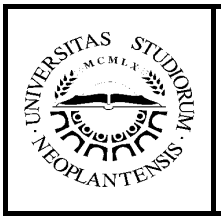


УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА

УНИВЕРЗИТЕТ У НОВОМ САД
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
НОВИ САД
Департман за рачунарство и аутоматику
Одсек за рачунарску технику и рачунарске комуникације

МАСТЕР РАД

Кандидат: **Тамара Купрешанин**
Број индекса: **E2102/2020**

Тема рада: **Једно решење ланца-блокова**

Ментор рада: **проф. др Мирослав Поповић**

Нови Сад, јун, 2022

УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, **РБР**:			
Идентификациони број, **ИБР**:			
Тип документације, **ТД**:		Монографска документација	
Тип записа, **ТЗ**:		Текстуални штампани материјал	
Врста рада, **ВР**:		Мастер рад	
Аутор, **АУ**:		Тамара Купрешанин	
Ментор, **МН**:		проф. др Мирослав Поповић	
Наслов рада, **НР**:		Једно решење ланца-блокова	
Језик публикације, **ЈП**:		Српски / латиница	
Језик извода, **ЈИ**:		Српски	
Земља публиковања, **ЗП**:		Република Србија	
Уже географско подручје, **УГП**:		Војводина	
Година, **ГО**:		2022	
Издавач, **ИЗ**:		Ауторски репринт	
Место и адреса, **МА**:		Нови Сад; трг Доситеја Обрадовића 6	
Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога)		7/26/14/5/17/0/0	
Научна област, **НО**:		Електротехника и рачунарство	
Научна дисциплина, **НД**:		Рачунарска техника	
Предметна одредница/Кqучне речи, ПО:		ланац-блокова, хеш функција, рударење, дигитално потписивање, интернет ствари	
УДК			
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад	
Важна напомена, **ВН**:			
Извод, **ИЗ**:		Ланац-блокова у последње време привлачи све већу пажњу широм света због свог потенцијала да обезбеди сигурност и верификацију за различите врсте података помоћу децентрализоване мреже која се не може изменити. Иако је ова технологија првобитно служила за криптовалуте, временом је нашла примену у различитим областима, од здравства, транспорта до уметности. У овом раду је представљена имплементација ланца-блокова, у Python-у, која омогућава стварање и верификацију трансакција, њихово додавање у блокове, који се затим међусобно дигитално повезују. Експериментални резултати, који су представљени, показују начин на који пораст броја чворова у мрежи утиче на пораст броја додатих трансакција по секунди у ланац-блокова.	
Датум прихватања теме, **ДП**:			
Датум одбране, **ДО**:			
Чланови комисије, **КО**:	Председник:	проф. др Силвиа Гилезан	
	Члан:	доц. др Миодраг Ђукић	Потпис ментора
	Члан, ментор:	проф. др Мирослав Поповић	

UNIVERSITY OF NOVI SAD ● **FACULTY OF TECHNICAL SCIENCES**
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Accession number, **ANO**:			
Identification number, **INO**:			
Document type, **DT**:		Monographic publication	
Type of record, **TR**:		Textual printed material	
Contents code, **CC**:		Master Thesis	
Author, **AU**:		Tamara Kuprešanin	
Mentor, **MN**:		PhD Miroslav Popović	
Title, **TI**:		One solution of blockchain	
Language of text, **LT**:		Serbian	
Language of abstract, **LA**:		Serbian	
Country of publication, **CP**:		Republic of Serbia	
Locality of publication, **LP**:		Vojvodina	
Publication year, **PY**:		2022	
Publisher, **PB**:		Author's reprint	
Publication place, **PP**:		Novi Sad, Dositeja Obradovica sq. 6	
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		7/26/14/5/17/0/0	
Scientific field, **SF**:		Electrical Engineering	
Scientific discipline, **SD**:		Computer Engineering, Engineering of Computer Based Systems	
Subject/Key words, **S/KW**:		blockchain, hash function, mining, digital signiture, internet of things	
UC			
Holding data, **HD**:		The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, **N**:			
Abstract, **AB**:		Blockchain has recently drawn more and more attention around the world because of its potential to provide security and verification for different types of data using a decentralized network that cannot be changed. Although this technology was originally used for cryptocurrencies, eventually it has found application in various areas, from healthcare to transportation and art. This paper presents our implementation of blockchain, in Python, which enables the creation and verification of transactions, their addition to blocks, which are then digitally interconnected. The experimental results, which are presented, show the way in which the increase in the number of network nodes affects the increase in the number of added transactions in the blockchain per second.	
Accepted by the Scientific Board on, **ASB**:			
Defended on, **DE**:			
Defended Board, **DB**:	President:	PhD Silvia Gilezan	
	Member:	PhD Miodrag Đukić	Menthor's sign
	Member, Mentor:	PhD Miroslav Popović	

Захвалност

Дугујем захвалност ментору проф. др Мирославу Поповићу, на стручним саветима, помоћи и стрпљењу у току израде овог рада.

Такође посебну захвалност дугујем породици и пријатељима на несебичној подршци у току целих студија.

САДРЖАЈ

Списак слика

Списак табела

Скраћенице

P2P - *Peer-to-Peer*, мрежа равноправних чворова

HTTP - *HyperText Transfer Protocol*, протокол

TCP - *Transmission Control Protocol*, трансмисиони контролни протокол

IP - *Internet Protocol*, интернет протокол

API - *Application Programming Interface*, програмски интерфејс апликације

UML - *Unified Modeling Language*, обједињени језик за моделовање

SHA - *Secure Hash Algorithm*, сигурносни хеш алгоритам

RSA - *Rivest–Shamir–Adleman*

LAN - *Local Area Network*, локална рачунарска мрежа

UTP - *Unshielded Twisted Pair*,

CPU - *Central Processing Unit*, централна процесорска јединица

RAM - R*andom Access Memory*, меморија са случајним приступом

HDD - *Hard Disk Drive*, хард-диск

MB - *MotherBoard*, матична плоча

OS - *Operating System*, оперативни систем

1. Увод

Ланац-блокова у последње време привлачи све већу пажњу широм света због свог потенцијала да обезбеди сигурност и верификацију за различите врсте података помоћу децентрализоване мреже при чему се претходно унети подаци накнадно не могу изменити. Задатак овог рада је пројектовање, имплементација и експериментална евалуација једног система на бази ланца-блокова трансакција (енг. blockchain) у програмском језику Python. Развијено решење архитектуре омогућава стварање и верификацију трансакција, њихово додавање у блокове и њихово међусобно дигитално повезивање у ланац-блокова трансакција. Циљ експерименталне евалуације развијеног решења је да се измери пропусност система, која је дефинисана као број трансакција у секунди, у зависности од броја чворова у мрежи.

Овај рад представља проширену верзију почетног рада Даниела ван Флајмен-а [1]. У свом раду Флајмен је омогућио стварање једноставног ланца-блокова, додавање иницијалног празног блока, стварање нових трансакција и њихово додавање у блокове, израчунавање хеш функција блокова и додавање блокова у ланац, као решење консензуса проналази се најдужи ланца од понуђених. За комуникацију са ланцем користио је HTTP захтеве, *get* и *post*. Касније је своју идеју проширио и изменио у књизи [2].

У овом раду уведен је серверски процес руковалац који омогућава пријављивање унапред одређеног броја чворова, чување свих адреса и њихово прослеђивање пријављеним чворовима. Комуникација чворова се одвија на нивоу транспортног слоја помоћу TCP протокола. Чворови се повезују у потпуно повезану мрежу. Сваки чвор осим, горе наведених, основних функционалности има додате могућности: генерисање приватног и јавног кључа, дигиталног потписивања трансакција помоћу приватног кључа, верификацију пристиглих трансакција на основу јавног кључа пошиљаоца пре њиховог хронолошког

додавања у блок, рударење случајног броја (енг. nonce) и валидацију случајног броја. За генерисање кључева имплементиран је RSA алгоритам, који користи Милер-Рабинов тест за потребе проналажења великих простих бројева [3,4]. Децентрализација и сигурност података обезбеђене су помоћу криптографске хеш функције, дигиталних потписа и механизма консензуса.

Остатак рада организован је на следећи начин. У другом поглављу дат је преглед теоријских основа које су послужиле као темељ за реализацију решења.

У трећем поглављу представљена је архитектура развијеног решења и понашање.

Четврто поглавље садржи опис реализованих класа и програмских функција, док пето поглавље садржи опис испитивања.

Завршно поглавље садржи закључак где су описане главне предности и недостаци конкретне имплементације и правци даљег развоја.

2. Теоријске основе

У овом поглављу описани су теоријски појмови неопходни за разумевање даљег рада. У првом делу поглавља описани су повезани радови, као и опис првог ланца-блокова Bitcoin-а. У наставку је дат опис кориштених апстракција из Python модула multiprocessing.

2.1 Ланац-блокова

Ланац-блокова представља базу података или јавни записник свих трансакција и дигиталних радњи које су извршене и дељене између учесника. Свака трансакција у овој бази мора бити одобрена од већине учесника у систему. Једном унешена информација не може никада бити избрисана, па самим тим, ланац-блокова садржи поуздан и верификован запис сваке трансакције која је икада настала. Свака трансакција, уговор, процес и извршени задатак се дигитално потписује ради верификације и провере исправности. Једна од кључних карактеристика ланца-блокова је да се ова дигитална књига не чува на једном месту, већ на свим рачунарима који чине потпуно повезану мрежу.

2.1.1 Преглед повезаних радова

Стјуарт Хејбр и В. Скот Сторнета су још 1991. осмислили концепт који је касније послужио као инспирација за стварање технологије ланца-блокова [5]. У свом раду представили су сигуран начин за временско означавање дигиталних докумената употребом криптографске хеш функције и међусобно повезивање докумената у ланац. Појединац или група под називом Сатоши Накамото је 2008. у јавном раду (енг. white paper) објавио основне концепте технологије ланца-блокова и представио прву употребу ове технологије створивши прву криптовалуту Bitcoin [6]. Та иновативна технологија базирала се на кључним карактеристикама као што су децентрализација, постојаност, анонимност и сигурност.

Иако је Bitcoin најпознатија апликација ланца-блокова, ова технологија може се применити у различитим апликацијама изван криптовалута, као што су паметни уговори, јавне услуге, интернет ствари и службе безбедности. Тако на пример, С. Симић, М. Марковић, С. Гостојић су у свом раду представили примену паметних уговора у хотелијерству [7]. Представили су паметни уговор о расподели смештајних капацитета који омогућава представницима хотела и туристичким агенцијама да преговарају, закључују и комуницирају. Док Ванг, Су, Занг дају занимљиву могућност примене ове технологије у интернету ствари (енг. IoT) за транспорт енергије [8]. Они су искористили безбедност ланца-блокова за бежични пренос обновљиве енергије на великом географском подручију помоћу електричних возила. Ал-Јароди, Мохамед представили су предности и изазове коришћења ланца-блокова за финансијске, здравствене, енергетске, телекомуникационе и забавне апликације [9]. Сличну студију су спровели и приказали у свом раду Јаоуде, Саде, где су дали систематски преглед литературе о технологији ланца-блокова, њене кључне карактеристике, као и области примене [10]. Док је Холбл спровео систематски преглед примене у здравству [11]. Описао је најсавременија истраживања ланца-блокова у здравству, истичући изазове и могуће правце даље примене ове технологије у области.

Технологија интернета ствари омогућава повезивање са светом преко мреже сензора и уређаја који комуницирају и размењују информације о подацима. Примена ланац-блокова у интернету ствари омогућава сигурност података, поузданост и ефикасно управљање. У раду Д. Миноли и Б. Окиогросо приказали су окружења интернет ствари у којима ланци-блокова играју важну улогу, у комбинацији са другим сигурносним механизмима [12]. Док су А. Дори, С. С. Канхере, Р. Јурдак у свом раду представили поједностављену архитектуру засновану на ланцу-блокова за примену у интернету ствари [13]. Представљена архитектура елиминише недостатке класичног ланца-блокова, попут велике пропусности и кашњења, уз задржавање већине његових предности у погледу безбедности и приватности.

Што се тиче безбедносних претњи и претњи приватности, Ли, Јианг, Чен, Луо и Вен су истражили различите претње као што су дупло плаћање (енг. double spending), напади од 51%, безбедност приватног кључа, злонамерне активности и приватност података у ланцу-блокова [14]. Такође дали су преглед решења за побољшање безбедности за развој различитих система.

2.1.2 Bitcoin

Bitcoin је прва децентрализована криптовалуту на свету, највећа је те врсте по тржишној вредности, и представља прву конкретну примену технологије ланца-блокова. Не постоји никакав централни ентитет коме би корисници морали да верују, сви учесници у

мрежи су равноправни, па се тако постиже децентрализација. Криптовалута значи да се користи криптографија да се обезбеди сигуран пренос и верификација дигиталних трансакција. Први пут се као идеја појављује 2008, а као програм отвореног кода у јануару 2009, када је и почела емисија bitcoin-а са генесис блоком од 50 токена [6]. У питању је мрежа равноправних чворова (P2P), која је децентрализована, дистрибуирана, са анонимним плаћањем, и која функционише помоћу сложеног алгоритма, а уједно и валута коју та мрежа користи.

Bitcoin се осим рударењем може купити новцем или добити у замену за робу и услуге. Bitcoin се уз незнатну накнаду може послати дигиталним путем помоћу новчаника. Нови bitcoin-и су награда за обраду трансакција корисницима укљученим у активности одржавања мреже ланца-блокова.

Појавом bitcoin-а решен је проблем дуплог плаћања увођењем дистрибуиране књиге, која садржи информације о свим трансакцијама које су извршене и дистрибуирана је међу свим корисницима мреже. Свака нова трансакција се проверава у односу на ланац-блокова и тако се избегава могућност дуплог плаћања. Трансакције су заштићене и верификоване употребом крипторафије са јавним кључем. Сваком кориснику мреже додељују се два кључа: приватни, који служи за очување анонимности корисника и јавни, који служи рударима за верификацију. Рудари су равноправни чворови који гарантују сигурност система, вршећи верификацију трансакција, стварања блокова и додавања блокова у ланац. Верификоване трансакције се додају у блок и израчунава се хеш код помоћу криптографске функције. Једном када је израчунат хеш код и блок додат у ланац блокова, немогуће је изменити трансакције.

Давањем награда рударима за решавање доказа о раду избегава се и проблем византијских генерала. Рудари имају више користи од поштеног извршавања посла, него да пробају да додају неисправне трансакције и блокове. Да би успели да наруше сигурност ланца-блокова потребно је да непоштени рудари поседују више од пола укупне рачунарске моћи система.

Свако може да инсталира овај програм отвореног кода и постане део bitcoin мреже, што представља разлог популарности ове криптовалуте.

2.1.3 Врсте ланца-блокова

Јавни ланац-блокова нема апсолутно никакво ограничење приступа. Свако ко има приступ интернет мрежи може му приступити, слати трансакције па тако може постати и валидатор, односно може учествовати у одлучивању на основу консензуса. Оваква врста ланца-блокова је отпорна на потенцијалне нападе. Тренутно једни од најпознатијих јавних

ланаца-блокова су Bitcoin, Ethereum итд. Главни недостатак је ограниченост капацитета ове врсте ланца-блокова.

Приватни ланац-блокова јављају се ограничења за учеснике и валидаторе. Сваки учесник овакве мреже мора бити одобрен од стране администратора. За стварање трансакција и стварање блокова је потребна дозвола. Капацитет оваквог ланца-блокова може бити изузетно велики. Недостатак је губљење децентрализације због превелике моћи администратора.

Хибридни ланац-блокова је комбинација различитих карактеристика, како јавних тако и приватних. Омогућава да се изабере које информације ће остати приватне, а које ће се објавити. Даља децентрализација у односу на примарно централизоване приватне блокове може се постићи на различите начине. Уместо да се трансакције чувају у сопственој мрежи приватних чворова, хеш може да се постави на потпуно децентрализоване ланце-блокова.

2.2 Python модул multiprocessing

Multiprocessing модул омогућава паралелно покретање више процеса из истог програма користећи API сличан модулу за обраду нити. Multiprocessing нуди и локални и удаљени паралелизам, избегавајући закључавање (енг. Global Interpreter Lock) коришћењем подпроцеса уместо нити. Овај модул омогућава да се у потпуности искористи више језгара на датој машини. У овом делу су описане основне апстракције из multiprocessing модула које су коришћене у имплементацији, као и кратки примери њихове употребе.

2.2.1 Класа Process

Класа Process је апстракција која омогућава стварање више процеса и њихово паралелно извршавање. Објекат процес представља активност која се извршава на засебном језгру. За разлику од нити процеси не деле меморију. Две важне методе су start() и join().

Прво је потребно написати функцију коју ће процес извршавати. Затим се мора инстрацирати објекат процеса помоћу конструктора:

*Process(group=None, target=None, name=None, args=(), kwargs={}, *, daemon=None)*

Ако створимо процесни објекат, ништа се неће десити док му не кажемо да започне извршавање преко функције start(). *Target* представља функцију коју процес извршава, *args* су аргументи који се прослеђују фунцији target. Могуће је дати име процесу помоћу аргумента конструктора *name*. Након тога се чека завршетак процеса позивом фунције join().

Једноставан пример коришћења класе Process дат је на слици 2.1. Главни процес main прави, покреће и чека процес потомак (помоћу конструктора Process и метода *start* и *join*).

Процес потомак извршава функцију *func*. Процес main прослеђује параметар процесу потомак ('wold'). Сваки процес се извршава на свом језгру.

```
from multiprocessing import Process

def func(name):
    print('hello ', name)

if __name__=='__main__':
    p = Process(target = func, args=('wold ',))
    p.start()
    p.join()
```

Резултат извршавања

```
hello world
```

Слика 2.1 Пример коришћења класе Process

2.2.2 Класа BaseManager

Руковаоци пружају начин за одржавање података који се могу делити између различитих процеса, укључујући дељење преко мреже између процеса који се покрећу на различитим машинама. Објекат руковаоца контролише серверски процес који управља дељеним објектима.

Прво је потребно створити објекат руковаоца помоћу конструктора BaseManager (параметри су: пар (IP адреса, порт) и аутентификациони кључ), затим прави објекат сервер помоћу методе *get_server()* који је под контролом руковаоца и на крају покреће серверски процес руковаоца. Аутентификациони кључ се користи за проверу исправности долазних веза са серверским процесом руковаоца.

За повезивање са серверским процесом руковаоца користи се функција *connect()*.

```
manager = BaseManager(address = ('127.0.0.1', 5000), authkey = b'abc')
server = manager.get_server()
server.serve_forever()
```

```
m = BaseManager(address = ('127.0.0.1', 5000), authkey = b'abc')
m.connect()
```

Слика 2.2 Пример коришћења класе BaseManager

2.2.3 Класе Client i Listener

Апстракција клијент-слушалац одговара класичној TCP/IP клијент-сервер архитектури, са тим што додатно обезбеђује заштићену везу клијент-слушалац на бази аутентификације.

Објекат клијента покушава да успостави везу са слушаоцем, који је на адреси која је прослеђена. Слушалац помоћу конструктора прави објекат, прослеђени параметри су адреса слушаоца и аутенификациони кључ на основу ког се прихватају везе са клијентима. Везе са клијентима се прихватају помоћу функције *accept()*. Након повезивања клијент-слушалац могуће је размењивање порука. Пример једноставног повезивања дат је на слици 2.2.

Слушалац

```
address = ('localhost', 6000)

def main():
    with Listener(address, authkey = b'secret password') as listener:
        with listener.accept() as conn:
            print('connection accepted from', listener.last_accepted)

if __name__=='__main__':
    main()
```

Клијент

```
address = ('localhost', 6000)

def main():
    with Client(address, authkey =  b'secret password') as conn:

if __name__=='__main__':
    main()
```

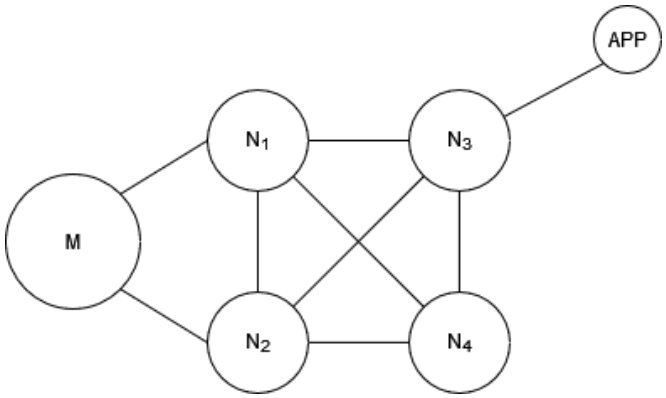
Слика 2.3 Пример коришћења класа Client i Listener

3. Пројекат решења

У овом поглављу описана је архитектура и понашање реализованог решења система на бази ланца-блокова трансакција.

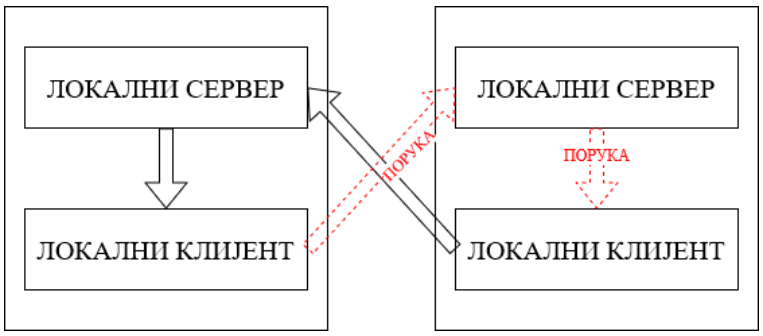
3.1 Архитектура

Архитектура система на бази ланца-блокова састоји се од: једног серверског процеса руковаоца (M), више чворова (N_1, N_2,...., Nn) и једне или више корисничких апликација (APP), као на слици 3.1. Руковаоц је изведен из класе BaseManager и служи за чување адреса свих чворова из мреже.



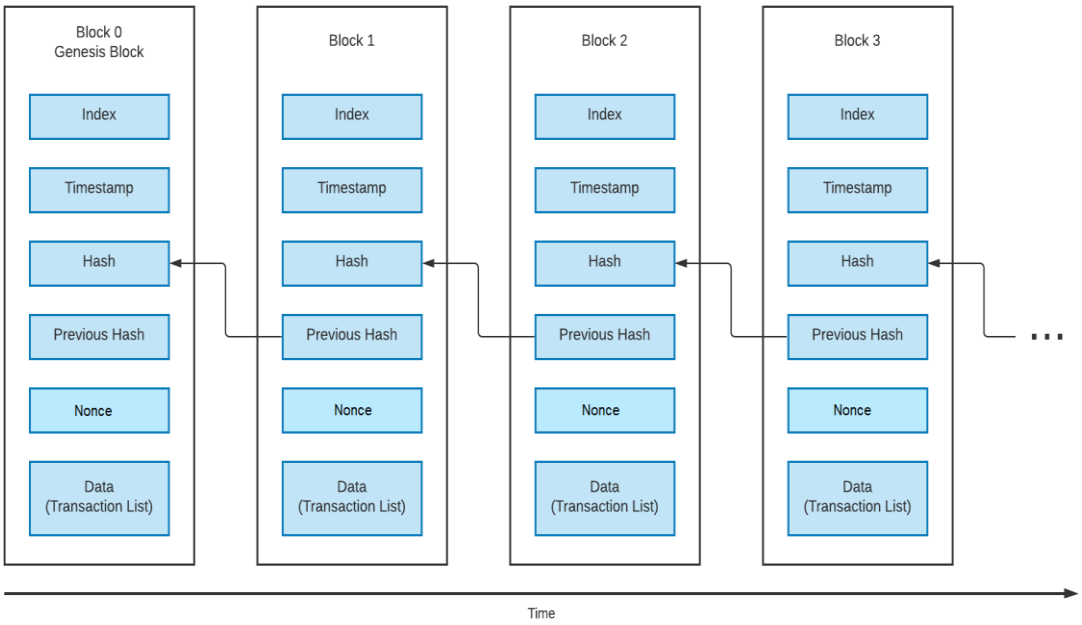
Слика 3.1 Архитектура система на бази ланца-блокова

Чворови су међусобно једнаки и повезују се у потпуно повезану мрежу. Сваки чвор се састоји од процеса локалног клијента и процеса локалног сервера. Клијент и сервер су повезани са редом queue, у који сервер уписује примљене поруке од удаљеног клијента, а локални клијент преузима поруке које је локални сервер уписао. Локални клијент је задужен за слање порука удаљеном серверу.



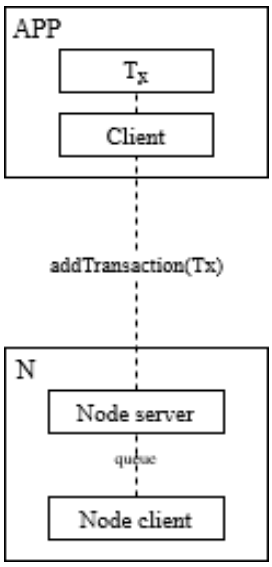
Слика 3.2 Реализација комуникације чворова

Сваки једнаки чвор у мрежи има могућност чувања ланца-блокова. Архитектура ланца-блокова дата је на слици 3.3. Ланац-блокова започиње иницијалним блоком и састоји се од хронолошки поређаних блокова (у односу на време настанка), који се међусобно дигитално повезују помоћу хеш функције. Блок садржи и листу верификованих трансакција, као и случајан број (nonce).



Слика 3.3 Архитектура ланца-блокова

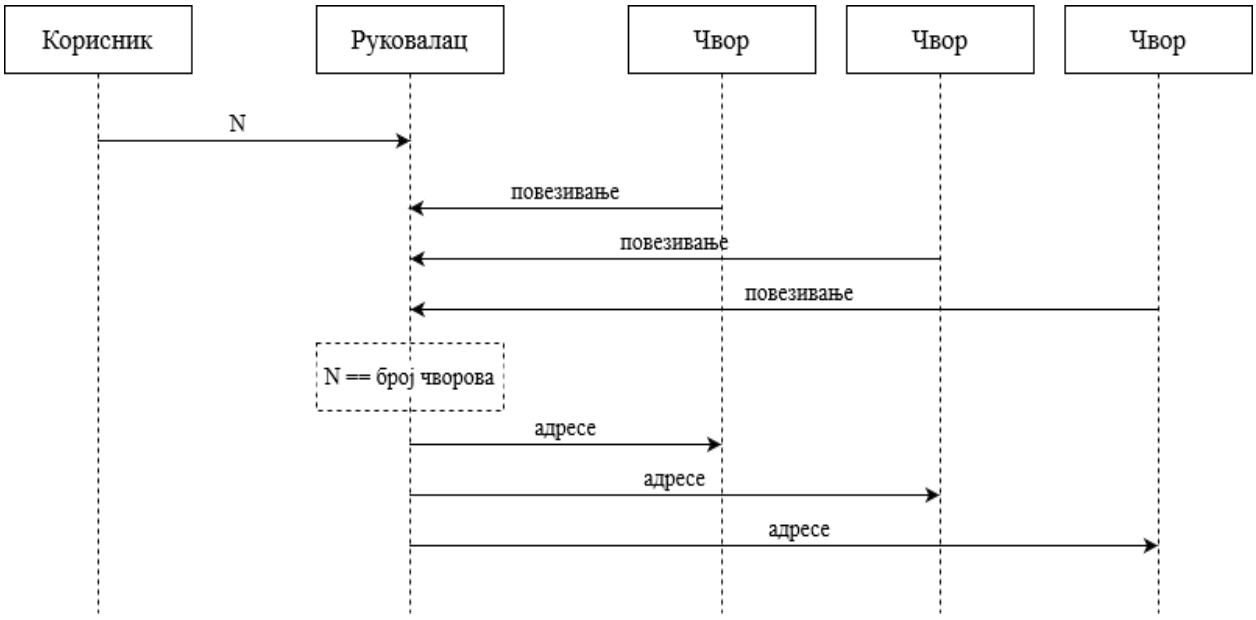
Чворови нуде могућност корисницима да стварају нове трансакције (T_x) помоћу функције addTransaction. Поједностављени UML дијаграм је приказан на слици 3.4.



Слика 3.4 UML дијаграм класа система на бази ланца-блокова

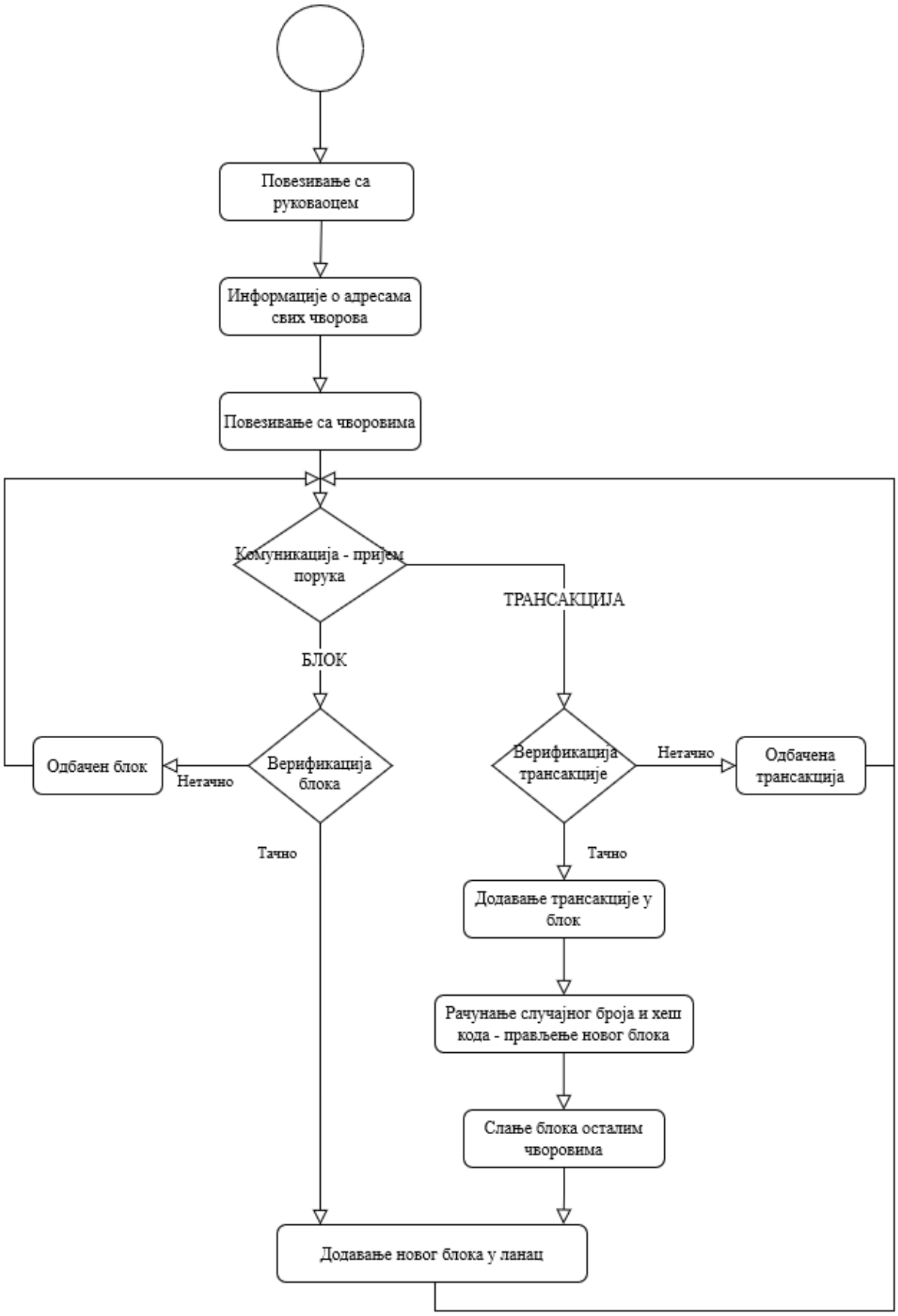
3.2 Понашање

Приликом покретања програма руковаоца потребно је да корисник унесе број, који представља колико чворова ће чинити потпуно повезану мрежу. Чворови се повезују са руковаоцем, записујући своје адресе, све док се не достигне жељени број чворова. Затим сви чворови добијају информације о адресама свих преосталих чворова у мрежи и међусобно се повезују у потпуно повезану мрежу, види слику 3.5.



Слика 3.5 Пријављивање чворова руковаоцу

Када су адресе свих чворова у мрежи познате започиње међусобно повезивање свих чворова у потпуно повезану мрежу. Након повезивања свих чворова започиње комуникација између чворова и размена информација. Сви чворови су равноправни у мрежи и обављају исте послове, види слику 3.6.



Слика 3.6 Понашање чворова

Сваки корисник има могућност стварања трансакција. Трансакције се састоје од: пошиљаоца, примаоца, вредности и временске ознаке када је трансакција настала. Да би

трансакција била створена потребно је да корисник унесе примаоца и вредност. На место пошиљаоца се поставља корисникова адреса, затим се трансакција временски означава када је настала. Пре стварања трансакција неопходно је генерисање дигиталних кључева: приватног и јавног. Приватни кључ представља тајни кључ који је познат само оном коме припада, док јавни кључ служи за верификацију трансакције и он је познат свима. За њихово генерисање користи се RSA алгоритам, кораци алгоритма приказани су на слици 3.7.

Генерисање RSA кључева

1. Насумичан избор два велика цела проста броја p и q
2. Рачунање модула n = p*q
3. Избор целог броја e, да испуњава $1 < e < \varphi(n)$, и да e и $\varphi(n)$ немају заједничких делиоца осим броја 1, где је $\varphi(n) = (n - 1)*(q - 1)$
4. Рачунање броја d, тако да задовољава конгруенцију $d * e \equiv 1 (\mathrm{mod}\ \varphi(n))$, односно d је модуларни мултипликативни инверз целог броја e по модулу $\varphi(n)$
5. Резултат - израчунати кључеви

   Јавни кључ: модуо n и јавни експонент e, public_key = (n, e)

   Приватни кључ: модуо n и приватни експонент d, private_key = (n, d)

Слика 3.7 Генерисање кључева RSA алгоритмом

У првом кораку алгоритма за потребе проналажења великих простих бројева користи се рандом генератор, за проверу да ли је изгенерисани број прост користи се Милер-Рабинов тест, псеудокод је приказан испод на слици 3.8.

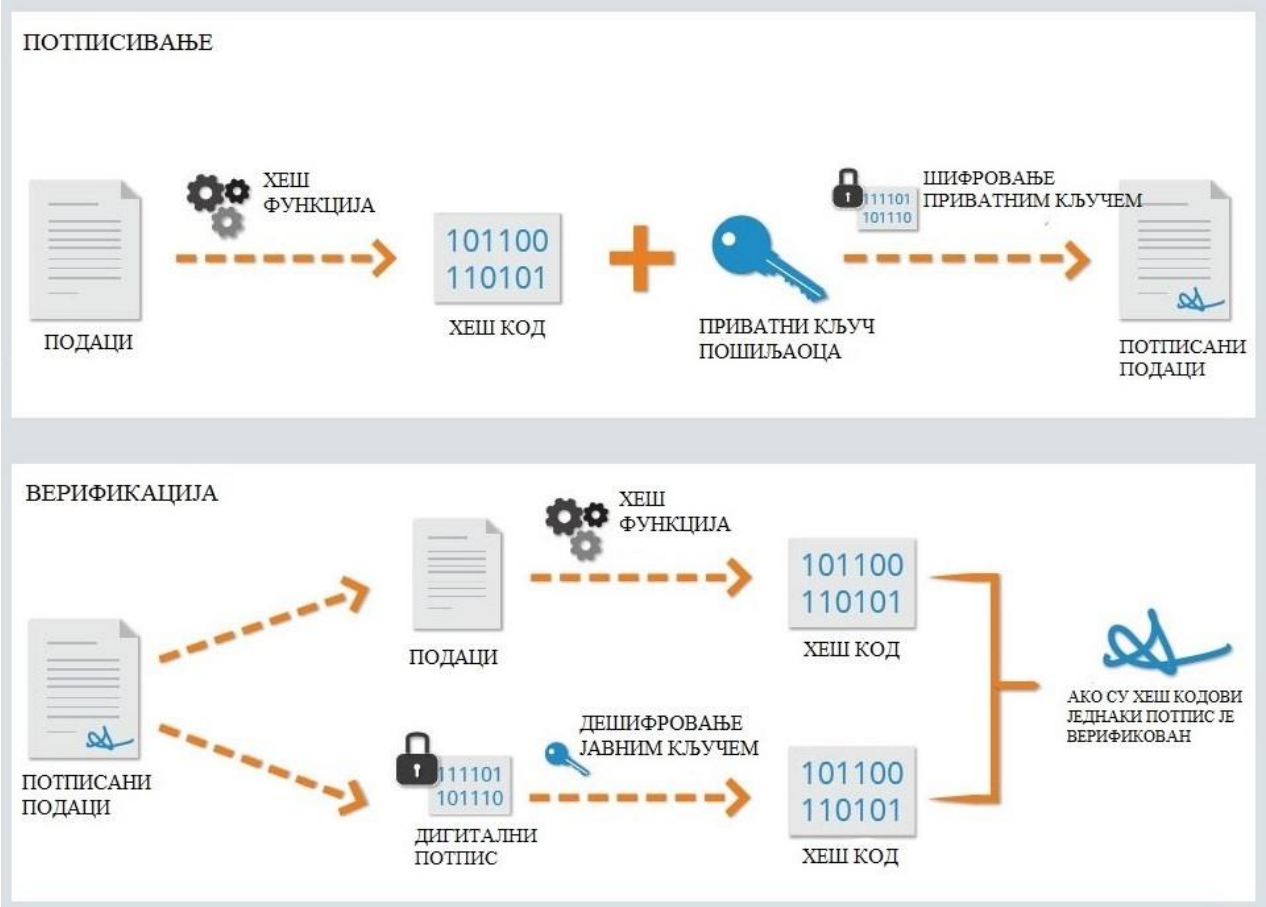
```
n       // улазна вредност, број који се проверава да ли је прост
k       //параметар који одређује тачност теста

Запиши n − 1 као (2^s)*d, са непарним d, фактором степена 2 од n − 1
WitnessLoop: понови к пута:
    избор случајног целог броја a у опсегу од [2, n − 2]
    x ← a^d mod n
    if x = 1 or x = n − 1 then continue WitnessLoop
    поновити s − 1 пута:
      x ← x^2 mod n
      if x = 1 then број сложен
      if x = n − 1 then урадити следећи WitnessLoop
    return сложен
  return вероватно прост
```

Слика 3.8 Псеудокод за Милер-Рабинов тест

Анонимност и сигурност трансакција се обезбеђује употребом криптографије са дигиталним потписивањем. Пре слања трансакције свима неопходно је да корисник потпише трансакцију својим приватним кључем. За стварање дигиталног потписа потребно је прво применити хеш функцију, затим се врши шифровање тако добијеног хеш кода корисниковим приватним кључем. Заједно са трансакцијом прослеђује се и дигитални потпис свим чворовима у мрежи.

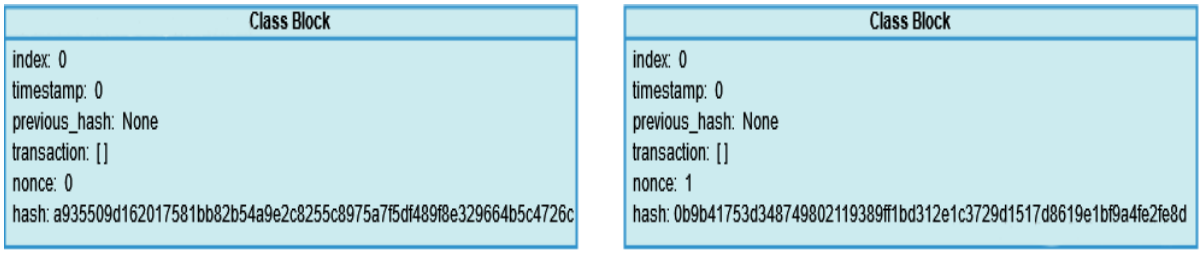
Верификацију трансакција врше чворови након пријема, на основу јавног кључа пошиљаоца. Врши се на супротан начин од потписивања. Корисник примењује јавни кључ пошиљаоца дешифрујући примљени дигитални потпис и добија хеш код трансакције, који пореди са резултатом примене хеш функције на примљену трансакцију. Ако су хеш кодови једнаки, трансакција је верификована. Процеси дигиталног потписивања и верификације трансакције су приказани на слици 3.9.



Слика 3.9 Дигитално потписивање

Само верификоване трансакције се хронолошки, на основу временске ознаке трансакције, додају у блок. Сви блокови се састоје од: индекса, листе верификованих трансакција, временске ознаке када је створен блок, случајног броја (nonce), хеш кода блока и хеш кода претходног блока. Ови параметри су јединствени за све блокове.

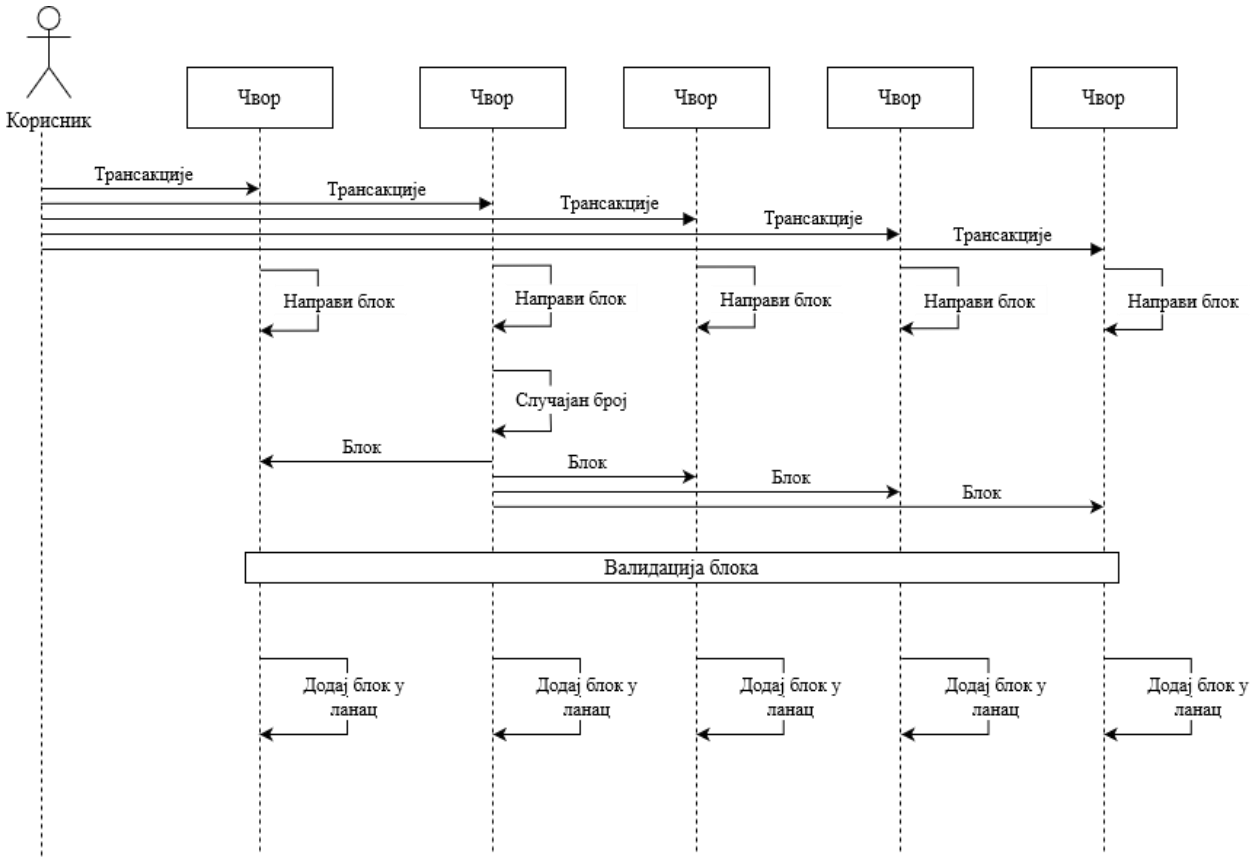
Приликом стварања хеш кода одређеног блока користи се SHA-256 функција и подаци: индекс, трансакције, временска ознака, случајан број (nonce) и хеш код претходног блока, што онемогућава промену садржаја једног блока, а да се не промени садржај свих блокова који иду након њега. Криптографска хеш функција је математички алгоритам који мапира податке произвољне дужине у излаз фиксне дужине (256 бита). За сваку поруку на улазу хеш функција генерише различиту излазну вредност. Свака промена улазне поруке изазива драстичну промену излазног хеш кода. На слици 3.10 је дат пример добијених резултата применом хеш функције на два блока, који се разликују у вредности случајног броја.



Слика 3.10 Пример уптребе хеш функције

Хеш функција је једносмерна и не постоји инверзна функција, не постоји начин да се зна који податак је произвео дати хеш код, такође је и немогуће из добијеног хеш кода пронаћи улазни податак, што омогућава сигурност. Немогуће је изменити већ уланчане податке.

Да би се направио нови блок неопходно је пронаћи случајан број (nonce). Случајан број представља број који рудари треба да пронађу да би се добио хеш који почиње са унапред задатим бројем нула. Проналажење случајног броја није једноставан посао, због особине једносмерности хеш функције. Случајан број се добија испробавањем различитих вредности. Када чвор пронађе случајан број и створи нови блок, шаље га остатку мреже на верификацију. Верификација блока је једноставна и врше је остали чворови у мрежи на основу пронађеног случајног броја. Неопходно је да чвор први пронађе случајан број и ствара блок, да би се његов блок додао у ланац. Након утврђивања исправности случајног броја и верификације блока, блок се додаје у ланац-блокова на свим чворовима, види слику 3.11.



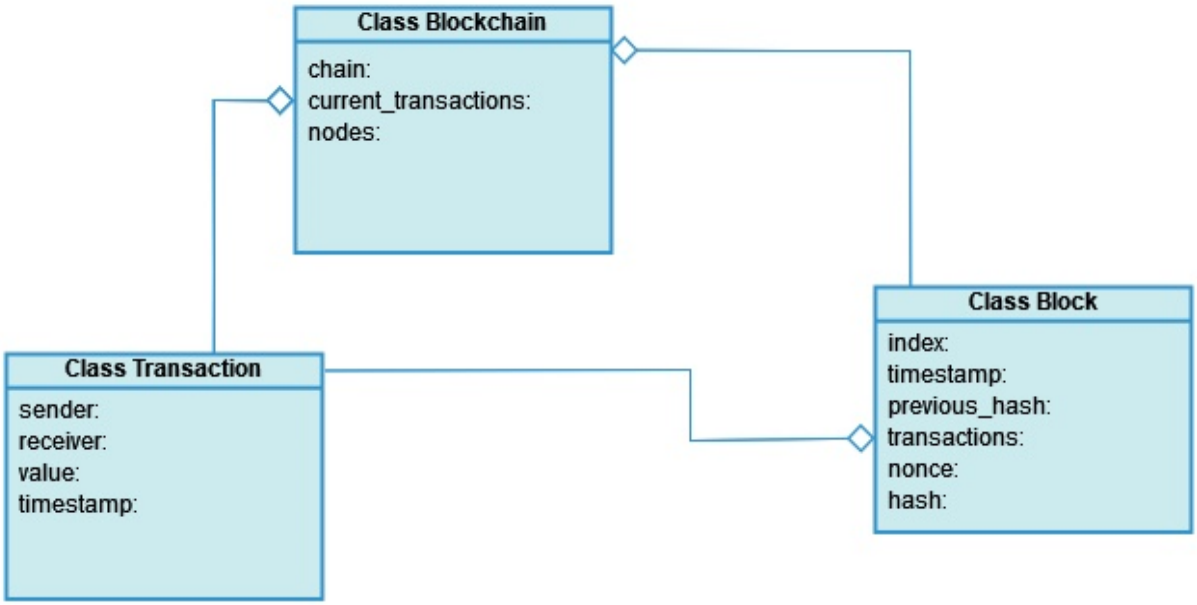
Слика 3.11 Додавање блокова у ланац

Блокови се међусобно дигитално повезују, помоћу хеш кода претходног блока у ланац. Тако се ствара јединствен ланац-блокова који је немогуће изменити.

4. Програмско решење

У овом поглављу описана је програмска имплементација ланца-блокова. Практични део рада рађен је у програмском језику Python. Описане су класе које су коришћене као и реализоване функције.

Класе које су реализоване приказане су на слици 4.1, као и њихова међусобна повезаност.



Слика 4.1 Приказ класа ланца-блокова

Приликом стварања чворова, они праве објекат класе Blockchain. Сви чворови се прво повезују са руковаоцем од кога сазнају информације о адресама свих осталих чворова у мрежи (nodes) са којима се затим повезују и размењују информације. Сви чворови чувају створени ланац-блокова, за шта служи поље chain, који се састоји из дигитално повезаних

блокова. Такође сваки чвор чува информације о верификованим трансакцијама које је тек потребно додати у нови блок у посебној листи (current_transaction).

Направљена је и класа Block која представља један блок у ланцу-блокова, и садржи информације: индекс датог блока, временску ознаку када је блок створен, хеш претходног блока који омогућава дигитално повезивање у ланац-блокова, листу верификованих трансакција, случајан број који су чворови морали да пронађу, као и хеш код самог блока.

Класа Transaction представља једну трансакцију коју корисник ствара, потписује и прослеђује чворовима на верификацију, само верификоване трансакције се додају у блок. Ова класа садржи поља која дају информације о: адреси корисника, примаоцу, вредности и времену настанка трансакције.

У следећим табелама дат је преглед реализованих функција.

Руковалац има једноставну улогу, он се повезује са свим чворовима из мреже, служи за чување адреса чворова и њихово прослеђивање када је пријављен довољан број чворова, видети табелу 4.1.

Табела 4.1 Функције руковаоца

Руковалац	
add(address)	Додавање прослеђене адресе чвора у листу адреса
get_nodes()	Враћа адресе свих чворова који су се повезали са њим

Табела 4.2 садржи преглед функција које се позивају над трансакцијама.

Табела 4.2 Функције трансакција

Трансакције	
generate_key(key_size)	Генерисање приватног и јавног кључа RSA алгоритмом
miller_rabin(num)	Провера да ли је прослеђени број прост, за потребе генерисања кључева
sign_transaction(private_key)	Потписивање трансакције помоћу прослеђеног приватног кључа корисника
verify_transaction(sign, public_key)	Врши верификацију трансакције на основу јавног кључа пошиљаоца и дигиталног потписа
readFromFile()	Стварање нове трансакције, читањем вредности из датотеке, направљене за потребе тестирања

consoleInput()	Стварање трансакције на основу корисникових унетих вредности за пошиљаоца и вредност

Табела 4.3 нам даје преглед свих функција које су потребне за обезбеђивање ланца-блокова.

Табела 4.3 Функције ланца-блокова

Ланац-блокова	
register_node(address)	Уписивање прослеђене адресе чвора из мреже у nodes
sort_add(transaction)	Додавање верификоване трансакције на одговарајуће место у current_transactions на основу времена настанка трансакције
new_transaction(transaction)	Додавање верификоване трансакције на крај current_transactions
create_block(index, transactions, previous_hash, proof, hash, timestamp)	Стварање новог блока помоћу прослеђених информација
validate_block(previous_block, block)	Провера исправности блока на основу свих његових вредности, користи функцију за проверу исправности случајног броја и хеш кода претходног блока
last_block()	Враћа последњи блок из ланца-блокова
calculate_hash(block)	Рачуна и враћа вредност хеш кода прослеђеног блока
add_block(block)	Додаје нови блок у ланац-блокова
proof_of_work(block, difficulty)	Проналази случајан број за прослеђени блок, мора се проследити и број нула са којим је потребно да започиње хеш код
validate_proof(block, difficulty)	Проверава исправност случајног броја помоћу прослеђеног потребног броја нула
mine_block()	Рудари нови блок позивајући функцију за проналажење случајног броја
valid_chain(chain)	Обавља проверу исправности целог прослеђеног ланца-блокова
resolve_conflicts(chains)	Проналази најдужи исправан ланац-блоква од свих примљених и ако је дужи од локалног ланца замењује га

Табела 4.4 нам даје преглед функција које користе процеси локалног сервера и локалног клијента сваког чвора.

Табела 4.4 Функције клијента и сервера

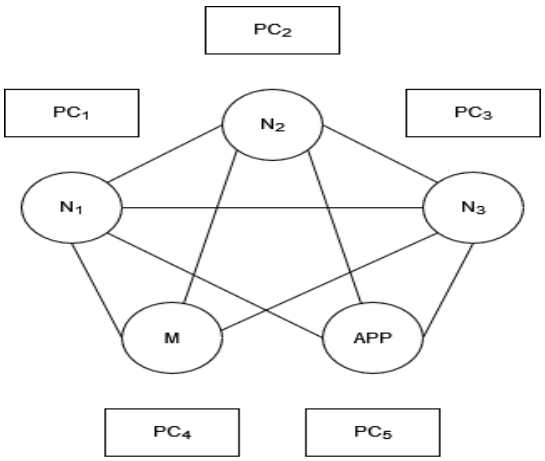
Клијент и сервер	
server_fun(my_address,local_port, queue)	Позива се приликом стварања процеса локалног сервера, за прихватање везе са удаљеним клијентима и уписивање примљених порука у queue
sendMsg(remote_server_address, msg)	Позива је локални клијент за слање msg поруке удаљеном серверу који се налази на прослеђеној адреси
broadcastMsg(list_of_remote_server_address, msg)	Позива је локални клијент за слање msg поруке свим удаљеним серверима, чије се адресе налазе у листи list_of_remote_server_address
rcvMsg(queue)	Позива је локални клијент за пријем поруке које је локални сервер уписао у queue
rcvMsgs(queue, no_of_messages_to_receive)	Позива је локални клијент за пријем прослеђеног броја порука које је локални сервер уписао у queue

5. Експериментална евалуација

Циљ овог одељка је да се покажу добијени експериментални резултати, измери пропусност система, која је дефинисана као број трансакција у секунди, у зависности од броја чворова развијене архитектуре.

За потребе тестирања направљен је тестни програм *test*, који служи за израчунавање оствареног броја трансакција у секунди. Експеримент је изведен на потпуно повезаној мрежи са 3, 5, 8, 10, 12 и 15 чворова. Тест програм се повезује са руковаоцем, од којег сазнаје информације о броју чворова и њихове адресе, затим се са чворовима повезује на исти начин на који се они међусобно повезују. Адреса рачунара на коме се покреће тест апликација је у напред позната и сви чворови у мрежи имају ту информацију.

Тест апликација број понуђених трансакција на улазу, *num_trans* = [50, 150, 200, 350, 450, 550, 650, 700, 900], дели у односу на број чворова у тренутној мрежи и прослеђује добијену вредност чворовима. Чворови обављају уобичајене функције: стварања трансакција, дигиталног потписивања, прослеђивања, верификације трансакција, додавања нових трансакција у блокове, верификације блокова и додавања блокова у ланац. Затим након стварања ланца-блокова, чворови прослеђују информације о дужини ланца-блокова, као и сам ланац-блокова. На основу информација добијених са свих чворова, тест апликација израчунава остварену пропусност система. Дељењем укупног броја трансакција и потрошеног времена добија се број трансакција по секунди. Експеримент смо спроводили на локалној мрежи у фирми Транспортгас. Тест апликација је покретана на засебном рачунару, као и руковалац и сваки чвор у мрежи.

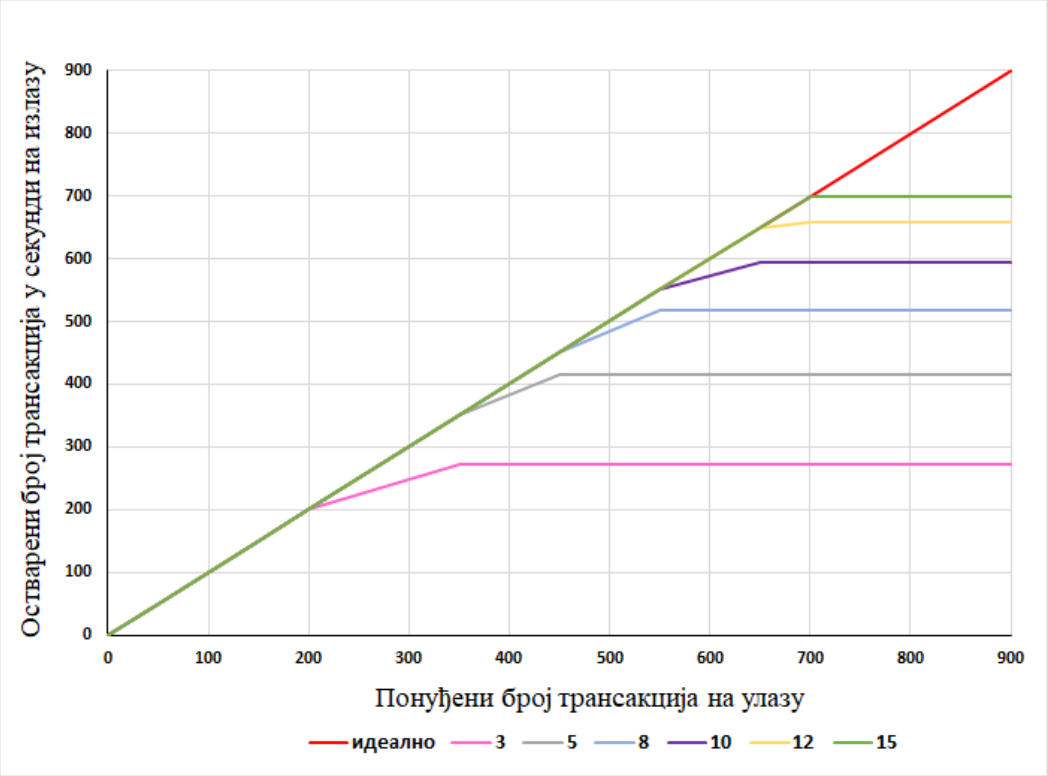


Слика 5.1 Пример тестне мрежне конфигурације

Карактеристике експерименталне поставке су следеће. LAN карактеристике: рачунарске утичнице су повезане на један комутатор Juniper EX-4300. Овај комутатор обезбеђује корисничке приступне портове од 10/100/1000Mbps и везе ка надређеним комутаторима од 10Gbps. Мрежни каблови су UTP категорије CAT5e. Карактеристике рачунара: CPU - Intel Corei7-10700 CPU @ 2.90GHz (8 cores x64), MB - Dell 0x8DxD, RAM - 2 module x 16 GB DDR4 @ 2666MHz, HDD - 1TB (Toshiba), Graphics - Intel(R) UHD Graphics 630. Верзија софтвера: OS - Windows 10 Enterprise (x64). Python верзија 3.10.5 on win32.

Табела 5.1 Добијени експериментални резултати

	3	*5*	*8*	*10*	*12*	*15*
50	50	50	50	50	50	50
150	150	150	150	150	150	150
200	200	200	200	200	200	200
350	272	350	350	350	350	350
450	272	415	450	450	450	450
550	272	415	518	550	550	550
650	272	415	518	593	650	650
700	272	415	518	593	659	698
900	272	415	518	593	659	698



Слика 5.2 Експериментално добијени дијаграм

Добијени експериментални резултати приказани су у табели 5.1 као и на слици 5.2 која представља дијаграм изведен на основу резултата из табеле. Сва тестирања су поновљена 20 пута ради сигурности. У приказаној табели представљене су средње вредности, одступања су била занемарљива. Колоне у табели 5.1 садрже информације о броју чворова у мрежи, док нам редови говоре о понуђеном броју трансакција на улазу. Табела 5.1 нам даје преглед оствареног броја трансакција у секунди на излазу у зависности од броја чворова у мрежи. Експериментално је утврђено да повећање броја чворова у мрежи резултира повећању броја додатих трансакција у секунди у ланац-блокова.

6. Закључак

Главни допринос овог мастер рада је једно решење ланца-блокова (енг. blockchain). У мастер раду је пројектована, имплементирана и експериментално евалуирана архитектура система за одржавање ланца-блокова трансакција у програмском језику Python. Развијено решење архитектуре омогућава стварање и верификацију трансакција, њихово додавање у блокове, и њихово међусобно дигитално повезивање у ланац-блокова трансакција. Резултати експерименталне евалуације развијеног решења архитектуре показују да пропусност система (тј. број трансакција у секунди) линеарно расте са бројем захтеваних трансакција до извесне границе, која природно расте са бројем чворова предложене архитектуре, након чега природно наступа засићење.

Главно ограничење приказаног решења је што је унапред потребно одредити број чворова који ће учествовати у стварању мреже, тај број корисник прослеђује руковаоцу приликом његовог покретања. Једном када је мрежа створена и чворови повезани није могуће додавати нове чворове у мрежу, из чега произилази да није постигнуто потпуно динамичко повезивање чворова.

Главна предност описаног решења је што је ланац-блокова имплементиран у Python-у без употребе додатних пакета, представљајући потпуно независно и самостално решење. Независност од пакета и техника омогућава лако интегрисање овог решења у друге програме и системе, представљајући основу за његово даље доградњивање.

Због корисних особина саме технологије ланца-блокова и независности овог решења, могућа је његова примена у различитим областима. Главни правац будућег рада је употреба ланца-блокова у интернету ствари и паметним кућама. Једна од идеја је да се рад примени на електричним бројилима који би чинили чворове у мрежи и уписивали своје тренутне вредности.

7. Литература

[1] https://github.com/dvf/blockchain?ref=hackernoon.com [приступљено у марту 2021.]

[2] Daniel van Flymen: *Learn Blockchain by Building One: A Concise Path to Understanding Cryptocurrencies*, Apress, 2020

[3] R.L. Rivest, A. Shamir, and L. Adleman, "*A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*", 1978

[4] Rabin, Michael O., "*Probabilistic algorithm for testing primality*", Journal of Number Theory, Vol. 12, No. 1, pp. 128–138, 1980

[5] Stuart Haber, W. Scott Stornetta, "*How to Time-Stamp a Digital Document*", In Journal of Cryptology, Vol. 3, No. 2, pp. 99-111, 1991.

[6] Satoshi Nakamoto: "*Bitcoin: A Peer-to-Peer Electronic Cash System*", 2008.

[7] S. Simić, M. Marković, S. Gostojić, "*Smart Contract and Blockchain Based Contract Management System*", ECBS, 2021

[8] Wang, Su, Zhang, "*BSIS for Energy Delivery in Vehicular Energy Network*", IEEE Transactions on Industrial Informatics, Vol. 15, Iss. 6, June 2019

[9] J. Al-Jaroodi, N. Mohamed, "*Blockchain in Industries: A Survey*", IEEE Access, Vol. 7, 2019

[10] J. A. Jaoude, R. Saade, "*Blockchain Applications – Usage in Different Domains*", IEEE Access, Vol. 7, 2019

[11] M. Hölbl, "*A Systematic Review of the Use of Blockchain in Healthcare*", In Journal Symmetry, Vol. 10, Iss. 10, 2018

[12] D. Minoli, B. Occhiogrosso, "*Blockchain mechanisms for IoT security* ", In Journal Internet of Things, Vol. 1, pp. 1–13, 2018

[13] A. Dorri, S. S. Kanhere, R. Jurdak, "*Towards an optimized blockchain for IoT*", In Proceedings of the Second International Conference on Internet-of-Things Design and Implementation, Pittsburgh, pp. 173–178, 2017

[14] Li, Jiang, Chen, Luo, Wen, "*A survey on the security of blockchain systems*", in Jurnal Future Generation Computer Systems, Vol. 107, pp. 841-853, 2017