# Raspodijeljene glavne knjige i kriptovalute

## Pametni ugovori na Ethereum platformi, jezik Solidity

Ante Đerek, Zvonko Konstanjčar

14. prosinca 2023.

## Literatura

- Mastering Ethereum, Andreas M. Antonopoulos, Gavin Wood, dostupna na `https://github.com/ethereumbook/ethereumbook/`
- Solidity documentation, dostupna na `https://solidity.readthedocs.io/en/latest/`

## Također zanimljivo:

- Ethereum whitepaper, Vitalik Buterin, dostupan na `https://ethereum.org/en/whitepaper/`
- Ethereum yellowpaper, Gavin Wood, dostupan na `https://ethereum.github.io/yellowpaper/paper.pdf`

## Nick Szabo (1996)

*A set of promises, specified in digital form, including protocols within which the parties perform on other promises.*

## ethereum.org (2018)

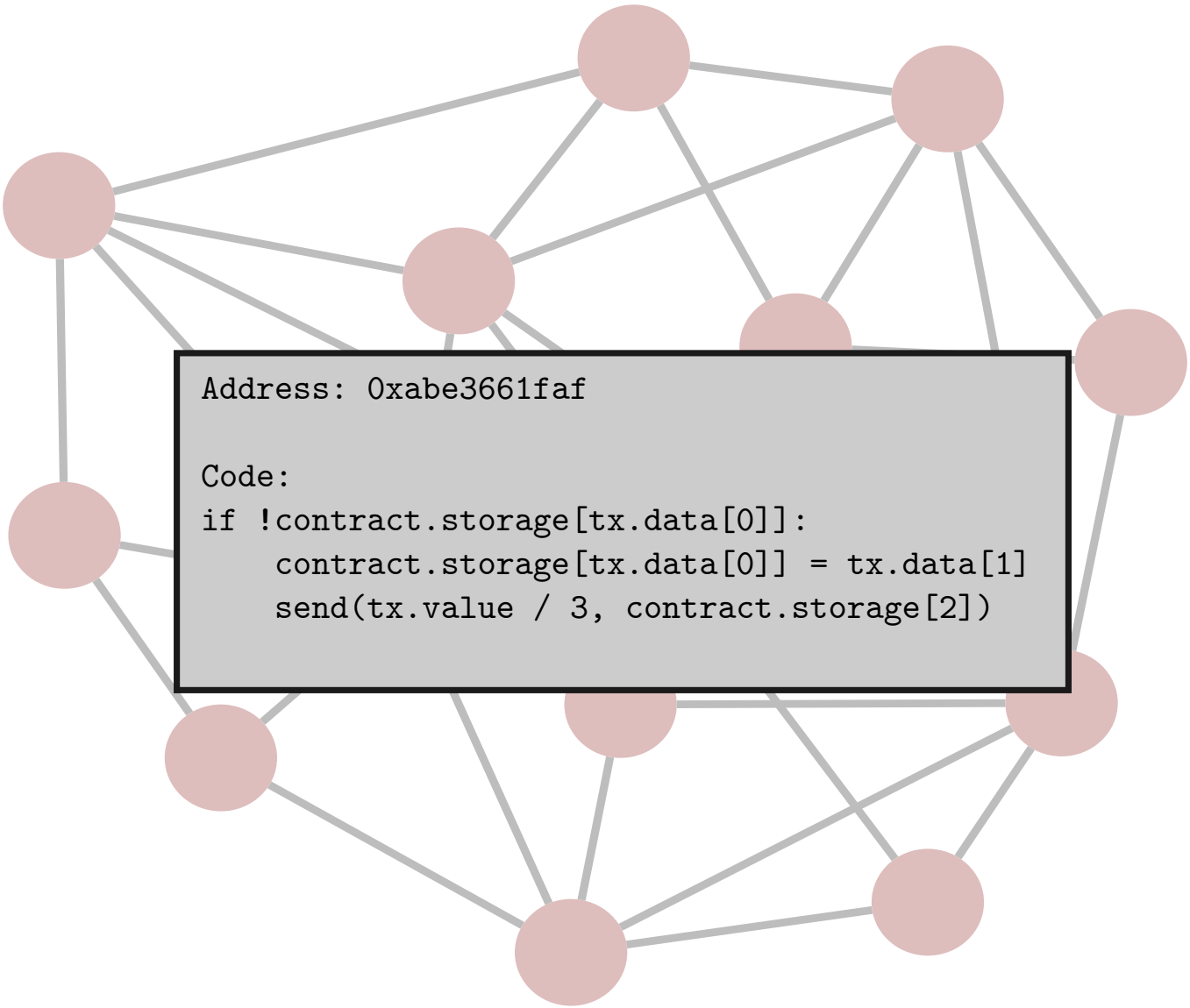
*Applications that run exactly as programmed without any possibility of downtime, censorship, fraud or third-party interference.*

## Definicija

*Pametni ugovor je javni i nepromjenjivi računalni program koji je pohranjen na kriptografskom lancu blokova i koji se može javno i pouzdano izvršavati koristeći kriptografski lanac blokova i distribuirani konsenzus.*

# Pametni ugovori – javni i nepromijenjivi



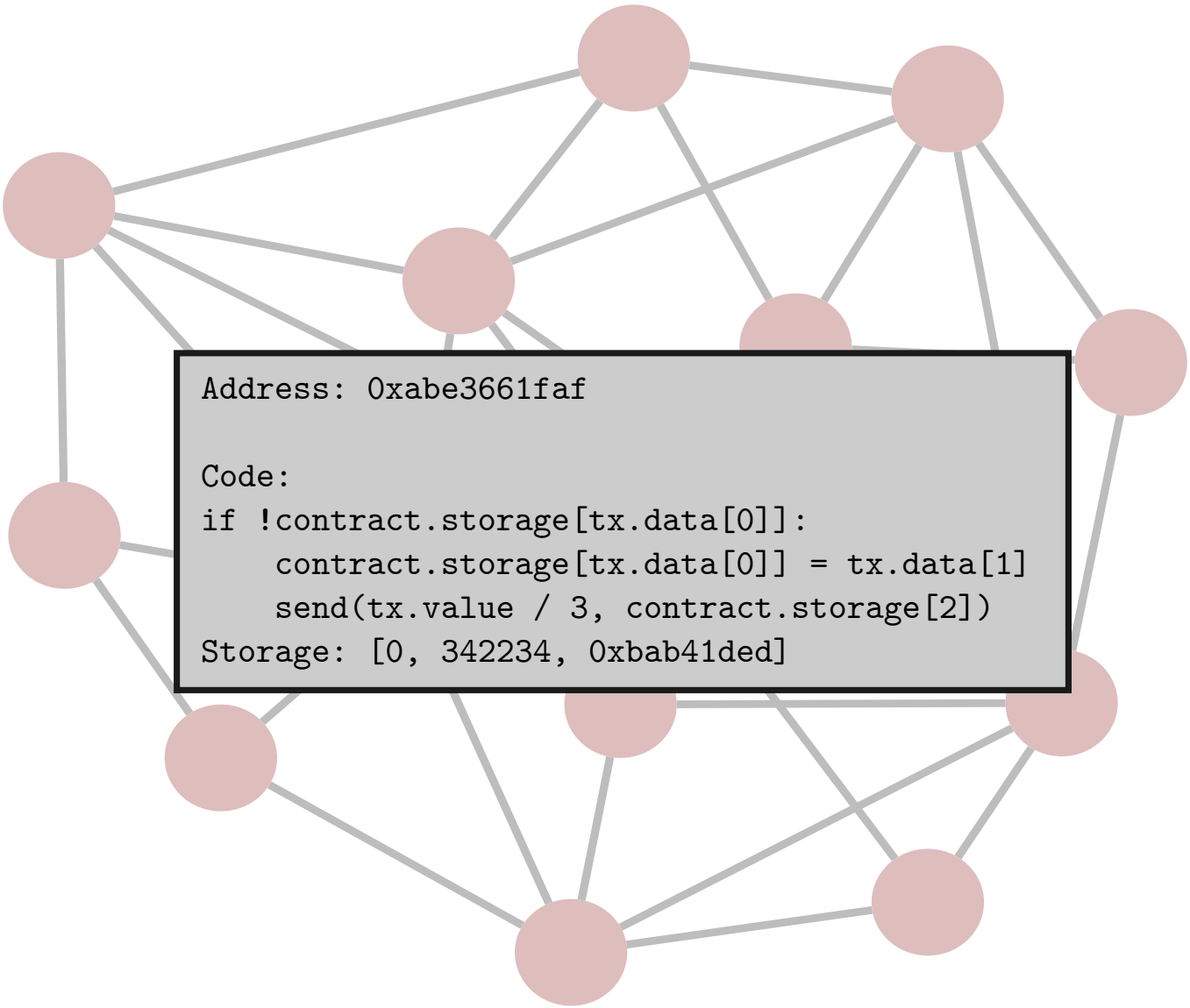


```
Address: 0xabe3661faf

Code:
if !contract.storage[tx.data[0]]:
    contract.storage[tx.data[0]] = tx.data[1]
    send(tx.value / 3, contract.storage[2])
```

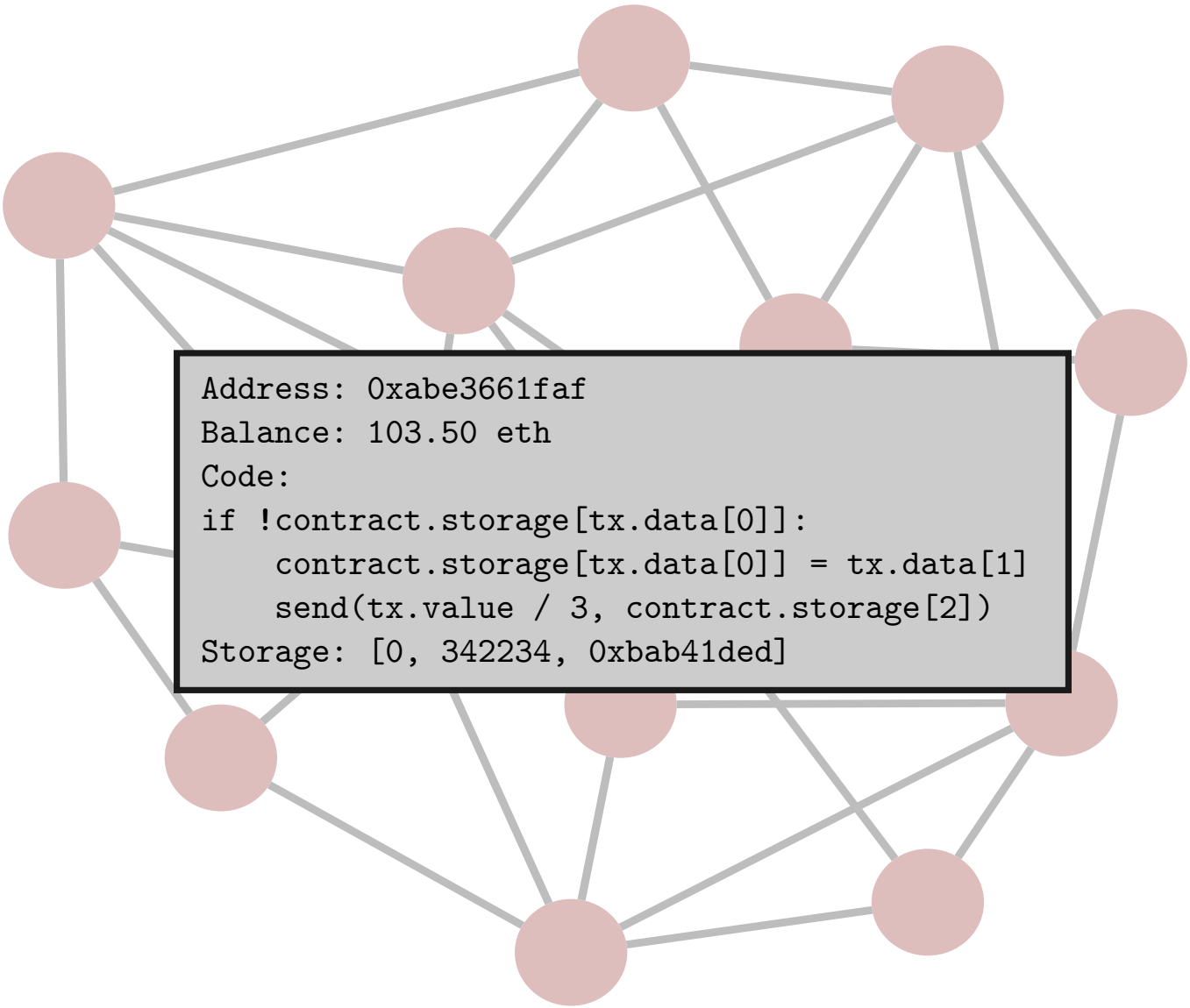
# Pametni ugovori – mogu pohraniti informacije



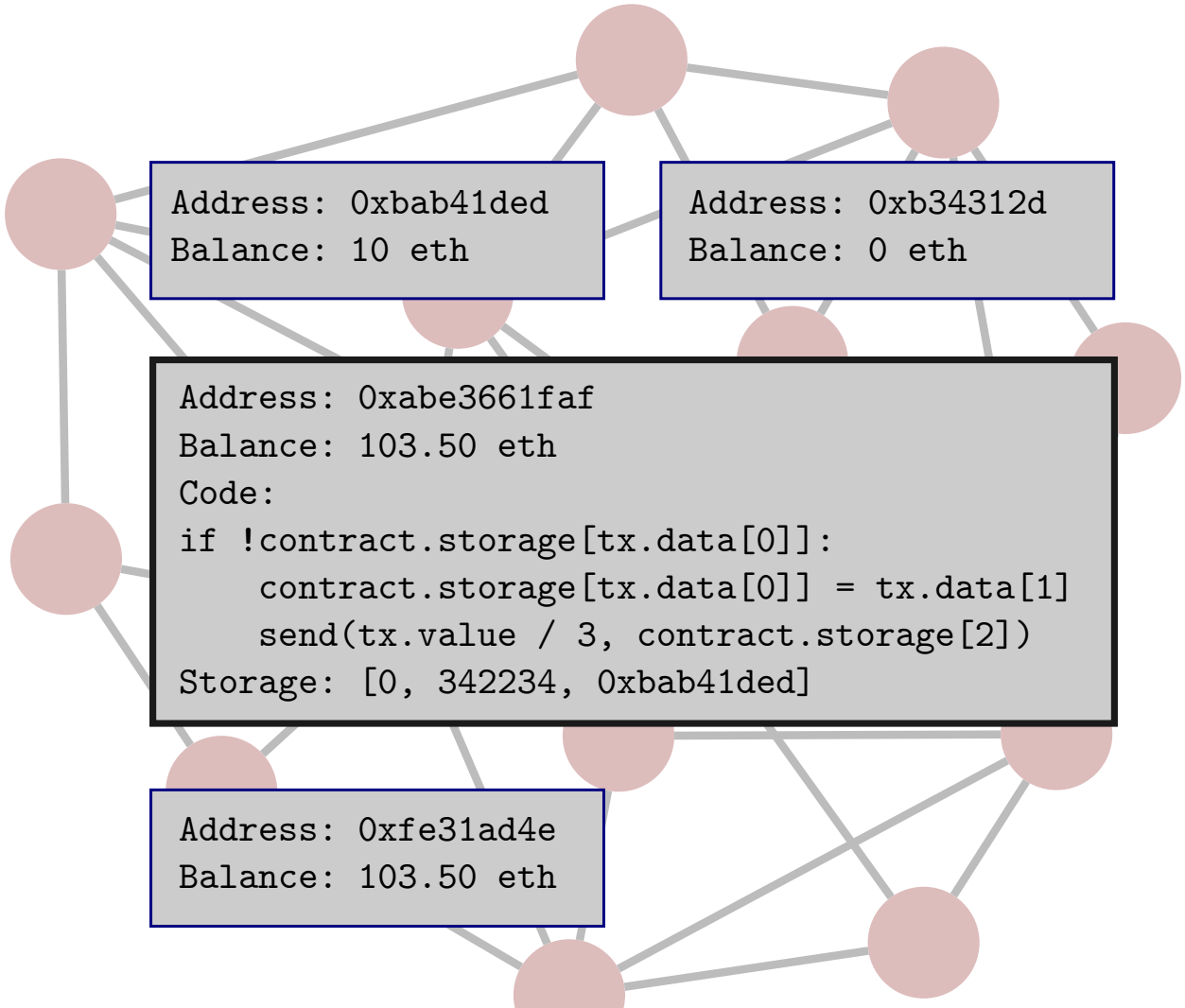


# Pametni ugovori – vlasnici kriptovalute

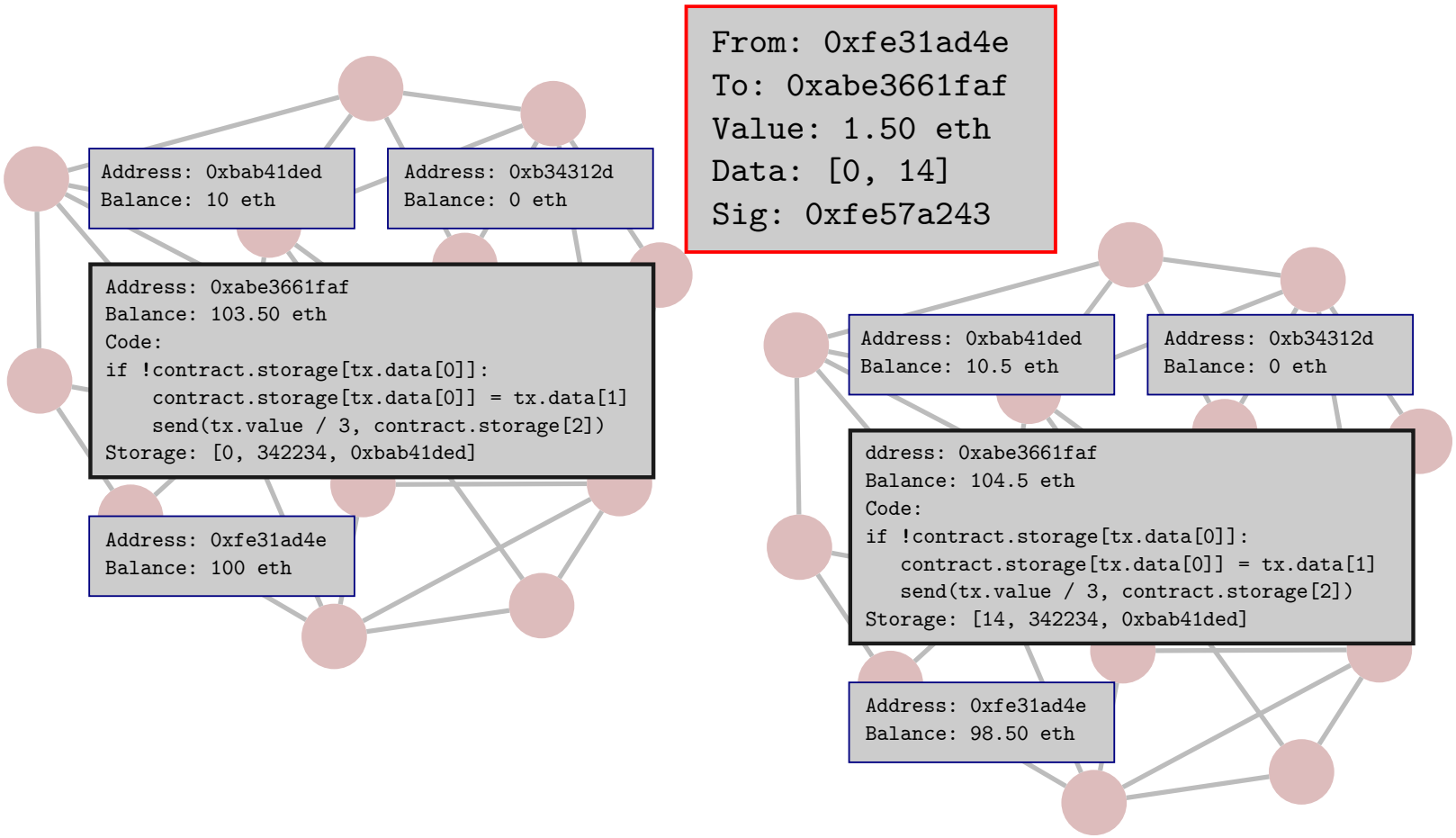




- Vanjski račun (*externally owned account*)
- Pametni ugovor (*contract*)



# Pametni ugovori – izvršavanje koda



## Globalno stanje kriptografskog lanca blokova

- Za svaki vanjski račun:
  - Iznos kriptovalute.
- Za svaki ugovor:
  - Iznos kriptovalute.
  - Sadržaj memorije.

## Transakcija

Transakcija mijenja globalno stanje.

- Obične transakcije: prebacuju kriptovalutu.
- Pozivi ugovora: izvršavaju program ugovora.

Izvor transakcije je uvijek *vanjski* račun!

**Zadatak**

*Osmislite pametni ugovor koji će služiti kao fond za Mirkovo fakultetsko obrazovanje i koji će imati sljedeća svojstva:*

- *Svatko može uplatiti novac u fond.*
- *Mirko može podići novac kada navrši 18 godina.*
- *Mirkovi roditelji mogu podići novac bilo kada, ako to oboje zatraže.*

**Zadatak**

*Tko je vlasnik ugovora kojeg ste osmislili i kako možemo implementirati vlasništvo ugovora?*

**Zadatak**

*Proširite problem na proizvoljan način i prilagodite dizajn.*

## Izražajnost ugovora (najčešće) nije ograničena

Prilikom izvršavanja ugovor može:

- računati bilo što (jezik je *Turing potpun*),
- slati kriptovalutu,
- pozivati druge ugovore.

## Ugovor se izvršava u *ograničenom* kontekstu

Ako transakcija $T$ poziva ugovor $C$, ugovoru su dostupni samo:

- trajna memorija ugovora $C$,
- podaci iz transakcije $T$,
- izvor transakcije $T$ (autentificiran digitalnim potpisom),
- metapodaci iz trenutnog i nedavnih blokova,
- rezultati eventualnih poziva drugih ugovora od strane $C$-a.

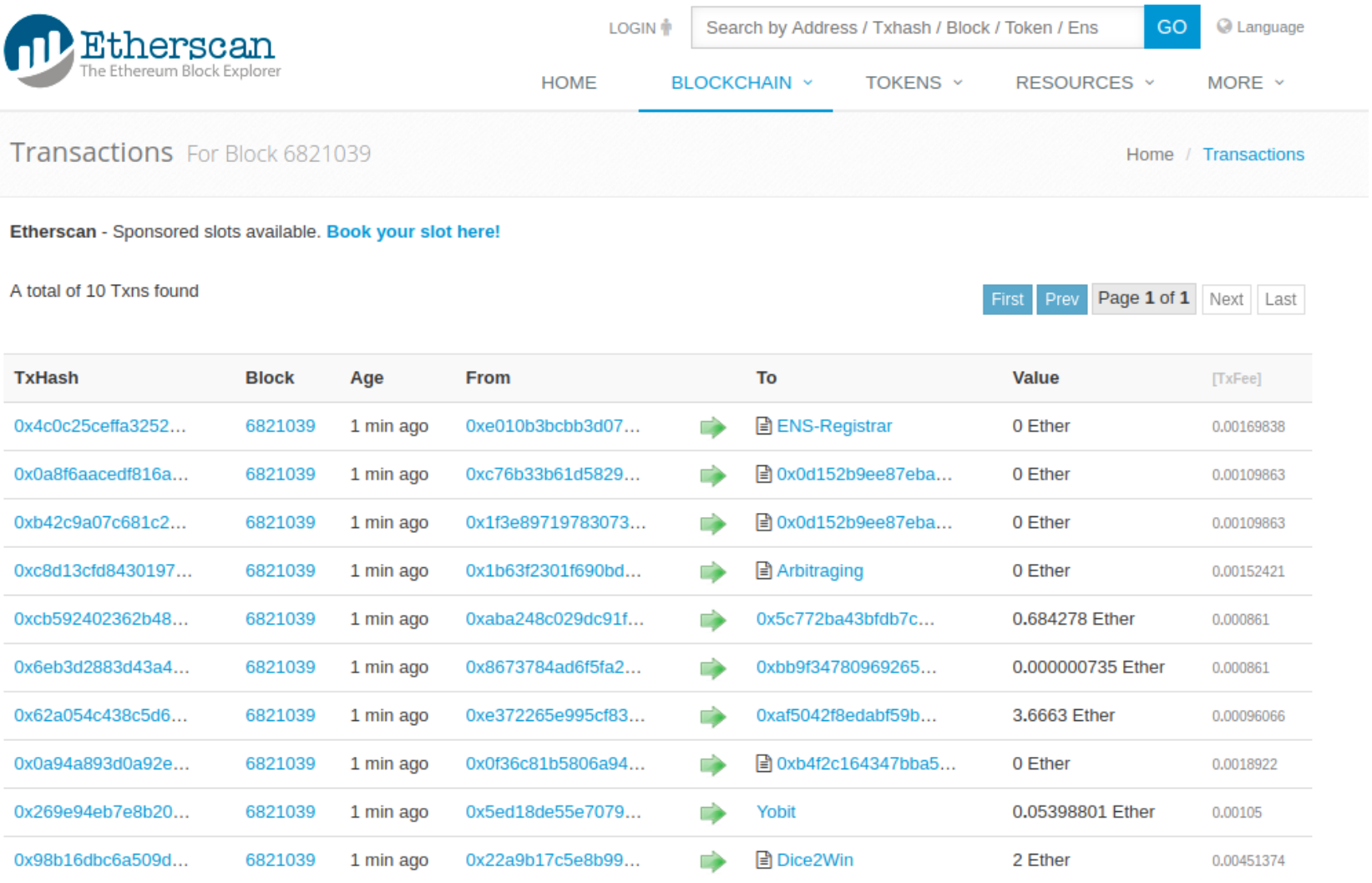
## Želimo konsenzus oko ispravnog izvršavanja!

Čvorovi mreže se moraju usaglasiti:

- je li poziv svaki ugovora izvršen ispravno,
- kojim su redoslijedom izvršene transakcije,
- koji je rezultat izvršavanja transakcija odnosno ugovora.

## Kako postići raspodijeljeni konsenzus?

- Niz transakcija zajedno s krajnjim globalnim stanjem čini *blok*.
- Rudari izvršavaju pozive ugovora prilikom slaganja bloka.
- Distribuirani konsenzus slično kao kod "običnih" kriptovaluta (*proof-of-work* ili *proof-of-stake*).


Etherscan
The Ethereum Block Explorer
LOGIN
Search by Address / Txhash / Block / Token / Ens
GO
Language
HOME
BLOCKCHAIN
TOKENS
RESOURCES
MORE
Transactions For Block 6821039
Home / Transactions
Etherscan - Sponsored slots available. Book your slot here!
A total of 10 Txns found
First Prev Page 1 of 1 Next Last
TxHash Block Age From To Value [TxFee]
0x4c0c25ceffa3252... 6821039 1 min ago 0xe010b3bcbb3d07... ENS-Registrar 0 Ether 0.00169838
0x0a8f6aacedf816a... 6821039 1 min ago 0xc76b33b61d5829... 0x0d152b9ee87eba... 0 Ether 0.00109863
0xb42c9a07c681c2... 6821039 1 min ago 0x1f3e89719783073... 0x0d152b9ee87eba... 0 Ether 0.00109863
0xc8d13cfd8430197... 6821039 1 min ago 0x1b63f2301f690bd... Arbitraging 0 Ether 0.00152421
0xcb592402362b48... 6821039 1 min ago 0xaba248c029dc91f... 0x5c772ba43bfdb7c... 0.684278 Ether 0.000861
0x6eb3d2883d43a4... 6821039 1 min ago 0x8673784ad6f5fa2... 0xbb9f34780969265... 0.000000735 Ether 0.000861
0x62a054c438c5d6... 6821039 1 min ago 0xe372265e995cf83... 0xaf5042f8edabf59b... 3.6663 Ether 0.00096066
0x0a94a893d0a92e... 6821039 1 min ago 0x0f36c81b5806a94... 0xb4f2c164347bba5... 0 Ether 0.0018922
0x269e94eb7e8b20... 6821039 1 min ago 0x5ed18de55e7079... Yobit 0.05398801 Ether 0.00105
0x98b16dbc6a509d... 6821039 1 min ago 0x22a9b17c5e8b99... Dice2Win 2 Ether 0.00451374

## Posljedice raspodijeljenog konzenzusa

- Svi čvorovi moraju izvršavati kod i provjeriti rezultat.
- Izvršavanje mora biti deterministično.
- Svaki poziv se mora izvršiti do kraja ili se ne izvršiti uopće.

## Zadatak

*Možete li:*

- *dizajnirati pametni ugovor koji simulira klađenje na ishod bacanja novčića?*
- *dizajnirati pametni ugovor koji simulira klađenje na igru "par-nepar"?*

## Gdje su poticaji za izvršavanje ugovora!

- Zašto bi rudar trošio resurse za izvršavanje ugovora kad može samo obrađivati *obične* transakcije?
- Što ako se poziv ugovora dugo (ili beskonačno) izvršava? Što ako troši nerazumnu količinu memorije?

## Rješenje: gorivo

Inicijator transakcije plaća resurse potrebne za izvršavanje *gorivom*. U svakoj transakciji inicijator navodi:

- Gornji limit na količinu goriva predviđenog za izvršavanje transakcije.
- Cijenu naknade koju je voljan platiti po jedinici goriva.

# Pametni ugovori – neka preostala pitanja



- Kako nastaje ugovor?
- Može li ugovor nestati?
- Može li ugovor kreirati transakciju?
- Ima li ugovor kriptografske ključeve?
- Kako točno izvršavanje troši gorivo?
- Što se točno događa ako se pozove beskonačna petlja?

Jednostavan jednodretveni virtualni stroj baziran na stogu.

## Podaci

- Bytecode programa koji se izvršava (*read only*).
- Trajna memorija (dio globalnog stanja).
- Privremena memorija.
- Privremeni stog.

## Instrukcije

- Aritmetičko-logičke, operacije (`ADD`, `MUL`, `SHA3`, ...).
- Operacije sa stogom (`POP`, `LOADM`, `DUP2`, ...).
- Kontrola toka programa (`STOP`, `JUMP`, `JUMPI`, ...).
- Sistemske operacije (`CREATE`, `CALL`, `RETURN`, ...).
- Dohvat informacija iz transakcije ili bloka (`CALLER`, `BALANCE`, `NUMBER`, ...)

**Bug-ovi su skupi!**

Pametni ugovor vjerojatno ne želite ručno pisati u bytecode-u.

**Specijalizirani programski jezici**

- Solidity: objektno-orijentirani jezik, sintaksom sličan Javi.
- Vyper: funkcijski jezik, sintaksom sličan Python-u.

*Ugovor* se sastoji od funkcija i podataka.

**Storage**

```
contract SimpleStorage {
    uint storedData;
    function set(uint x) public {
        storedData = x;
    }
    function get() public view returns (uint) {
        return storedData;
    }
  }
```

Izvor: `solidity.readthedocs.io`

Ugovor može slati i primati sredstva.

Faucet.sol

```
contract Faucet {
    // Give out ether to anyone who asks
    function withdraw(uint withdraw_amount) public {
        // Limit withdrawal amount
        require(withdraw_amount <= 10000000000000000);
        // Send the amount to the address that requested it
        msg.sender.transfer(withdraw_amount);
    }
    // Accept any incoming amount
    function () public payable {}
}
```

Izvor: github.com/ethereumbook

Jesu li pozivi funkcija dio specifikacije virtualnog stroja?

## The Contract Application Binary Interface (ABI)

*Dogovoreni* način interakcije s ugovorima na Ethereum platformi – prilikom pozivanja ugovora iz transakcija i iz drugog ugovora.

## Specifikacija (otprilike):

- Prva četiri byte-a *data* polja u transakciji označavaju funkciju koja se poziva.
- Oznaka funkcije su prva četiri byte-a Keccak-256 sažetka njezinog prototipa.
- Ostatak *data* polja sadrži enkodirane parametre funkcije koja se zove.

## Tipovi podataka

- Boolean (bool)
- Integer (int, uint, uint8, . . . , uint256)
- Fixed point (fixed, ufixed)
- Address (A 20-byte Ethereum address)
- Byte array (fixed or dynamic)
- Enum
- Arrays
- Struct
- Mapping (hash tablica)

## Izražavanje konstantnih vrijednosti

- Vremenske jedinice (seconds, minites, hours, days)
- Ether novčane jedinice (wei, finney, szabo, ether)

## Coin

```
contract Coin {
    address public minter;
    mapping (address => uint) public balances;

    constructor() public {
        minter = msg.sender;
    }

    function mint(address receiver, uint amount) public {
        require(msg.sender == minter);
        require(amount < 1e60);
        balances[receiver] += amount;
    }

    function send(address receiver, uint amount) public {
        require(amount <= balances[msg.sender], "Insufficient balance.");
        balances[msg.sender] -= amount;
        balances[receiver] += amount;
    }
}
```

Izvor: `solidity.readthedocs.io`

**Pazi!**

Baratanje s pogreškama je važan dio logike ugovora i česti izvor skupih grešaka.

Ako izvršavanje ugovora rezultira exceptionom onda transakcija ne mijenja globalno stanje. Funkcije: `assert`, `require`, `revert`.

```
contract Sharer {
    function sendHalf(address payable addr) public payable
      returns (uint balance) {
        require(msg.value % 2 == 0, "Even value required.");
        uint balanceBeforeTransfer = address(this).balance;
        addr.transfer(msg.value / 2);
        assert(address(this).balance ==
            balanceBeforeTransfer - msg.value / 2);
        return address(this).balance;
    }
}
```

Izvor: `solidity.readthedocs.io`

**Pazi!**

Bitno je znati što rade funkcije koje zovete u slučaju pogreške.

```
addr.transfer(funds / 2);
funds = funds - funds / 2;
```

```
addr.send(funds / 2);
funds = funds - funds / 2;
```

**Pazi!**

Bitno je pratiti novosti :)

Don't use transfer() or send() ...

Izvor: `consensys.github.io/smart-contract-best-practices`

## Stvaranje novog ugovora i interakcija s njim

```
import "Faucet.sol";

contract Token {
    Faucet _faucet;

    constructor() {
        _faucet = (new Faucet).value(0.5 ether)();
    }

    function destroy() ownerOnly {
        _faucet.destroy();
    }
}
```

Izvor: github.com/ethereumbook

## Interakcija s postojećim ugovorom

```
import "Faucet.sol";

contract Token is mortal {
    Faucet _faucet;
    constructor(address _f) {
        _faucet = Faucet(_f);
        _faucet.withdraw(0.1 ether)
    }
}
```

Izvor: `github.com/ethereumbook`

## Pazi!

Svaka interakcija s drugim ugovorom je opasna.

- Je li poznat uopće kod drugog ugovora?
- Što ako pozvani ugovor opet pozove originalni ugovor?
- ...

Nismo pričali o puno toga:

- Destruktori.
- `private`, `public`, `payable`, ....
- *Function modifiers*.
- Nasljeđivanje ugovora.
- Eventi.
- Razvoj i testiranje.
- Kako procijeniti potrebno gorivo.
- Sigurnosni propusti i dobra praksa pisanja ugovora.
- Dapps.
- Oracles.
- ...

**Zadatak**

*Istražite kojih su najpopularniji ugovori na Ethereum platformi.*

**Zadatak**

*Pronađite neki pametni ugovor na Etherscan-u pisan u Solidity-u te proučite što radi i kako radi.*

**Zadatak**

*Istražite koliko je koštao najskuplji sigurnosni propust u pametnom ugovoru.*

**Zadatak**

*Istražite koliki su računalni resursi potrebni za čisto računanje na Ethereum platformi u usporedbi s vašim laptopom.*