UiO : University of Oslo

Ethereum: A blockchain-based smart contract platform

Tien Dat Le

Questions

- Bitcoin vs Ethereum ?
- Why Ethereum and Decentralized application (Dapps) ?
- How Ethereum work ?
- What is new challenges in Dapps context ?

Layout

- Motivation
- How Ethereum work?
  - Smart contract
  - Transactions
  - Block state
  - Datastructure
  - Mining
- Ethereum application
- Research challenge
- Discussion

Existing blockchain protocols were designed with script language

bitcoin

OR
THIS

MONERO

Why not make a protocols like this

OR
THIS

OR
THIS

Ethereum

- Blockchain with expressive programming language
  - Programming language makes it ideal for **smart contracts**
- Why?
  - Most public blockchains are cryptocurrencies
    - Can only transfer coins between users
  - Smart contracts enable much more applications

A **smart contract** is a computer program executed in a **secure environment** that directly controls **digital assets**

What are digital assets?

- A broad category
  - Domain name
  - Website
  - Money
  - Anything tokenisable (e.g. gold, silver, stock share etc)
  - Game items
  - Network bandwidth, computation cycles

How Ethereum Works

- Two types of account:
  - **Normal account** like in Bitcoin
    - has balance and address
  - **Smart Contract account**
    - like an object: containing (i) code, and (ii) private storage (key-value storage)
    - Code can
      - Send ETH to other accounts
      - Read/write storage
      - Call (ie. start execution in) other contracts

DNS: The “Hello World” of Ethereum

```
data domains[](owner, ip)
```

Private Storage

```
def register(addr):
    if not self.domains[addr].owner:
        self.domains[addr].owner = msg.sender
```

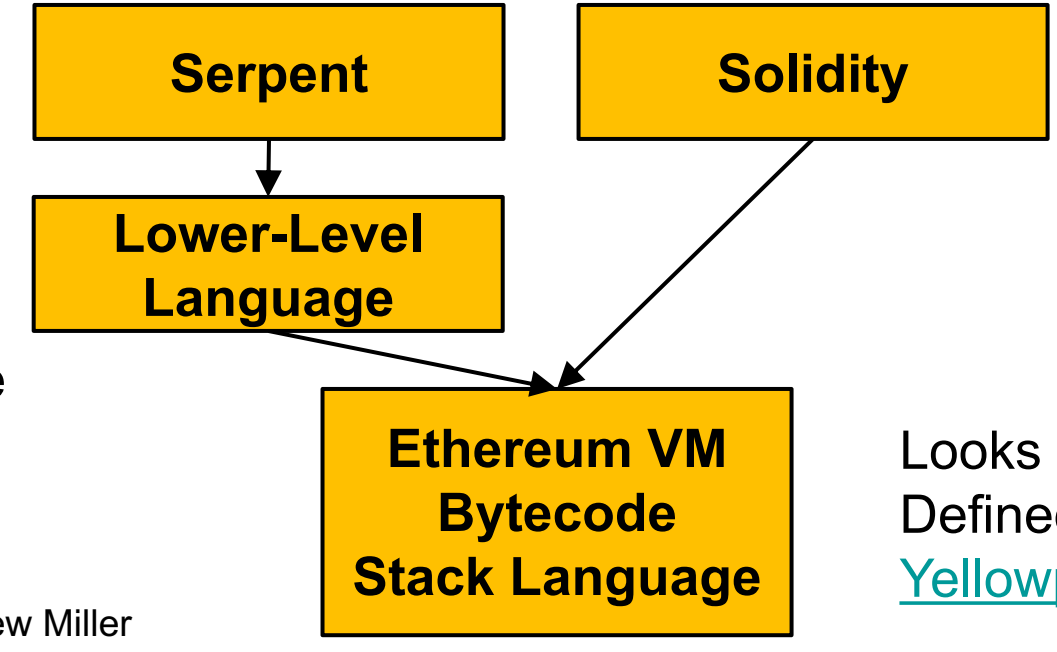
Can be invoked by other accounts

```
def set_ip(addr, ip):
    if self.domains[addr].owner == msg.sender:
        self.domains[addr].ip = ip
```

Ethereum Languages

Looks like python

Types, invariants, looks like Javascript



Functional, macros, looks like scheme

Looks like Forth. Defined in Yellowpaper

Slide is courtesy of Andrew Miller

Example

```
1  contract Greetings {
2      string greeting;
3      function Greetings (string _greeting) public {
4          greeting = _greeting;
5      }
6
7      /* main function */
8      function greet() constant returns (string) {
9          return greeting;
10     }
11 }
```

What you write

What other see on the blockchain

```
6060604052604051610250380380610250833981016040528.....
```

```
PUSH 60
    PUSH 40
    MSTORE
    PUSH 0
    CALLDATALOAD
```

What people get from the disassembler

Transactions in Ethereum

- Normal transactions like Bitcoin transactions
  - Send tokens between accounts
- Transactions to contracts
  - like function calls to objects
  - specify which object you are talking to, which function, and what data (if possible)
- Transactions to create contracts

Transactions

- **`nonce`** (anti-replay-attack)
- **`to`** (destination address)
- **`value`** (amount of ETH to send)
- **`data`** (readable by contract code)
- **`gasprice`** (amount of ether per unit gas)
- **`startgas`** (maximum gas consumable)
- **`v, r, s`** (ECDSA signature values)

How to Create a Contract?

- Submit a transaction to the blockchain
  - `nonce:` previous `nonce + 1`
  - `to:` empty
  - `value:` value sent to the new contract
  - `data:` contains the code of the contract
  - `gasprice` (amount of ether per unit gas)
  - `startgas` (maximum gas consumable)
  - `v, r, s` (ECDSA signature values)
- If tx is successful
  - Returns the address of the new contract

How to Interact With a Contract?

- Submit a transaction to the blockchain
  - `nonce:` previous `nonce + 1`
  - `to:` contract address
  - `value:` value sent to the new contract
  - `data:` data supposed to be read by the contract
  - `gasprice` (amount of ether per unit gas)
  - `startgas` (maximum gas consumable)
  - `v, r, s` (ECDSA signature values)
- If tx is successful
  - Returns outputs from the contract (if applicable)

Blockchain State

Bitcoin's state consists of key value mapping addresses to account balance

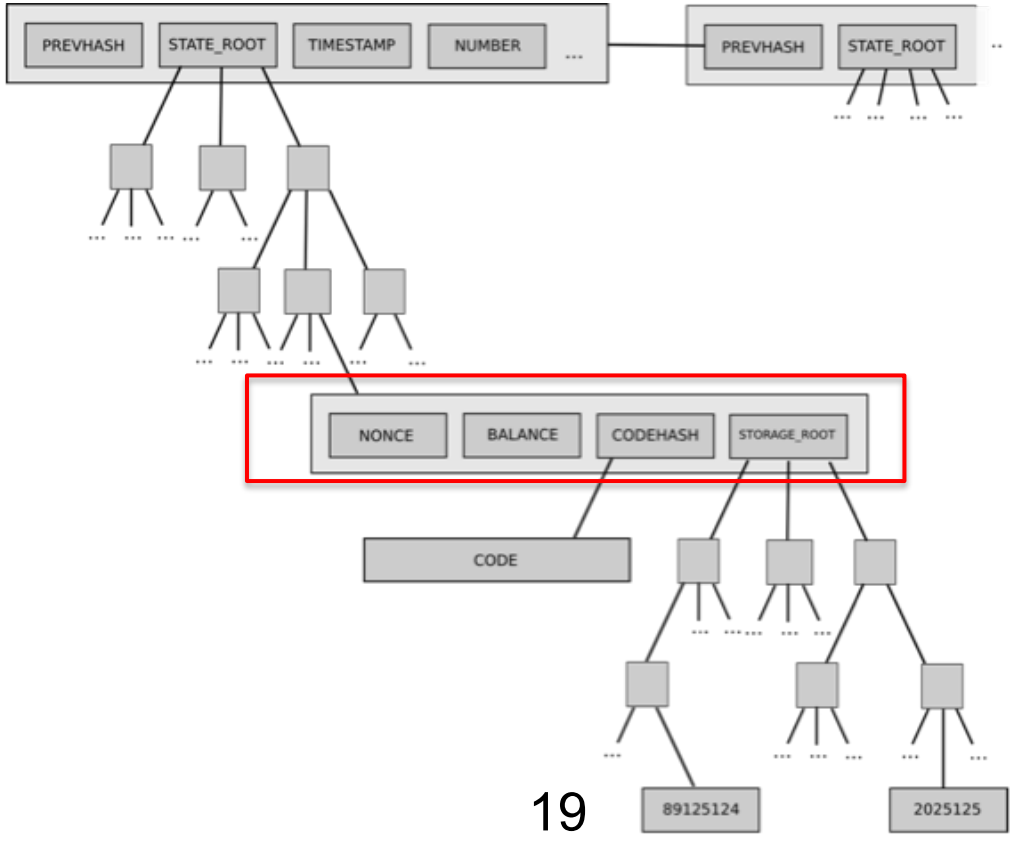
Address	Balance (BTC)
0x123456 …	10
0x1a2b3f …	1
0xab123d …	1.1

Ethereum's state consists of key value mapping addresses to account objects

Address	Object
0x123456 …	X
0x1a2b3f …	Y
0xab123d …	Z

Account Object

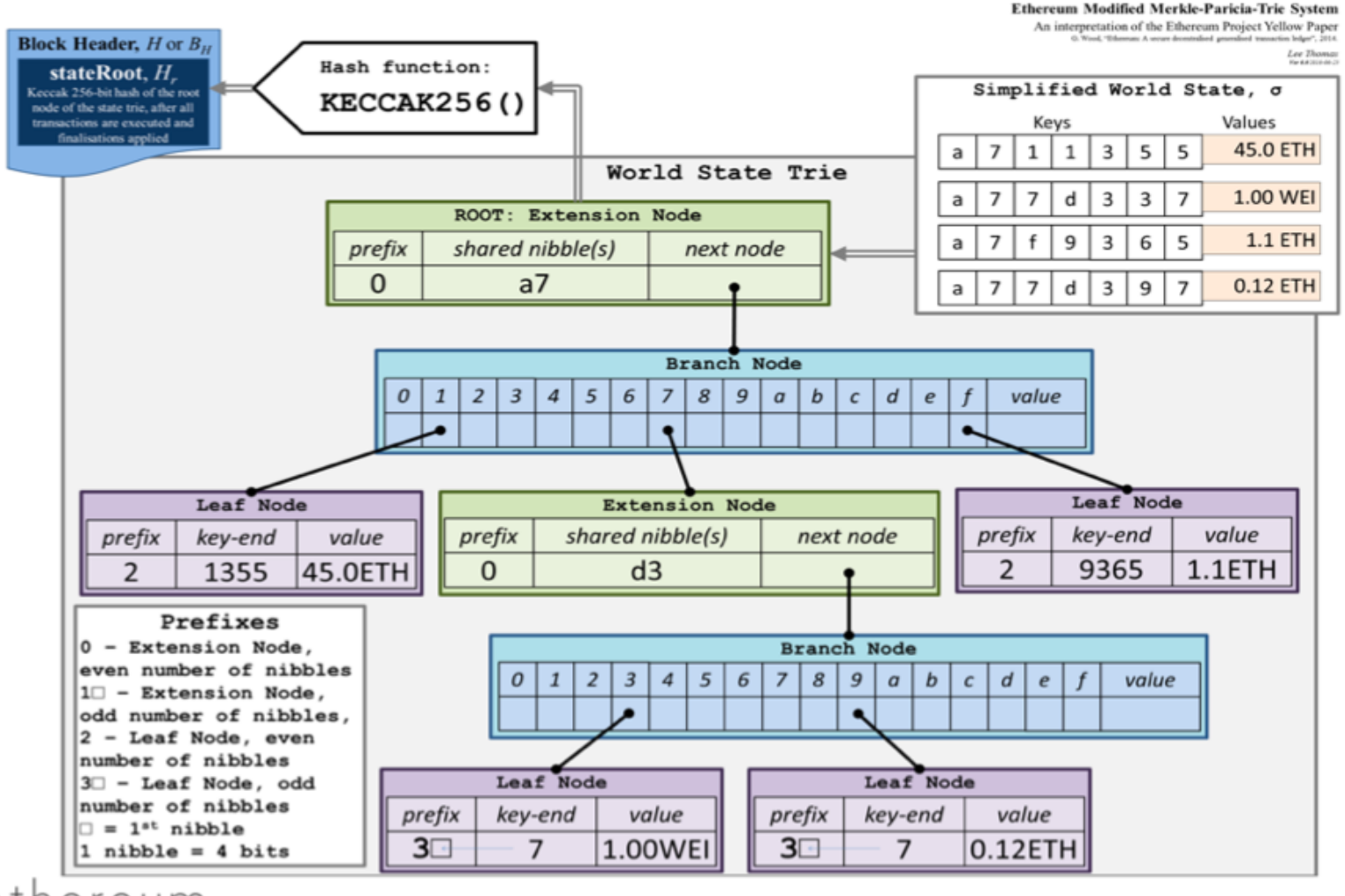
- Every account object contains 4 pieces of data:
  - Nonce
  - Balance
  - Code hash (code = empty string for normal accounts)
  - Storage trie root



19

Merkle Patricia Trie

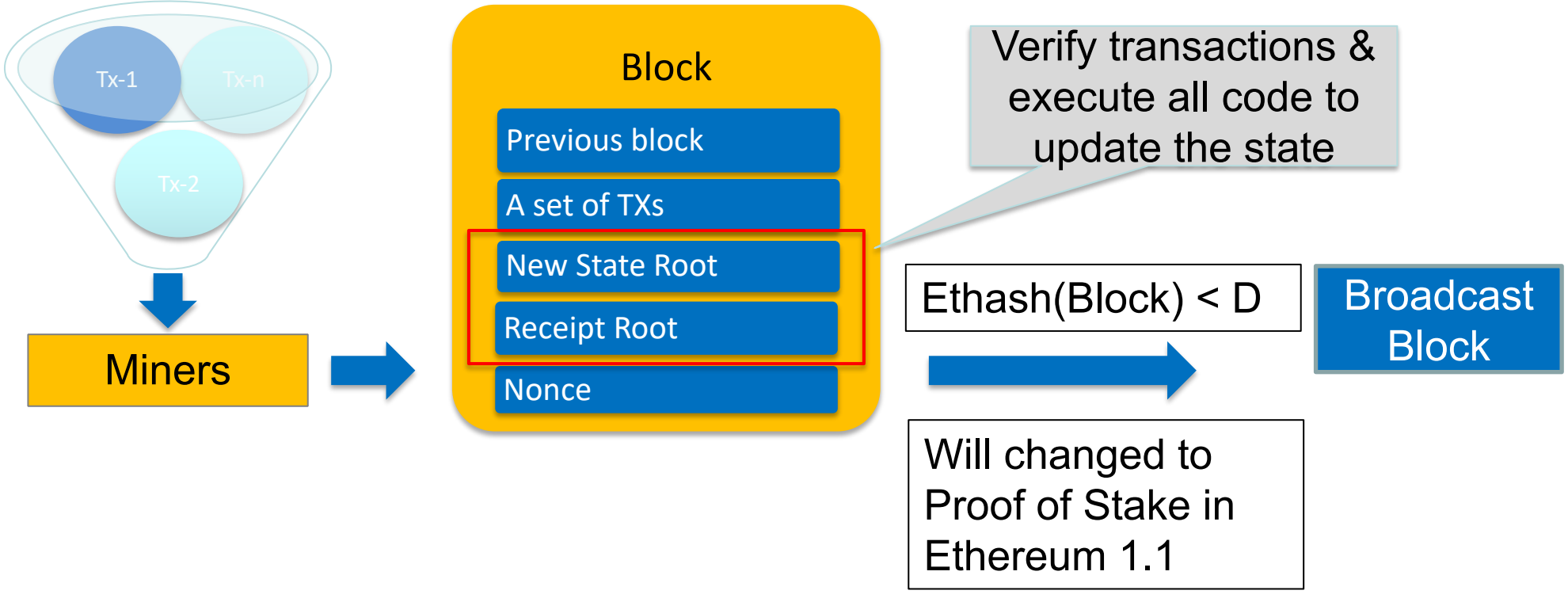
Appendix - Merkle Patricia Tree



ethereum

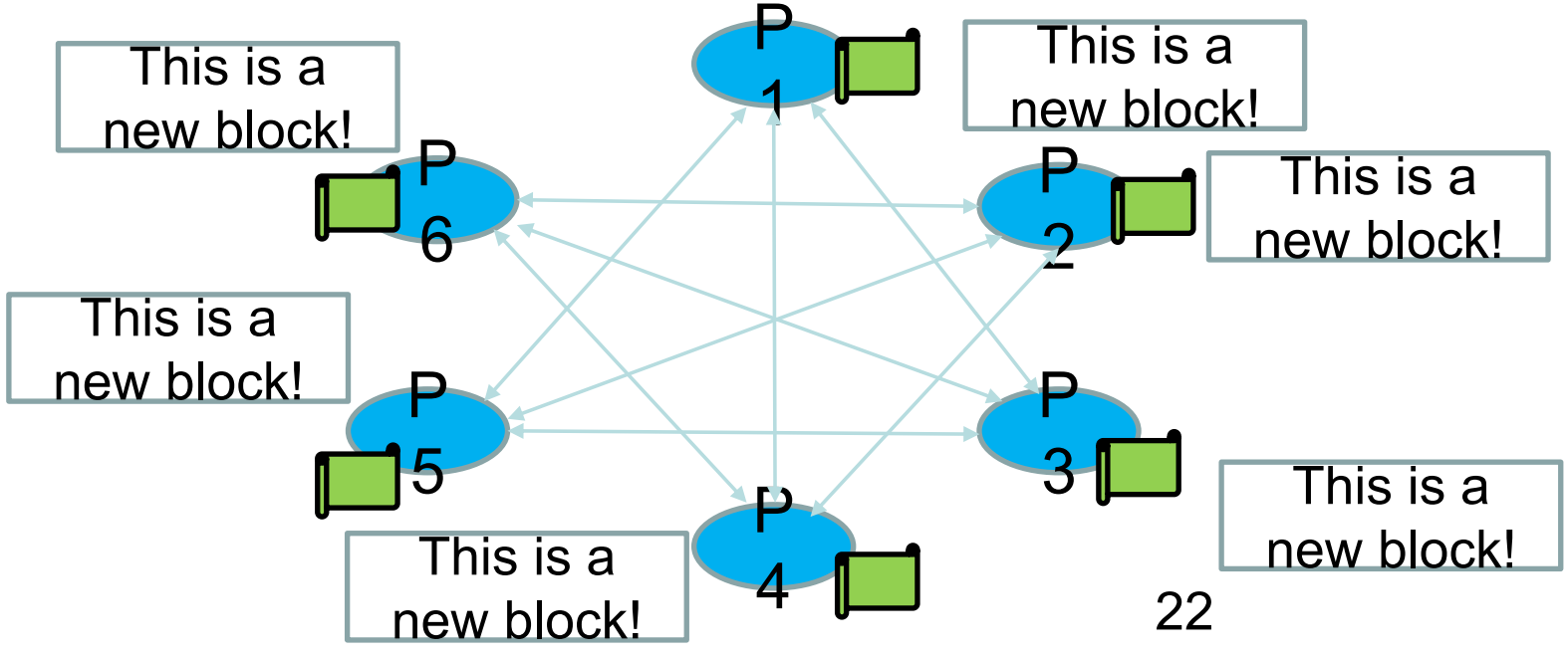
https://ethereum.stackexchange.com/questions/6415/eli5-how-does-a-merkle-patricia-trie-tree-work

Block Mining

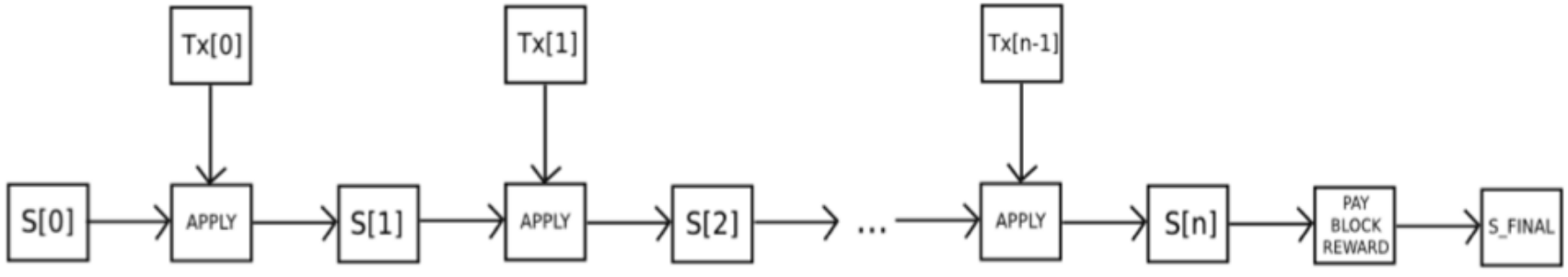


Code execution

- Every (full) node on the blockchain processes every transaction and stores the entire state



Code execution



Dos Attack Vector

- Halting problem
  - Cannot tell whether or not a program will run infinitely
  - A malicious miner can DoS attack full nodes by including lots of computation in their txs
    - Full nodes attacked when verifying the block

```
uint i = 1;
while (i++ > 0) {
    donothing();
}
```

Solution: Gas

- Charge fee per computational step (“gas”)
  - Special gas fees for operations that take up storage

Operation	Gas	GasCost
PUSH1	111741	3
PUSH1	111738	3
MSTORE	111726	12
CALLDATASIZE	111724	2
ISZERO	111721	3
PUSH2	111718	3
JUMPI	111708	10

Sender has to pay for the gas

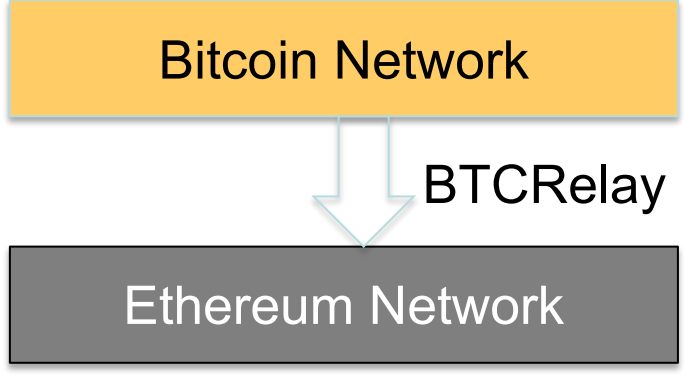
- **`gasprice`**: amount of ether per unit gas
- **`startgas`**: maximum gas consumable
  - If **startgas** is less than needed
    - Out of gas exception, revert the state as if the TX has never happened
    - Sender still pays all the gas
- TX fee = gasprice * consumedgas
- Gas limit: similar to block size limit in Bitcoin
  - Total gas spent by all transactions in a block < Gas Limit

Application build in Ethereum ecosystem

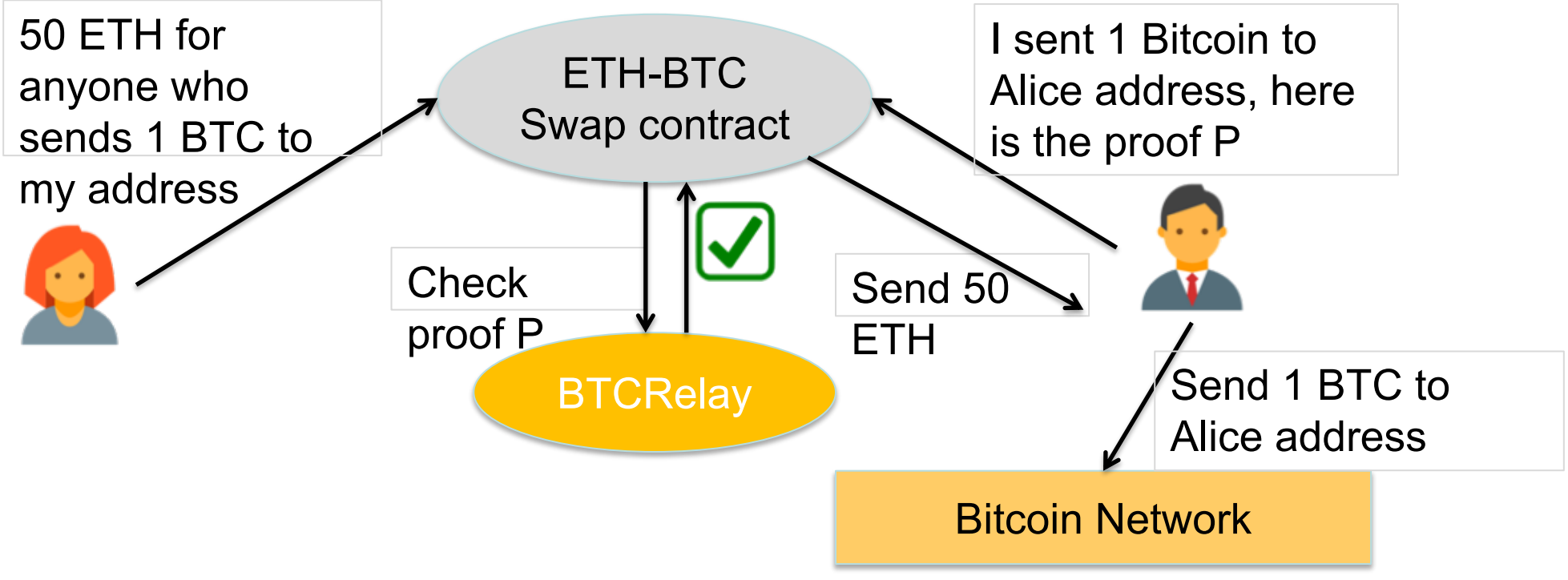
- ERC20 Token
- 0x
  - A protocol for building decentralized exchange on ETH
- TownCrier and Oraclize
  - allow contracts to fetch external data from real websites
  - Enable a lots of applications: betting, insurance, bounty based on real world event
- Augur and Gnosis
  - Prediction market: predict the outcome of real world event to get reward

BTCRelay

- A bridge between the Bitcoin blockchain & the Ethereum blockchain
  - Allow to verify Bitcoin transactions within Ethereum network
  - Allow Ethereum contracts to read information from Bitcoin blockchain



BTCRelay Application: ETH-BTC atomic swaps



Can we build any blockchain practical relays on ETH ?

- **Dogecoin, litecoin** relay on Ethereum ?

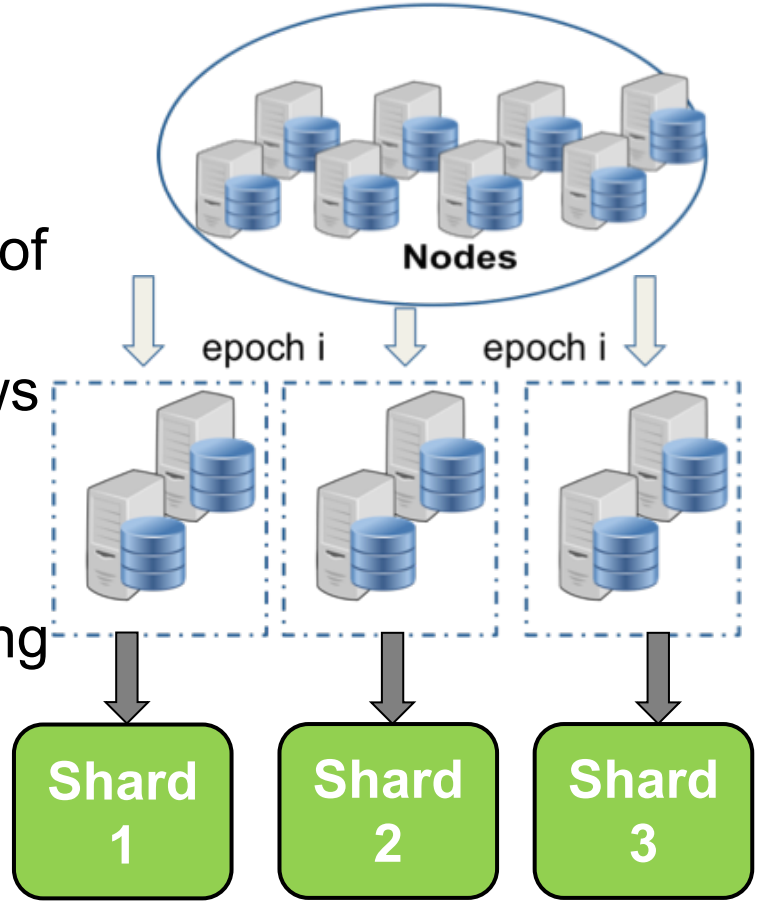
Research challenges

Scalability

- Resources on blockchain are expensive
  - Full nodes perform the same on-chain computations
  - Full nodes store the same data
- Gas-limit is relatively small
  - Can’t run an OS on blockchain
  - Can’t increase gas-limit: DoS vector

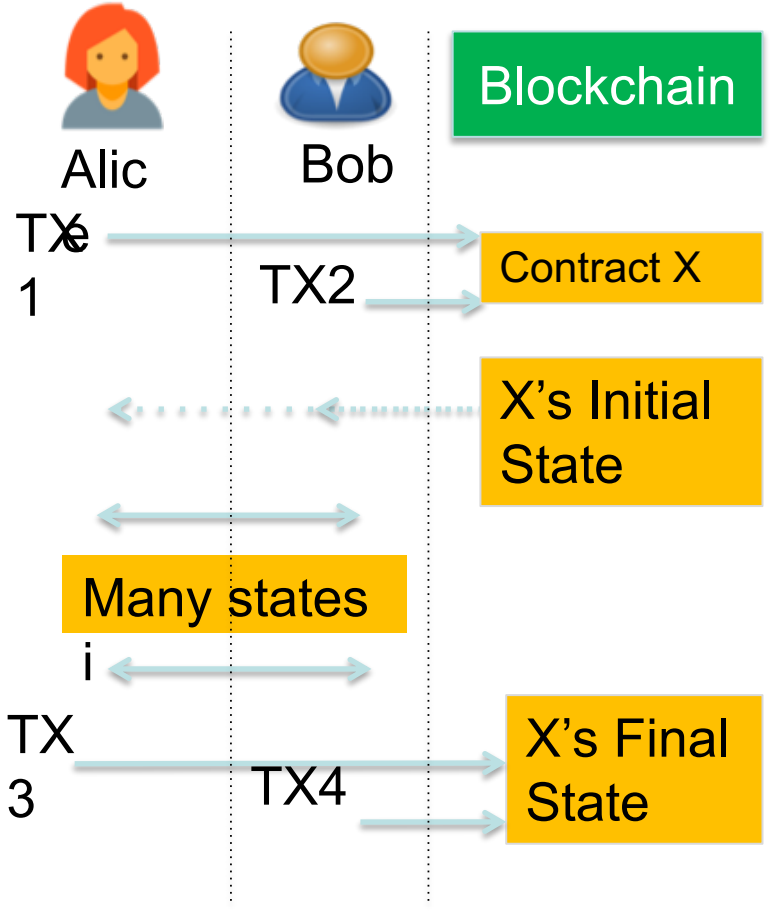
Scalability Solution 1: Sharding

- Divide the network into sub-networks
  - each stores and manages a fraction of the blockchain (a shard)
  - Allow scaling up as the network grows
- There is a catch
  - May affect usability or performance
  - May not be compatible with all existing applications



Scalability Solution 2: State Channel

- Similar to payment channel (e.g. lightning network) but for states
  - Scaling by using off-chain transactions
  - Can update the state multiple times
  - Only settlement transactions are on-chain
- Challenges
  - Cannot create state channel for all applications
  - Still early research, more work needed



Security Flaws

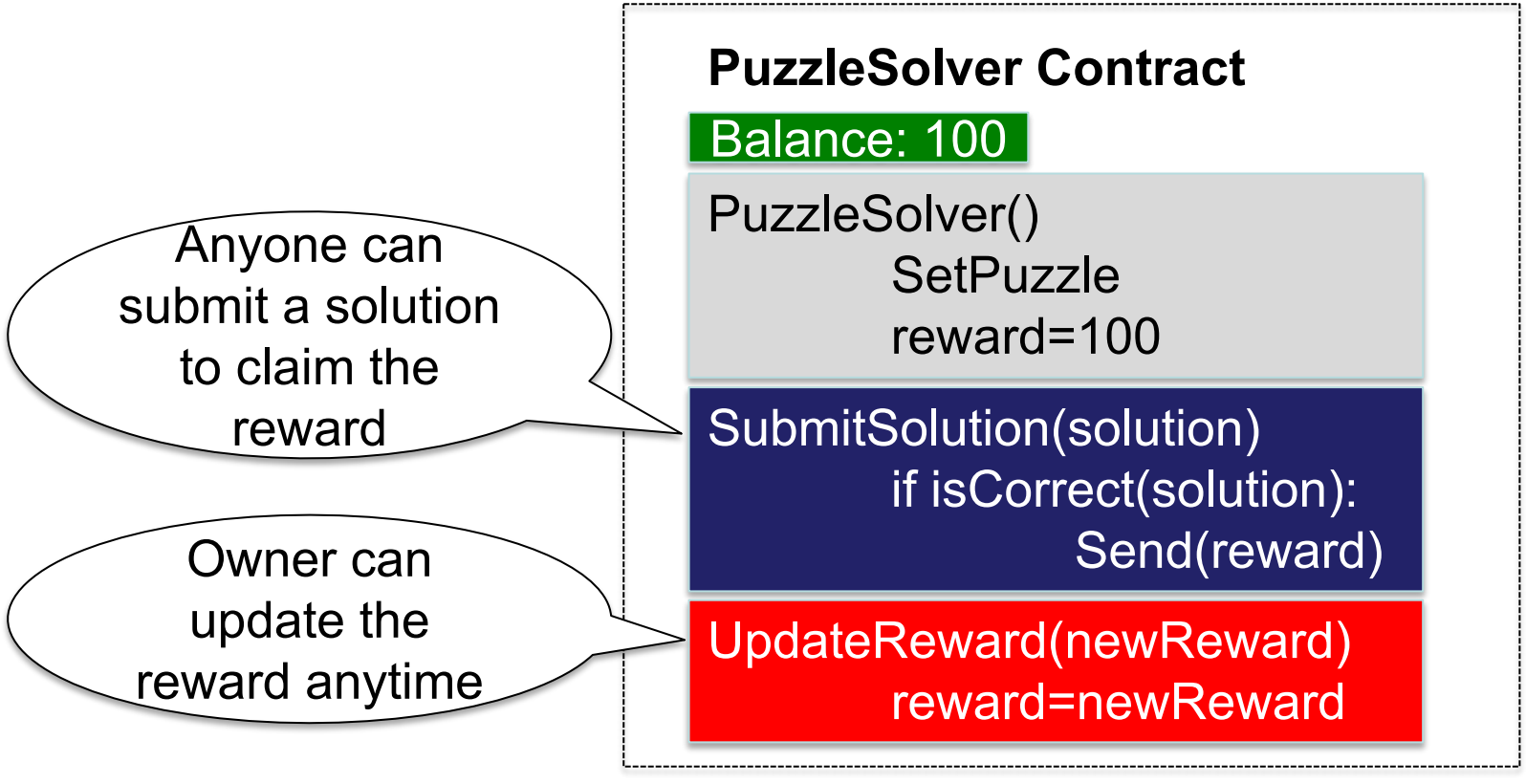
- Due to abstraction of semantic
  - Transaction ordering dependence
  - Reentrancy bug
    - Which exploited the DAO
- Obscure VM rules
  - Maximum stack depth is 1024: not many devs know
  - Inconsistent Exception Handling in EVM


The DAO Attacked: Code Issue Leads to $60 Million Ether Theft

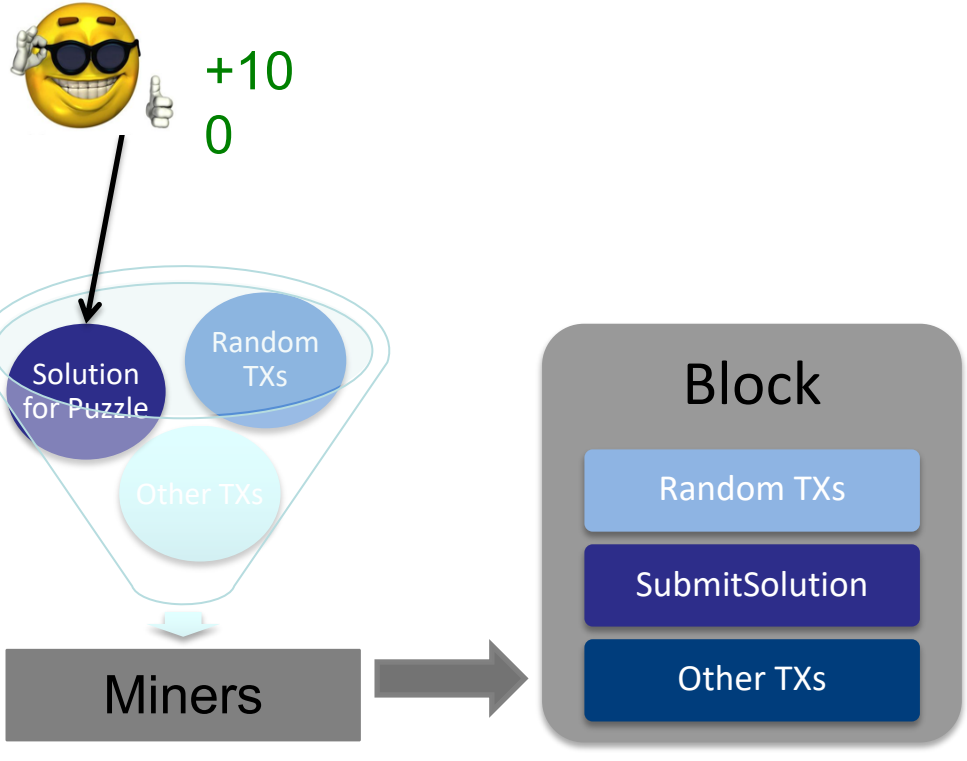
Michael del Castillo (@DelRayMan) | Published on June 17, 2016 at 14:00 GMT

NEWS

Example 1: Transaction Ordering Dependence



Scenario 1: SubmitSolution is trigerred



PuzzleSolver Contract

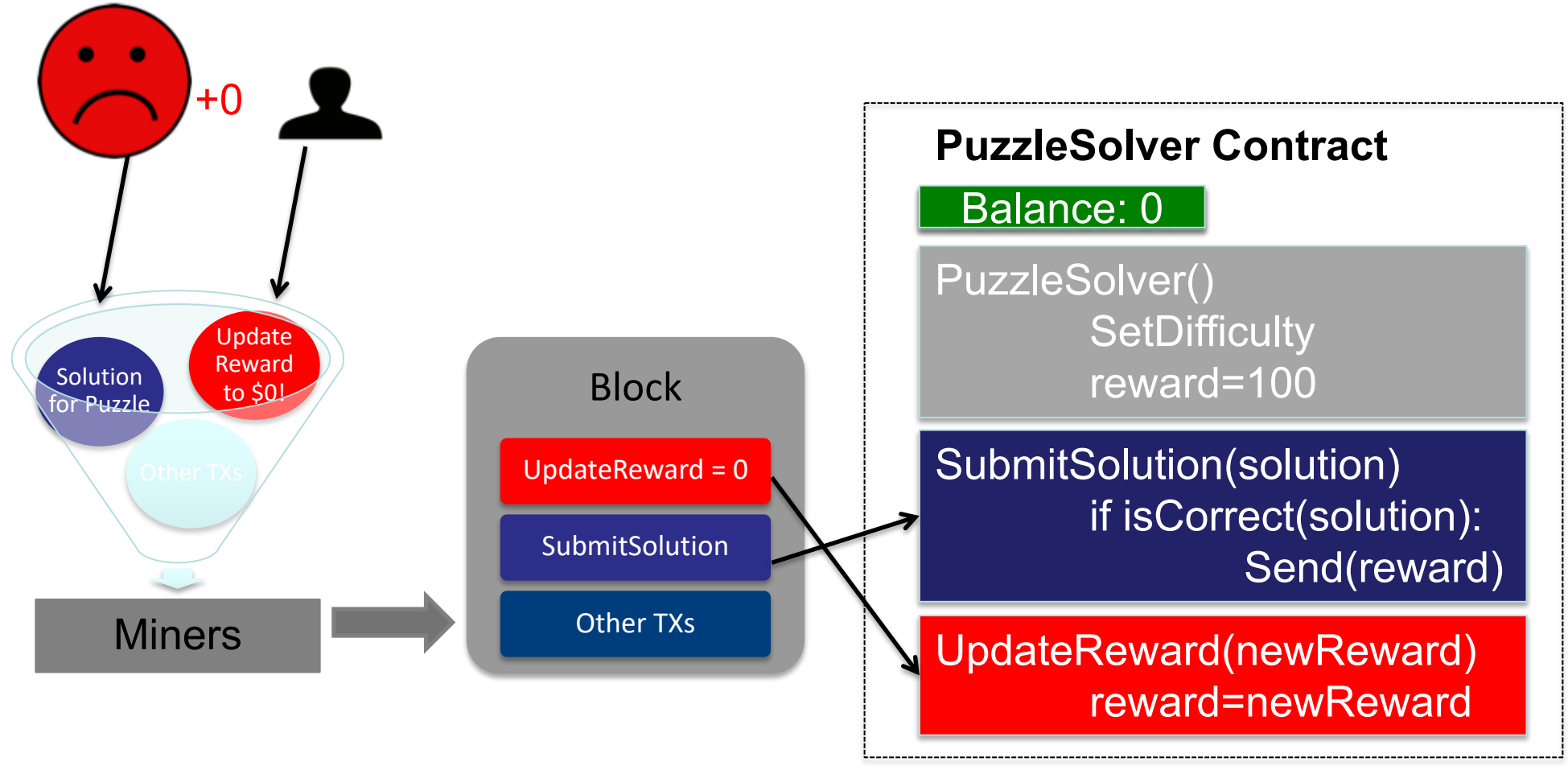
Balance: 0

```
PuzzleSolver()
        SetDifficulty
        reward=100
```

```
SubmitSolution(solution)
        if isCorrect(solution):
                Send(reward)
```

```
UpdateReward(newReward)
        reward=newReward
```

Scenario 2: Both SubmitSolution and UpdateReward are triggered

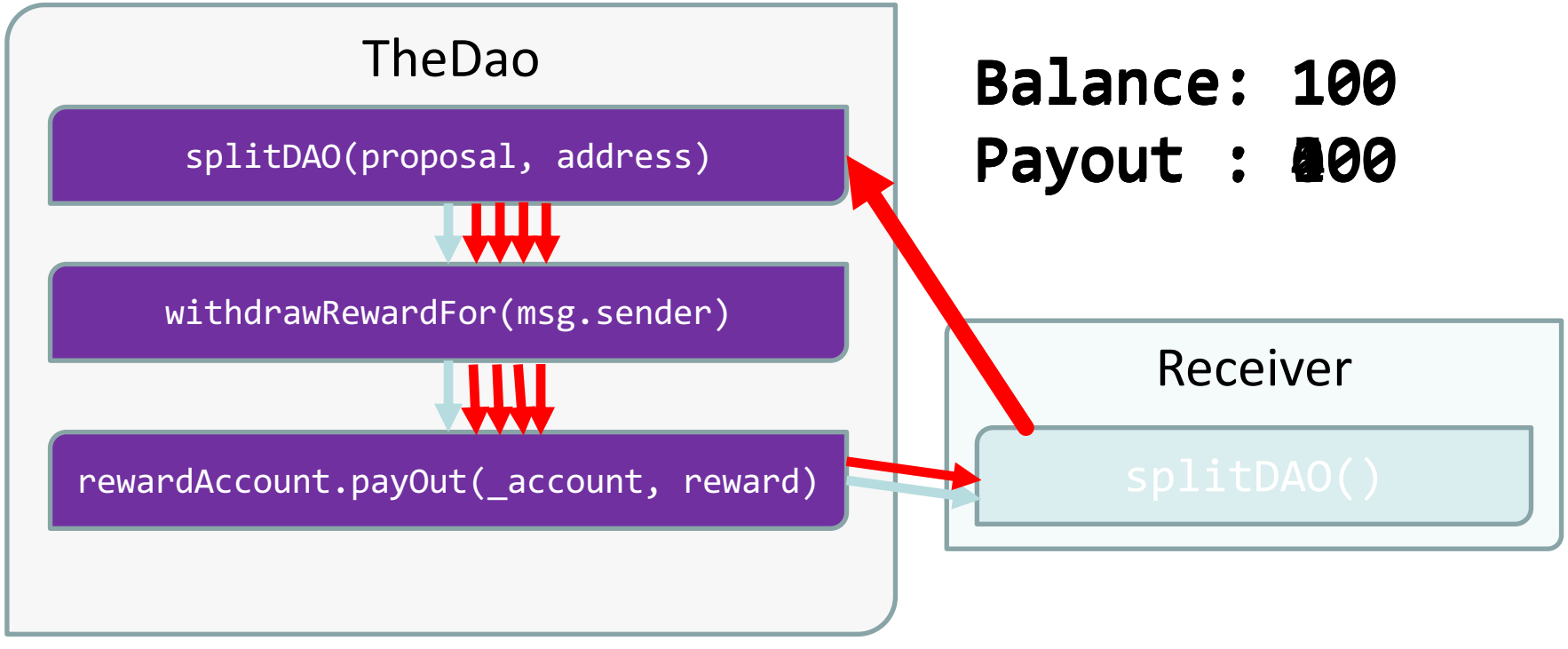


Example 2: Reentrancy Bug --- TheDAO Bug

- Reentrancy vulnerability
  - Lead to ETH hardfork
- Call before balance update

```
...
// Burn DAO Tokens
if (balances[msg.sender] == 0)
        throw;
withdrawRewardFor(msg.sender);
totalSupply -= balances[msg.sender];
balances[msg.sender] = 0;
paidOut[msg.sender] = 0;
return true;
```

TheDAO Bug: Attack Scenario



Solutions to Resolve Security Flaws

- Create developer tools
  - Smart contract analyser based on symbolic exec: Oyente
  - Testing and deployment framework: truffle
  - Formal verification for smart contracts: eth-isabelle, why3
- Design better semantic [CCS'16]
- Educate users

Discussion

- Is gas system really prevent DDoS attack ? Is there any case that DDoS attack is free of cost
- Why Scalability is much more severe problem in Ethereum?
- What often happened when an ICO on Ethereum have a limited quota for participants to compete?
- Why Ethereum have to introduce uncle blocks to blockchain?

GHOST - The "Greedy Heaviest Observed Subtree"

- ETH reduces block confirmation time to 10s.
- Suffer from reduced security due to a high stale rate as block propagation take time.
- GHOST solves the first issue of network security loss by including stale blocks in the calculation of which chain is the "longest"

GHOST - The "Greedy Heaviest Observed Subtree"

- A block must specify a parent, and it must specify 0 or more uncles
- An uncle included in block `B` must have the following properties:
  - It must be a direct child of the `k`-th generation ancestor of `B`, where `2 <= k <= 7`.
  - It cannot be an ancestor of `B`
  - An uncle must be a valid block header, but does not need to be a previously verified or even valid block
  - An uncle must be different from all uncles included in previous blocks and all other uncles included in the same block (non-double-inclusion)
- For every uncle `U` in block `B`, the miner of `B` gets an additional 3.125% added to its coinbase reward and the miner of U gets 93.75% of a standard coinbase reward.

References

- Bitcoin and Cryptocurrency Technologies
  - Chapter 10.7
- https://github.com/ethereum/wiki/wiki/White-Paper
- https://en.wikipedia.org/wiki/Ethereum
- https://www.coindesk.com/research/understanding-ethereum-report/
- Luu, Loi, Jason Teutsch, Raghav Kulkarni, and Prateek Saxena.Demystifying incentives in theconsensus compute
- Luu, L., Chu, D.H., Olickel, H., Saxena, P., Hobor, A.: Making smart contractssmarter. In: ACM CCS (2016)