Smart Contracts II & Solidity

Alexander Schönhuth

Bielefeld University
June 15, 2022

Recap Lecture 7

- *Bitcoin Scripts Syntax*
  - Introduction
  - Pay-to-PubKeyHash
  - Opcodes
  - Pay-to-ScriptHash
  - Multisig
- *Bitcoin Scripts Applications*
  - Escrow Transactions
  - Micro Payments
  - Lock Time
- *Ethereum Introduction*
  - Transition Function
  - Turing-Complete Cryptocurrency
  - Blockchain Layers; Ethereum Virtual Machine
- *Smart Contracts*
  - Definition
  - Accounts
  - Account State Transitions

Ethereum
–
Gas

Ethereum
–
Mining

Smart Contracts
Basic Examples

Smart Contracts
Advanced
Examples

Overview

Introduction

- *Ethereum: Gas*
  - Introduction
  - Definition
  - Calculation
- *Ethereum: Mining*
  - Blockchain Blocks
  - Mining
- *Solidity: Basic Examples*
  - Solidity: Introduction
  - Contract "Storage"
  - Contract "Conference"
  - Contract "Coin"
- *Solidity: Advanced Examples*
  - Contract "Ballot"
  - Contract "Conference"
  - Contract "Coin"

Ethereum
–
Gas

Ethereum
–
Mining

Smart Contracts
Basic Examples

Smart Contracts
Advanced
Examples

Ethereum: Gas I

- *Motivation:*
  - Strate transition function Turing-complete
  - Prevent execution of computationally expensive transaction
  - Pay reasonably calculated transaction fees to miner
- *Solution:*
  - Executing owning node pays for computational operations
  - Ensures that programs are used reasonably
  - Transaction fees put into context execution expenses
  - *Gas:* unit that measures costs of computational operations

Ethereum: Gas II

- *Gas – Properties:*
  - Gas costs vary per operation, but are constant over time
    - Additions or comparisons cost 1 gas
    - Computing hash costs 20 gas
    - Writing 256-bit word to storage costs 100 gas
    - Every transaction costs 21 000 gas per se
    - Etc ...
  - Gas price in ETH subject to varying exchange rates
  - *Used in transactions: Wei* = 10^{-18} ETH; *Gigawei* = 10^{-9} ETH
- *Transaction Fees:*
  - Gas costs of transaction converted into ETH (or wei, gigawei)
  - Fees calculated according to (complicated) formula
  - Fees paid to miner

Gas: Definitions I

- *GasPrice:* Gas-to-Wei conversion price for a Tx (= transaction)
  - GasPrice varies per Tx
  - GasPrice depends on other variables specified in transaction
  - Formula for calculation will follow
- *GasLimit:* Maximum total gas allowed for Tx
  - Specified in Tx by account launching Tx
- *MaxTxFee:* Maximal transaction fee for a Tx, computes as

$$MaxTxFee = GasLimit \times GasPrice$$

Gas: Definitions II

- *BaseFee:* Minimum GasPrice required for each Tx in a block
  - BaseFee is re-calculated for each block
  - Earlier blocks at block gas limit (= 30M Gas): BaseFee increases
  - Earlier blocks ”empty”: BaseFee decreases
- *MaxFee:* Maximum allowed GasPrice for Tx
  - Specified in Tx by account launching Tx
- *MaxPriorityFee:* Additional tip to be paid for miner
  - MaxPriorityFee indicates Gas-to-Wei conversion rate
  - Specified in Tx by account launching Tx
- *GasPrice Calculation:*

$$GasPrice = \min\{MaxFee, BaseFee + MaxPriorityFee\} \qquad (1)$$

Gas: Calculation

- *msg.sender* is the account sending the Tx
- *msg.sender* specified *GasLimit, MaxFee, MaxPriorityFee*
- *BaseFee* was determined based on earlier blocks
- *GasPrice* has been computed according to (1)

Algorithm for determining validity of Tx in terms of gas:

1. **If** $GasPrice < BaseFee$: Abort
   # Tx invalid: conversion rate should be at least BaseFee
2. **If** $GasPrice < msg.sender.balance$: Abort
   # Sender account cannot afford Tx
3. Adjust $msg.sender.balance \leftarrow msg.sender.balance - MaxTxFee$
   # Deduct maximal Tx fee from sender's account balance

Gas: Calculation II

Algorithm for determining validity of Tx in terms of gas (cont'd):

4. Set $Gas \leftarrow GasLimit$
5. Execute Tx: Deduct *gas* for each instruction from *Gas*
   # Subtract factual gas costs for Tx from maximal gas costs specified
6. **If** $Gas < 0$: Abort
   # Tx invalid: more gas spent than allowed by sender
   # Paid to miner: $GasLimit \times GasPrice$

Gas: Calculation III

Algorithm for determining validity of Tx in terms of gas (cont'd II):

7. $msg.sender.balance \leftarrow msg.sender.balance + Gas \times GasPrice$
8. $GasUsed \leftarrow GasLimit - Gas$

    8.1 *"Burn" GasUsed* $\times$ *BaseFee*
    # Burning prevents creation of fake Tx's and offline agreements
    # Burning destroys coins ☞ possible deflation
    8.2 Send $GasUsed \times (GasPrice - BaseFee)$ to miner

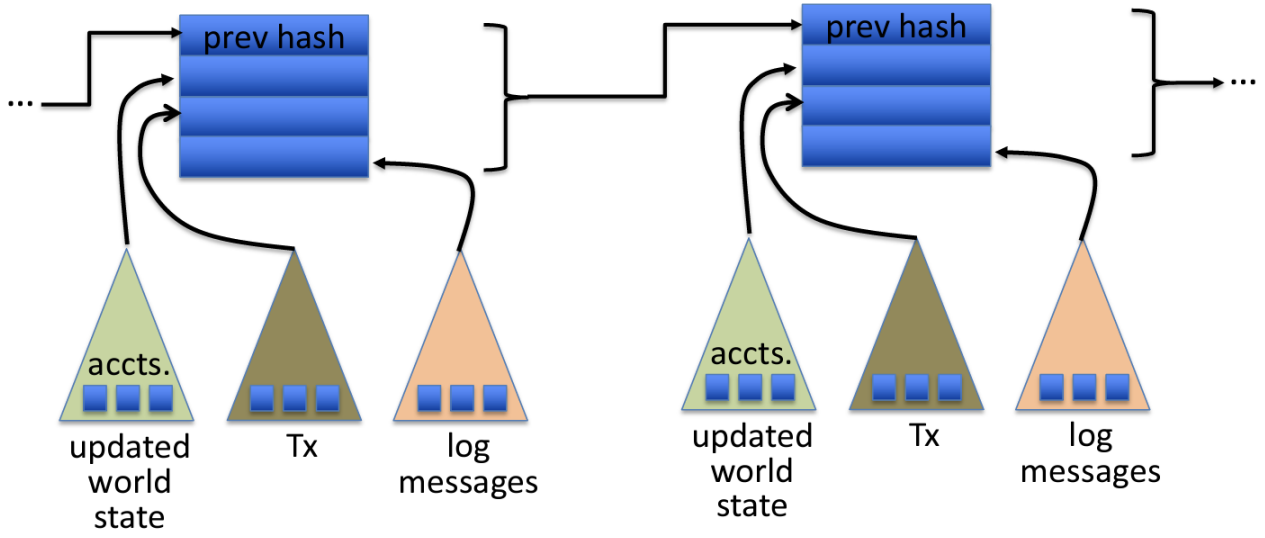
Ethereum
–
Gas

Ethereum
–
Mining

Smart Contracts
Basic Examples

Smart Contracts
Advanced
Examples

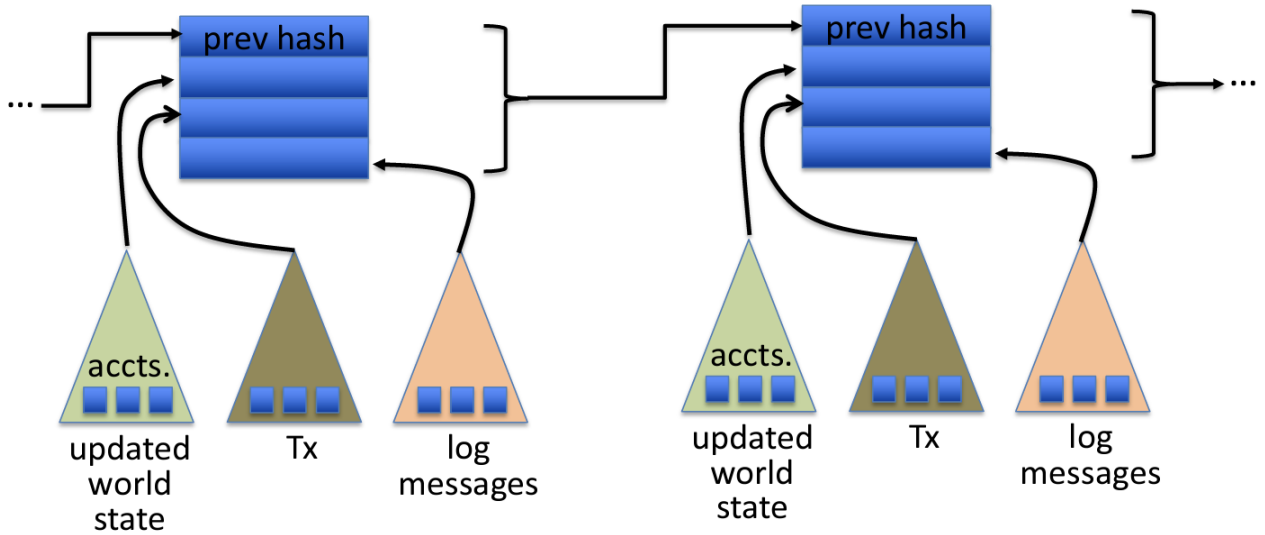
Ethereum: Blockchain



From `cs251.stanford.edu`

- Each block records:
  - Full, updated account based state
  - Transactions that lead to updated state
  - Log messages resulting from performing transactions

Ethereum: Mining



From `cs251.stanford.edu`

- Miner collects transactions into block
- Executes all transactions and computes updated state
  ☞ Note that order of transactions could matter
- Other nodes re-execute transactions to verify block

Ethereum
–
Gas

Ethereum
–
Mining

Smart Contracts
Basic Examples

Smart Contracts
Advanced
Examples

Smart Contracts: Resources

- *Solidity Documentation:* `https://docs.soliditylang.org/en/v0.8.14/`
- *Remix:* `https://remix.ethereum.org/`
- *Remix Documentation:* `https://remix-ide.readthedocs.io/en/latest/`

Smart Contracts: Basic Examples

- *Storage:* Simple smart contract template
- *Conference:* Organizing a conference, purchasing and refunding tickets
- *Coin:* Implement a (sub-)currency

Basic Examples: Storage

- *Description:* Storing and retrieving a value
- *Ingredients:*
  - Licensing and versioning
  - Variables and functions
  - Deploying contracts
  - Calling functions

Basic Examples: Conference

- *Description:* Organizing a conference
  - Participants purchase tickets
  - Keeping track of participants; upper limit on number
  - Refunding participants if necessary
  - Cancel conference
- *Ingredients:*
  - More advanced types and functions
  - Events: writing to log space
  - Deploying via constructor
  - Transferring ether

Basic Examples: Coin

- *Description:* Simulate currency
  - One "minter" can create coins
  - Owners can transfer coins to each other
  - Keep track of balances of coin owners
- *Ingredients:*
  - Address type
  - Error handling

Ethereum
–
Gas

Ethereum
–
Mining

Smart Contracts
Basic Examples

Smart Contracts
Advanced
Examples

Advanced Examples: Resources

- *Solidity by Examples:* `https://docs.soliditylang.org/en/v0.8.14/solidity-by-example.html`
- *Contracts:*
  - *Ballot*
  - *Purchase*

Advanced Examples: Ballot

- *Description:* Simulate voting process
  - Voters can vote for other votes
  - Voters can accumulate votes by receiving delegated votes
  - Voters can delegate vote(s) to other voters
  - Eventually, winner is determined
- *Ingredients:*
  - Struct type
  - Conditions and loops

Advanced Examples: Escrow

- *Description:* Buyer wants to pay seller only upon having received goods in order
  - If goods are fine, pay
  - If goods are not Ok, refund buyer
- *Ingredients:*
  - Function modifiers
  - Enum type
  - Advanced error handling

Materials / Outlook

- See *Bitcoin and Cryptocurrency Technologies*, 10.7
- See `cs251.stanford.edu`, Lecture 7 & 8
- See also
  - `https://bitcoinbook.cs.princeton.edu/`
  - `https://ethdocs.org/en/latest/index.html`
  - `https://ethereum.org/en/developers/docs/`
  - `https://docs.soliditylang.org/en/v0.8.14/`
  - `https://remix.ethereum.org/`
  - `https://remix-ide.readthedocs.io/en/latest/`

  for further resources
- Next lecture: “Ethereum Mechanics & Solidity”