УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ

Мијановић Бојан

Дизајн и имплементација *Blockchain* механизма са криптовалутом

ДИПЛОМСКИ РАД
- Основне академске студије -

Нови Сад, 2024

Образац **Q2.НА.15-04** - Издање 1

Садржај

1 Увод

Blockchain технологија представља дистрибутивну, децентрализовану и јавну базу свих трансакција [1].

Први концепт *blockchain* технологије помиње се у 1982. години, када је Давид Чаум у својој дисертацији описао дистрибуирану базу података која користи криптографију [2]. Овај рани рад није био директно повезан са дигиталним валутама, али је поставио темеље за будући развој *blockchain* - а.

Права револуција долази 2008. године када Сатоши Накамото објављује рад "*Bitcoin*: *Peer-to-peer* електронски готовински систем", уводећи први модерни *blockchain* и криптовалуту *Bitcoin*. Генесис блок, први блок *Bitcoin blockchain* - а, ископан је 3. јануара 2009. године, означавајући почетак *blockchain* технологије какву данас познајемо [3].

Etherium, лансиран 2015. године од стране Виталика Бутерина, увео је паметне уговоре који омогућавају сложеније трансакције и аутоматизацију различитих процеса. Овај развој проширио је примену *blockchain* технологије далеко изван дигиталних валута, омогућавајући креирање децентрализованих апликација [4].

Blockchain технологије су се од свог настанка имплементирале у различитим програмским језицима и окружењима. У својим раним фазама, *blockchain* технологије су се углавном развијале користећи језике као што су *C++* и *Java*, захваљујући њиховој ефикасности и широкој употреби у индустрији. Касније, с појавом паметних уговора, *Solidity* је постао стандард за развој на *Etherium* платформи.

Овај рад се фокусира на имплементацију основних концепата *blockchain* технологије у програмском језику *Rust*, који је познат по својој сигурности, перформансама и могућности превенције грешака при руковању меморијом.

Рад је структуиран X целина

2 Основе *Rust* програмског језика

Rust је савремени програмски језик који је развијен да буде безбедан и брз. Развијен од стране *Mozilla Research*-а, *Rust* је од свог настанка привукао велику пажњу због својих изузетних безбедносних карактеристика и перформанси [5].

2.1 Увод у *Rust* програмски језик

Rust је системски програмски језик, а уместо интерпретираног језика, као што су *JavaScript* или *Ruby*, има компајлер, као што имају *Go*, *C* или *Swift*. Не комбинује активни *runtime*, али обезбеђује језичку ергономију. Све је ово могуће захваљујући компајлеру који спречава грешке било којег типа и осигурава да не дође до проблема у меморији пре него што се покрене апликација [6].

Rust обезбеђује перформансе (нема *runtime*, нити прикупљање "смећа"), безбедност (компајлер осигурава да је све безбедно за меморију, чак и у асинхроним окружењима) и продуктивност (његове уграђене алатке за тестирање, документацију и "менаџер"пакета чине га лаким за израдз и одржавање) [6].

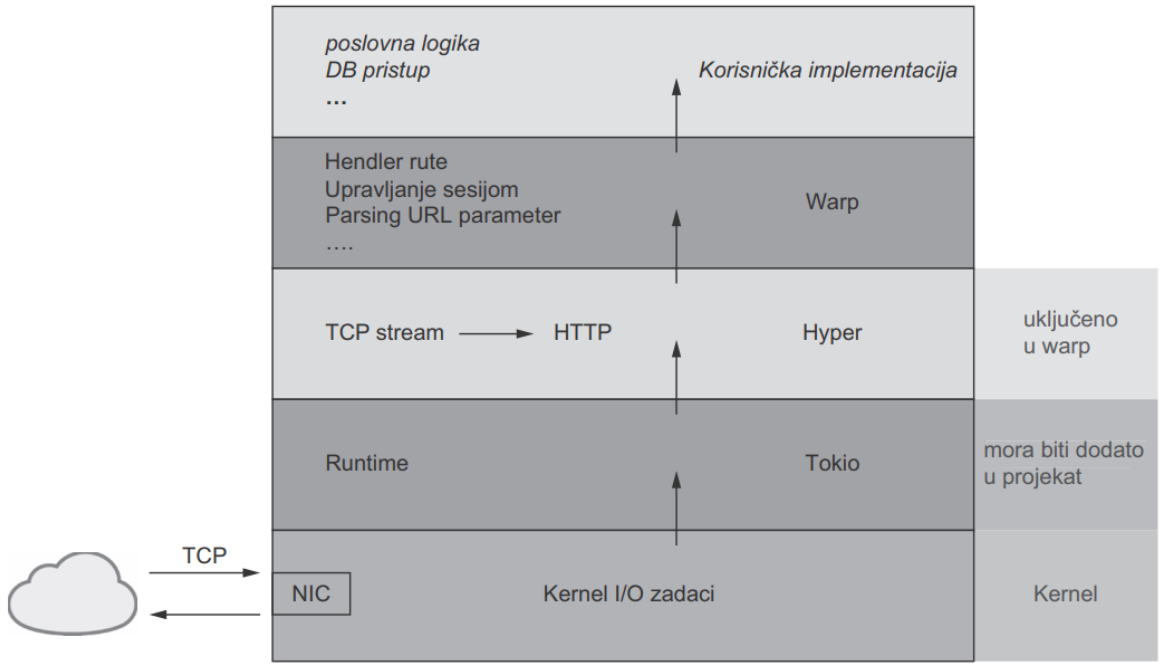
2.2 Зашто *Rust* за *blockchain*?

Када је у питању развој blockchain апликација, *Rust* се истиче као одличан избор из неколико разлога:

1. **Безбедност меморије**: *Rust*-ов систем власништва и провера за време компилације осигуравају да програмери избегну уобичајене грешке у раду са меморијом, што је критично за сигурност *blockchain* система [7].
2. **Перформансе**: *Rust* је дизајниран да буде брз и ефикасан. Његов минималан *overhead* и високо оптимизован компајлер резултирају брзим извршавањем кода, што је важно за обраду великог броја трансакција у реалном времену [7].
3. **Паралелизам и конкурентност**: *Rust* нуди снажну подршку за паралелно и конкурентно програмирање, омогућавајући оптимално коришћење мултијезгарних процесора [7].

Rust нуди низ алата и библиотека које олакшавају развој сложених апликација. Две од најзначајнијих библиотека за развој *blockchain* апликација су *Tokio* и *libp2p*. Ове библиотеке пружају подршку за асинхроне позиве и *peer-to-peer* комуникацију, што је кључно за функционалност и ефикасност *blockchain* система.

Слика 2.1 прииказује технички стек који је укључен у одабир радног оквира. ***Warp*** је довољно мали да се "склони са пута", довољно се користи да се њиме управља активно и има активну заједницу. Заснован је на *Tokio runtime* - у.



Слика 2.1: *Warp* веб радни оквир

libp2p је модуларни мрежни стек који омогућава *peer-to-peer* комуникацију. У контексту *blockchain*-а, *libp2p* се користи за омогућавање комуникације између различитих чворова у мрежи. Ова библиотека је флексибилна и подржава различите протоколе за пренос података, што је чини идеалном за развој децентрализованих апликација.

3 Увод у *blockchain* технологију

Blockchain технологија представља савремен приступ складиштењу и дистрибуцији података. Основни принципи и концепти *blockchain* технологије нуде дубоку промену у начину на који се информације похрањују, проверавају и дистрибуирају путем децентрализоване мреже рачунара.

3.1 Основни принципи и концепти

Blockchain се може дефинисати као дистрибуисана дигитална књига трансакција. Основна идеја је стварање низа блокова који садрже податке. Блкови су криптографски повезани тако да је немогуће мењати податке у претходним блоковима без мењања свих следећих блокова [8].

Кључни елементи *blockchain*-а укључују:

1. **Децентрализација**: Подаци се похрањују и управљају путем мреже чворова уместо централизованог ауторитета, што осигурава транспарентност и отпорност на цензуру.
2. **Дистрибуираност**: Сваки чвор у мрежи садржи комплетан или део *blockchain*-а, омогућујући свима у мрежи да виде исте податке. Ово спречава појединачне тачке квара и повећава отпорност на нападе.
3. **Сигурност**: Криптографски алгоритми осигуравају да је свака промена у *blockchain*-у лако проверљива, а трансакције се потврђују кроз консензус мреже.
4. **Неповратност**: Након што је трансакција забележена у *blockchain*-у, тешко ју је променити или обрисати без сагласности већине чворова у мрежи, чиме се осигурава поверење и интегритет података.

3.2 Поређење са традиционалним базама података

Насупрот традиционалним базама података које су често централизоване и ослањају се на поверење у једну јединицу, *blockchain* нуди неколико кључних разлика:

1. **Централизација у односу на децентрализацију**: Традиционалне базе података често су централизоване под контролом једне организације. *Blockchain* дистрибуише податке широм мреже, елиминишући потребу за централним ауторитетом.
2. **Транспарентност и проверљивост**: *Blockchain* омогућава свим корисницима увид у све трансакције које су се догодиле, што повећава транспарентност и смањује могућност манипулације.

3. **Сигурност и отпорност**: Због своје дистрибуиране природе, *blockchain* је отпорнији на нападе и кварове у поређењу са традиционалним базама података које су осетљиве на појединачне тачке квара.

4. **Ефикасност и трошкови**: Иако *blockchain* може бити спорији у обради података у поређењу са централизованим базама података, његова сигурност и транспарентност могу надмашити трошкове и ризике традиционалних система.

4 Архитектура апликације

Blockchain технологија се састоји од неколико кључних компоненти које омогућавају њено функционисање. Основне јединице података су блокови, који садрже информације о трансакцијама, временским ознакама и криптографским хеш функцијама претходних блокова. Ови блокови су повезани у секвенцијални ланац, познат као *blockchain*, који осигурава неповредивост података.

Дистрибуирана мрежа чворова заједнички одржава и верификује *blockchain*, омогућавајући децентрализацију. Консензус алгоритми, као што су *Proof of Work (PoW)* и *Proof of Stake (PoS)*, омогућавају учесницима мреже да се сложе око валидности нових блокова. Криптографија осигурава сигурност и приватност података унутар *blockchain* -а, користећи хеш функције и дигиталне потписе.

У наредним подсецијама, детаљно ћемо описати сваку од ових компоненти, укључујући процесе као што су PoW и мајнинг, који су кључни за додавање нових блокова у ланац.

4.1 Блокови

Блокови су основне јединице података у *blockchain* технологији. Сваки блок садржи скуп података који су повезани са трансакцијама и другим важним информацијама. У контексту *blockchain*-а, блокови су организовани у ланац, где сваки блок садржи хеш претходног блока, што обезбеђује интегритет и сигурност података. Следећи код приказује структуру блока у Rust програмском језику:

```rust
pub struct Block {
    pub timestamp: DateTime<Utc>,
    pub last_hash: String,
    pub hash: String,
    pub data: Vec<Transaction>,
    pub nonce: u64,
    pub difficulty: u64,
}
```

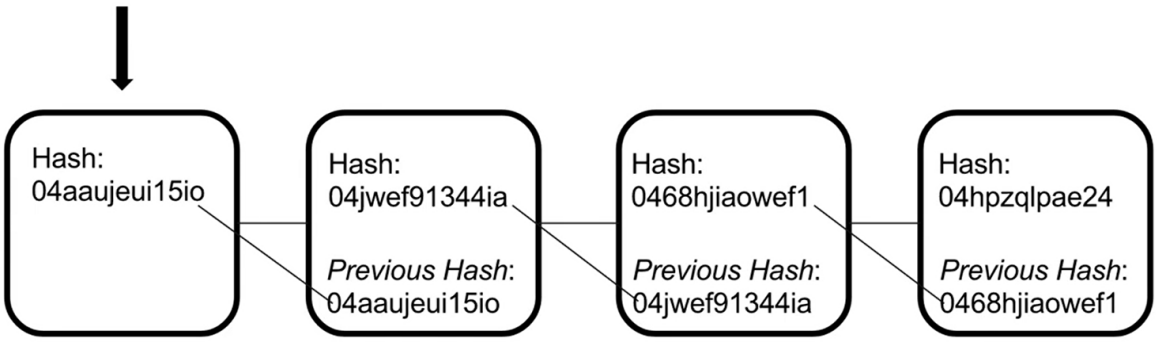
Атрибути блока су:

- **timestamp**: Време када је блок креиран. Овај атрибут омогућава праћење хронологије трансакција у *blockchain*-у.
- **last_hash**: Хеш вредност претходног блока у ланцу. Овај атрибут обезбеђује да сваки блок буде повезан са својим претходником, чиме се осигурава интегритет ланца.

- **hash**: Хеш вредност тренутног блока. Ова вредност се добија применом хеш функције на садржај блока и служи као јединствени идентификатор блока.
- **data**: Податке у блоку, који обично укључују трансакције. У овом случају, то је вектор трансакција *(Vec<Transaction>)*.
- **nonce**: Произвољни број који рудари мењају током процеса рударења како би добили хеш вредност блока која задовољава критеријуме тешкоће.
- **difficulty**: Ниво тежине који одређује колико је сложено пронаћи важећи хеш за блок. Тежина рударења се прилагођава да би се одржала константна брзина креирања блокова у мрежи.

4.1.1 Генесис блок

Генесис блок је први блок у ланцу блокова и служи као темељ целокупног *blockchain* система (Слика 4.1). Он нема претходника и обично је ручно креиран од стране креатора *blockchain*-а. Генесис блок обично садржи посебне параметре и почетне вредности које су специфичне за дат *blockchain*. Његова важност лежи у чињеници да сваки наредни блок у ланцу зависи од њега кроз хеш вредности.



Слика 4.1: Приказ генесис блока у *blockchain*-у

4.1.2 Рударење

Рударење је процес додавања нових блокова у *blockchain*. Рудари користе своју рачунарску снагу да реше комплексне математичке проблеме који су потребни за валидацију нових трансакција и креирање нових блокова. Овај процес захтева значајну количину енергије и ресурса, али је кључан за одржавање безбедности и децентрализације *blockchain* мреже. У процесу рударења, рудари се такмиче да пронађу одговарајући *nonce* који ће произвести хеш вредност која испуњава одређене критеријуме тешкоће.

4.1.3 Хеш функција

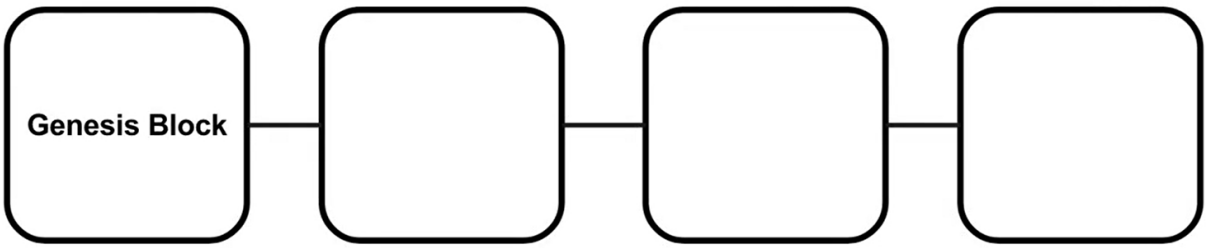
Хеш функција је критичан елемент у *blockchain* технологији, јер обезбеђује сигурност и интегритет података у блоковима. Хеш функција узима улазне податке произвољне дужине и генерише фиксну дужину излазне вредности, која је јединствена за те улазне податке. У контексту *blockchain*-а, хеш функција се користи да повезује сваки блок са претходним блоком, чиме се обезбеђује да свака промена у подацима било ког блока одмах утиче на све наредне блокове, што чини *blockchain* изузетно отпорним на манипулацију.

4.2 Ланац

Blockchain је структура података која се састоји од низа повезаних блокова, где сваки блок садржи хеш претходног блока, чиме се обезбеђује интегритет и сигурност ланца. Следећи код приказује структуру *blockchain*-а у *Rust* програмском језику:

```
pub struct Blockchain {
    pub chain: Vec<Block>,
}
```

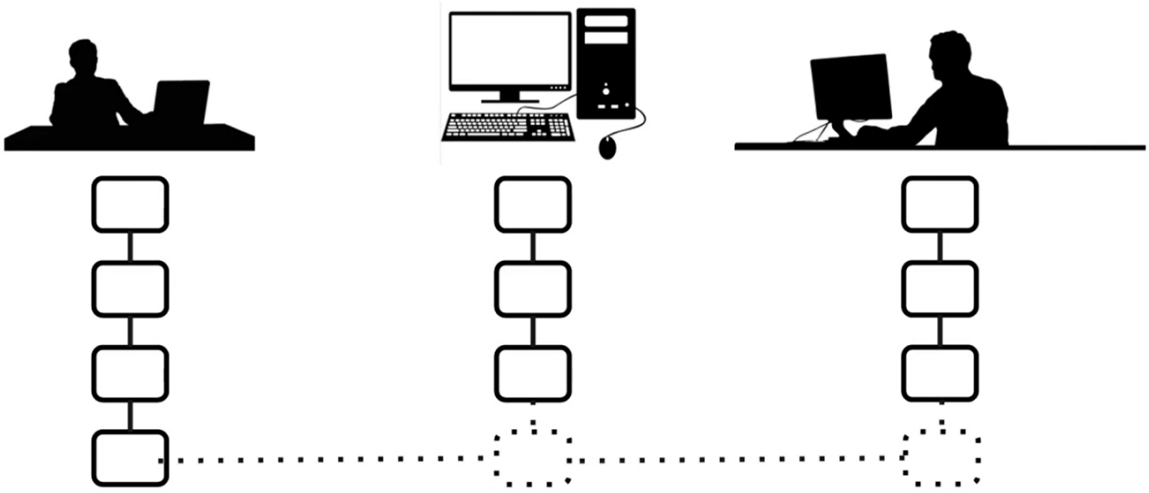
Атрибут ***chain*** је вектор који чува редоследно повезане блокове, формирајући ланац блокова. Слика 4.2 приказује структуру ланца са генесис блоком на почетку.



Слика 4.2: Приказ идеје *blockchain*-а

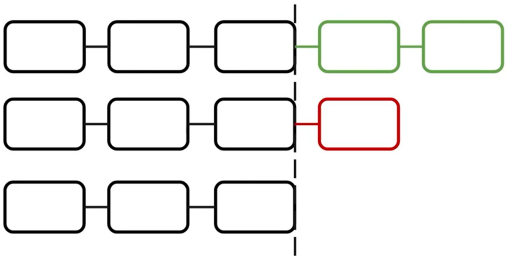
4.2.1 Валидација више ланаца

Идеја овог механизма је да подржи више доприносиоца, при чему ће више доприносиоца додавати блокове у *blockchain*. Сваки рудар ће имати своју верзију истог ланца. Када један рудар дода нови блок у ланац, мораће да пошаље тај нови блок осталим ланцима у систему како би они прихватили ту промену и ажурирали целокупни систем. На тај начин сви добијају ажурирану копију са тим новим блоком, чиме се осигурава да сви ланци буду конзистентни.

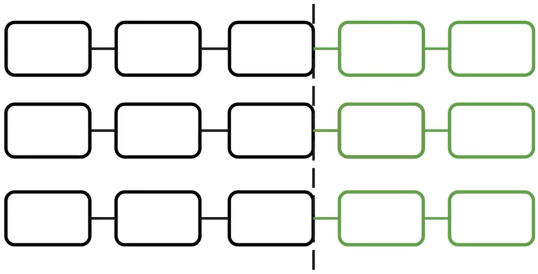


Слика 4.3: Приказ дељења ланаца

Међутим, да би сви рудари прихватили ове нове ланце, мора постојати неки облик валидације који ће осигурати да је нови блок валидан и да треба да буде прихваћен. Главни облик валидације је прихватање дужих ланаца који стигну. На пример, ако сви имају договорени *blockchain* који је већ дуг три блока, и један рудар дода два блока у ланац, док други рудар дода само један блок у исто време, систем ће прихватити дужи ланац. На тај начин се осигурава да договорени ланац за све увек буде онај који садржи највише података.

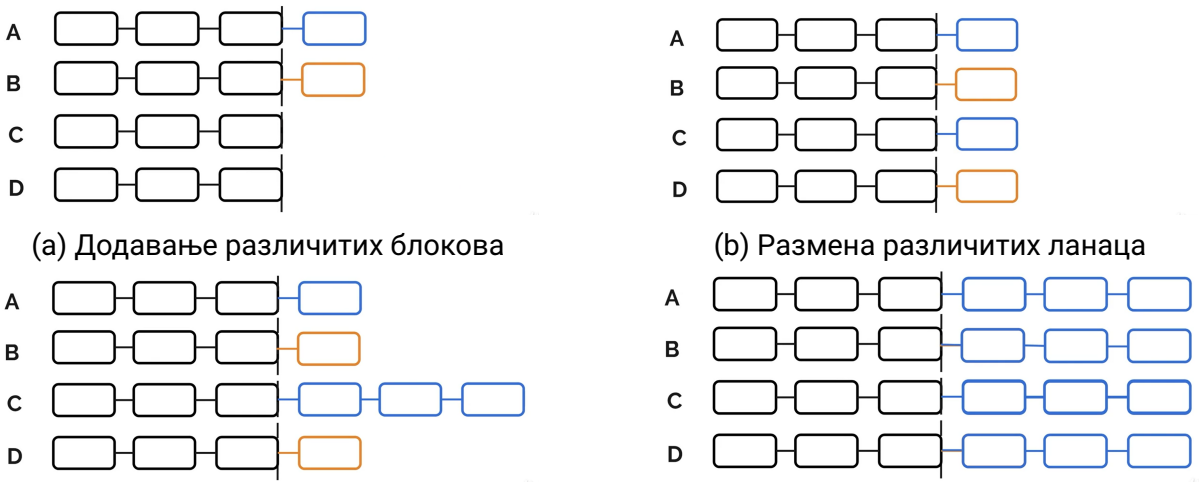


Слика 4.4: Стање пре валидације



Слика 4.5: Стање после валидације

Ово такође решава проблем рачвања у ланцу. На пример, ако две одвојене инстанце *blockchain*-а истовремено произведу један блок на основу претходног блока, настаје рачвање у систему где оба рудара производе блок на основу истог претходног блока (слика 4.6a). Пола рудара ће имати ланац који је произвео рудар А, а друга половина ће имати ланац који је произвео рудар Б (слика 4.6b). Коначно, систем треба да дође до договора о томе који ланац ће прихватити. Ако неко дода неколико блокова на ланац рудара А, тај ланац ће сада бити дужи од свих осталих у систему (слика 4.6c). Сви ће морати да прихвате најдужи ланац, који садржи блок од рудара А, чиме се решава рачвање прихватањем оригиналног блока од рудара А (слика 4.6d).



(a) Додавање различитих блокова

(b) Размена различитих ланаца

(c) Додавање нових блокова

(d) Прихватање најдужег ланца

Слика 4.6: Решавање рачвања

Ово не значи да блокови са ланца рудара Б губе оригиналне податке, јер се блок који није укључен у рачвање може сада додати на крај новоприхваћеног ланца.

Други облик валидације је провера вредности хеша произведених за сваки блок ланца. Сваки *blockchain* има приступ хеш функцији која генерише хеш на основу података блока. Када *blockchain* прими нови ланац, може осигурати да је хеш исправно генерисан тако што ће сам поново генерисати тај хеш. Ако се хешеви не поклапају, вероватно су подаци мењани, и због тога *blockchain* неће прихватити нови ланац.

4.3 *Proof of work*

Proof of work систем је систем који захтева од рудара да троше време на рачунарски рад како би додали блокове у ланац. Ово има корист у одвраћању неискрених учесника од замене *blockchain*-а корумпираним и неважећим подацима.

Да би се објаснило како се неискрени чворови одвраћају, треба узети у обзир да у децентрализованом *blockchain*-у сваки чвор има могућност подношења новог ланца систему. Све док је тај ланац довољно дуг и садржи важеће хеш податке,

почевши од генезис блока, тај ланац ће бити прихваћен од свих чворова у мрежи *blockchain*-а.

Међутим, како би се одвратили неискрени појединци у мрежи од преузимања целог *blockchain*-а са корумпираним ланцем у њихову корист, који заправо има важеће хешеве, систем доказивања рада чини то рачунарски скупим.

Proof of work систем чини да појединачни чворови у *blockchain*-у морају уложити велику количину рачунарског рада како би додали блок. Међутим, за неискрене чворове, доказивање рада чини веома непродуктивним покушај преузимања са потпуно ново генерисаним ланцем.

Proof of work систем функционише на следећи начин. У сваком тренутку постоји ниво тежине у систему *blockchain*-а. У зависности од ове тежине, неко ко покушава да дода нови блок мора пронаћи хеш вредност за тај блок која одговара овој тежини. За ово поклапање, рудари морају пронаћи исти број водећих нула као што је тренутна тежина за генерисани хеш новог блока који треба додати у ланац.

Difficulty = 6

Hash = 000000haxi2910jasdflk

Слика 4.7: Приказ задовољеног хеша

Пронаћи одређени број водећих нула насумично постаје експоненцијално теже како се сама тежина повећава. Дакле, да би се решио доказ рада, рудар ће морати генерисати огроман број хешева како би на крају пронашао један који задовољава тежину. Они ће генерисати нове хешеве за исти блок на основу података блока као и његове прилагођене вредности зване *nonce*.

4.3.1 *Nonce*

Променом вредности нонса рудар генерише нове важеће хешеве за текући блок и његове податке. *Nonce* је термин у криптографији који се односи на вредност која се може користити само једном. Пошто свака јединствена вредност *nonce*-а генерише јединствен хеш, *nonce*-і се могу користити само једном за генерисање тог новог хеша за блок.

Вредност нонса почиње од нуле и повећава се док се не пронађе *nonce* који има одговарајући број водећих нула у складу са подешеном тежином, што генерише хеш. Ова вредност *nonce*-а се затим чува као део блока.

4.3.2 Процес рударења

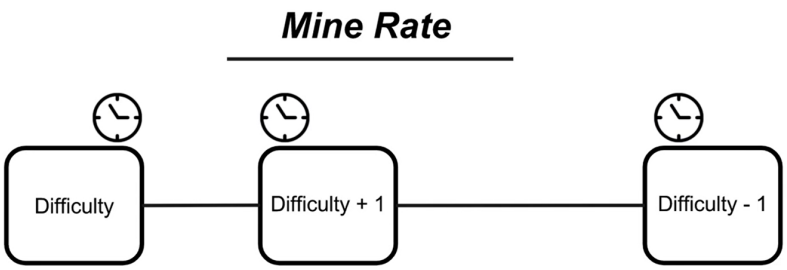
Овај процес генерисања нових хешева са променљивим вредностима захтева доста рачунарског рада. Због тога се акт трошења овог рачунарског рада назива рударење. Када рудар успешно ископа блок, они ће поднети свој блок са пронађеном вредношћу нонса осталим рударима.

Када остали рудари сазнају ову вредност нонса, они могу брзо проверити ваљаност тог решења и додати нови блок у ланац. На тај начин, они не морају поново да раде исти рачунарски рад када приме нови блок за додавање у свој *blockchain* и остану ажурни.

4.3.3 Динамичка тежина додавања блокова

Са вредношћу *nonce*-а у блоку, имплементирали смо *Proof of work* систем, где рудари морају уложити одређену количину рачунарске снаге како би додали блокове у ланац. Међутим, како се у мрежу *blockchain*-а додаје све више учесника, блокови ће се откривати брже, јер ће постојати већа шанса да бар један рудар пронађе валидан хеш при одређеној тежини. Због тога нам је потребан динамичан систем који прилагођава ниво тежине *blockchain*-а како се у систем додаје све више рудара.

Да бисмо то постигли, можемо додати вредност тежине сваком блоку. Поред тога, такође постављамо временску вредност која представља стопу којом желимо да се сваки блок рудари. Механизам за прилагођавање тежине функционисаће на следећи начин: проверићемо временски жиг ново изрудареног блока и упоредити га са временским жигом претходно изрудареног блока. Ако је разлика између оба временска жига мања од постављене стопе рударења, знамо да је блок изрударен сувише брзо, те ће се тежина повећати за 1. Слично томе, ако је разлика у временским жиговима између нашег најновијег блока и блока који је претходио већа од стопе рударења, знамо да је блок изрударен сувише споро, те ће се тежина смањити за 1. Са овим системом, увек ћемо прилагођавати тежину за сваки блок и требао би имати систем који рудари блокове у одређеним интервалима.



Слика 4.8: Динамичка промена тежине рударења блока

4.4 Новчаник и трансакције

Када проширујемо *blockchain* да укључује криптовалуту, уводимо неколико нових концепата.

4.4.1 Новчаник

Дигитални новчаник је повезан са кључевима и трансакцијама. Следећи код приказује структуру новчаника у *Rust* програмском језику:

```
pub struct Wallet {
    pub balance: u64,
    pub signing_key: SigningKey<Secp256k1>,
    pub verifying_key: VerifyingKey<Secp256k1>,
    pub public_key: String,
}
```

Атрибути новчаника су:

- **balance**: Стање представља укупну количину валуте која припада појединцу у криптовалутном *blockchain*-у.
- **signing_key**: Приватни кључ користи се за генерисање јединствених дигиталних потписа за сваку трансакцију
- **verifying_key**: Јавни кључ омогућава другима да верификују те потписе.
- **public_key**: Јавна адреса новчаника је иста као и његов јавни кључ, и омогућава другима да шаљу валуту на тај новчаник.

4.4.2 Креирање кључева

Сваки појединац који жели да обави трансакцију у *blockchain*-у мора потписати податке о трансакцији својим приватним кључем. Потпис се базира на два кључа: приватном, који само корисник има, и јавном, који је свима видљив. Кључеви су јединствени низови бројева који омогућавају кориснику да потпише податке стварањем шифроване хеш вредности. Ова вредност се генерише комбиновањем података о трансакцији и приватног кључа. Свака промена у оригиналним подацима резултује новим потписом.

4.4.3 Трансакције

Трансакције представљају размену валуте између два појединца у криптовалутном систему. Следећи код приказује структуру трансакције у *Rust* програмском језику:

```
pub struct Transaction {
    pub id: TransactionId,
    pub input: Option<TransactionInput>,
    pub outputs: Vec<TransactionOutput>,
}

pub struct TransactionId(pub Uuid);
```

Свака трансакција има два дела: улаз и излаз.

Улаз трансакције је кључна компонента која описује детаље о извору средстава коришћених у трансакцији. Следећи код приказује структуру улаза трансакције у *Rust* програмском језику:

```
pub struct TransactionInput {
    pub timestamp: DateTime<Utc>,
    pub amount: u64,
    pub address: String,
    pub signature: String,
}
```

Атрибути улаза трансакције су:

- **timestamp**: Ово поље представља тачан тренутак када је трансакција креирана. Време је обично записано у *UTC* формату, што омогућава прецизно праћење и синхронизацију у међународним оквирима.
- **amount**: Износ представља количину криптовалуте која се преноси са једне адресе на другу.
- **address**: Адреса представља адресу новчаника са ког средства долазе.
- **signature**: Потпис обезбеђује интегритет и аутентичност трансакције

Излаз трансакције писује детаље о одредишту средстава која се преносе у трансакцији. Свака трансакција може имати један или више излаза, што омогућава податке о томе где се средства шаљу. У наставку је пример структуре излаза трансакције у програмском језику *Rust*:

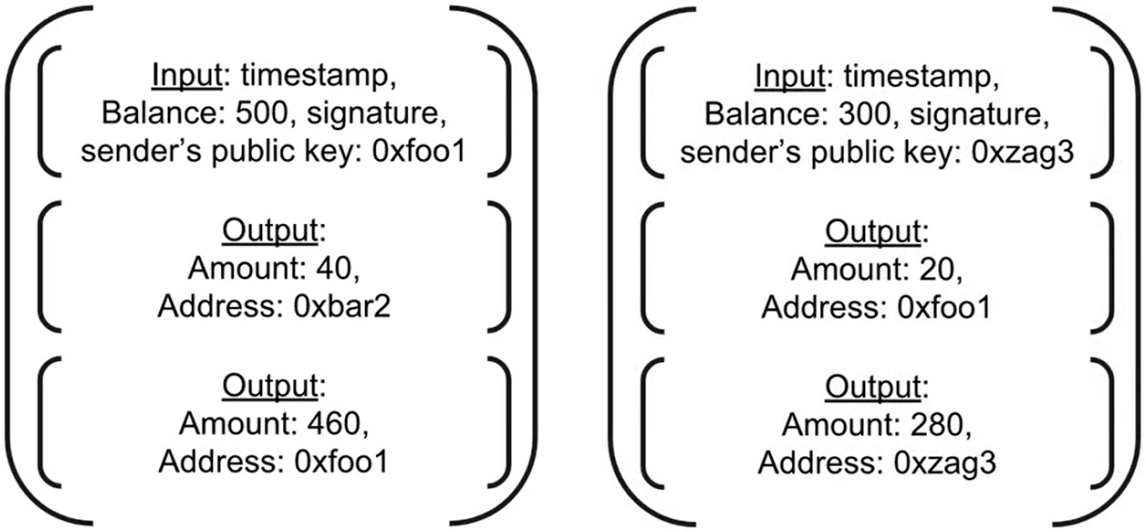
```rust
pub struct TransactionOutput {
    pub amount: u64,
    pub address: String,
}
```

Атрибути излаза трансакције су:

- **amount**: Износ представља количину криптовалуте која се преноси на одређену адресу.
- **address**: Ова адреса одређује где се средства шаљу.

Пример трансакција приказан на слици 4.9 илуструје две различите трансакције у *blockchain* мрежи. У првој трансакцији, улаз садржи податке као што су временски печат, баланс од 500 јединица, дигитални потпис и јавни кључ пошиљаоца (0xfoo1). Ова трансакција има два излаза: први излаз преноси 40 јединица на адресу 0xbar2, а други излаз приказује колико ће јединица валуте остати на адреси пошиљаоца (0xfoo1) после извршене трансакције. Ова структура показује како се улазни баланс дели између различитих адреса, где један део иде новом кориснику, а преостали део се враћа пошиљаоцу као кусур.

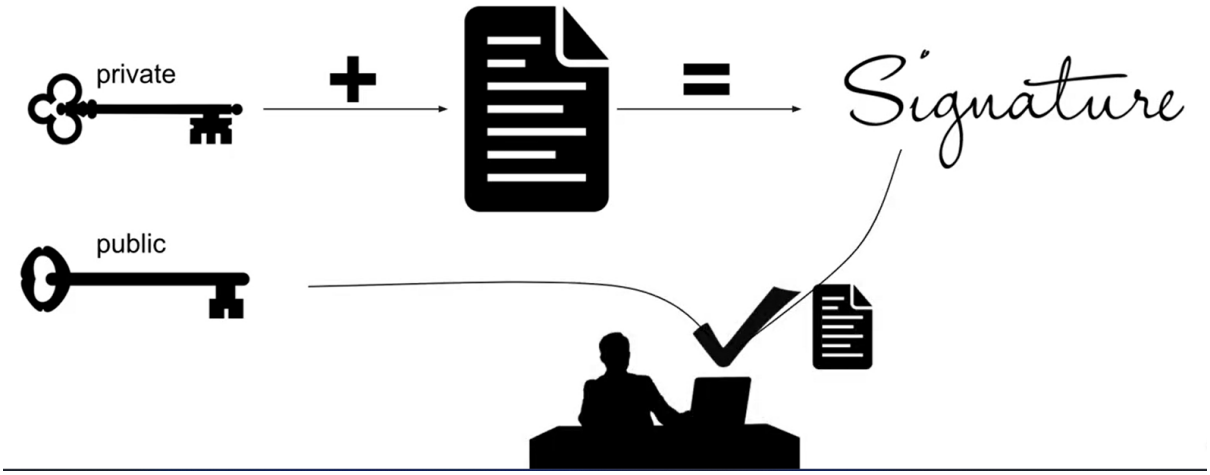
У другој трансакцији, улаз укључује временски печат, баланс од 300 јединица, дигитални потпис и јавни кључ пошиљаоца (0xzag3). Ова трансакција такође има два излаза: први излаз преноси 20 јединица на адресу 0xfoo1, а други излаз преноси преосталих 280 јединица назад на адресу пошиљаоца (0xzag3). Оваква структура обезбеђује да се трансакције у *blockchain* мрежи извршавају на транспарентан и безбедан начин, чувајући интегритет и прецизност финансијских токова.



Слика 4.9: Пример трансакција

4.4.4 Потписивање трансакција

Свака трансакција мора бити потписана дигиталним потписом корисника. Ово осигурава да подаци нису мењани након потписивања и да потпис одговара јавном кључу пошиљаоца.



Слика 4.10: Дигитални потписи

Слика 4.10 илуструје рад приватног и јавног кључа. Приватни кључ је познат само кориснику и служи за стварање потписа, док је јавни кључ доступан свима и користи се за верификацију тог потписа. Ови кључеви су јединствени низови бројева. Корисник потписује податке стварањем потписа који је шифровани хеш вредност. Хеш вредност се генерише комбинацијом података о трансакцији и приватног кључа корисника. Пошто је хеш вредност заснована на оригиналним подацима, свака промена у оригиналним подацима, чак и један знак, резултоваће новом хеш вредношћу и новим потписом.

Сада, када подаци имају потпис, свако може користити јавни кључ корисника који је потписао податке да би верификовао тај потпис. Јавни кључ се користи за дешифровање потписа и читање оригиналних података. Ако дешифровани подаци не одговарају оригиналним подацима из трансакције, знамо да је или оригинални податак измењен након потписивања или је потпис генерисан приватним кључем који не одговара презентованом јавном кључу. У оба случаја, трансакција је неважећа.

4.4.5 Ажурирање трансакција

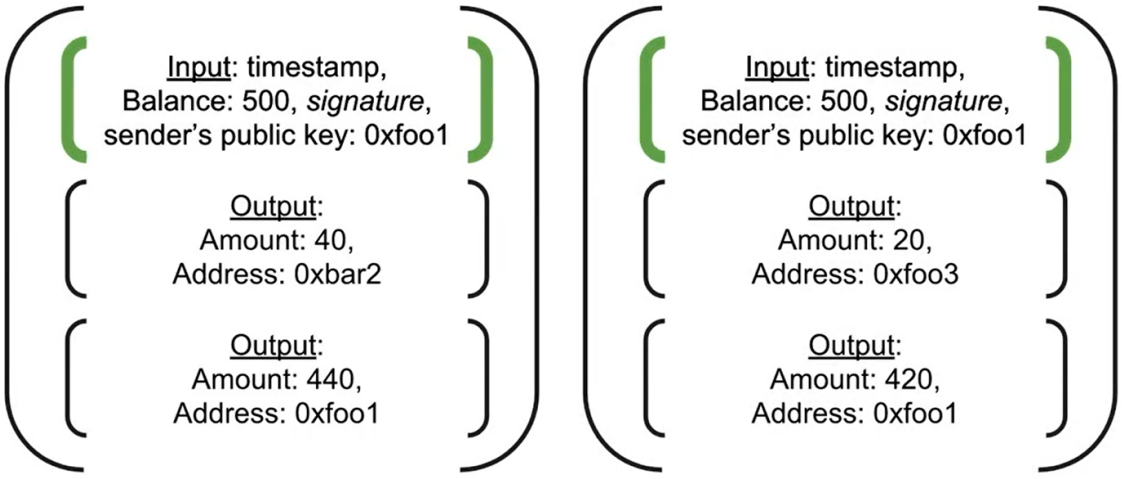
Оптимизација трансакција у *blockchain* систему се може спровести кроз ажурирање постојећих трансакција уместо креирања нових. Овај приступ омогућава бољу ефикасност у управљању трансакцијама и смањује потребу за генерисањем нових објеката за сваку нову трансакцију.

Традиционално, свака трансакција у *blockchain* систему садржи улазне и излазне вредности. Улазна вредност обухвата информације о пошиљаоцу, као што

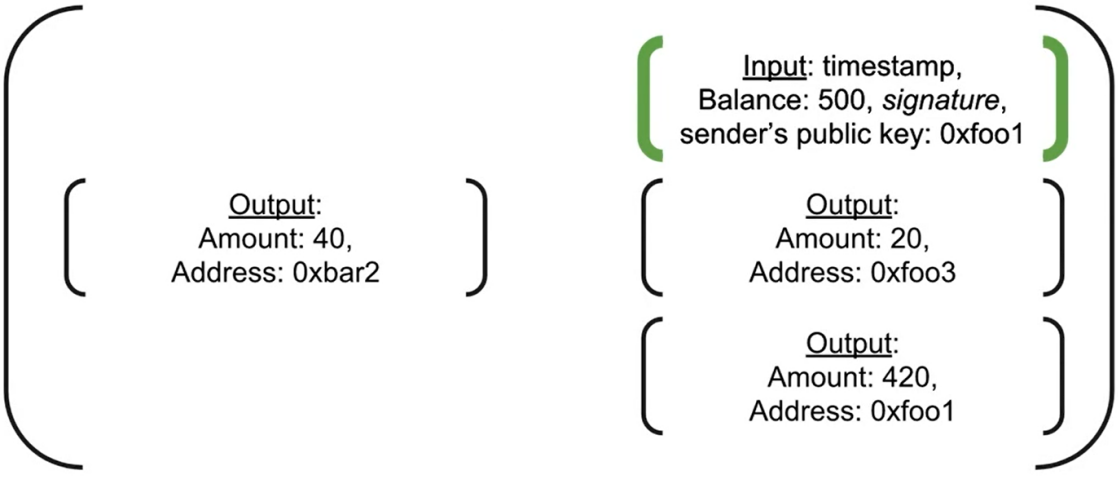
су временска ознака, почетно стање и дигитални потпис. Излазне вредности укључују износ који се шаље и адресу примаоца. Међутим, када појединац жели да обави више трансакција у кратком временском периоду, креирање потпуно нових трансакција може бити неефикасно.

Да бисмо оптимизовали овај процес, предлажемо ажурирање постојећих трансакција са новим излазима који представљају нове размене. На тај начин, један објекат трансакције може садржати више излаза који омогућавају слање валуте на више прималаца. Приликом ажурирања трансакције, потребно је поново потписати трансакцију како би се укључили нови подаци и обезбедила верификација валидности.

Конкретно, сваки нови излаз ће одражавати нову размену валуте, док ће укупни износ свих излаза бити ажуриран да би се одразила нова стања. Овај метод омогућава једноставнију и бржу размену валуте, смањујући потребу за креирањем и управљањем великим бројем трансакцијских објеката.



Слика 4.11: Пример две трансакције са истим улазом



Слика 4.12: Пример ажуриране трансакције

5 Литература

[1] Zheng, Zibin, Shaoan Xie, Hong Ning Dai, Xiangping Chen, i Huaimin Wang: *An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends*. strana 1, Jun 2017.

[2] Sherman, Alan, Farid Javani, Habin Zhang, i Enis Golaszewski: *On the Origins and Variations of Blockchain Technologies*. University of Maryland, Baltimore County (UMBC) Baltimore, Maryland 21250, Octobar 2018.

[3] Nakamoto, Satoshi: *Bitcoin: A Peer-to-Peer Electronic Cash System*. Cryptography Mailing list at https://metzdowd.com, Mart 2009.

[4] Sixt, Elfriede: *Ethereum*, strane 189–194. Januar 2017, ISBN 978-3-658-02843-5.

[5] *Rust (programming language)*. `https://en.wikipedia.org/wiki/Rust_(programming_language)`. прегледано 16. јул 2024.

[6] Gruber, Bastian: *Rust Web Development*. Manning, 2023, ISBN 978-1617299001.

[7] *The Rust Reference*. `https://doc.rust-lang.org/reference`. прегледано 16. јул 2024.

[8] Aggarwal, Shubhani i Neeraj Kumar: *Chapter Seven - Basics of blockchain*. U: Aggarwal, Shubhani, Neeraj Kumar, i Pethuru Raj (urednici): *The Blockchain Technology for Secure and Smart Applications across Industry Verticals*, tom 121 iz *Advances in Computers*, strane 129–146. Elsevier, 2021. `https://www.sciencedirect.com/science/article/pii/S0065245820300620`.

6 Подаци о кандидату

Кандидат Бојан Мијановић је рођен 2002. године у Зрењанину. Завршио је средњу школу у Зрењанину, 2020. године као ђак генерације. Факултет Техничких Наука у Новом Саду је уписао 2020. године. Испунио је све обавезе и положио је све испите предвиђеним студијским програмом са просечном оценом од 9.75.