**NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY**

(AN AUTONOMOUS INSTITUTION, AFFILIATED TO VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAUM, APPROVED BY AICTE & GOVT.OF KARNATAKA



**PHASE-III REPORT**

## on

**CRYPTOKEN – A BLOCKCHAIN TRANSACTION SYSTEM**

*Submitted in partial fulfilment of the requirement for the award of Degree of*

*Bachelor of Engineering*

*in*

*Computer Science and Engineering*

*Submitted by:*

DEON SEQUEIRA SESHA SAI S SHISHIR S HEGDE

SIDDHANTH KARANTH

1NT17CS172

1NT17CS173

1NT17CS178

1NT17CS184

Under the Guidance of

Dr. Thippeswamy M N

Professor, Head, Dept. of CSE, NMIT



## Department of Computer Science and Engineering

#### (Accredited by NBA Tier-1)

2020-2021

**NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY**

(AN AUTONOMOUS INSTITUTION, AFFILIATED TO VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAUM

, APPROVED BY AICTE & GOVT.OF KARNATAKA)

Department of Computer Science and Engineering

**(Accredited by NBA Tier-1)**



#### CERTIFICATE

This is to certify that the Phase II Report on **CRYPTOKEN – A BLOCKCHAIN TRANSACTION SYSTEM** is an authentic work carried out by DEON SEQUEIRA (1NT17CS172), SESHA SAI S **(1NT17CS173),** SHISHIR S HEGDE **(1NT17CS178)** and

SIDDHANTH KARANTH **(1NT17CS184)** bonafide students of **Nitte Meenakshi Institute of Technology**, Bangalore in partial fulfilment for the award of the degree of ***Bachelor of Engineering*** in COMPUTER SCIENCE AND ENGINEERING of Visvesvaraya Technological University, Belagavi during the academic year ***2020-2021.*** It is certified that all corrections and suggestions indicated during the internal assessment has been incorporated in the report.

**Internal Guide Signature of the HOD Signature of Principal**

Dr. Thippeswamy M. N.

Professor, Head, Dept. CSE, NMIT Bangalore

Dr.Thippeswamy M. N. Professor, Head, Dept. CSE, NMIT Bangalore

Dr. H. C.Nagaraj Principal,NMIT, Bangalore

#### Signature of Examiners

1.

2.

**DECLARATION**

We hereby declare that

1. The project work is our original work
2. This Project work has not been submitted for the award of any degree or examination at any other university/College/Institute.
3. This Project Work does not contain other persons’ data, pictures, graphs or other information, unless specifically acknowledged as being sourced from other persons.
4. This Project Work does not contain other persons’ writing, unless specifically acknowledged as being sourced from other researchers. Where other written sources have been quoted, then:
   1. their words have been re-written but the general information attributed to them has been referenced;
   2. where their exact words have been used, their writing has been placed inside quotation marks, and referenced.
5. This Project Work does not contain text, graphics or tables copied and pasted from the Internet, unless specifically acknowledged, and the source being detailed in the thesis and in the References sections.

|  |  |  |
| --- | --- | --- |
| **NAME** | **USN** | **Signature** |
| DEON SEQUEIRA | 1NT17CS172 |  |
| SESHA SAI S | 1NT17CS173 |  |
| SHISHIR HEGDE | 1NT17CS178 |  |
| SIDDHANTH KARANTH | 1NT17CS184 |  |

## Date:

**ACKNOWLEDGEMENT**

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crowned our effort with success. I express my sincere gratitude to our Principal **Dr.**

**H. C. Nagaraj**, Nitte Meenakshi Institute of Technology for providing facilities.

We wish to thank our HoD**, Dr.Thippeswamy M.N.** for the excellent environment created to further educational growth in our college. We also thank him for the invaluable guidance provided which has helped in the creation of a better project.

I hereby like to thank our ***Hod, Dr, Thippeswamy M.N.***, Department of Computer Science & Engineering on **his** periodic inspection, time to time evaluation of the project and help to bring the project to the present form.

Thanks to our Departmental Project coordinators. We also thank all our friends, teaching and non- teaching staff at NMIT, Bangalore, for all the direct and indirect help provided in the completion of the project.

|  |  |  |
| --- | --- | --- |
| **NAME** | **USN** | **Signature** |
| DEON SEQUEIRA | 1NT17CS172 |  |
| SESHA SAI S | 1NT17CS173 |  |
| SHISHIR HEGDE | 1NT17CS178 |  |
| SIDDHANTH KARANTH | 1NT17CS184 |  |

## Date:

**ABSTRACT**

A Blockchain, is a growing list of records, called *blocks* that are linked using cryptography. Each block contains a hash value of the previous block, timestamp, and transaction data generally represented as a Merkle tree.

By design, a blockchain is resistant to modification of its data. This is because once recorded, the data in any given block cannot be altered retroactively without alteration of all subsequent blocks. For use as a distributed ledger, a blockchain is typically managed by a peer-to-peer network collectively adhering to a protocol for inter-node communication and validating new blocks.

Blockchain Transaction System is provided by Blockchain and is an E-wallet that allows individuals to store and transfer crypto currencies. In this project we are making use of the open source currency Simple Coin. Once the wallet is created, the user is provided with a Wallet address or a public key, along with a private key. This private key is a unique identifier, similar to a bank account number.

Once the wallet is created, the user can send money to other users. Once their transaction is mined, it is then added to the Blockchain network and the user can view the completed and mined transactions.

#### DECLARATION ACKNOWLEDGEMENT ABSTRACT

**TABLE OF CONTENTS CHAPTER 1: INTRODUCTION**

* 1. Background
  2. Brief history of Technology/concept
  3. Research motivation and Problem statement
     1. Research Motivation
     2. Statement of the Problem
  4. Research objectives and contributions
     1. Primary objectives
     2. Main contributions
  5. Organization of the report
  6. Summary

#### CHAPTER 2: LITERATURE SURVEY

* 1. Introduction
  2. Related work
  3. Study of Tools/Technology
  4. Summary

#### CHAPTER 3: SYSTEM REQUIREMENTS SPECIFICATIONS

* 1. General Description
     1. Product Perspective
  2. System Requirements
     1. Hardware Requirements
     2. Software Requirements
        1. Functional Requirments & Non-functional Requirements

#### CHAPTER 4: DESIGN AND IMPLEMENTATION CHAPTER 6: RESULTS

**CHAPTER 7: CONCLUSIONS**

**CHAPTER 8: REFERENCES**

**APPENDIX: Survey Paper**

**Chapter 1: INTRODUCTION**

A blockchain is a type of database. A database is a collection of information that is stored electronically on a computer system. Information, or data, in databases is typically structured in table format to allow for easier searching and filtering for specific information. Large databases achieve this by housing data on servers that are made of powerful computers.

These servers can sometimes be built using hundreds or thousands of computers in order to have the computational power and storage capacity necessary for many users to access the database simultaneously. While a spreadsheet or database may be accessible to any number of people, it is often owned by a business and managed by an appointed individual that has complete control over how it works and the data within it.

Blockchain does not store any of its information in a central location. Instead, the blockchain is copied and spread across a network of computers. Whenever a new block is added to the blockchain, every computer on the network updates its blockchain to reflect the change. By spreading that information across a network, rather than storing it in one central database, blockchain becomes more difficult to tamper with. If a copy of the blockchain fell into the hands of a hacker, only a single copy of the information, rather than the entire network, would be compromised.

Blockchain Wallet is provided by Blockchain and is an E-wallet that allows individuals to store and transfer crypto currencies. In this project we are making use of the open source currency Simple Coin. Once the wallet is created, the user is provided with a Wallet address or a public key, along with a private key. This private key is a unique identifier, similar to a bank account number.

# Brief history of Technology/concept

In the year 1976, a paper was released on “New Directions in Cryptography” discussed the concept of distributed ledger. With the advancement in the field of Cryptography, another paper entitled as “Hot to Time-Stamp a Digital Document” by Stuart Haber and Scott Stornetta which laid out the concept to timestamp the data instead of the medium. Another important concept called as “Electronic cash” or “Digital Currency” which came into existence based on a model proposed by David Chaum also contributed towards the development of the concept of Blockchain which was followed by Protocols such as e-cash schemes that introduced double spending detection.

Satoshi Nakamoto is considered as the inventor of blockchain technology when he published a paper on bitcoin in 2008 as “Bitcoin: A Peer-to-Peer Electronic Cash System,”. The abstract of the paper was on the direct online payment from one source to another source without relying on a third-party source. The paper described an electronic payment system based on the concept of cryptography. Nakamoto’s paper provided a solution to the double spending where a digital currency cannot be duplicated, and no one can spend it more than once. The paper stated the concept of public ledger where an electronic coin transaction history can be traced and confirmed if the coin has not been spent before and to prevent double spending issue.

There are hundreds of different cryptocurrencies such as Litecoin, Dogecoin etc., but bitcoins hold the lion share of the market it has become the most popular cryptocurrency among the others.

# PROBLEM STATEMENT

The current transaction system consists of a few flaws like depicted below:

1. Service fees: Payment gateways and third-party payment processors charge service fees.
2. Inconvenient for offline sales: Online payment methods are inconvenient for offline sales.
3. Vulnerability to cybercriminals: Cybercriminals can disable online payment methods or exploit them to steal people’s money or information. Visit the Australian Cybercrime Reporting Network’s Learn about cybercrime page to learn more about cybercrime.
4. Reliance on telecommunication infrastructure: Internet and server problems can disable online payment methods.
5. Technical problems: Online payment methods can go down due to technical problems.
6. The transaction system: CrypToken, aims to solve the above mentioned problems by making use of Blockchain technology.

# OBJECTIVES

1. By using Blockchain technology, there is no use of servers for updating and storing the data, which makes the transactions offline.
2. This use of transactions using Blockchain technology can be used in rural areas with no proper internet facilities or cellular connectivity.
3. Cryptoken does not charge any service fees for transactions.
4. Blockchain transaction system is more secure since it uses cryptographic algorithms for encryption.
5. Since there is no use of servers, there is no question of internet issues or server connectivity issues to be worried by the user.

# CHAPTER 2: LITERATURE SURVEY

Blockchain may well be viewed as a public ledger and each submitted dealings is place during a list of blocks. This chain develops as new blocks are mounted to that incessantly. With an awfully designed data storage structure, transactions in Bitcoin system might occur with no any third party and therefore the core innovation to construct Bitcoin is blockchain that was initial planned in 2008 and dead in 2009[13].

These days digital cash has become a stylish expression in each trade and profound world. In concert of the foremost eminent digital cash, Bitcoin has delighted an enormous success with its capital market achieving ten billion [12].

Asymmetric cryptography and distributed accord calculation are dead for consumer security and record consistency. The blockchain technology has key qualities of decentralization, persistence, anonymity and auditability. With these attributes, blockchain will considerably spare the price and enhance the productivity. As a matter of 1st importance blockchain is permanent. Dealings cannot be altered once it's stuffed into the blockchain. Organizations that need high responsibility and honesty will utilize blockchain to draw in purchasers. Moreover, blockchain is distributed and may avoid the only purpose of disappointment circumstance. Blockchain are often utilised in several money services as an example, advanced resources, settlement and on-line payment. Additionally, it may be applied into alternative fields as well as sensible contracts, public services, Internet of Things (IoT), name systems and security services[11].

Those fields favour blockchain in multiple ways in which. It’s been proved that miners might come through larger revenue than their justifiable share through inconsiderate mining strategy. Blockchain could be a sequence of blocks that holds an entire list of dealing records like standard public ledger. With a previous block hash contained within the block header, a block has just one parent block. Its price noting that uncle blocks (children of the block’s ancestors) hashes would even be hold on in ethereum blockchain. The primary block of a blockchain is named genesis block that has no parent block. We tend to then justify the internals of blockchain in details[14].

#### Characteristics of Blockchain:

1. Decentralization. In standard centralized group action systems, every group action must be valid through the central trustworthy agency (e.g., the central bank), inevitably ensuing to the value and therefore the performance bottlenecks at the central servers. Distinction to the centralized mode, third party is not any longer required in blockchain. Accord algorithms in blockchain are accustomed maintain information consistency in distributed network.
2. Persistency. Transactions are often valid quickly and invalid transactions wouldn't be admitted by honest miners. It’s nearly not possible to delete or rollback transactions once they're enclosed within the blockchain. Blocks that contain invalid transactions may well be discovered directly.
3. Anonymity. Every user will act with the blockchain with a generated address, that doesn't reveal the $64000 identity of the user.
4. Auditability. Bitcoin blockchain stores knowledge regarding user balances supported the unexpended dealings. Any dealings must ask some previous unexpended transactions. Once this dealing is recorded into the blockchain, the state of these referred unexpended transactions switch from unexpended to spend. Therefore, transactions may well be simply verified and tracked. [13]

#### Consensus Algorithms:

1. PoW: (Proof of work) could be an accord strategy employed in the Bitcoin network. In PoW, every node of the network is shrewd a hash worth of the block header. The block header contains a nowadays and miners would modification the nowadays often to induce completely different hash values. The accord needs that the calculated worth should be adequate to or smaller than a specific given worth.
2. PoS: (Proof of stake) is a vitality sparing option in contrast to PoW. Diggers in PoS need to demonstrate the responsibility for measure of money. Specifically, Blockchain utilizes randomization to anticipate the next generator. It utilizes an equation that searches for the most minimal hash an incentive in blend with the span of the stake. Numerous blockchains embrace PoW toward the start and change to PoS bit by bit.
3. PBFT: (Practical byzantine fault tolerance) is a replication calculation to endure byzantine issues. Hyper ledger Fabric uses the PBFT as its accord calculation since PBFT could deal with up to 1/3 malignant byzantine reproductions.
4. DPOS: (Delegated proof of stake) is agent fair. Partners choose their agents to produce and approve squares. Casted a ballot out effectively. DPOS is the foundation of Bitshares.
5. Ripple: Ripple is an accord calculation that uses by and large confided in sub networks inside the bigger system. In the system, hubs are separated into two kinds: server for taking an interest accord process and customer for just exchanging assets.[18]

# CHAPTER 3: SYSTEM REQUIREMENTS SPECIFICATIONS

#### Hardware requirements:

* + System: Intel i5 1.9 GHz
  + Hard disk: 500 GB
  + Ram: 6 GB
  + Any/Desktop with above configuration.

#### Software requirements:

* + Virtual Machine Hosts: Virtual Box or VM Ware
  + Any Linux Distribution (Kali Linux 2020 used in this case)
  + Python 3
  + Simple Coin

# FUNCTIONAL REQUIREMENTS

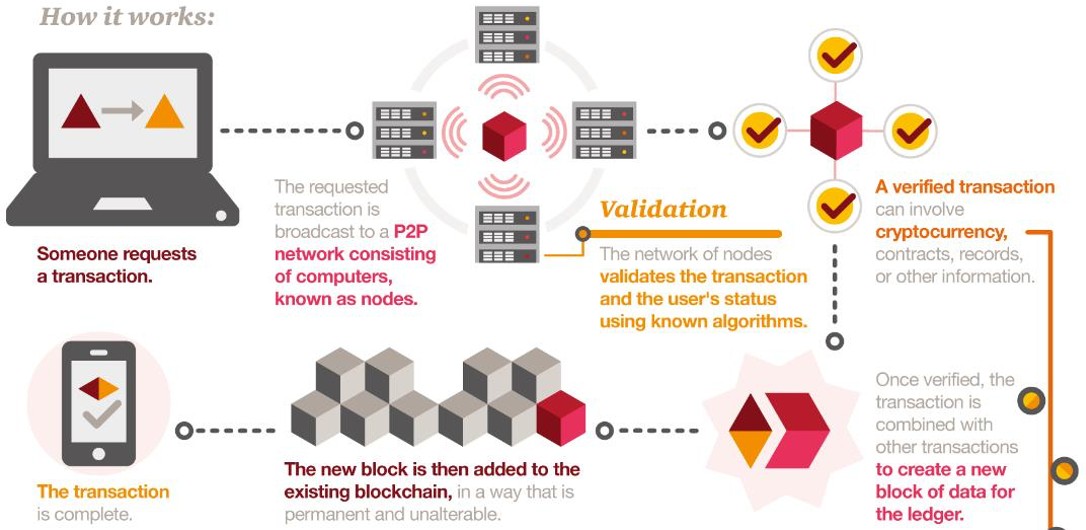
The project consists of 2 main features:

1. Miner.py
2. Wallet.py

**Miner.py**: This file is probably the most important. Running it will create a node (like a server). From here you can connect to the Blockchain and process transactions (that other users send) by mining. The more nodes exist, the more secure the Blockchain gets.

**Wallet.py**: This file is for those who don't want to be nodes but simple users. Running this file allows you to generate a new address, send coins and check your transaction history (keep in mind that if you are running this in a local server, you will need a "miner" to process your transaction). When creating a wallet address, a new file will be generated with all your security credentials. You are supposed to keep it safe.

# CHAPTER 4: DESIGN AND IMPLEMENTATION



1. **:** [**https://www.developcoins.com/how-are-blockchain-transactions-verified**](http://www.developcoins.com/how-are-blockchain-transactions-verified)

Blockchain consists of three important concepts: blocks, nodes and miners.

#### Blocks

Every chain consists of multiple blocks and each block has three basic elements:

* + The **data** in the block.
  + A 32-bit whole number called a **nonce.** The nonce is randomly generated when a block is created, which then generates a block header hash.
  + The **hash** is a 256-bit number wedded to the nonce. It must start with a huge number of zeroes (i.e., be extremely small).

When the first block of a chain is created, a nonce generates the cryptographic hash. The data in the block is considered signed and forever tied to the nonce and hash unless it is mined.

#### Miners

Miners create new blocks on the chain through a process called mining.

In a blockchain every block has its own unique nonce and hash, but also references the hash of the previous block in the chain, so mining a block isn't easy, especially on large chains.

Miners use special software to solve the incredibly complex math problem of finding a nonce that generates an accepted hash. Because the nonce is only 32 bits and the hash is 256, there are roughly four billion possible nonce-hash combinations that must be mined before the right one is found. When that happens miners are said to have found the "golden nonce" and their block is added to the chain.

Making a change to any block earlier in the chain requires re-mining not just the block with the change, but all of the blocks that come after. This is why it's extremely difficult to manipulate blockchain technology. Think of it is as "safety in math" since finding golden nonce requires an enormous amount of time and computing power.

When a block is successfully mined, the change is accepted by all of the nodes on the network and the miner is rewarded financially.

#### Nodes

One of the most important concepts in blockchain technology is decentralization. No one computer or organization can own the chain. Instead, it is a distributed ledger via the nodes connected to the chain. Nodes can be any kind of electronic device that maintains copies of the blockchain and keeps the network functioning.

Every node has its own copy of the blockchain and the network must algorithmically approve any newly mined block for the chain to be updated, trusted and verified. Since blockchains are transparent, every action in the ledger can be easily checked and viewed. Each participant is given a unique alphanumeric identification number that shows their transaction.

# CHAPTER 5: IMPLEMENTATION

#### Wallet.py

* Generate a new address (public and private key). You are going

to use this address (public key) to send or receive any transactions. You can have as many addresses as you wish, but keep in mind that if you

lose its credential data, you will not be able to retrieve it.

* Send coins to another address
* Retrieve the entire blockchain and check your balance

If this is your first time using this script don't forget to generate a new address and edit miner config file with it (only if you are going to mine).

Timestamp in hashed message. When you send your transaction it will be received by several nodes. If any node mine a block, your transaction will get added to the blockchain but other nodes still will have it pending. If any node see that your transaction with same timestamp was added, they should remove it from the node\_pending\_transactions list to avoid it get processed more than 1 time.

"""

import requests import time import base64 import ecdsa

def wallet(): response = None

while response not in ["1", "2", "3"]:

response = input("""What do you want to do?

1. Generate new wallet
2. Send coins to another wallet
3. Check transactions\n""") if response == "1":

# Generate new wallet print("""=========================================\n

IMPORTANT: save this credentials or you won't be able to recover your wallet\n

=========================================\n""")

generate\_ECDSA\_keys() elif response == "2":

addr\_from = input("From: introduce your wallet address (public key)\n") private\_key = input("Introduce your private key\n")

addr\_to = input("To: introduce destination wallet address\n")

amount = input("Amount: number stating how much do you want to send\n") print("=========================================\n\n")

print("Is everything correct?\n")

print("From: {0}\nPrivate Key: {1}\nTo: {2}\nAmount: {3}\n".format(addr\_from, private\_key, addr\_to, amount))

response = input("y/n\n") if response.lower() == "y":

send\_transaction(addr\_from, private\_key, addr\_to, amount) else: # Will always occur when response == 3.

check\_transactions()

def send\_transaction(addr\_from, private\_key, addr\_to, amount):

"""Sends your transaction to different nodes. Once any of the nodes manage to mine a block, your transaction will be added to the blockchain. Despite that, there is a low chance your transaction gets canceled due to other nodes having a longer chain. So make sure your transaction is deep into the chain before claiming it as approved!

"""

# For fast debugging REMOVE LATER #

private\_key="181f2448fa4636315032e15bb9cbc3053e10ed062ab0b2680a37cd8cb51f53f2" # amount="3000"

#

addr\_from="SD5IZAuFixM3PTmkm5ShvLm1tbDNOmVlG7tg6F5r7VHxPNWkNKbzZfa+Jd KmfBAIhWs9UKnQLOOL1U+R3WxcsQ=="

#

addr\_to="SD5IZAuFixM3PTmkm5ShvLm1tbDNOmVlG7tg6F5r7VHxPNWkNKbzZfa+JdK mfBAIhWs9UKnQLOOL1U+R3WxcsQ=="

if len(private\_key) == 64:

signature, message = sign\_ECDSA\_msg(private\_key) url = 'http://localhost:5000/txion'

payload = {"from": addr\_from, "to": addr\_to, "amount": amount,

"signature": signature.decode(), "message": message}

headers = {"Content-Type": "application/json"}

res = requests.post(url, json=payload, headers=headers)

else:

print("Wrong address or key length! Verify and try again.")

def check\_transactions():

"""Retrieve the entire blockchain. With this you can check your

wallets balance. If the blockchain is to long, it may take some time to load. """

res = requests.get('http://localhost:5000/blocks') print(res.text)

def generate\_ECDSA\_keys():

"""This function takes care of creating your private and public (your address) keys.

It's very important you don't lose any of them or those wallets will be lost

forever. If someone else get access to your private key, you risk losing your coins.

private\_key: str

public\_ley: base64 (to make it shorter) """

sk = ecdsa.SigningKey.generate(curve=ecdsa.SECP256k1) #this is your sign (private key) private\_key = sk.to\_string().hex() #convert your private key to hex

vk = sk.get\_verifying\_key() #this is your verification key (public key) public\_key = vk.to\_string().hex()

#we are going to encode the public key to make it shorter public\_key = base64.b64encode(bytes.fromhex(public\_key))

filename = input("Write the name of your new address: ") + ".txt" with open(filename, "w") as f:

f.write("Private key: {0}\nWallet address / Public key: {1}".format(private\_key, public\_key.decode()))

print("Your new address and private key are now in the file {0}".format(filename))

def sign\_ECDSA\_msg(private\_key): """Sign the message to be sent private\_key: must be hex

Return

signature: base64 (to make it shorter) message: str

"""

# Get timestamp, round it, make it into a string and encode it to bytes message = str(round(time.time()))

bmessage = message.encode()

sk = ecdsa.SigningKey.from\_string(bytes.fromhex(private\_key), curve=ecdsa.SECP256k1) signature = base64.b64encode(sk.sign(bmessage))

return signature, message

if name == ' main ':

print(""" =========================================\n CRYPTOKEN - BLOCKCHAIN SYSTEM\n

=========================================\n\n

\n""") wallet()

input("Press ENTER to exit...")

#### Miner.py

#### import time

#### import hashlib

#### import json

#### import requests

#### import base64

#### from flask import Flask, request

#### from multiprocessing import Process, Pipe

#### import ecdsa

#### from miner\_config import MINER\_ADDRESS, MINER\_NODE\_URL, PEER\_NODES

#### node = Flask(\_\_name\_\_)

#### class Block:

#### def \_\_init\_\_(self, index, timestamp, data, previous\_hash):

#### """Returns a new Block object. Each block is "chained" to its previous

#### by calling its unique hash.

#### Args:

#### index (int): Block number.

#### timestamp (int): Block creation timestamp.

#### data (str): Data to be sent.

#### previous\_hash(str): String representing previous block unique hash.

#### Attrib:

#### index (int): Block number.

#### timestamp (int): Block creation timestamp.

#### data (str): Data to be sent.

#### previous\_hash(str): String representing previous block unique hash.

#### hash(str): Current block unique hash.

#### """

#### self.index = index

#### self.timestamp = timestamp

#### self.data = data

#### self.previous\_hash = previous\_hash

#### self.hash = self.hash\_block()

#### def hash\_block(self):

#### """Creates the unique hash for the block. It uses sha256."""

#### sha = hashlib.sha256()

#### sha.update((str(self.index) + str(self.timestamp) + str(self.data) + str(self.previous\_hash)).encode('utf-8'))

#### return sha.hexdigest()

#### def create\_genesis\_block():

#### """To create each block, it needs the hash of the previous one. First

#### block has no previous, so it must be created manually (with index zero

#### and arbitrary previous hash)"""

#### return Block(0, time.time(), {

#### "proof-of-work": 9,

#### "transactions": None},

#### "0")

#### # Node's blockchain copy

#### BLOCKCHAIN = [create\_genesis\_block()]

#### """ Stores the transactions that this node has in a list.

#### If the node you sent the transaction adds a block

#### it will get accepted, but there is a chance it gets

#### discarded and your transaction goes back as if it was never

#### processed"""

#### NODE\_PENDING\_TRANSACTIONS = []

#### def proof\_of\_work(last\_proof, blockchain):

#### # Creates a variable that we will use to find our next proof of work

#### incrementer = last\_proof + 1

#### # Keep incrementing the incrementer until it's equal to a number divisible by 9

#### # and the proof of work of the previous block in the chain

#### start\_time = time.time()

#### while not (incrementer % 7919 == 0 and incrementer % last\_proof == 0):

#### incrementer += 1

#### # Check if any node found the solution every 60 seconds

#### if int((time.time()-start\_time) % 60) == 0:

#### # If any other node got the proof, stop searching

#### new\_blockchain = consensus(blockchain)

#### if new\_blockchain:

#### # (False: another node got proof first, new blockchain)

#### return False, new\_blockchain

#### # Once that number is found, we can return it as a proof of our work

#### return incrementer, blockchain

#### def mine(a, blockchain, node\_pending\_transactions):

#### BLOCKCHAIN = blockchain

#### NODE\_PENDING\_TRANSACTIONS = node\_pending\_transactions

#### while True:

#### """Mining is the only way that new coins can be created.

#### In order to prevent too many coins to be created, the process

#### is slowed down by a proof of work algorithm.

#### """

#### # Get the last proof of work

#### last\_block = BLOCKCHAIN[-1]

#### last\_proof = last\_block.data['proof-of-work']

#### # Find the proof of work for the current block being mined

#### # Note: The program will hang here until a new proof of work is found

#### proof = proof\_of\_work(last\_proof, BLOCKCHAIN)

#### # If we didn't guess the proof, start mining again

#### if not proof[0]:

#### # Update blockchain and save it to file

#### BLOCKCHAIN = proof[1]

#### a.send(BLOCKCHAIN)

#### continue

#### else:

#### # Once we find a valid proof of work, we know we can mine a block so

#### # ...we reward the miner by adding a transaction

#### # First we load all pending transactions sent to the node server

#### NODE\_PENDING\_TRANSACTIONS = requests.get(url = MINER\_NODE\_URL + '/txion', params = {'update':MINER\_ADDRESS}).content

#### NODE\_PENDING\_TRANSACTIONS = json.loads(NODE\_PENDING\_TRANSACTIONS)

#### # Then we add the mining reward

#### NODE\_PENDING\_TRANSACTIONS.append({

#### "from": "network",

#### "to": MINER\_ADDRESS,

#### "amount": 1})

#### # Now we can gather the data needed to create the new block

#### new\_block\_data = {

#### "proof-of-work": proof[0],

#### "transactions": list(NODE\_PENDING\_TRANSACTIONS)

#### }

#### new\_block\_index = last\_block.index + 1

#### new\_block\_timestamp = time.time()

#### last\_block\_hash = last\_block.hash

#### # Empty transaction list

#### NODE\_PENDING\_TRANSACTIONS = []

#### # Now create the new block

#### mined\_block = Block(new\_block\_index, new\_block\_timestamp, new\_block\_data, last\_block\_hash)

#### BLOCKCHAIN.append(mined\_block)

#### # Let the client know this node mined a block

#### print(json.dumps({

#### "index": new\_block\_index,

#### "timestamp": str(new\_block\_timestamp),

#### "data": new\_block\_data,

#### "hash": last\_block\_hash

#### }) + "\n")

#### a.send(BLOCKCHAIN)

#### requests.get(url = MINER\_NODE\_URL + '/blocks', params = {'update':MINER\_ADDRESS})

#### def find\_new\_chains():

#### # Get the blockchains of every other node

#### other\_chains = []

#### for node\_url in PEER\_NODES:

#### # Get their chains using a GET request

#### block = requests.get(url = node\_url + "/blocks").content

#### # Convert the JSON object to a Python dictionary

#### block = json.loads(block)

#### # Verify other node block is correct

#### validated = validate\_blockchain(block)

#### if validated:

#### # Add it to our list

#### other\_chains.append(block)

#### return other\_chains

#### def consensus(blockchain):

#### # Get the blocks from other nodes

#### other\_chains = find\_new\_chains()

#### # If our chain isn't longest, then we store the longest chain

#### BLOCKCHAIN = blockchain

#### longest\_chain = BLOCKCHAIN

#### for chain in other\_chains:

#### if len(longest\_chain) < len(chain):

#### longest\_chain = chain

#### # If the longest chain wasn't ours, then we set our chain to the longest

#### if longest\_chain == BLOCKCHAIN:

#### # Keep searching for proof

#### return False

#### else:

#### # Give up searching proof, update chain and start over again

#### BLOCKCHAIN = longest\_chain

#### return BLOCKCHAIN

#### def validate\_blockchain(block):

#### """Validate the submitted chain. If hashes are not correct, return false

#### block(str): json

#### """

#### return True

#### @node.route('/blocks', methods=['GET'])

#### def get\_blocks():

#### # Load current blockchain. Only you should update your blockchain

#### if request.args.get("update") == MINER\_ADDRESS:

#### global BLOCKCHAIN

#### BLOCKCHAIN = b.recv()

#### chain\_to\_send = BLOCKCHAIN

#### # Converts our blocks into dictionaries so we can send them as json objects later

#### chain\_to\_send\_json = []

#### for block in chain\_to\_send:

#### block = {

#### "index": str(block.index),

#### "timestamp": str(block.timestamp),

#### "data": str(block.data),

#### "hash": block.hash

#### }

#### chain\_to\_send\_json.append(block)

#### # Send our chain to whomever requested it

#### chain\_to\_send = json.dumps(chain\_to\_send\_json)

#### return chain\_to\_send

#### @node.route('/txion', methods=['GET', 'POST'])

#### def transaction():

#### """Each transaction sent to this node gets validated and submitted.

#### Then it waits to be added to the blockchain. Transactions only move

#### coins, they don't create it.

#### """

#### if request.method == 'POST':

#### # On each new POST request, we extract the transaction data

#### new\_txion = request.get\_json()

#### # Then we add the transaction to our list

#### if validate\_signature(new\_txion['from'], new\_txion['signature'], new\_txion['message']):

#### NODE\_PENDING\_TRANSACTIONS.append(new\_txion)

#### # Because the transaction was successfully

#### # submitted, we log it to our console

#### print("New transaction")

#### print("FROM: {0}".format(new\_txion['from']))

#### print("TO: {0}".format(new\_txion['to']))

#### print("AMOUNT: {0}\n".format(new\_txion['amount']))

#### # Then we let the client know it worked out

#### return "Transaction submission successful\n"

#### else:

#### return "Transaction submission failed. Wrong signature\n"

#### # Send pending transactions to the mining process

#### elif request.method == 'GET' and request.args.get("update") == MINER\_ADDRESS:

#### pending = json.dumps(NODE\_PENDING\_TRANSACTIONS)

#### # Empty transaction list

#### NODE\_PENDING\_TRANSACTIONS[:] = []

#### return pending

#### def validate\_signature(public\_key, signature, message):

#### """Verifies if the signature is correct. This is used to prove

#### it's you (and not someone else) trying to do a transaction with your

#### address. Called when a user tries to submit a new transaction.

#### """

#### public\_key = (base64.b64decode(public\_key)).hex()

#### signature = base64.b64decode(signature)

#### vk = ecdsa.VerifyingKey.from\_string(bytes.fromhex(public\_key), curve=ecdsa.SECP256k1)

#### # Try changing into an if/else statement as except is too broad.

#### try:

#### return vk.verify(signature, message.encode())

#### except:

#### return False

#### def welcome\_msg():

#### print(""" =========================================\n

#### CRYPTOKEN - BLOCKCHAIN SYSTEM\n

#### =========================================\n\n

#### \n""")

#### if \_\_name\_\_ == '\_\_main\_\_':

#### welcome\_msg()

#### # Start mining

#### a, b = Pipe()

#### p1 = Process(target=mine, args=(a, BLOCKCHAIN, NODE\_PENDING\_TRANSACTIONS))

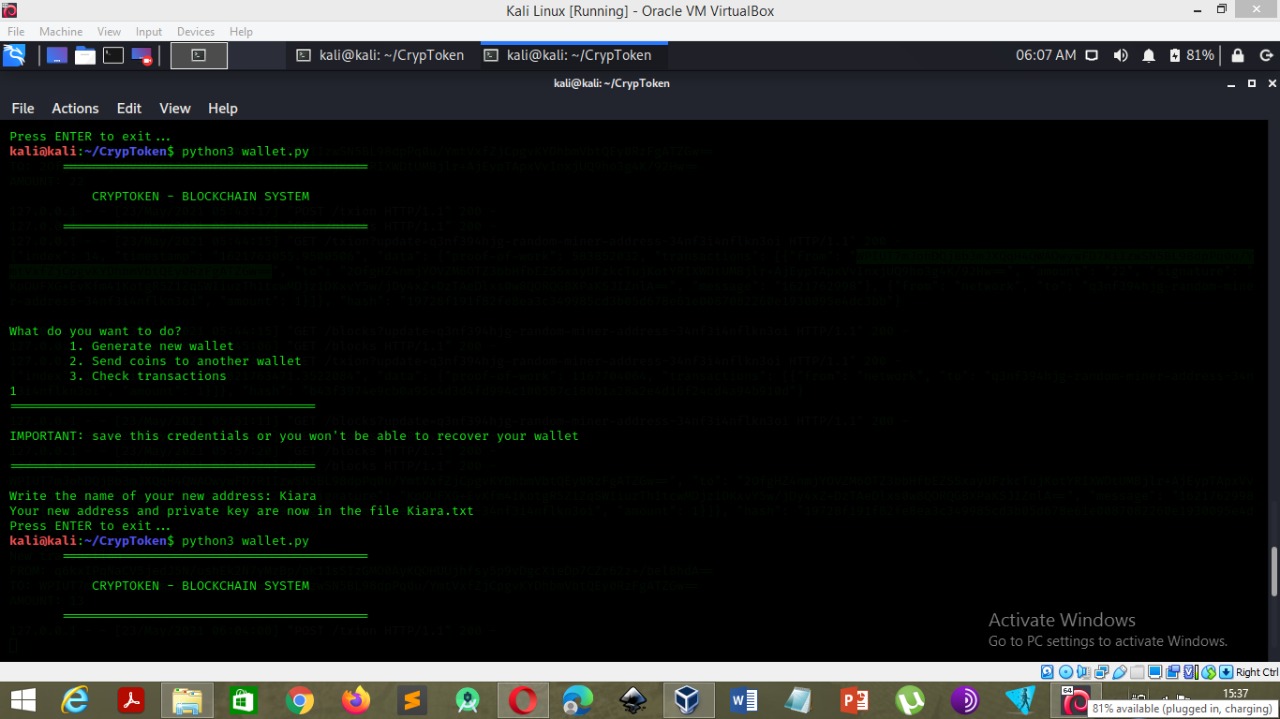
#### p1.start()

#### # Start server to receive transactions

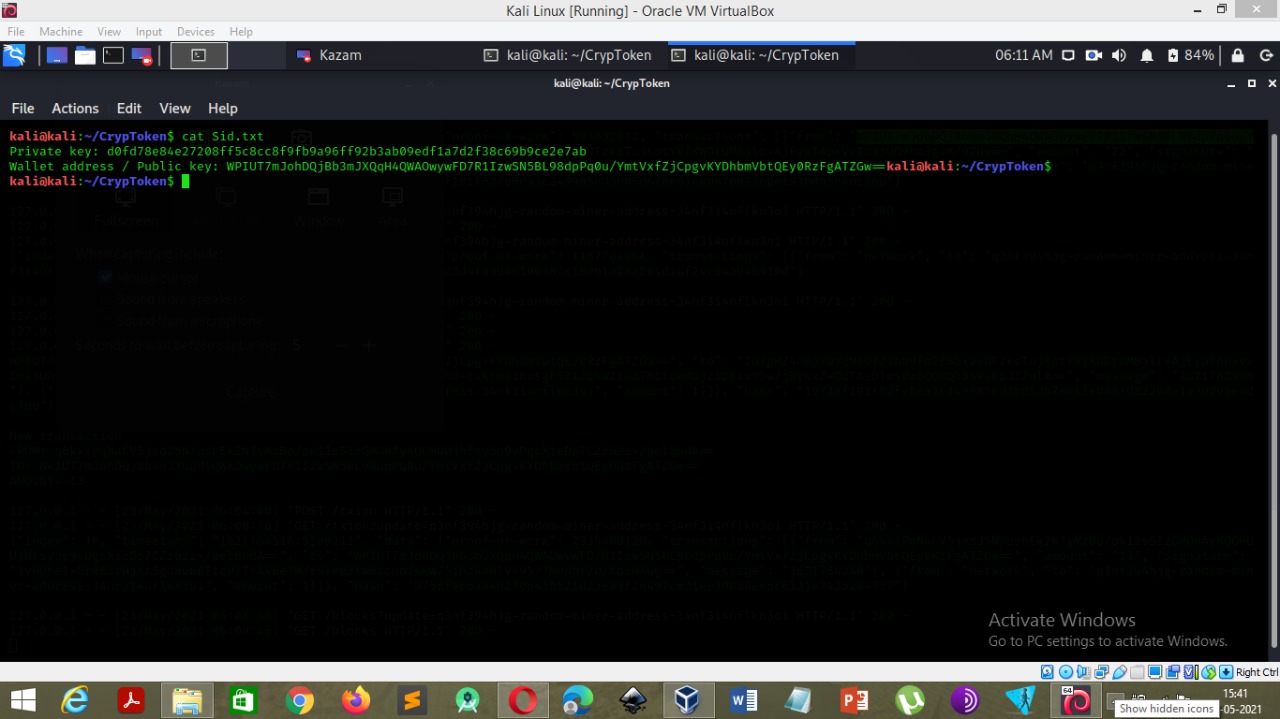
#### p2 = Process(target=node.run(), args=b)

#### p2.start()

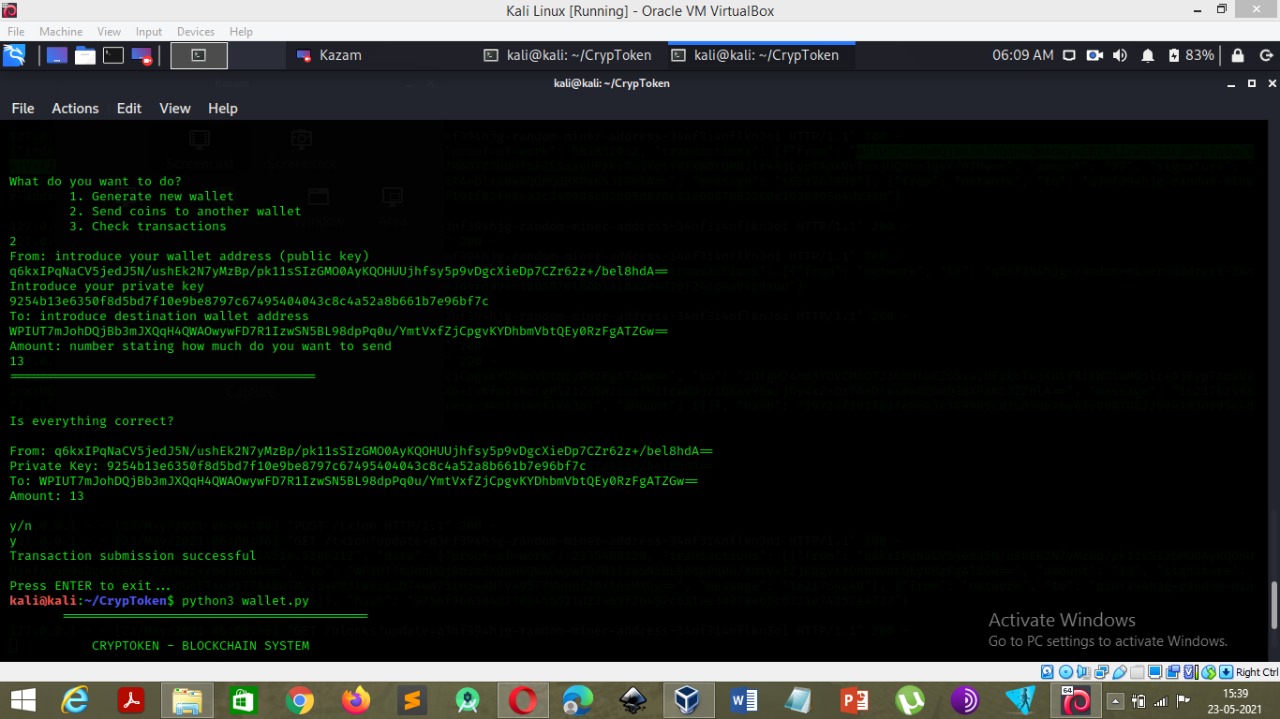
# CHAPTER 6: RESULTS



**Fig 01**: Generate new wallet: It generates the new wallet when you click on option 1, while creating a wallet it creates public key/ wallet address and private key, both of these are 256 bit generated hex code. Further the private is converted into base 64 in order to minimize the key length.

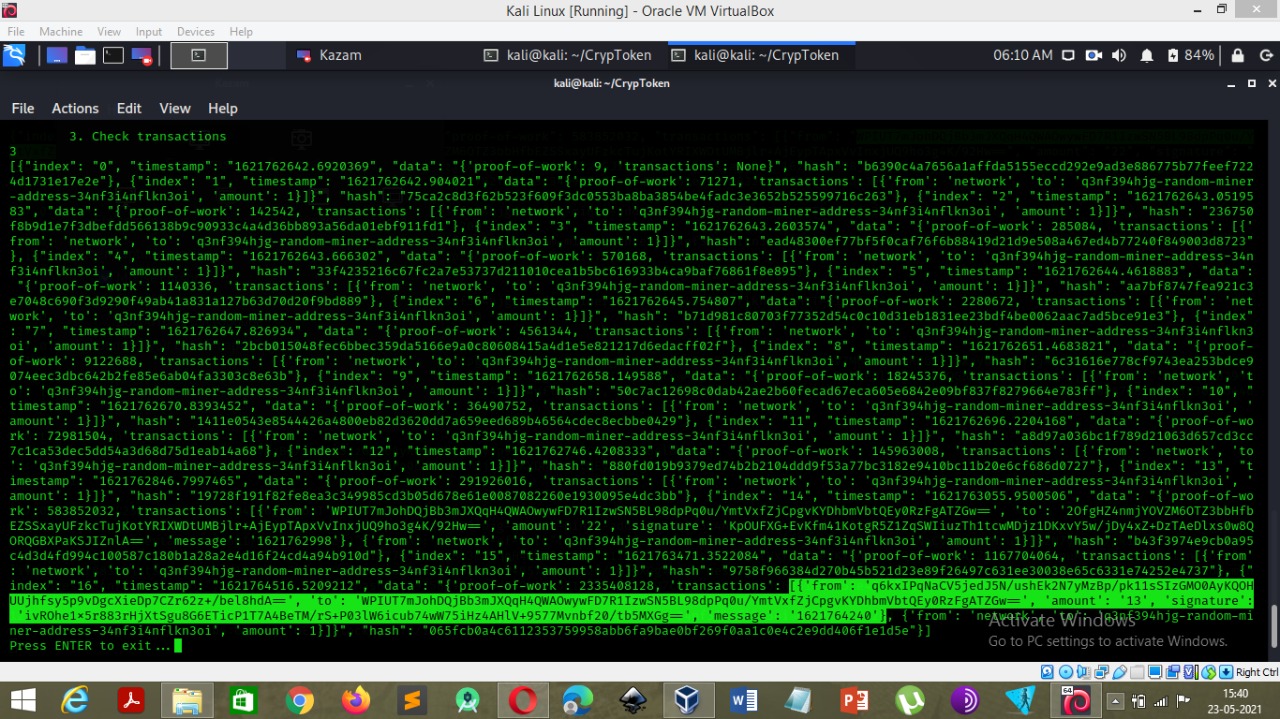


**Fig 02:** This image shows the credentials of the wallet generated. A private key as well as a public key / wallet address is generated in a txt file whenever a new wallet is generated.



**Fig 03**: Send coins to another wallet: For ease of demonstration purposes we assume that whenever a wallet is generated there are indefinite mount of coins for disposition.

Firstly we have to enter the sender’s wallet address and in order to authenticate that the sender is legitimate he/she should introduce a private key. Then introduce the destination wallet address and enter the amount to be sent. Then you get a prompt to verify the details and once its done the transaction is done instantaneously.



**Fig 04**: Check Transactions: The 3rd option allows us to check the transactions which has been performed by the sender. The highlighted transaction gives us information of the entire transaction which has been stored in a block. It consists of the senders and receivers addresses as well as the amount and the digital signature of the transaction.

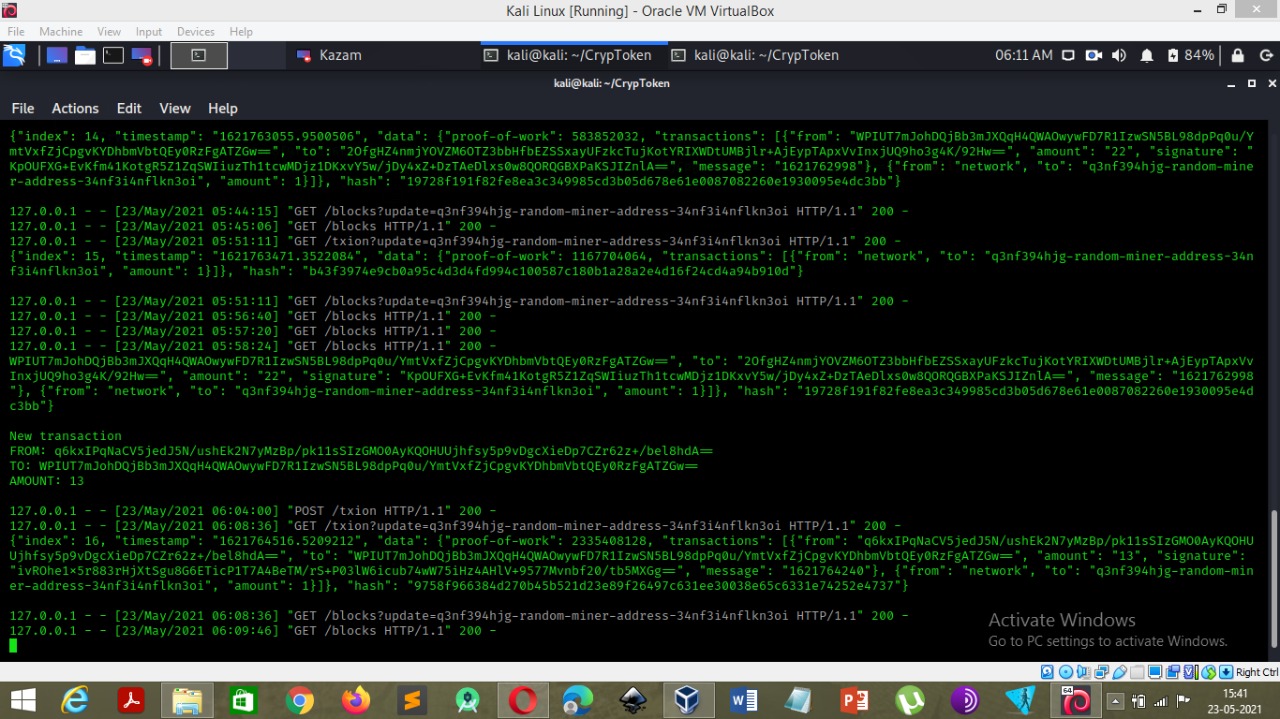


Fig 05: This image shows the recent transaction which has been mined by the miner node. The transaction which is illustrated is the recent block which is added to the blockchain.

# CHAPTER 7: CONCLUSION

Blockchain is a revolutionary technology which has changed the way people interact with the Internet. In this cyber world, where nothing is private, and no data is safe, Blockchain has shown promising potentials of being the best bet of people dealing with the value-sensitive commodities.

This project addresses the shortcomings of the present transactional system and also gives us a sneak peek into the transactions carried out in the Blockchain system. By doing so we have gained knowledge in the application of Blockchain for transactional application.

Blockchain technology stands to revolutionize the way money can be handled. This technology efficiently bypasses the need to have an arbiter and allows people to move and collect their money without the involvement of banks and other financial entities. This is, in and of itself revolutionary because it provides an avenue for people who wouldn’t otherwise have access to the global economy.

The number of digital wallet apps has exploded from 9.2 million to over 17.5 million. This gives us a glimpse of the future possibilities that cryptocurrency can have on Cryptocurrency Wallet Application Development.

The future of blockchain technology will continue to drive cryptocurrency wallet application development. This technology adds security, transparency, and convenience to many financial transactions. Plus, it’s applications continue to expand.

# CHAPTER 8: REFERENCES

1. ht[tps://www.freecodecamp.org/news/tag/blockchain](http://www.freecodecamp.org/news/tag/blockchain)
2. https://en.wikipedia.org/wiki/Blockchain
3. ht[tps://www.investopedia.com/terms/b/blockchain.asp](http://www.investopedia.com/terms/b/blockchain.asp)
4. ht[tps://www.blockcypher.com/dev/bitcoin/#introduction](http://www.blockcypher.com/dev/bitcoin/#introduction)
5. ht[tps://www.computerworld.com/article/3191077/what-is-blockchain-the-](http://www.computerworld.com/article/3191077/what-is-blockchain-the-) completeguide.
6. https://onezero.medium.com/how-does-the-blockchain-work-98c8cd01d2ae
7. https://bitcoin.org/en/
8. ht[tps://www.pwc.com/us/en/industries/financial-services/fintech/bitcoin-](http://www.pwc.com/us/en/industries/financial-services/fintech/bitcoin-) blockchaincryptocurrency.html
9. https://blockchainhub.net/blockchain-glossary/
10. ht[tps://www.irjet.net/archives/V5/i10/IRJET-V5I10284.pdf](http://www.irjet.net/archives/V5/i10/IRJET-V5I10284.pdf)
11. Understanding Blockchain Technology Simanta Shekhar Sarmah, Business Intelligence Architect, Alpha Clinical Systems, USA
12. The Architectural Design Requirements of a Blockchain-Based Port Community System Elnaz Irannezhad
13. Blockchain Technology for the Advancement of the Future, Quoc Khanh Nguyen, Quang Vang Dang
14. Research on network security protection technology of energy industry based on blockchain
15. Blockchain Technology: Practical P2P Computing
16. Blockchain based Data Security for Financial Transaction System, *Sujatha Kumari B A* Department of Electronics and Communication, *Sadaf Farheen* Department of Electronics and Communication
17. A Cluster Based System for Analyzing Ethereum Blockchain Transaction Data

BaranKılıc ,CanO turanandAlperSen Department of Computer Engineering

1. College Fees Transaction Using Hash Functions of Blockchain Model Aswini.R1, Kiruba.K2 Assistant Professor1, PG Scholar2 Department of Computer Science and Engineering

# SURVEY PAPER

