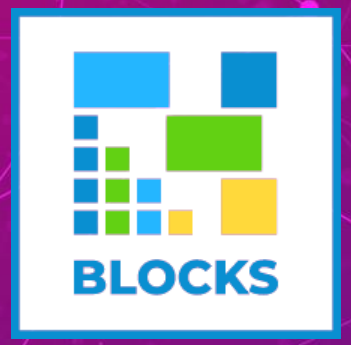


blockchain.taltech.ee

BASICS OF SMART CONTRACTS AND CREATION OF NEW TOKENS

27.03.2019

AGENDA

- INTRODUCTION TO BLOCKCHAIN
  - Fundamentals of Blockchain
  - Transactions in Blockcain
  - Data structure in blockchain
  - Blockchain Peer 2 peer network
  - Decentralized consensus
- INTRODUCTION TO SMART CONTRACTS
  - History
  - Transactions from smart contracts
  - State machine for executing smart-contracts
  - Transaction fees
  - Sample smart contracts applications

Fundamentals of Blockchain

- • First, Bitcoin at its most fundamental level is a breakthrough in
  - computer science – one that builds on 20 years of research into
  - cryptographic currency, and 40 years of research in cryptography, by thousands of researchers around the world.
- • Financial crisis: 2014
- • Bitcoin whitepaper was released
- • Email was the first killer app for Internet, similarly Bitcoin is one of the popular apps (or in fact the killer app) for blockchain technology

Fundamentals of Blockchain: Ledger

- What is a blockchain?
  - Blockchain is essentially a Distributed Transaction Ledger
- What is a ledger ?
  - Essentially a list of transactions
  - E.g Listing money going in/out
  - But can also be used for other things such as maintaining a record of all the changes made to a house, all the owners and etc. or an “auto scheckheft”

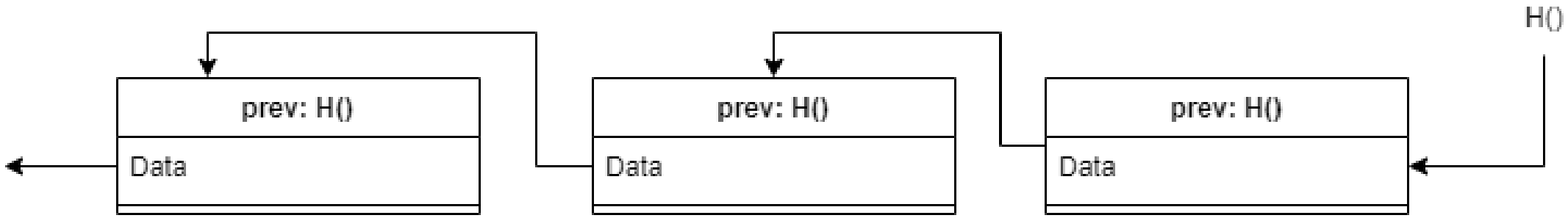
Fundamentals of Blockchain: transactions

- Transactions
  - Physical/face-to-face exchange between two or more people
- How about performing transactions online
  - Double spend problem
    * Alice gives the same money to two people and order items from them
  - Solving Double spending problem
    - All transactions are recorded. But no one can alter it. This ledger is publicly visible for anyone to verify Or privately to all those concerned. Multiple copies exists.
    - Prevent double spend
    - No one person has decision making authority

Traditional method of solving double spending problem

Bank and other intermediaries (Amazon, Ebay) provide this service
- However, at a significant cost
- But the main problem is that we have to trust someone
- The person can tamper with it
- Difficult to do in a global scale

Blockchain Data structure: Pointer Based



• Blockchain uses Cryptographic Hash pointers (a data structure):
• While a regular pointer points to where something is
• Hash Pointer points to where some information is stored • and cryptographic hash of that information

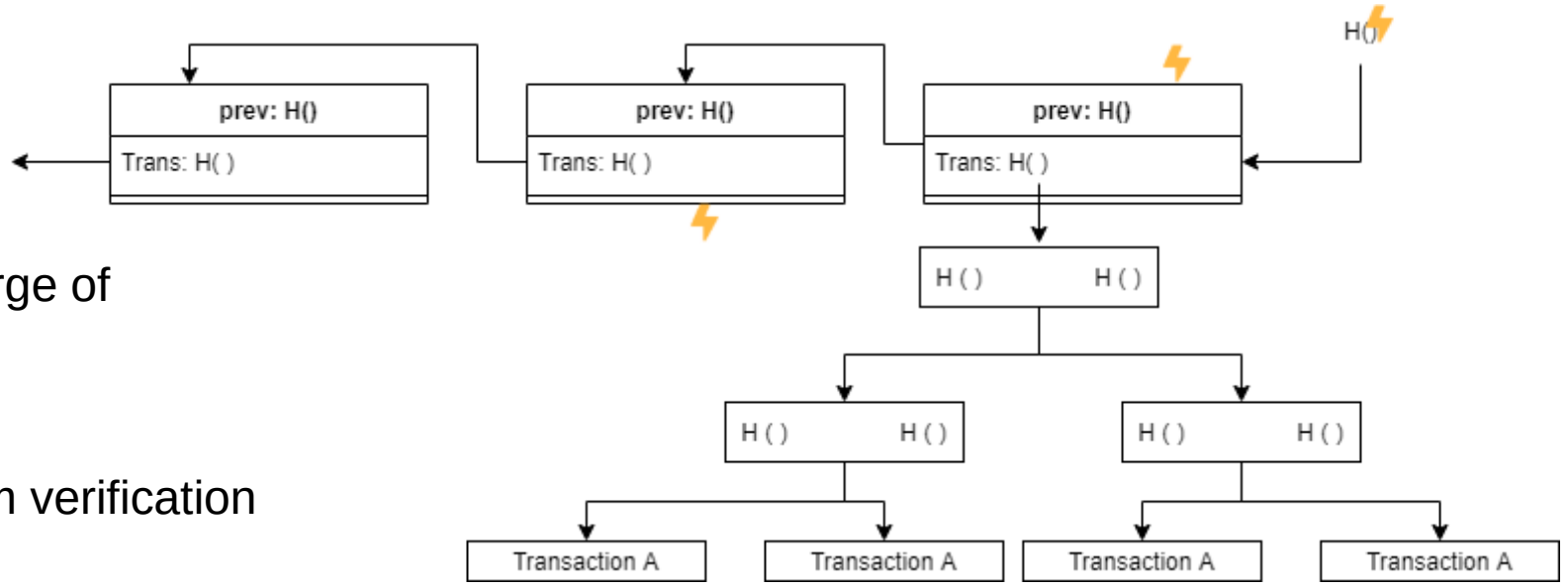
Note: “prev: H()” is a cryptographic hash pointer to the hash of the full block (Data, prev: H() and etc.) and not just the data.

Cryptographic hash pointers make sure that if any member of the list is modified in any way, to any extent, the chain is verifiably invalid by anyone who checks.

Distributed ledger: Bitcoin P2P network

- All nodes are equal, random topology
- New nodes can join at any time
  - Append-only ledger
  - Anyone can view/validate transactions
  - Decentralized/Distributed Consensus

Distributed/Decentralized Consensus

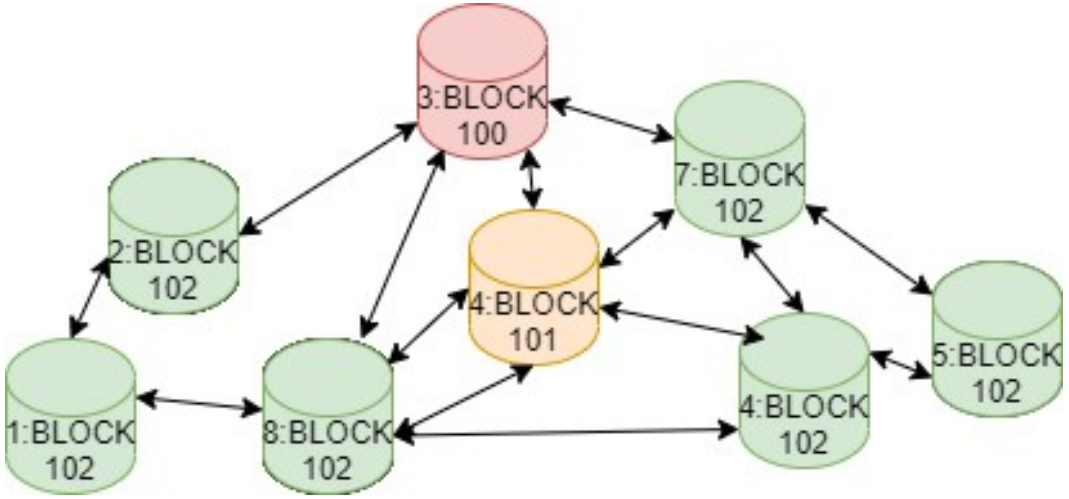


Imagine one central node is in charge of verifying the contents

- It can still edit the chains
- Why should we trust this node?
- Might be the bottleneck to perform verification

Distributed/Decentralized Consensus

- Append-only ledger
- Anyone can validate
- Transactions Decentralized/
- Distributed Consensus
  - Longest Chain prevails: the chain with the consistent data is the chosen chain
  - “Blockchain distributed consensus model is the most important invention since the Internet itself”, Marc Andreessen



Smart Contracts: history

- Buterin was introduced and intrigued by blockchain technology when he got involved in Bitcoin as a 17-year-old programmer in 2011 and co-founded Bitcoin Magazine[1]
- Ethereum's co-founder, Vitalik Buterin said, "I thought [those in the Bitcoin community] weren't approaching the problem in the right way. I thought they were going after individual applications; they were trying to kind of explicitly support each [use case] in a sort of Swiss Army knife protocol."
- In 2013, Vitalik Buterin started working on the first version of the ethereum white paper

[1] https://bitcoinmagazine.com/articles/vitalik-buterin-on-his-long-term-goals-for-ethereum-1462381147

Ethereum blockchain

- Ethereum
- Aim to create an alternate protocol for building decentralized applications
- Emphasis on development time, security and scaling
  - Own language: solidity
  - Turing complete Ethereum Virtual Machine (EVM)

Ethereum Virtual Machine

- The EVM specifies an execution model for state changes of the blockchain
- Formally, the EVM can be specified by the following tuple: (block_state, transaction, message, code, memory, stack, pc, gas)
- The block_state represents the global state of the whole blockchain including all accounts, contracts and storage

Ethereum Transactions

- What is a transaction
  - It is an agreement between a buyer and a seller
  - Or a provider and a consumer
  - That there would be exchange of assets/services/products/currency
  - In lieu of currency/crypto-currency or some other asset and In the present or in the future
- What does a transaction have?
  - From: The address originating the transaction and The one paying for executing the transaction
  - To: Receiver of payment, In case of contracts, this can be left blank)
  - Value: Ether
  - Input: refers to the compiled contract bytecode, is used during contract deployment in EVM. It is also used for storing data related to smart contract function calls along with its parameters

Types of Transactions in Ethereum

- Types of Transactions in Ethereum

  Type 1: Transfer of ether

- Type 2: Deployment of a smart contract

  An externally owned account can deploy a contract using a transaction in Ethereum virtual machine.

- Type 3: Using or invoking a function within a smart contract

  Executing a function in a contract that changes state are considered as transactions in Ethereum.

  If executing a function does not change state, it does not require a transaction

What is a “smart” contract?

- A contract is a legal document that binds two or more parties • who agree to execute a transaction immediately or in future
- Smart contracts are digitization of the legal contracts.
- In Ethereum Smart contracts are deployed, stored and executed within the Ethereum Virtual machine (EVM)

Source: https://medium.com/coinmonks/https-medium-com-ritesh-modi-solidity-chapter1-63dfaff08a11

What is a “smart” contract?

- What is a “smart” contract?

  Smart contracts can also store data

  The data stored can be used to record

  information, fact, associations, balances

  or any other information needed to implement logic for real world contracts
- Smart contracts are similar to Object oriented classes
- A smart contract can call another smart contract just like an Object-oriented
  - object to create and use objects of another class.
  - Think of smart contract as a small program consisting of functions.
  - You can create an instance of the contract and invoke functions to view and
  - update contract data along with execution of some logic

Writing Smart contracts: solidity

- Writing contracts
  - Contracts are written in solidity language
  - Create the UI for contracts in HTML/JS
  - Remix (https://remix.ethereum.org/) is a good starting point
- Syntax similar to javascript
  - Minimize entry barrier
  - Leverage existing skills
  - Contracts are the main focus
  - Each contract is a class

Smart contracts

- Smart contract
  - It is a set of functions that can be called by other users of contracts
  - They can be used to execute functions, send ether or store data
  - Each smart contract is an account holding object, i.e., has its own address
- Security
  - Once deployed, a contract is publicly accessible by anyone on the network with the following information
  - Address of the smart contract
- OPCODE
  - Number of public functions and their hash signature
  - Moreover, the whole transaction history is accessible (function calls + actual arguments)
  - Smart contracts, once deployed, cannot be changed or patched anymore

Examples of smart contracts

- Some possible contracts
  - Altcoins (alternate cryptocurrencies launched after Bitcoin), Tokens, Assets
    - Tokens are currently the largest use-case for smart contracts
  - Mostly used to collect money via Initial Coin Offerings (ICOs)
  - Crowdfunding, fan incentive schemes
  - Identity and reputation systems
    - Smart contracts can be used as a decentralized identity management system like e.g. uPort
  - Voting systems, prediction markets, lotteries
    - Blockchain provides a tamper-proof data structure for storing votes
    - A smart contract can ensure that a specific wallet can only vote once
- Access control
  - Sites, games, doors, cars
- Decentralized Autonomous Organizations (DAOs)
  - Organizations on the blockchains
  - They are basically a generalization of multi signature wallets
  - The members vote to trigger certain methods in the smart contract, like transfer on money

Decentralized Applications: introduction

- Decentralized apps have existed even before blockchain
- E.g. BitTorrent is a decentralized P2P application without any blockchain

- In the case of blockchain
  - Applications on top of blockchain (e.g. Ethereum) are known as dApps
  - Implies that most of the application data resides on blockchain
  - Changes to those data must also be recorded, e.g. via smart contracts

Blockchain DApps

- Distributed Applications
  - Open-source software that leverage on the blockchain technology are called DApps.
  - One could say that “Bitcoin” is the first DApp
  - Bitcoin is a self-sustaining public ledger that allows efficient transactions without intermediaries and centralized authorities.
  - Ethereum helps achieve it via smart contracts

Dapps: Pro and Cons

- Pros
  - It depends on the use-case
  - Some benefits:
    - Trust: The source code of any verified smart contract can be checked by anyone
    - Payment: Payment is implemented in a trusted manner via ether
    - Accounts: DApps can be built on top of the Ethereum accounts, no need for a separate user account management system
    - Storage: DApps can leverage blockchain for secure, trusted and redundant storage

Dapps: pro and cons

- Cons
  - It depends on the use-case
- Some benefits
  - Costs: Any change to state costs money (transaction fees) and computation (almost all nodes will have to have a verified record of it)
  - Time: Block time is approx. 14 seconds (https://etherscan.io/chart/blocktime)
  - Availability: Although it is usually a pro, in a decentralized environment due to churn (nodes turning on and off) and load, this could also become an issue

Examples of popular Dapps

- Cryptokitties: Dapp for creation and collection virtual kitties
- DAO: a fully digital (virtual) organization, uses smart contracts to interact with its share holders, employees, customers, suppliers, partners and public authorities