IJIRCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 9, Issue 6, June 2021

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 7.542

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.542 |

|| Volume 9, Issue 6, June 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0906273 |

Secured E-Transactions Using Blockchain Technology

E.Sri Divya, A.Yoga Vyshnavi, Ch.Sri Valli, B.Chandana, B. Rechal Priyanka

UG Student, Department of Electronics and Communication Engineering, Vasireddy Venkatadri Institute of Technology, Guntur, AP, India

ABSTRACT: Now–a day's Electronic transactions has become a common ,everyone purchases products over the internet and also pay the bill ,which becoming an easier and every day necessity, secured methods of payment gateway for such online purchase product have become very important so that it need to protect from unauthorized use. Therefore security for such transactions is necessary because its confidential data like credit card numbers, pin numbers and other information for that security is required. Blockchain technology is utilized to build a platform that secures theft's identity and drastically reduces fraudulent activity. The technology is also possible to assist businesses in developing strong blockchains capable of dealing with authentication and reconciliation issues that arise in a variety of industries. Furthermore, it can give people the ability to construct identities that are encrypted can serve as a substitute for various passwords and login. The Raspberry Pi 3 Model B is used to power the entire system.

KEYWORDS: Block chain, Authentication, digital identification, Security, Raspberry pi and transactions.

I. INTRODUCTION

"Block chain Identity Management is a decentralized and secure system that reclaims control for users through a distributed trust model." Several sectors benefit from block chain technology because of its transparency, security, and other aspects, which bring value to their company. As a result, it's safe to assume that it's on track to transform the existing state of identity management in a highly secure manner. The current identity management system is insecure and untrustworthy. At every turn, you'll be required to identify yourself using a variety of government-issued identification cards, such as a voter ID, passport, or Pan Card. Privacy risks and data breaches arise when numerous IDs are shared. As a result, through decentralized networks, the block chain can pave the way to self-sovereign identity. Identity papers are guarded, verified, and authorized by permissioned participants in a self-sovereign identity, which ensures privacy and trust.

II. CONTEMPORARY TECHNOLOGY'S DIFFICULTIES

THEFT OF PERSONAL INFORMATION:

Identity theft occurs when people share their personal information online through unknown sources or use services that allow their identification documents to fall into the wrong hands. Furthermore, online applications use centralized servers to store data, Hackers will find it easy to gain access to the servers and steal important data.

MULTIFACTOR AUTHENTICATION:

When signing up for different online platforms, Each time, users must generate a new login and password.It is tough for an person to recall a username and password combination for accessing several services. Maintaining several authentication profiles is a difficult task.

INABILITY TO CONTROL:

Users now have no control over their information personally identifiable due to a lack of control (PII). They have no idea how many times their private information has been shared or utilised without their permission. Furthermore, most people are unaware of where all of their personal data is held. As a result, an innovative adjustment to the current state of identity administration approach is required.

number of hops will help in reducing the range of the transmission power. Route discovery has been done in the same way as being done in on-demand routing algorithms. After packet has been reached to the destination, destination will wait for time δt and collects all the packets. After time δt it calls the optimization function to select the path and send RREP. Optimization function uses the individual node's battery energy; if node is having low energy level then optimization function will not use that node.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.542 |

|| Volume 9, Issue 6, June 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0906273 |

III. THE IMPACT OF BLOCKCHAIN ON THE SECURITY OF E TRANSACTIONS

Ethereum is a blockchain platform that is open source. We can construct, deploy, and access the Ethereum blockchain using ethereum clients. Different architectural components work together to create a fully functional ethereum network that is globally dispersed and decentralised, with no central authority. Ethereum Virtual Machine (EVM), miner, block, transaction, consensus algorithm, account, smart contract, mining, Ether, and gas are some of the most significant ethereum components.

Because blockchain uses a decentralised ledger to store digital assets that are protected by strong cryptographic keys, hackers will find it difficult to change the information. On the basis of blockchain network, the data is kept on numerous computers. In order to gain access to critical data assets, attackers must infiltrate every system on the network, which is nearly impossible. Even if hackers get access to the network, any modifications they make to the data will be reflected in the systems, which will alert all participants. All of this distinguishes blockchain as a one-of-a-kind solution to security challenges.

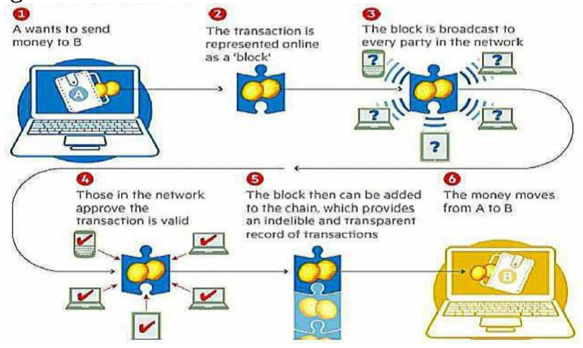

1 A wants to send money to B
2 The transaction is represented online as a 'block'
3 The block is broadcast to every party in the network
4 Those in the network approve the transaction is valid
5 The block then can be added to the chain, which provides an indelible and transparent record of transactions
6 The money moves from A to B

Figure1: Example of a transaction in Blockchain

IV. BLOCKCHAIN TECHNOLOGY

A blockchain is a series of blocks that each contain a legitimate transaction. The hash to the previous block in the blockchain is included in each block. It operates on a peer-to-peer network, which means that every node in thwe network is linked to every other node.The deal is transmitted to the network and added to everyone's copy of the blockchain once it has been confirmed.

A block refers to files that permanently store data related to the blockchain network. A block is comparable to the pages of a ledger or a book of accounts. When a block is finished, it makes room for the next block. The data in blocks cannot be changed.

Blockchains are distributed (i.e., without a single repository) and usually decentralised digital ledgers that are tamper obvious and resistant to tampering (i.e., a bank, company or government). At their most basic level, they allow a group of users to record transactions in a shared ledger within that group, with the result that no transaction can be modified once it has been published, as long as the blockchain network is operational.

V. ETHEREUM'S BLOCKCHAIN

Ethereum is a software platform based in blockchain used primarily to support the world's second-largest cryptocurrency, after Bitcoin, in terms of market value. Ethereum Blockchain stores the entire history of all smart contracts. The Ethereum's blockchain is structured similarly to the Bitcoin blockchain in that it is a data base that everyone has a access to all smart contracts and transaction history. Hundreds of people volunteered from all across the world have volunteered to keep a backup of the Ethereum blockchain, which is quite large. This is just one of the characteristics that distinguishes Ethereum as a decentralised platform. In Ethereum's network, these are all referred to as a "node." When an smart contract on ethereum is used, thousands of computers, are processed through a network which ensures that the user is abiding by the regulations. These nodes are all linked together.

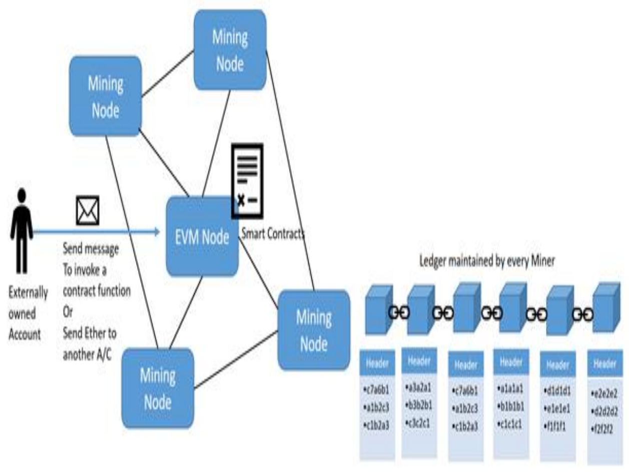


Fig.1. Ethereum's Architecture

Here's a rundown of what each node contain:

Accounts: Every user can create an account, which displays their Ether balance. Ethereum contains smart contracts, which outline the rules that must be followed in order for funds to be moved and unlocked.

ETHER:

Ethereum apps and smart contracts both require money in the form of ether, Ethereum's native coin. Ether is required for almost everything on Ethereum, and it's referred to as "gas" when it's utilised to execute smart contracts on the network. Ethereum can be used to invoke smart contracts. To hold the ether, Ethereum uses accounts, which are similar to bank accounts.

Two kinds of accounts exist:

Externally owned accounts (EOAs): These are accounts that are not owned by the company. Normal users' accounts for storing and sending ether.

Contract accounts: These are the accounts where smart contracts are stored, which can be activated by ether transactions from EOAs or other events.

ETHEREUM WALLETS:

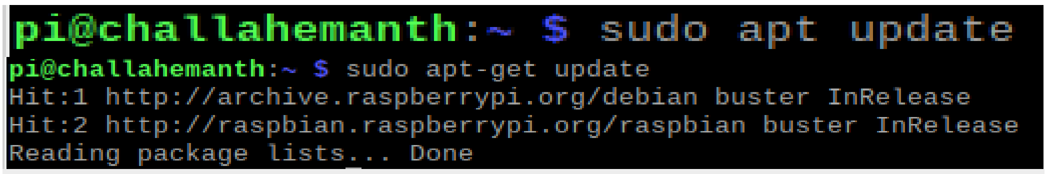
An Ethereum wallet stores a user's private keys, which are secret keys that can be used to access ether. Every single key is a long, jumbled string of characters as well as numbers that appear to be something like this:
073d9dbe8875e7c91422d80413c85ba5e8e9fe7cad5dc001871dac882d07f2f

Only the owners of the private keys can spend money associated with the keys. These days, it's all about ethereum wallets. For storing private keys, there are a few distinct types of Ethereum wallets:

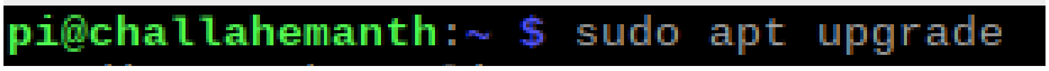
- Wallets for the desktop
- Wallets for mobile devices
- Wallets made of metal
- Wallets made of paper

SYSTEM DESIGN:

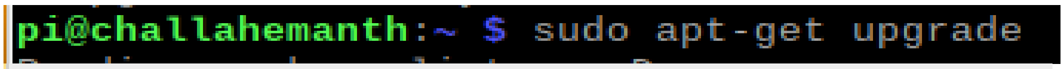
Assume we've already installed Raspbian and are now updating and installing packaged software to the most recent versions.



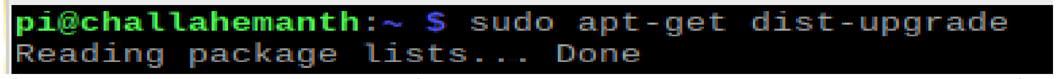
```
pi@challahemanth:~ $ sudo apt update
pi@challahemanth:~ $ sudo apt-get update
Hit:1 http://archive.raspberrypi.org/debian buster InRelease
Hit:2 http://raspbian.raspberrypi.org/raspbian buster InRelease
Reading package lists... Done
```



```
pi@challahemanth:~ $ sudo apt upgrade
```

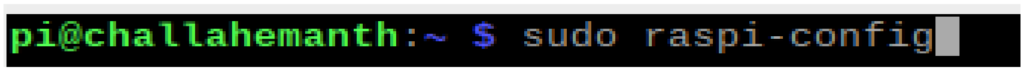


```
pi@challahemanth:~ $ sudo apt-get upgrade
```



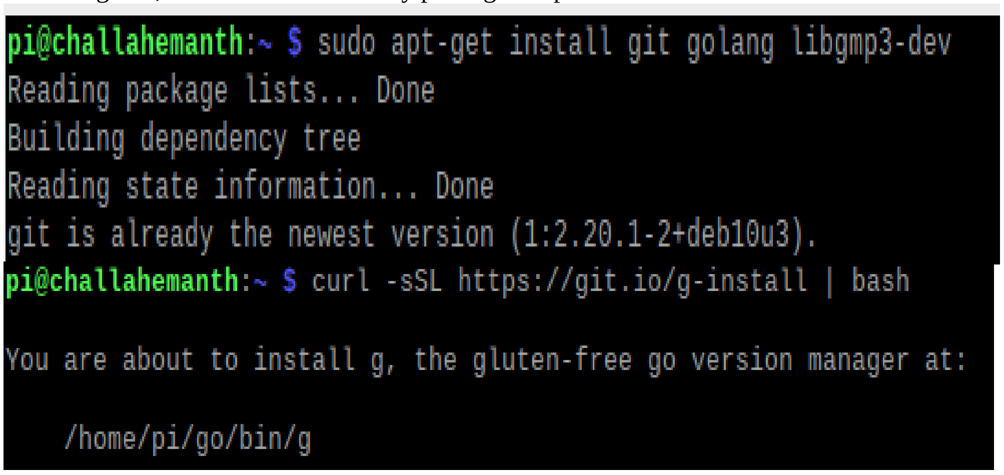
```
pi@challahemanth:~ $ sudo apt-get dist-upgrade
Reading package lists... Done
```

If you don't need a graphical user interface, set your machine to boot to the command line interface



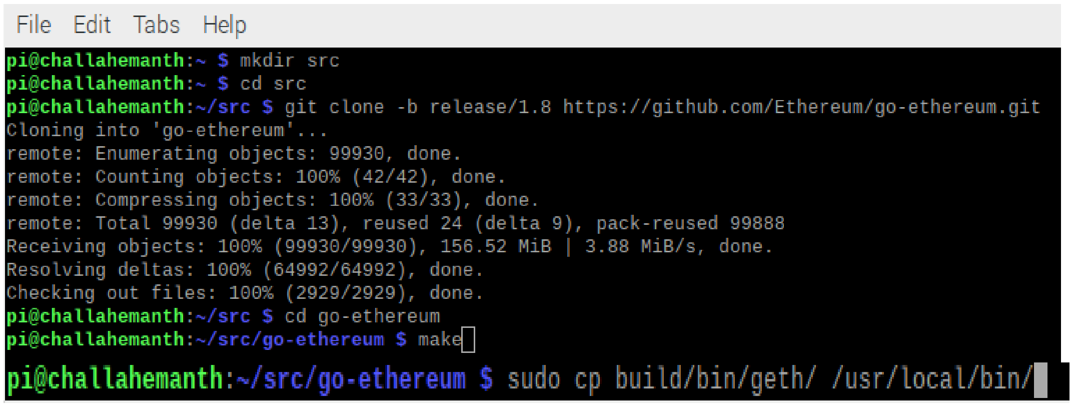
```
pi@challahemanth:~ $ sudo raspi-config
```

Following that, we instal the necessary packaged dependencies.



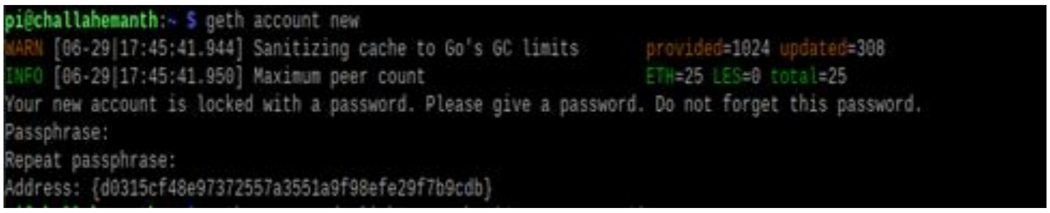
```
pi@challahemanth:~ $ sudo apt-get install git golang libgmp3-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
git is already the newest version (1:2.20.1-2+deb10u3).
pi@challahemanth:~ $ curl -sSL https://git.io/g-install | bash

You are about to install g, the gluten-free go version manager at:

    /home/pi/go/bin/g
```

Then we download the sources for geth, the official Go language implementation of an Ethereum node, compile it, and copy the executable to/usr/local/bin/.



```
pi@challahemanth:~ $ mkdir src
pi@challahemanth:~ $ cd src
pi@challahemanth:~/src $ git clone -b release/1.8 https://github.com/Ethereum/go-ethereum.git
Cloning into 'go-ethereum'...
remote: Enumerating objects: 99930, done.
remote: Counting objects: 100% (42/42), done.
remote: Compressing objects: 100% (33/33), done.
remote: Total 99930 (delta 13), reused 24 (delta 9), pack-reused 99888
Receiving objects: 100% (99930/99930), 156.52 MiB | 3.88 MiB/s, done.
Resolving deltas: 100% (64992/64992), done.
Checking out files: 100% (2929/2929), done.
pi@challahemanth:~/src $ cd go-ethereum
pi@challahemanth:~/src/go-ethereum $ make
pi@challahemanth:~/src/go-ethereum $ sudo cp build/bin/geth/ /usr/local/bin/
```

Make an account and begin testing:
First, let's see whether we can establish a new account with geth.



```
pi@challahemanth:~ $ geth account new
WARN [06-29|17:45:41.944] Sanitizing cache to Go's GC limits       provided=1024 updated=308
INFO [06-29|17:45:41.950] Maximum peer count                       ETH=25 LES=0 total=25
Your new account is locked with a password. Please give a password. Do not forget this password.
Passphrase:
Repeat passphrase:
Address: {d0315cf48e97372557a3551a9f98efe29f7b9cdb}
```

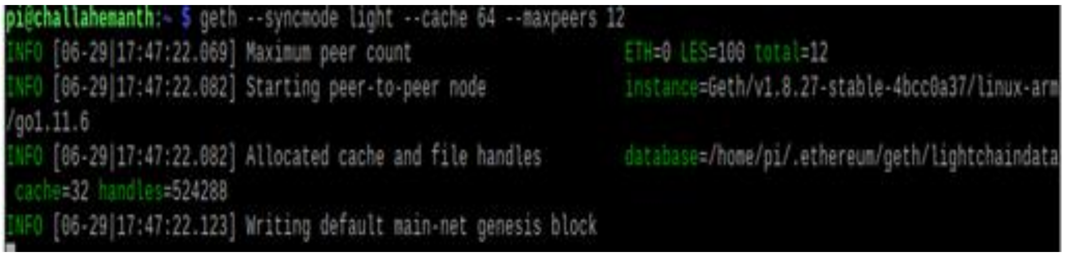
As shown in illustration, a new set of cryptographic keys will be generated for the newly established account. The private keys are secured with a password. It's worth noting that if you're going to use this account to mine bit coin and transact in any way, you'll want to back up your keys and protect your private key from being accessible.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.542 |

|| Volume 9, Issue 6, June 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0906273 |

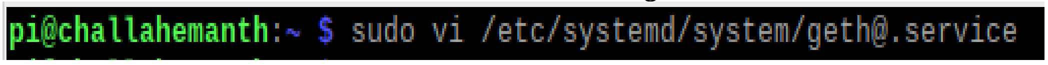
CONNECTING OUR PI BOARD TO MAIN NET:



```
pi@challahemanth:~ $ geth --syncmode light --cache 64 --maxpeers 12
INFO [06-29|17:47:22.069] Maximum peer count                       ETH=0 LES=100 total=12
INFO [06-29|17:47:22.082] Starting peer-to-peer node               instance=Geth/v1.8.27-stable-4bcc0a37/linux-arm
/go1.11.6
INFO [06-29|17:47:22.082] Allocated cache and file handles         database=/home/pi/.ethereum/geth/lightchaindata
 cache=32 handles=524288
INFO [06-29|17:47:22.123] Writing default main-net genesis block
```

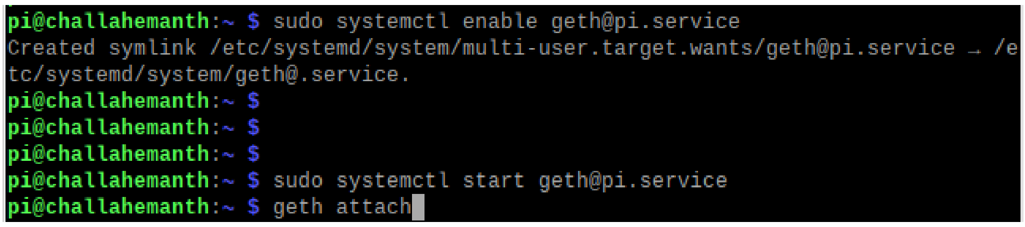
If we execute it without any arguments we would start 11 a node and try and synchronise the entire main network block chain. Which, at more than 50GB in size and expanding all the time, might not be the best option for an embedded computer? As a consequence, the node is started in light synchronisation mode. This just retrieves block headers as they appear, as well as other portions of the block chain as they become available.

With the assistance of this vi editor, we can now enable and start geth service on our node:



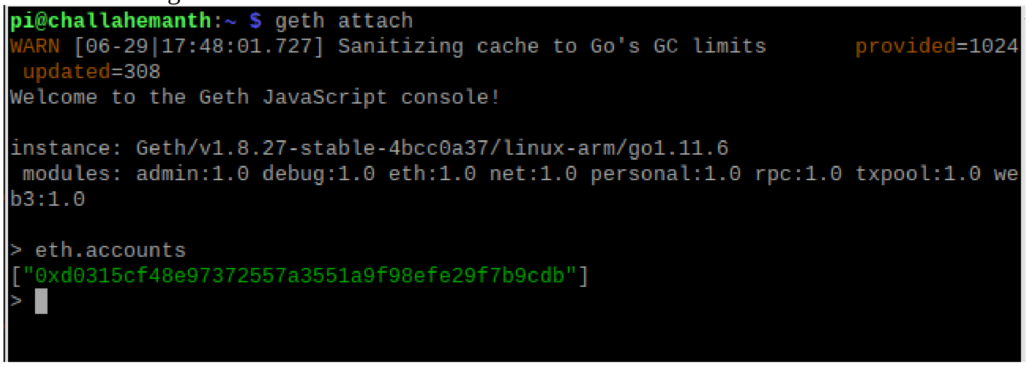
```
pi@challahemanth:~ $ sudo vi /etc/systemd/system/geth@.service
```

To run the Ethereum node as the "pi" user, do the following: It's used to make a service start when the system boots up.



```
pi@challahemanth:~ $ sudo systemctl enable geth@pi.service
Created symlink /etc/systemd/system/multi-user.target.wants/geth@pi.service → /e
tc/systemd/system/geth@.service.
pi@challahemanth:~ $
pi@challahemanth:~ $
pi@challahemanth:~ $
pi@challahemanth:~ $ sudo systemctl start geth@pi.service
pi@challahemanth:~ $ geth attach
```

It's utilized to restart a service that's been turned off. We can now connect to our Ethereum node that is running as a service using



```
pi@challahemanth:~ $ geth attach
WARN [06-29|17:48:01.727] Sanitizing cache to Go's GC limits       provided=1024
 updated=308
Welcome to the Geth JavaScript console!

instance: Geth/v1.8.27-stable-4bcc0a37/linux-arm/go1.11.6
 modules: admin:1.0 debug:1.0 eth:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0 we
b3:1.0

> eth.accounts
["0xd0315cf48e97372557a3551a9f98efe29f7b9cdb"]
>
```

This offers us a JavaScript console that we can interact with. We can call functions from here which will give you a list of your current accounts.

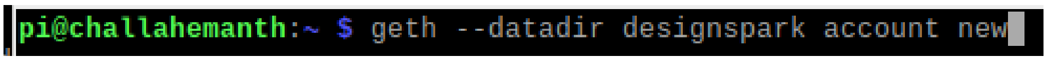
```
> eth.accounts
```

It should be noted that the light client protocol is still in its infancy, is somewhat experimental, and is dependent on full peers/nodes enabling support for it. As a result, it may not be totally practicable to transact on the Ethereum main-net block chain during the time of writing. However, things are changing quickly, and this situation might easily change in the near future.

Now we'll build a private block chain and use it to delve deeper into Ethereum. Because we will be starting with a brand new block chain, We can sync it completely and do not need to set nodes in the light sync mode.

Signing up for a new account:

We need a name for our new blockchain network, and for this example, we'll call it DesignSpark. Ethereum stores data by default in a subdirectory of your home directory called.ethereum, which is a hidden directory on Linux/BSD. To keep our private blockchain data separate, we'll use.designspark, as shown in Figure

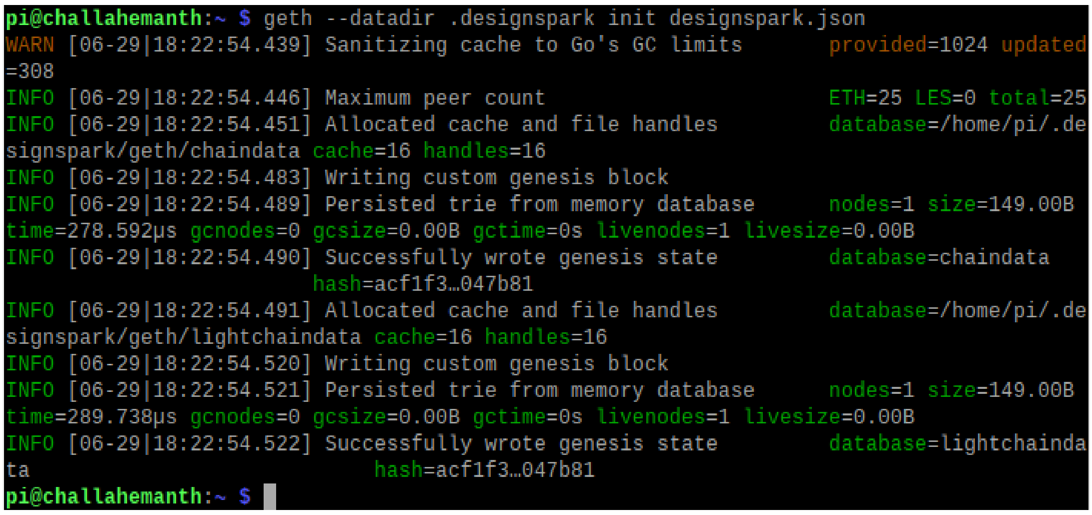


```
pi@challahemanth:~ $ geth --datadir designspark account new
```

If we start with a new account:

Make a note of the account's address, as we will need it when we initialise the new network if we want to pre-allocate any funds to it.

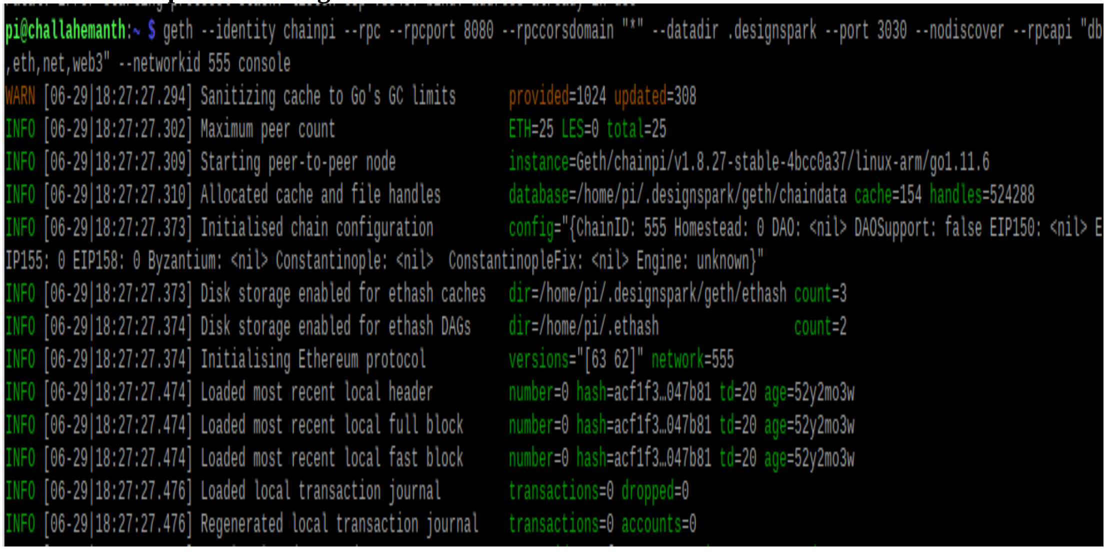
There was at Block zero at the start



```
pi@challahemanth:~ $ geth --datadir .designspark init designspark.json
WARN [06-29|18:22:54.439] Sanitizing cache to Go's GC limits       provided=1024 updated
=308
INFO [06-29|18:22:54.446] Maximum peer count                       ETH=25 LES=0 total=25
INFO [06-29|18:22:54.451] Allocated cache and file handles         database=/home/pi/.de
signspark/geth/chaindata cache=16 handles=16
INFO [06-29|18:22:54.483] Writing custom genesis block
INFO [06-29|18:22:54.489] Persisted trie from memory database      nodes=1 size=149.00B
time=278.592µs gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [06-29|18:22:54.490] Successfully wrote genesis state         database=chaindata
                         hash=acf1f3…047b81
INFO [06-29|18:22:54.491] Allocated cache and file handles         database=/home/pi/.de
signspark/geth/lightchaindata cache=16 handles=16
INFO [06-29|18:22:54.520] Writing custom genesis block
INFO [06-29|18:22:54.521] Persisted trie from memory database      nodes=1 size=149.00B
time=289.738µs gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [06-29|18:22:54.522] Successfully wrote genesis state         database=lightchainda
ta                            hash=acf1f3…047b81
pi@challahemanth:~ $
```

There must be a first link in a chain, and a blockchain is no exception, requiring the generation of a genesis block that will be used by the initial set of nodes that will participate in the network. This is accomplished by using a JSON file and the contents of the one that we used.

Getting the first node up and running:



```
pi@challahemanth:~ $ geth --identity chainpi --rpc --rpcport 8080 --rpccorsdomain "*" --datadir .designspark --port 3030 --nodiscover --rpcapi "db
,eth,net,web3" --networkid 555 console
WARN [06-29|18:27:27.294] Sanitizing cache to Go's GC limits       provided=1024 updated=308
INFO [06-29|18:27:27.302] Maximum peer count                       ETH=25 LES=0 total=25
INFO [06-29|18:27:27.309] Starting peer-to-peer node               instance=Geth/chainpi/v1.8.27-stable-4bcc0a37/linux-arm/go1.11.6
INFO [06-29|18:27:27.310] Allocated cache and file handles         database=/home/pi/.designspark/geth/chaindata cache=154 handles=524288
INFO [06-29|18:27:27.373] Initialised chain configuration          config="{ChainID: 555 Homestead: 0 DAO: <nil> DAOSupport: false EIP150: <nil> E
IP155: 0 EIP158: 0 Byzantium: <nil> Constantinople: <nil>  ConstantinopleFix: <nil> Engine: unknown}"
INFO [06-29|18:27:27.373] Disk storage enabled for ethash caches   dir=/home/pi/.designspark/geth/ethash count=3
INFO [06-29|18:27:27.374] Disk storage enabled for ethash DAGs     dir=/home/pi/.ethash                  count=2
INFO [06-29|18:27:27.374] Initialising Ethereum protocol           versions="[63 62]" network=555
INFO [06-29|18:27:27.474] Loaded most recent local header          number=0 hash=acf1f3…047b81 td=20 age=52y2mo3w
INFO [06-29|18:27:27.474] Loaded most recent local full block      number=0 hash=acf1f3…047b81 td=20 age=52y2mo3w
INFO [06-29|18:27:27.474] Loaded most recent local fast block      number=0 hash=acf1f3…047b81 td=20 age=52y2mo3w
INFO [06-29|18:27:27.476] Loaded local transaction journal         transactions=0 dropped=0
INFO [06-29|18:27:27.476] Regenerated local transaction journal    transactions=0 accounts=0
```

Adding a second node:

A single node in a block chain network would be ineffective, so we'll build a second one, as shown in figure. This time, it is recommended that we use a computer with a bit more RAM, such as a laptop or desktop running Debian/Ubuntu, as this will most likely be required if we ever want to run a miner at some point.

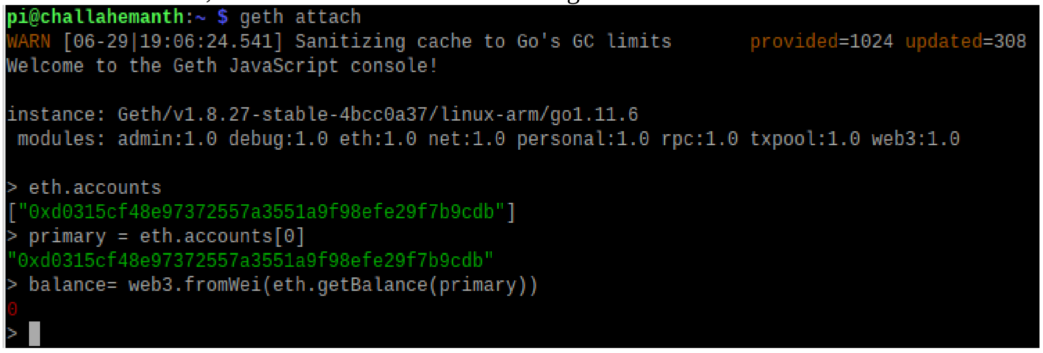
To summarize, the steps are as follows:

1. Set up geth.
2. Execute the above command to sign up for a new account.
3. Use the same JSON configuration file to get started.
4. Restart the node, but this time with a different identity!

After we've started the node and dropped it into the JavaScript console, we can check the new account and its balance again with:

Balance verification:

Once you have entered the console, we can use eth.accounts and eth.get Balance to check the balance



```
pi@challahemanth:~ $ geth attach
WARN [06-29|19:06:24.541] Sanitizing cache to Go's GC limits       provided=1024 updated=308
Welcome to the Geth JavaScript console!

instance: Geth/v1.8.27-stable-4bcc0a37/linux-arm/go1.11.6
 modules: admin:1.0 debug:1.0 eth:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0 web3:1.0

> eth.accounts
["0xd0315cf48e97372557a3551a9f98efe29f7b9cdb"]
> primary = eth.accounts[0]
"0xd0315cf48e97372557a3551a9f98efe29f7b9cdb"
> balance= web3.fromWei(eth.getBalance(primary))
0
> 
```

We should see a balance of 0 this time due to the fact that we did not pre-allocate any funds to the account.

VI. Conclusion

In order to conduct safe digital transactions, identity management is critical. We propose a mechanism for managing identities based on blockchain technology in this paper. Blockchain technology is the technology of the future, capable of dealing with security issues effectively. End-to-end traceability, resulting from the immutability of data records, authentication, and permission are built-in aspects that provide value to multidisciplinary technical solutions. The hashed addresses of the wallet are utilised as identities in this blockchain-based identity management system. Within the same system, we generated numerous identities and added them as peers to a miner. Only peer-to-peer transactions are permitted following adequate identification verification. The suggested approach is compatible with the Ethereum primary chain because it is based on open-source libraries. With minor adjustments, the intended system can be put into any real-time application.

References

[1] Zhang, Y.; Kasahara, S.; Shen, Y.; Jiang, X.; Wan, J. Smart Contract-Based Access Control for the Internet of Things. IEEE Internet Things J. 2019, 6, 1594– 1605.

[2] Ismail, L.; Materwala, H.; Zeadally, S. Lightweight Blockchain for Healthcare. IEEE Access 2019, 7, 149935–14995

[3] https://www.ibm.com/in-en/blockchain.

[4] Al-Fuqaha, A.; Guizani, M.; Mohammadi, M.; Aledhari, M.; Ayyash, M. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. IEEE Comm. Surveys &amp; Tutorials 2015, 17, 2347–2376.Chengwei Liu, Yixiang Chan, Syed Hasnain Alam Kazmi, Hao Fu, "Financial Fraud Detection Model: Based on Random Forest," International Journal of Economics and Finance, Vol. 7, Issue. 7, pp. 178-188, 2015.

[5] V. Buterin, ―On public and private blockchains,‖ Ethereum blog, vol. 7, 2015.

[6] K. Cameron, ―The laws of identity,‖ Microsoft Corp, 2005.

[7] https://www.rs-online.com/designspark/exploring-ethereum-with-raspberry-pi-part-2-creating-a-private-blockchain

[8] Received November 12, 2020, accepted November 25, 2020, date of publication November 30, 2020, date of current version December 14, 2020. Digital Object Identifier 10.1109/ACCESS.2020.3041317. A Block chain Ethereum Technology-Enabled Digital Content: Development of Trading and Sharing Economy Data UMAIR KHAN 1 , ZHANG YONG AN1 , AND AZHAR IMRAN 2 , (Graduate Student Member, IEEE)

[9] Oliva, G.A., Hassan, A.E. & Jiang, Z.M.(. An exploratory study of smart contracts in the Ethereum blockchain platform. *Empir Software Eng* **25,** 1864–1904 (2020). https://doi.org/10.1007/s10664-019-09796-5.

[10] Ethereum Transactions and Smart Contracts among Secure Identities Francesco Buccafurri, Gianluca Lax, Lorenzo Musarella, and Antonia Russo University Mediterranea of Reggio Calabria, Italy {bucca,lax,lorenzo.musarella,antonia.russo}@unirc.it

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.542 |

|| Volume 9, Issue 6, June 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0906273 |

[11] https://selfkey.org/introduction-to-blockchain-identity-management/

[12] Juliana Zanelatto Gavião Mascarenhas, Artur Ziviani, Klaus Wehmuth, Alex Borges Vieira, On the transaction dynamics of the Ethereum-based cryptocurrency, *Journal of Complex Networks*, Volume 8, Issue 4, August 2020, cnaa042, https://doi.org/10.1093/comnet/cnaa042

[13] https://ethereum.github.io/yellowpaper/paper.pdf

[14] S. Wang, R. Pei and Y. Zhang, "EIDM: A Ethereum-Based Cloud User Identity Management Protocol," in *IEEE Access*, vol. 7, pp. 115281-115291, 2019, doi: 10.1109/ACCESS.2019.2933989.

[15] https://tykn.tech/identity-management-blockchain/#Models_of_Digital_Identity_Management.

[16] The Block chain Technology for Secure and Smart Applications across Industry Verticals, Volume 121, **1st Edition. Serial Volume Editors:** Neeraj Kumar Shubhani Aggarwal Pethuru Raj. **eBook ISBN:** 9780128219928. **Hardcover ISBN:** 9780128219911. **Imprint:** Academic Press. **Published Date:** 23rd January 2021. **Page Count:** 516

[17] Metcalfe W. (2020) Ethereum, Smart Contracts, DApps. In: Yano M., Dai C., Masuda K., Kishimoto Y. (eds) Blockchain and Crypto Currency. Economics, Law, and Institutions in Asia Pacific. Springer, Singapore. https://doi.org/10.1007/978-981-15-3376-1_5

[18] RUI ZHANG and RUI XUE, State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, China and School of Cyber Security, University of Chinese Academy of Sciences, China LING LIU, School of Computer Science, Georgia Institute of Technology, USA.