Raspodijeljene glavne knjige i kriptovalute

Jezik Solidity, Ethereum protokol

Ante Đerek, Zvonko Konstanjčar

8. siječnja 2024.

Logistika – druga laboratorijska vježba

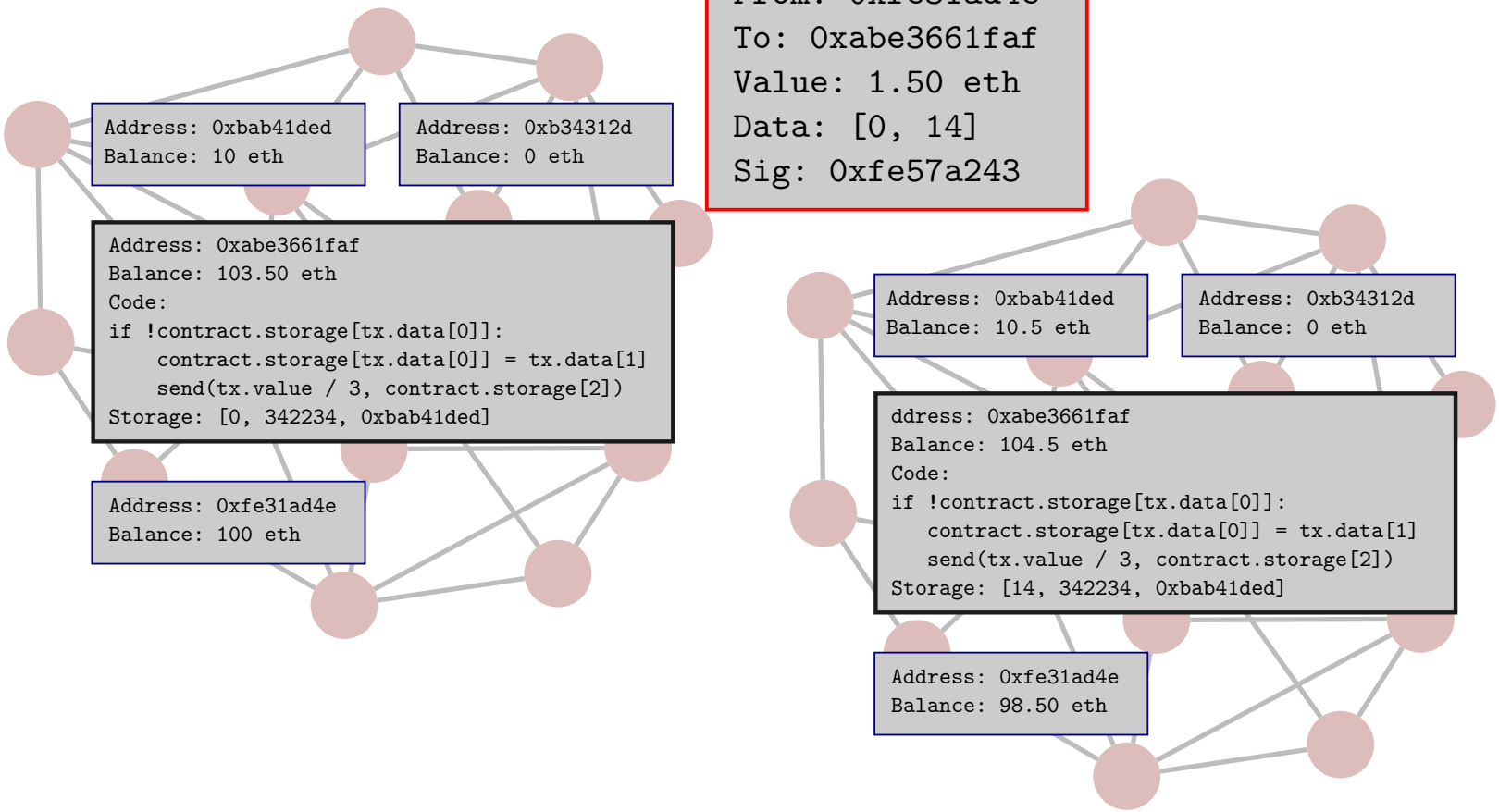
Druga vježba: Pametni ugovori u jeziku Solidity

- Objavljena početkom tjedna.
- Rok za predaju: 07.01.2024. u 23:59h.
- Obrana vježbe: 9.1.2024.–13.1.2024.

Naučite Solidity!

Završni ispit pretpostavlja znanje Solidity-ja potrebno za izradu vježbe.

Ponavljanje: Pametni ugovori



Ponavljanje: Pametni ugovori – konsenzus

Kako postići raspodijeljeni konsenzus?

- Niz transakcija zajedno s krajnjim globalnim stanjem čini *blok*.
- raspodijeljeni konsenzus slično kao kod "običnih" kriptovaluta (*proof-of-stake* ili *proof-of-work*).

Tko izvršava ugovore?

- Rudari izvršavaju pozive ugovora prilikom slaganja bloka.
- Svi čvorovi izvršavaju pozive ugovora prilikom prihvaćanja bloka.

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.

Ponavljanje: Pametni ugovori – izvršavanje

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.

Ponavljanje: Pametni ugovori – gorivo

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.

Ponavljanje: Programski jezik Solidity

```
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`

Primjene pametnih ugovora – podvalute

Ethereum token

Kriptovaluta implementirana kao pametni ugovor na Ethereum platformi.

ERC20

Tehnički standard za implementaciju tokena.

Primjeri

USDT, BNB, ...

Primjene pametnih ugovora – ERC20 Standard

```
...
function name() public view returns (string)
function symbol() public view returns (string)
function totalSupply() public view returns (uint256)
function balanceOf(address _owner) public view returns (uint256 balance)
...
function transfer(address _to, uint256 _value)
    public returns (bool success)
function approve(address _spender, uint256 _value)
    public returns (bool success)
function transferFrom(address _from, address _to, uint256 _value)
    public returns (bool success)
...
```

Primjene pametnih ugovora – DeFi

Decentralizirane burze

Pametni ugovori koji omogućuju razmjenu i trgovanje tokenima.

Krediti i ulaganje

Pametni ugovori koji omogućuju ulaganje tokena te uzimanje zajmova uz kamatu.

Primjeri

Uniswap, IDEX, Aave

Primjene pametnih ugovora – NFT

NFT

Omogućavaju bilježenje “vlasništva” i razmjenu digitalnih sadržaja.

Primjeri

OpenSea, Upland, . . .

Primjene pametnih ugovora – ERC721 Standard

```
...
function balanceOf(address _owner) external view returns (uint256);
function ownerOf(uint256 _tokenId) external view returns (address);
...
function safeTransferFrom(address _from, address _to, uint256 _tokenId,
   bytes data) external payable;
function safeTransferFrom(address _from, address _to, uint256 _tokenId)
   external payable;
function transferFrom(address _from, address _to, uint256 _tokenId)
   external payable;
function approve(address _approved, uint256 _tokenId) external payable;
...
```

Primjene pametnih ugovora – imena i reputacija

Sustavi za registraciju imena

Pametni ugovor održava bazu koja imenima pridružuje adrese.

Reputacijski sustavi

Pametni ugovor prikuplja certifikate na kojima se zasniva reputacija i povjerenje.

Primjeri

`https://ens.domains/`

Primjene pametnih ugovora – orakli

Sustavi za registraciju imena

Omogućavaju sučelje sa “stvarnim svijetom”.

Primjeri

ChainLink

Primjene pametnih ugovora – DAO

Raspodijeljena autonomna organizacija

Virtualni entitet, suvlasnici glasanjem kolektivno odlučuju na koji način djelovati, potrošiti, probati zaraditi novce.

Jednostavan primjer

- Skup pametnih ugovora čije se ponašanje može mijenjati i proširivati (npr. kreiranjem novih ugovora.)
- Svaki suvlasnik može predložiti novu funkcionalnost (npr. u obliku ugovora).
  - Suvlasnici glasaju o prijedlogu.
  - Ako više od dvije trećine suvlasnika podrži prijedlog, on se automatski izvršava.

Neslavni primjer:

"The DAO".

Primjene pametnih ugovora

Puno potencijalnih primjena ...

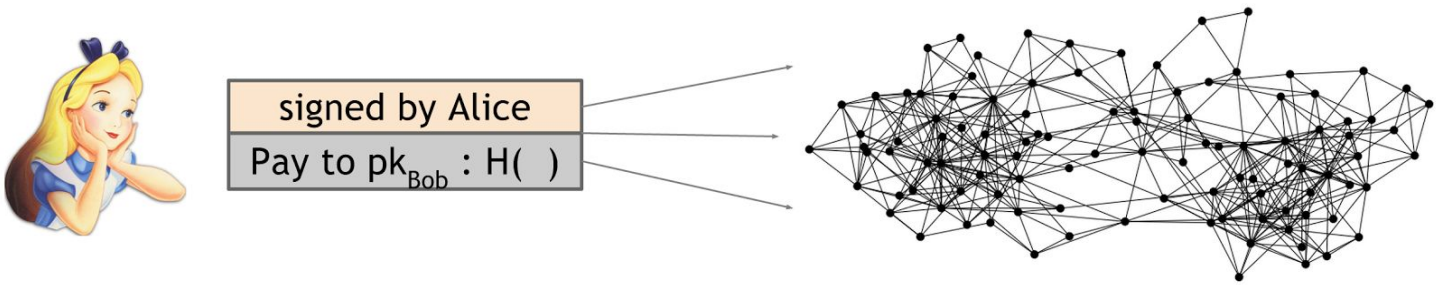
- Financijski derivati.
- Aukcije.
- Glasanje.
- Kockanje i klađenje.

... i puno problema.

- Zašto je uopće potrebno koristiti pametni ugovor?
- Kako se uvjeriti da je pametni ugovor ispravan?
- Kako na dobar način povezati ugovore sa stvarnim svijetom?
- ...

Arhitektura sustava

- Puno čvorova u "peer-to-peer" mreži.
- Svi čvorovi imaju skoro identične kopije lanca blokova.
- Svaki čvor održava skup transakcija koje treba dodati u lanac.
- Periodički se dodaje novi blok u lanac:
  - Koristi se *proof-of-stake* mehanizam konsenzusa.



Izvor: `bitcoinbook.cs.princeton.edu`

Ethereum protokol – ključevi i adrese

Kriptografski primitivi u Ethereumu

- Hash funkcija: Keccak256
- Digitalni potpis: ECDSA nad secp256k1 (kao i Bitcoin)

Adresa

- EOA: zadnjih 20 byte-ova sažetka javnog ključa.
- Ugovor: zadnjih 20 byte-ova sažetka para (A, n_A) gdje je A adresa stvaratelja, a n_A redni broj transakcije s adrese A koja je stvorila ugovor.

Ethereum protokol – transakcije

Unutar lanca blokova i prilikom komunikacije na Ethereum mreži:

- Transakcija je zapisana u kompaktnom binarnom obliku.
- Hash transakcije služi kao njezin identifikator.

```
{
 nonce: 2,
 gasPrice: 0.0000000765 ether,
 gasLimit: 21000,
 to: '0x0975ca9f986eee35f5cbba2d672ad9bc8d2a0844',
 value: 0.02893164 ether,
 data: '',
 v: 38,
 r: '0xd2b0d401b543872d2a6a50de92455decbb868440321bf63a13b310c069e2ba5b',
 s: '0x3c6d51bcb2e1653be86546b87f8a12ddb45b6d4e568420299b96f64c19701040'
}
```

Ethereum protokol – stanje računa

Bitcoin (UXTO)

Stanje računa je skup svih njegovih nepotrošenih novčića.

Ethereum (Account)

Stanje računa je broj — količina kriptovalute koju posjeduje.

Koje su razlike između dva pristupa?

- Gdje je veća doza privatnosti?
- Što je jednostavnije implementirati?
- Što bolje skalira?
- Što troši više prostora?

Ethereum protokol – nonce

Uloge

- Štiti od napada ponovnim slanjem iste transakcije.
- Omogućava korisniku da poreda istovremene transakcije.

Implementacija

- Globalno stanje za svaku adresu A uključuje i broj transakcija n_A kojima je ta adresa bila izvor.
- Transakcija s adrese A je ispravna (u trenutnom globalnom stanju) samo ako je njezin *nonce* jednak n_A.

Ethereum protokol – transakcije

Razlika između "obične" transakcije i poziva ugovora?

```
{
 nonce: 476,
 gasPrice: 0.00000008 Ether,
 gasLimit: 210000,
 to: '0x9041fe5b3fdea0f5e4afdc17e75180738d877a01',
 value: 0 Ether,
 data: '0xa9059cbb000000000000000000000051093b3e69a6ab781ad596866...',
 v: 37,
 r: '0x9e8915ec764e62c3903618742cb8fc5b2daeebbfd0f705d978c8e84ba3074f31',
 s: '0x649961191caa11a0f787dbe1597464b04d73fc9cf007eb927b3d48fa5ba5bdb5'
}
```

Stvaranje ugovora?

```
...
 to: ''
 data: '0x6060604052604080519081016040526006808252f50524f312e300000...'
...
```

Ethereum protokol – globalno stanje

Globalno stanje (*World State*)

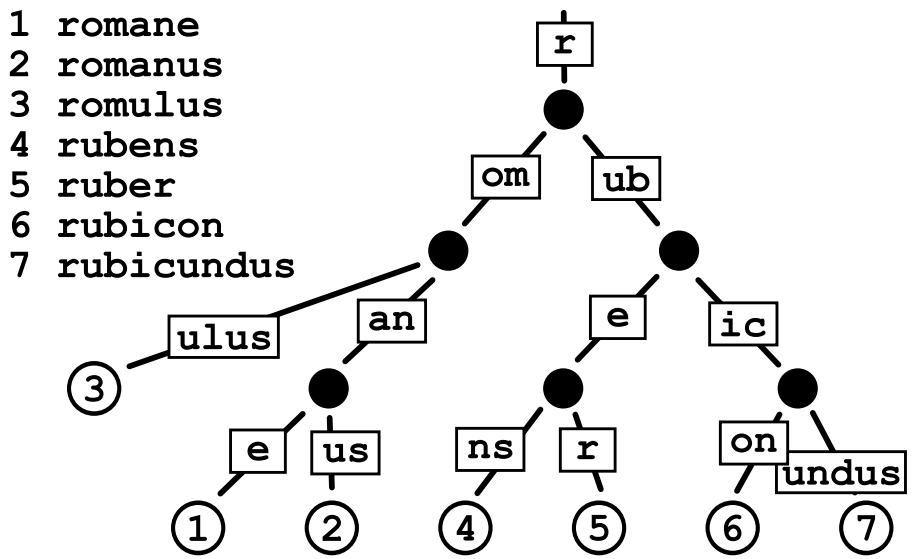
Preslikavanje između adresa i *stanja računa*.

Stanje računa:

- `nonce`: broj transkacija.
- `balance`: iznos kriptovalute.
- `storageRoot`: hash pokazivač na strukturu podataka u kojoj je pohranjena trajna memorija ugovora.
- `codeHash`: hash pokazivač na bytecode ugovora.

Strukture podataka

- *Recursive length prefix* (RLP) kodiranje za sve strukture (transakcija, stanje jednog računa, zaglavlje bloka, . . .).
- *Merkle-Patricia* stabla (MPT) za preslikavanja (globalno stanje, lista transakcija, trajna memorija ugovora, . . .).



Patricia (radix) stablo, Izvor: `wikipedia.org`

Potrošnja goriva – više detalja

- Na početku izvođenja ugovora `gasPrice * gasLimit` se skine s računa pošiljatelja transakcije.
- Svaka operacija prilikom izvođenja potroši neku količinu goriva.
- Ako na kraju izvođenja ostane `gasRem` goriva, onda se pošiljatelju transakcije vraća `gasPrice * gasRem`.
- Ostatak je naknada za rudara.
- Ako nestane goriva tijekom izvršavanja, stanje ugovora se vraća na početno, ali se transakcija smatra ispravnom i naknada ostaje rudaru.
- Kada ugovor zove drugi ugovor može mu ograničiti količinu goriva koju smije potrošiti.

Izvod iz cjenika:

$R_{selfdestruct}$	24000	Refund given (added into refund counter) for self-destructing an account.
$G_{selfdestruct}$	5000	Amount of gas to pay for a SELFDESTRUCT operation.
G_{create}	32000	Paid for a CREATE operation.
$G_{codedeposit}$	200	Paid per byte for a CREATE operation to succeed in placing code into state.
G_{call}	700	Paid for a CALL operation.
$G_{callvalue}$	9000	Paid for a non-zero value transfer as part of the CALL operation.
$G_{callstipend}$	2300	A stipend for the called contract subtracted from $G_{callvalue}$ for a non-zero value transfer.
$G_{newaccount}$	25000	Paid for a CALL or SELFDESTRUCT operation which creates an account.
G_{exp}	10	Partial payment for an EXP operation.
$G_{expbyte}$	50	Partial payment when multiplied by $\lceil \log_{256}(\text{exponent}) \rceil$ for the EXP operation.
G_{memory}	3	Paid for every additional word when expanding memory.
$G_{txcreate}$	32000	Paid by all contract-creating transactions after the Homestead transition.
$G_{txdatazero}$	4	Paid for every zero byte of data or code for a transaction.
$G_{txdatanonzero}$	68	Paid for every non-zero byte of data or code for a transaction.
$G_{transaction}$	21000	Paid for every transaction.
G_{log}	375	Partial payment for a LOG operation.
$G_{logdata}$	8	Paid for each byte in a LOG operation's data.
$G_{logtopic}$	375	Paid for each topic of a LOG operation.
G_{sha3}	30	Paid for each SHA3 operation.
$G_{sha3word}$	6	Paid for each word (rounded up) for input data to a SHA3 operation.
G_{copy}	3	Partial payment for *COPY operations, multiplied by words copied, rounded up.
$G_{blockhash}$	20	Payment for BLOCKHASH operation.

Izvor: `ethereum.github.io/yellowpaper`

Ethereum protokol – blok

Zaglavlje bloka (odabrana polja)

- `number`: visina bloka.
- `parentHash`: hash pokazivač na prethodni blok.
- `beneficiary`: adresa za nagradu.
- `transactionsRoot`: hash pokazivač na MPT transakcija.
- `receiptsRoot`: hash pokazivač na MPT potvrda.
- `stateRoot`: hash pokazivač na MPT globalnog stanja.
- `gasLimit`: trenutna najveća dozvoljena količina po bloku.
- `gasUsed`: ukupna količina potrošenog goriva.
- `difficulty`: (x) težina rudarenja.
- `mixHash`: (x) parametar za *proof-of-work* funkciju.
- `nonce`: (x) parametar za *proof-of-work* funkciju.

Ethereum 2.0

Ethereum 1

Proof-of-work sustav. Funkcija rudarenja dizajnirana tako da bude otporna na ASIC rudarenje.

Faza 0: *Beacon chain* (2020-2021)

Proof-of-stake blockchain koji trenutno radi paralelno Ethereum lancu i samo potvrđuje Ethereum blokove.

Faza 1: *The merge* (2022)

Prelazak Ethereum lanca na *proof-of-stake*.

Faza 2: *Shard Chains* (2023?)

Razdvajanje u 64 lanca koji se održavaju paralelno i sinkroniziraju pomoću *Beacon* lanca.

Ethereum protokol – konsenzus

Dva sloja

- *compute layer* (eth1) – dosadašnji Ethereum kriptografski lanac blokova, izvršava sve transakcije i pametne ugovore.
- *consensus layer* (eth2) – *Beacon* kriptografski lanac blokova, izvršava *proof-of-stake* konsenzus mehanizam i potvrđuje eth1 blokove.

Literatura:

- Ethereum Proof-of-Stake Consensus Specifications
  `https://github.com/ethereum/consensus-specs/`
- Ethereum Execution Client Specifications
  `https://github.com/ethereum/execution-specs/`

Ethereum protokol – *Beacon chain*

Validatori u Ethereum sustavu (pojednostavljeno)

- Čvorovi ulažu ether kako bi postali validatori (barem 32 ETH)
- Validatori provjeravaju i potpisuju blokove. Blok je ispravan kada skupi dovoljno potpisa validatora.
- Slučajno odabrani validator može predložiti novi blok.

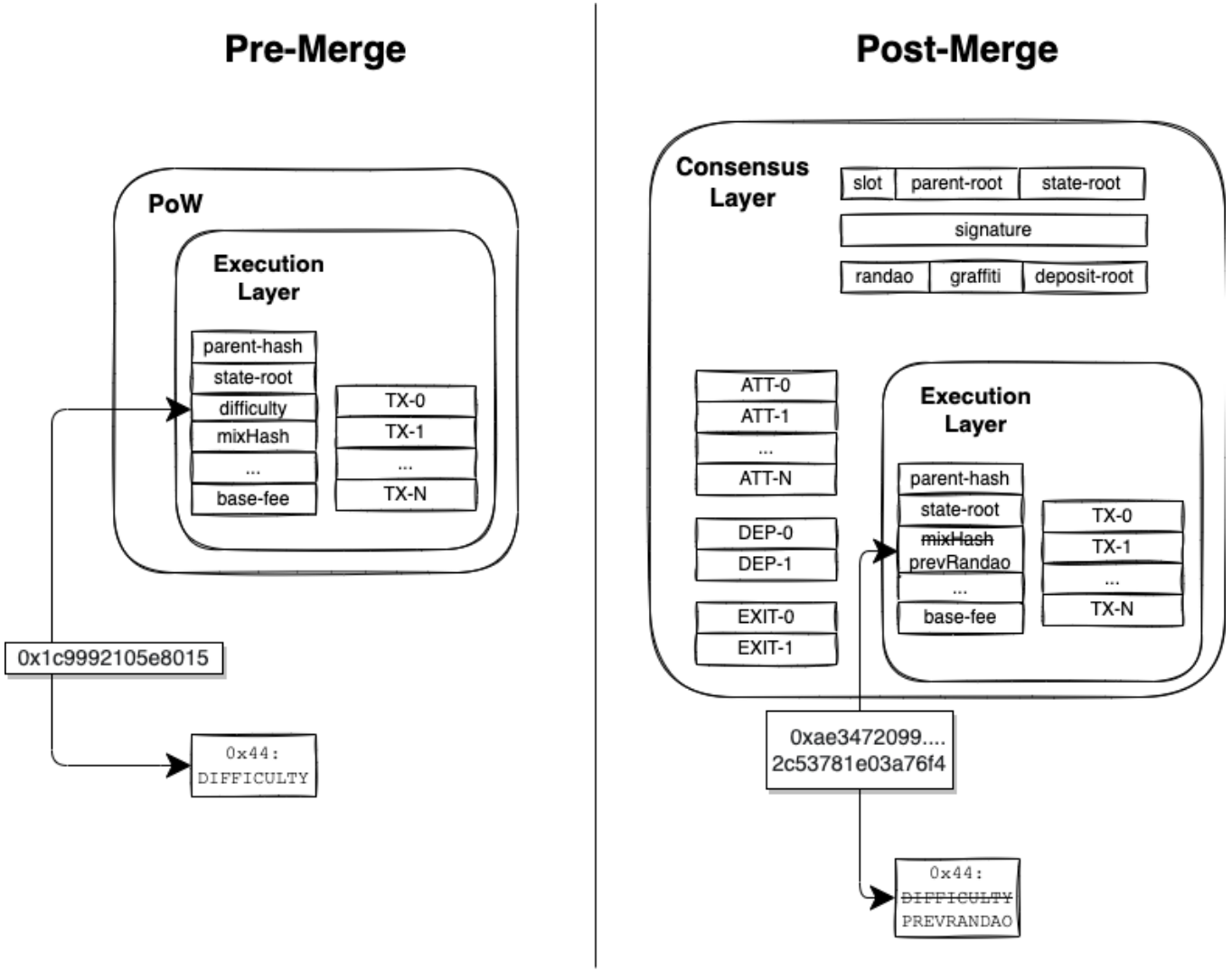
Ispravno ponašanje

Nagrade u ethereima.

Neispravno ponašanje

Oduzimanje (dijela) uloga (*slashing*)

Ethereum protokol – *Beacon chain*



Izvor: `blog.ethereum.org`