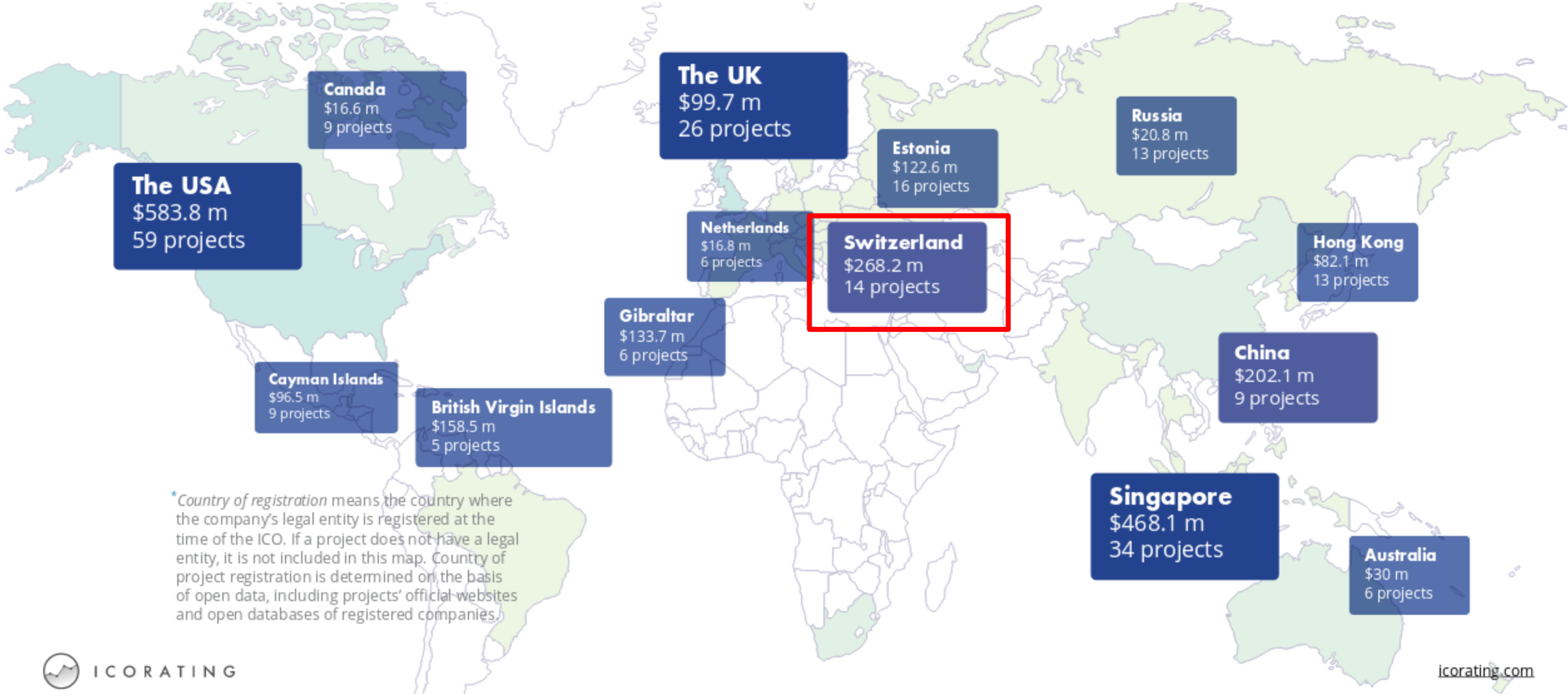
	Country	% of Projects	Total Raised
1	USA	15.94%	875m $
2	UK	9.6%	184m $
3	Singapore	8.3%	340m $
4	Russia	7.02%	1003m $
5	Switzerland	6.58%	547m $
6	Estonia	4.09%	34m $
7	Australia	3.36%	31m $

Total Raised

	Country	% of Projects	Total Raised
1	Russia	7.02%	1003m $
2	USA	15.94%	875m $
3	Switzerland	6.58%	547m $
4	Singapore	7.02%	340m $
5	UK	9.6%	184m $
6	Estonia	4.09%	34m $
7	Australia	3.36%	31m $

Source: icowatchlist

ICO countries and categories



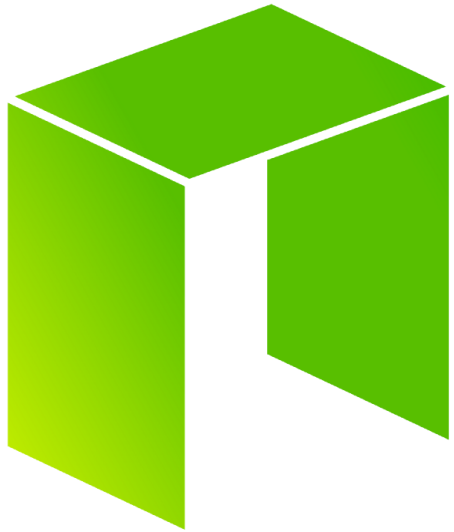
NEO

- **Hongfei Da, founder of NEO**
- **«Ethereum of China»**
  - Pushing engagement outside China
  - Opinion: NEO: Is China Holding Its Best Blockchain Asset Back?
    - China's ban on ICOs and cryptocurrency trading
    - Other opinion
- **Formerly AntShares (2014) – rebranded**
  - Shanghai-based blockchain R&D company "OnChain" – 2 ICOs (in China!!)
    - 1st 550K USD (2100BTC), 2nd 4.5m USD

NEO Political

- **"NEO Sponsor Giveback Plan" Announcement**
  - Give back in the amount equal to the fiat value they contributed back then
  - «ICO, a highly-efficient yet controversial financing method has its two sides.»
  - China not ICO friendly, NEO gives back ICO investment
    - “undoing ICO”, but keeps price gains

- **NEO is**
  - Digital Assets + Digital Identity + Smart Contract = Smart Economy
- **Active Blockchain**
  - Every block ~1-10 Tx
- **July 2018: NEO blockchain starts decentralization**
  - NEO == centralized consensus?
  - KPN and CoZ - decentralization
    - “City of Zion is a global, independent group of open source developers, designers and translators which support the NEO core and ecosystem.”



NEO
smart economy

NEO Token

- **NEO and NeoGas**
  - Pre-mined, max 100 million tokens, 15 sec block time
- **50/50**
  - 50m distributed during ICO, 50m is kept by NEO
  - ~350m USD investment power
  - 15% dedicated to other blockchain projects
- **GAS is generated with each new block**
  - Total limit of 100 million GAS will be achieved in about 22 years
  - GAS will be distributed proportionally in accordance with the NEO holding ratio
  - 8 GAS every block, reduced by 1 ever 2m blocks

#	Name	Market Cap	Price	Volume (24h)	Circulating Supply	Change (24h)	Price Graph (7d)
12	IOTA	$1,377,729,855	$0.495670	$18,078,573	2,779,530,283 MIOTA *	-3.15%	
13	Dash	$1,292,529,942	$154.06	$166,767,055	8,389,597 DASH	-3.36%	
14	Binance Coin	$1,269,918,405	$9.71	$25,522,164	130,799,315 BNB *	-2.12%	
15	NEO	$1,060,068,089	$16.31	$339,487,764	65,000,000 NEO *	-5.20%	
16	Ethereum Classic	$1,006,284,229	$9.55	$136,069,596	105,385,599 ETC	-1.98%	
17	NEM	$847,576,709	$0.094175	$5,161,489	8,999,999,999 XEM *	-1.67%	
18	Tezos	$793,965,923	$1.31	$2,501,514	607,489,041 XTZ *	-4.41%	

NEO Token / Technical

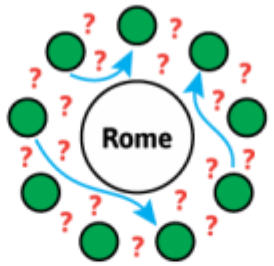
- **Consensus mechanism: dBFT**
  - Delegated Byzantine Fault Tolerant
  - Delegates create block, 2/3 need to approve through voting:
  - Works if more than 66% honest
- **Byzantine Generals' Problem**

#	Name	Market Cap	Price	Volume (24h)	Circulating Supply	Change (24h)	Price Graph (7d)
105	Mixin	$54,218,864	$122.19	$8,557	443,743 XIN *	-1.01%	
106	ZCoin	$54,150,119	$9.40	$555,789	5,757,841 XZC	-4.66%	
107	QuarkChain	$54,007,916	$0.067371	$5,095,024	801,649,919 QKC *	0.05%	
108	Gas	$52,051,444	$5.14	$581,217	10,128,375 GAS *	-4.42%	
109	Bitcoin Private	$50,922,778	$2.48	$95,935	20,524,490 BTCP	-2.32%	
110	Syscoin	$50,658,818	$0.093483	$221,244	541,902,755 SYS	-2.23%	
111	SALT	$48,036,029	$0.578418	$1,212,934	83,047,261 SALT *	-3.17%	

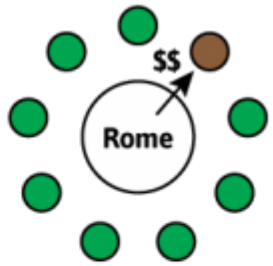
Imagine Rome being besieged by **nine armies**, each commanded by a (Byzantine) general.

In order to launch a succesful **attack** or retreat, all armies have to **do the same**, otherwise they will be decimated by Rome's armies.

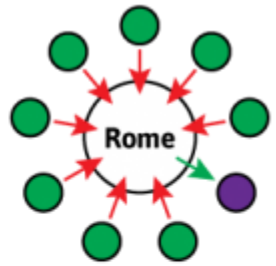
The decision to either **attack** or retreat is put up to a **vote**. Whichever option receives **more than 50%** of the votes, that's what the Generals will do (**retreat** in the example above).



Problem 1
The generals communicate by using **couriers**, who have to cross unknown areas controlled by the Romans, risking capture or their message becoming corrupt.



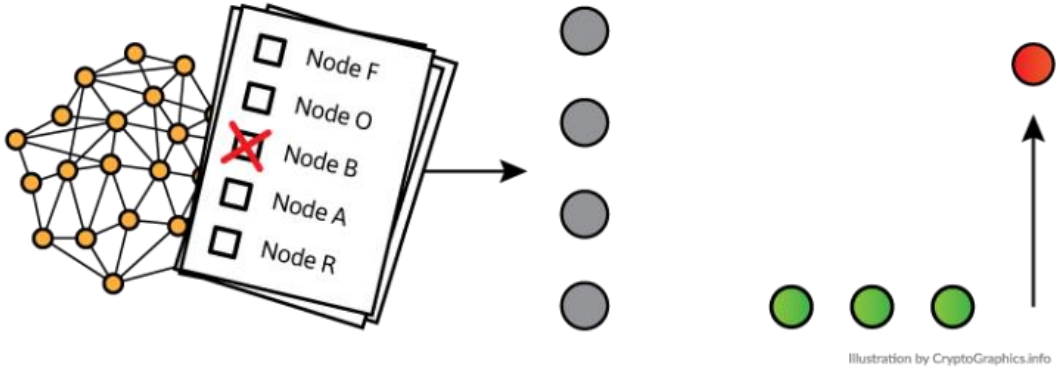
Problem 2
Each of the generals could be bribed by the Romans: **Traitorous Generals**.



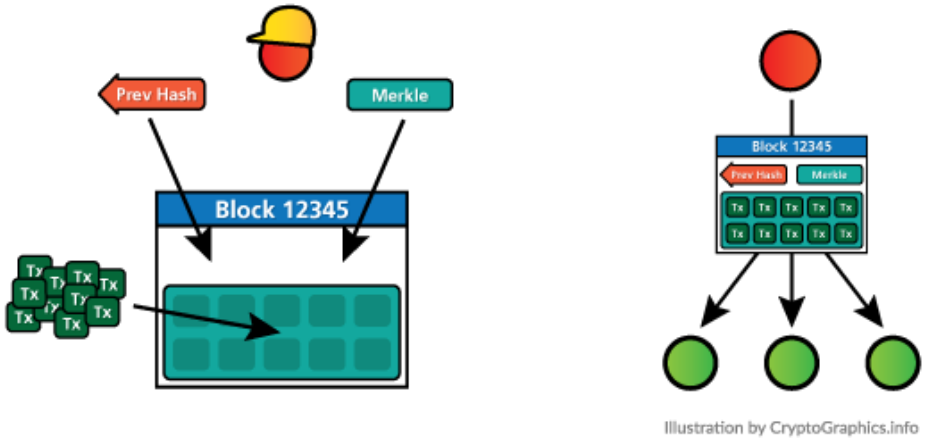
Problem 3
Any of the Generals can make the wrong decision, regardless of the vote: **Improperly Functioning Generals**.

Delegated Byzantine Fault Tolerant

- **Nodes in the network elect a group of consensus nodes (e.g. CoZ)**
- **Leader/speaker randomly chosen from consensus nodes, rest are delegates**



- **Leader/speaker creates new block, needs to be positively checked by 2/3 of delegates**
- **If 2/3 agrees, block added to the chain**

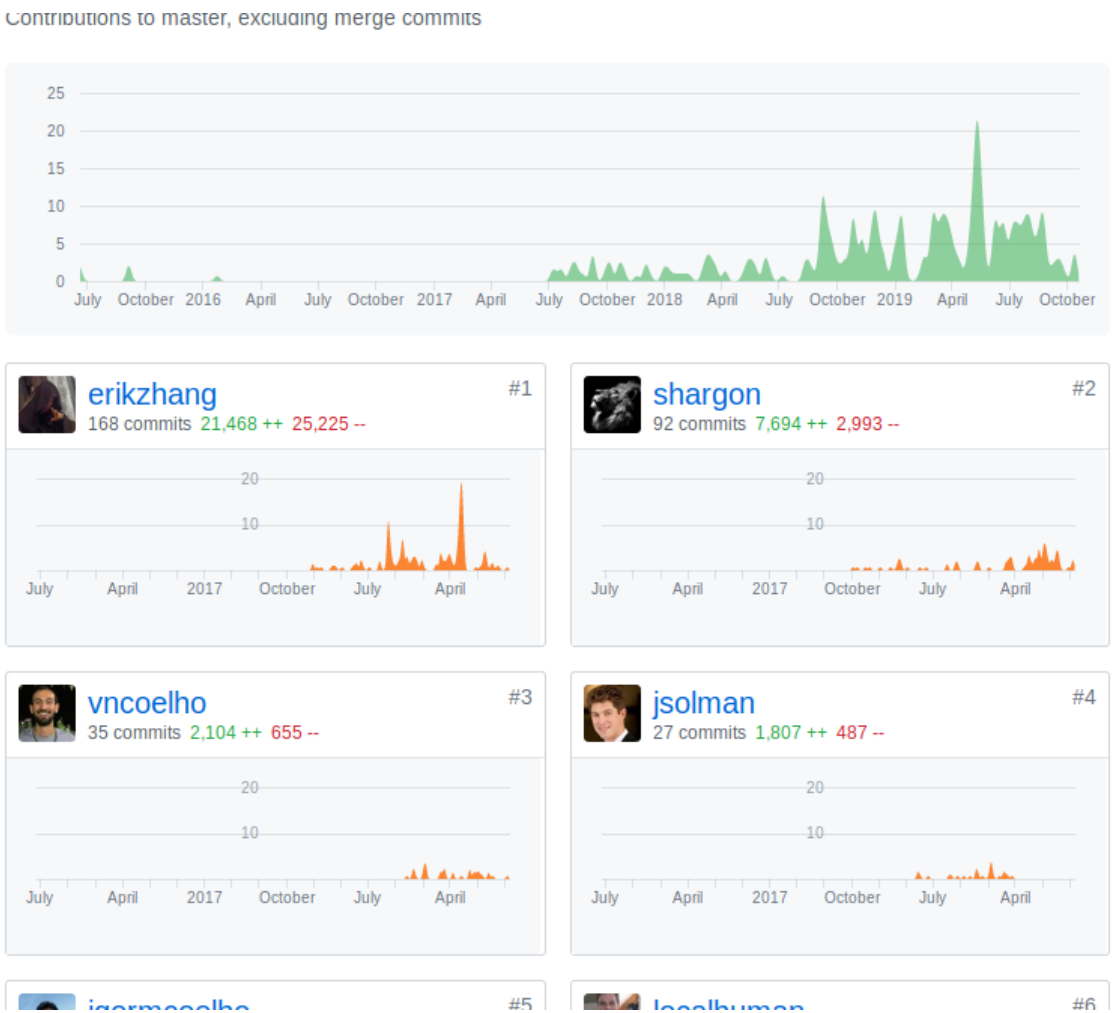


- **Countermeasures for dishonest leader/speakers or delegates**

(source)

NEO Technical

- **C# driven**
  - Github repos – shows activity, e.g., neo
- **Smart Contracts:**
  - C#, Python, Golang
    - “Ethereum requires developers to learn to program with Solidity. Neo, on the other hand, will support almost all programming languages via a compiler, including those on Microsoft.net, Java, Kotlin, Go and Python”
  - NeoVM – Stack-based VM
- **15-20s block time**
  - 1000 TPS, claiming to reach 10000 TPS
- **NeoVM JIT, NeoX (Cross-chain interoperability agreement) / NeoFS (Distributed Storage Protocol) / NeoQS (Anti-quantum cryptography mechanism)**
  - Ideas in the whitepaper



URL: b.socrative.com/student/

Room: HSR

Token Interface Comparison: Ethereum / NEO

ERC-20

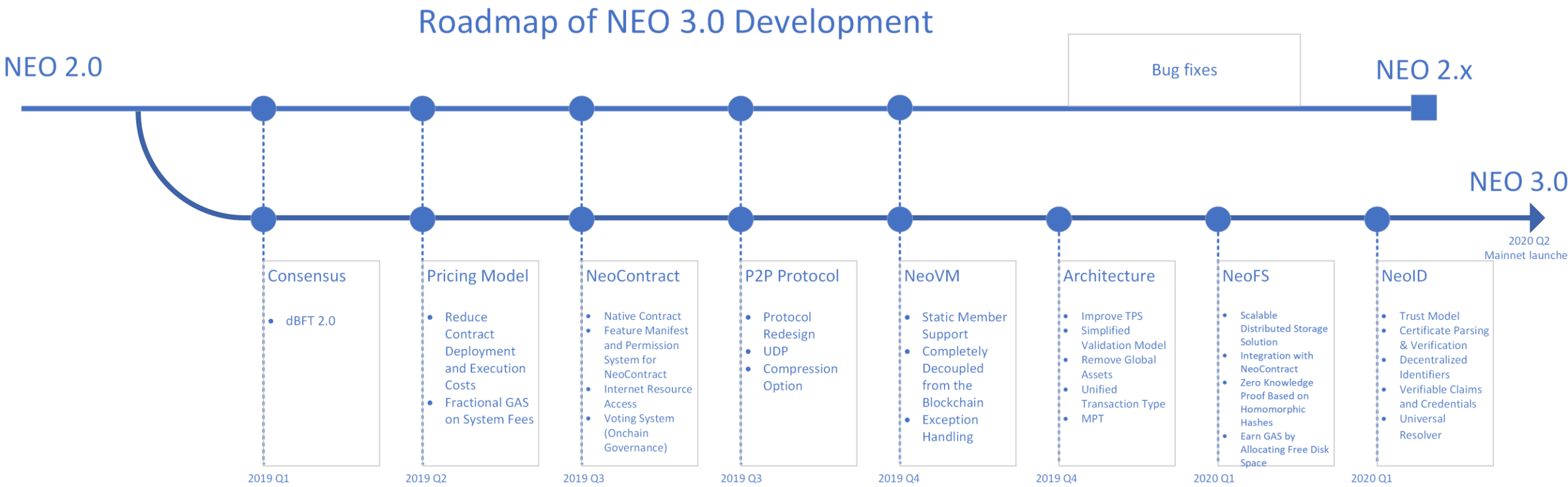
- Functions
  - `function totalSupply() public constant returns (uint);`
  - `function balanceOf(address tokenOwner) public constant returns (uint balance);`
  - `function allowance(address tokenOwner, address spender) public constant returns (uint remaining);`
  - `function transfer(address to, uint tokens) public returns (bool success);`
  - **`function approve(address spender, uint tokens) public returns (bool success);`**
  - **`function transferFrom(address from, address to, uint tokens) public returns (bool success);`**
- Events
  - `event Transfer(address indexed from, address indexed to, uint tokens);`
  - **`event Approval(address indexed tokenOwner, address indexed spender, uint tokens);`**

NEP-5 / NEP5(.1)

- Functions
  - `BigInteger totalSupply()`
  - `string name()`
  - `string symbol()`
  - `byte decimals()`
  - `BigInteger balanceOf(byte[] account)`
  - `bool transfer(byte[] from, byte[] to, BigInteger amount)`
- Events
  - `event transfer(byte[] from, byte[] to, BigInteger amount)`

NEO 3.0 Outlook

- UDP-based protocol?
- Reduce contract deployment cost (a contract deployment ~1000USD)
- Built-in Oracle implementation
- NEO3 changes from [UTXO-based to account-based]



Questions?

Dr. Thomas Bocek thomas.bocek@hsr.ch