INFSCI 2750 Mini Project 3

Blockchain-assisted Verifiable Cassandra

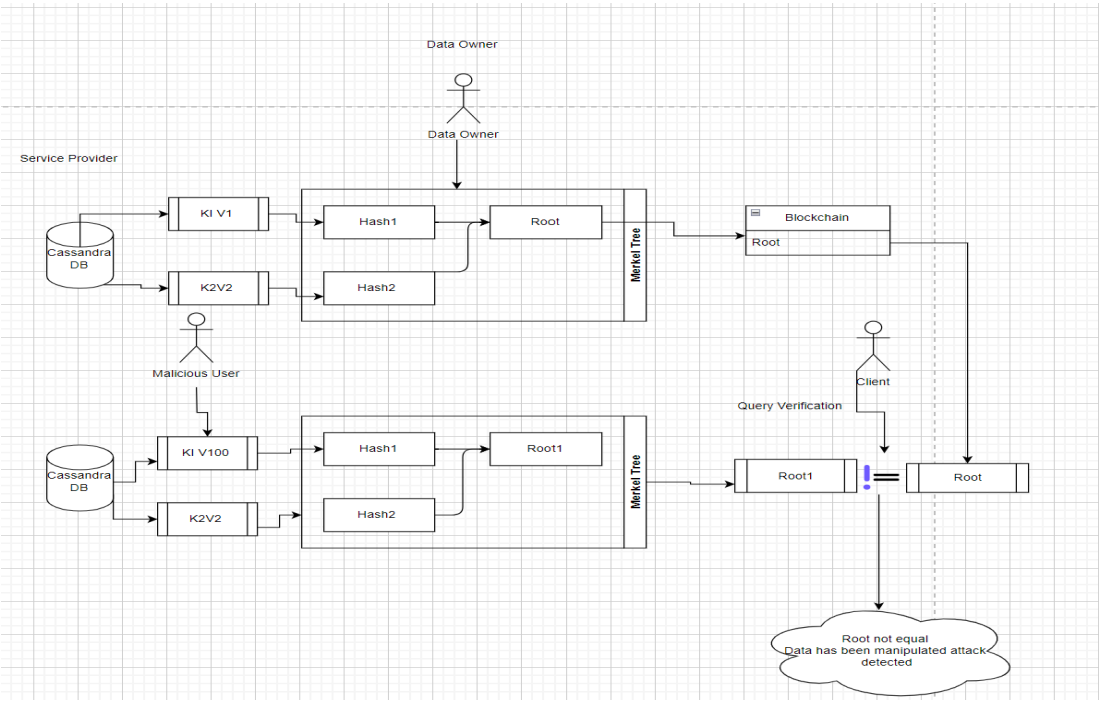
GROUP 7

GAV39 Ganesh Venkattaraman, KIS104 Kiran Shridhar Alva, JAP508 Pratibha Jallu

Description:

Blockchain-assisted Verifiable Cassandra merges the strengths of blockchain technology with the data storage capabilities of Apache Cassandra to ensure data integrity and auditability. By periodically calculating and storing the Merkle root hash of data stored in Cassandra on a blockchain, this approach offers a verifiable and tamper-resistant mechanism for ensuring the authenticity of stored data. This enables anyone with access to the blockchain to independently verify the integrity of the data by comparing the calculated Merkle root hash with the hash stored on the blockchain, providing cryptographic proof of data integrity without relying on a central authority.

Approach:



- Store data in Cassandra DB.
- Generate a mekle tree and store its root in the blockchain.
- Use a malicious user to edit the data in cassandra DB.
- Generate a mekle tree and obtain the new merkle tree's root.
- Compare that with the root stored in the blockchain and validate the attack.

The Set Up:

- ❖ Installing all the prerequisites : Java and Python
- ❖ Installing Cassandra
- ❖ Setting up nvm and Installing Ganache Ethereum and Solc
- ❖ Loading all the python files onto the VM
- ❖ Storing data in Cassandra Database

Get all the prerequisites Java and Python, Update the packages

```
ubuntu@group7:~$ sudo apt-get update
Hit:1 http://security.ubuntu.com/ubuntu bionic-security InRelease
Hit:2 http://nova.clouds.archive.ubuntu.com/ubuntu bionic InRelease
Get:3 http://nova.clouds.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Hit:4 http://nova.clouds.archive.ubuntu.com/ubuntu bionic-backports InRelease
Get:5 http://nova.clouds.archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [3045 kB]
Fetched 3134 kB in 2s (1304 kB/s)
Reading package lists... Done
ubuntu@group7:~$ sudo apt-get install python3-pip
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  grub-pc-bin
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu build-essential cpp cpp-7 dh-python dpkg-dev fakeroot g++ g++-7
  gcc gcc-7 gcc-7-base libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan4 libatomic1
  libbinutils libc-dev-bin libc6 libc6-dev libcc1-0 libcilkrts5 libdpkg-perl libexpat1-dev libfakeroot
  libfile-fcntllock-perl libgcc-7-dev libgomp1 libisl19 libitm1 liblsan0 libmpc3 libmpx2 libpython3-dev
  libpython3.6-dev libquadmath0 libstdc++-7-dev libtsan0 libubsan0 linux-libc-dev make manpages-dev python-pip-whl
  python3-crypto python3-dev python3-distutils python3-keyring python3-keyrings.alt python3-lib2to3
  python3-secretstorage python3-setuptools python3-wheel python3-xdg python3.6-dev
Suggested packages:
  binutils-doc cpp-doc gcc-7-locales debian-keyring g++-multilib g++-7-multilib gcc-7-doc libstdc++6-7-dbg
  gcc-multilib autoconf automake libtool flex bison gdb gcc-doc gcc-7-multilib libgcc1-dbg libgomp1-dbg libitm1-dbg
  libatomic1-dbg libasan4-dbg liblsan0-dbg libtsan0-dbg libubsan0-dbg libcilkrts5-dbg libmpx2-dbg libquadmath0-dbg
```

```
Processing triggers for libc-bin (2.27-3ubuntu1.3) ...
ubuntu@group7:~$ python3 -m pip install -U pip
Collecting pip
  Downloading https://files.pythonhosted.org/packages/a4/6d/6463d49a933f547439d6b5b98b46af8742cc03ae83543e4d7688c2420f8b/pip-21.3.1
-py3-none-any.whl (1.7MB)
    100% |████████████████████████████████| 1.7MB 586kB/s
Installing collected packages: pip
Successfully installed pip-21.3.1
ubuntu@group7:~$ pip3 install cassandra-driver merkletools py-solc-x
WARNING: pip is being invoked by an old script wrapper. This will fail in a future version of pip.
Please see https://github.com/pypa/pip/issues/5599 for advice on fixing the underlying issue.
To avoid this problem you can invoke Python with '-m pip' instead of running pip directly.
Defaulting to user installation because normal site-packages is not writeable
Collecting cassandra-driver
  Downloading cassandra-driver-3.29.1.tar.gz (292 kB)
     |████████████████████████████████| 292 kB 2.3 MB/s
  Preparing metadata (setup.py) ... done
Collecting merkletools
  Downloading merkletools-1.0.3.tar.gz (8.3 kB)
  Preparing metadata (setup.py) ... done
Collecting py-solc-x
```

Cassandra Installation:

```
Collecting requests<3,>=2.19.0
  Downloading requests-2.27.1-py2.py3-none-any.whl (63 kB)
     |████████████████████████████████| 63 kB 239 kB/s
Requirement already satisfied: click in /usr/lib/python3/dist-packages (from geomet<0.3,>=0.1->cassandra-driver) (6.7)
Requirement already satisfied: six in /usr/lib/python3/dist-packages (from geomet<0.3,>=0.1->cassandra-driver) (1.11.0)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/lib/python3/dist-packages (from requests<3,>=2.19.0->py-solc-x) (1.22)
Requirement already satisfied: certifi>=2017.4.17 in /usr/lib/python3/dist-packages (from requests<3,>=2.19.0->py-solc-x) (2018.1.1
8)
Requirement already satisfied: idna<4,>=2.5 in /usr/lib/python3/dist-packages (from requests<3,>=2.19.0->py-solc-x) (2.6)
Collecting charset-normalizer~=2.0.0
  Downloading charset_normalizer-2.0.12-py3-none-any.whl (39 kB)
Building wheels for collected packages: cassandra-driver, merkletools
  Building wheel for cassandra-driver (setup.py) ... done
  Created wheel for cassandra-driver: filename=cassandra_driver-3.29.1-cp36-cp36m-linux_x86_64.whl size=12414039 sha256=fab079bada0
26c4f35851424cabc311b8d7a8aae858f2ff3be0ba038ef6ed6db
  Stored in directory: /home/ubuntu/.cache/pip/wheels/08/f9/c3/be985d2759b3f38eee5d0ff6b5cde33b6f478281e42a410278
  Building wheel for merkletools (setup.py) ... done
  Created wheel for merkletools: filename=merkletools-1.0.3-py3-none-any.whl size=6799 sha256=b1de49b71271ae82d22fe61906be8b3063fbe
05786e60b9cd32f9e1b51b0516e
  Stored in directory: /home/ubuntu/.cache/pip/wheels/03/48/d4/8226cc53d844550d4ed782b6cce82fe7542135174fbc5f7801
Successfully built cassandra-driver merkletools
Installing collected packages: charset-normalizer, semantic-version, requests, pysha3, geomet, py-solc-x, merkletools, cassandra-dr
iver
  WARNING: The script normalizer is installed in '/home/ubuntu/.local/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
  WARNING: The script geomet is installed in '/home/ubuntu/.local/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed cassandra-driver-3.29.1 charset-normalizer-2.0.12 geomet-0.2.1.post1 merkletools-1.0.3 py-solc-x-1.1.1 pysha
```

```
 # Addresses of hosts that are deemed contact points.
 # Cassandra nodes use this list of hosts to find each other and learn
 # the topology of the ring.  You must change this if you are running
 # multiple nodes!
 - class_name: org.apache.cassandra.locator.SimpleSeedProvider
   parameters:
     # seeds is actually a comma-delimited list of addresses.
     # Ex: "<ip1>,<ip2>,<ip3>"
     - seeds: "10.254.2.54,127.0.0.1"

# For workloads with more data than can fit in memory, Cassandra's
# bottleneck will be reads that need to fetch data from
# disk. "concurrent_reads" should be set to (16 * number_of_drives) in
# order to allow the operations to enqueue low enough in the stack
# that the OS and drives can reorder them. Same applies to
# "concurrent_counter_writes", since counter writes read the current
# values before incrementing and writing them back.
#
# On the other hand, since writes are almost never IO bound, the ideal
# number of "concurrent_writes" is dependent on the number of cores in
```

```
ubuntu@group7:~$ nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address    Load        Tokens  Owns (effective)  Host ID                               Rack
UN  127.0.0.1  235.36 KiB  16      100.0%            77219bea-9c9b-4349-8844-2f1cec4430d2  rack1
```

Create a project 3 directory and store the python codes:

```
Successfully installed cassandra-driver-3.29.1 charset-normalizer-2.0.12 geomet-0.2.1.post1 merkletools-1.0.3 py-solc-x-1.1.1 pysha
3-1.0.2 requests-2.27.1 semantic-version-2.10.0
ubuntu@group7:~$ mkdir project3
ubuntu@group7:~$ ls
project3
```

```
ubuntu@group7:~/project3$ ls
__pycache__     data_owner.py   hs_err_pid10154.log  mer2.py       p.py               proj.py          storedata.py
adv_client.py   db_provider.py  malc.py              merkel.py     package-lock.json  query_client.py  storeleaf.py
blockchain.py   driver.py       mer.py               node_modules  package.json       retroot.py
ubuntu@group7:~/project3$
```

Ganache Installation:

```
ubuntu@group7:~$ touch ~/.bashrc
ubuntu@group7:~$ curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.37.2/install.sh | bash
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 13527  100 13527    0     0  45545      0 --:--:-- --:--:-- --:--:-- 45545
=> Downloading nvm from git to '/home/ubuntu/.nvm'
=> Cloning into '/home/ubuntu/.nvm'...
remote: Enumerating objects: 333, done.
remote: Counting objects: 100% (333/333), done.
remote: Compressing objects: 100% (283/283), done.
remote: Total 333 (delta 39), reused 143 (delta 25), pack-reused 0
Receiving objects: 100% (333/333), 177.04 KiB | 2.81 MiB/s, done.
Resolving deltas: 100% (39/39), done.
=> Compressing and cleaning up git repository

=> Appending nvm source string to /home/ubuntu/.bashrc
=> Appending bash_completion source string to /home/ubuntu/.bashrc
=> Close and reopen your terminal to start using nvm or run the following to use it now:

export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"  # This loads nvm
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion"  # This loads nvm bash_completion
ubuntu@group7:~$ source ~/.bashrc
ubuntu@group7:~$ nvm install 16.14.2
Downloading and installing node v16.14.2...
Downloading https://nodejs.org/dist/v16.14.2/node-v16.14.2-linux-x64.tar.xz...
######################################################################################################################################################## 100.0%
```

```
######################################################################################################################################################## 100.0%
Computing checksum with sha256sum
Checksums matched!
Now using node v16.14.2 (npm v8.5.0)
Creating default alias: default -> 16.14.2 (-> v16.14.2)
ubuntu@group7:~$ node -v
v16.14.2
ubuntu@group7:~$ sudo add-apt-repository ppa:ethereum/ethereum

 More info: https://launchpad.net/~ethereum/+archive/ubuntu/ethereum
Press [ENTER] to continue or Ctrl-c to cancel adding it.

Hit:1 http://nova.clouds.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://security.ubuntu.com/ubuntu bionic-security InRelease
Hit:3 http://nova.clouds.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:4 http://nova.clouds.archive.ubuntu.com/ubuntu bionic-backports InRelease
Err:5 http://ppa.launchpad.net/ethereum/ethereum/ubuntu bionic InRelease
  Cannot initiate the connection to ppa.launchpad.net:80 (2620:2d:4000:1::81). - connect (101: Network is unreachable) Could not co
nnect to ppa.launchpad.net:80 (185.125.190.80), connection timed out
Reading package lists... Done
W: Failed to fetch http://ppa.launchpad.net/ethereum/ethereum/ubuntu/dists/bionic/InRelease  Cannot initiate the connection to ppa.
launchpad.net:80 (2620:2d:4000:1::81). - connect (101: Network is unreachable) Could not connect to ppa.launchpad.net:80 (185.125.1
90.80), connection timed out
W: Some index files failed to download. They have been ignored, or old ones used instead.
ubuntu@group7:~$
```

```
ubuntu@group7:~$ touch ~/.bashrc
ubuntu@group7:~$ curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.37.2/install.sh | bash
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 13527  100 13527    0     0  53678      0 --:--:-- --:--:-- --:--:-- 53466
=> nvm is already installed in /home/ubuntu/.nvm, trying to update using git
=> => Compressing and cleaning up git repository

=> nvm source string already in /home/ubuntu/.bashrc
=> bash_completion source string already in /home/ubuntu/.bashrc
=> You currently have modules installed globally with `npm`. These will no
=> longer be linked to the active version of Node when you install a new node
=> with `nvm`; and they may (depending on how you construct your `$PATH`)
=> override the binaries of modules installed with `nvm`:

/home/ubuntu/.nvm/versions/node/v16.14.2/lib
├── corepack@0.10.0
├── ganache@7.9.2
=> If you wish to uninstall them at a later point (or re-install them under your
=> `nvm` Nodes), you can remove them from the system Node as follows:

     $ nvm use system
     $ npm uninstall -g a_module

=> Close and reopen your terminal to start using nvm or run the following to use it now:

export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"  # This loads nvm
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion"  # This loads nvm bash_completion
```

```
ubuntu@group7:~$ source ~/.bashrc
ubuntu@group7:~$ nvm install 16.14.2
v16.14.2 is already installed.
Now using node v16.14.2 (npm v8.5.0)
ubuntu@group7:~$
```

Ethereum and Solc installation:

```
ubuntu@group7:~$ node -v
v16.14.2
ubuntu@group7:~$ sudo add-apt-repository ppa:ethereum/ethereum

 More info: https://launchpad.net/~ethereum/+archive/ubuntu/ethereum
Press [ENTER] to continue or Ctrl-c to cancel adding it.

Hit:1 http://nova.clouds.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://security.ubuntu.com/ubuntu bionic-security InRelease
Hit:3 http://nova.clouds.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:4 http://nova.clouds.archive.ubuntu.com/ubuntu bionic-backports InRelease
Err:5 http://ppa.launchpad.net/ethereum/ethereum/ubuntu bionic InRelease
  Cannot initiate the connection to ppa.launchpad.net:80 (2620:2d:4000:1::81). - connect (101: Network is unreachable) Could not co
nnect to ppa.launchpad.net:80 (185.125.190.80), connection timed out
Reading package lists... Done
W: Failed to fetch http://ppa.launchpad.net/ethereum/ethereum/ubuntu/dists/bionic/InRelease  Cannot initiate the connection to ppa.
launchpad.net:80 (2620:2d:4000:1::81). - connect (101: Network is unreachable) Could not connect to ppa.launchpad.net:80 (185.125.1
90.80), connection timed out
W: Some index files failed to download. They have been ignored, or old ones used instead.
ubuntu@group7:~$ sudo apt-get update
Hit:1 http://security.ubuntu.com/ubuntu bionic-security InRelease
Get:2 http://ppa.launchpad.net/ethereum/ethereum/ubuntu bionic InRelease [15.4 kB]
Hit:3 http://nova.clouds.archive.ubuntu.com/ubuntu bionic InRelease
Hit:4 http://nova.clouds.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:5 http://nova.clouds.archive.ubuntu.com/ubuntu bionic-backports InRelease
Get:6 http://ppa.launchpad.net/ethereum/ethereum/ubuntu bionic/main amd64 Packages [2544 B]
Get:7 http://ppa.launchpad.net/ethereum/ethereum/ubuntu bionic/main Translation-en [772 B]
Fetched 18.7 kB in 1s (22.6 kB/s)
Reading package lists... Done
ubuntu@group7:~$ sudo apt-get install solc
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  grub-pc-bin
Use 'sudo apt autoremove' to remove it.
The following NEW packages will be installed:
  solc
```

```
ubuntu@group7:~$ sudo apt-get install solc
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  grub-pc-bin
Use 'sudo apt autoremove' to remove it.
The following NEW packages will be installed:
  solc
0 upgraded, 1 newly installed, 0 to remove and 75 not upgraded.
Need to get 5950 kB of archives.
After this operation, 22.6 MB of additional disk space will be used.
Get:1 http://ppa.launchpad.net/ethereum/ethereum/ubuntu bionic/main amd64 solc amd64 1:0.8.25-0ubuntu1~STATIC [5950 kB]
Fetched 5950 kB in 2s (2981 kB/s)
Selecting previously unselected package solc:amd64.
(Reading database ... 97575 files and directories currently installed.)
Preparing to unpack .../solc_1%3a0.8.25-0ubuntu1~STATIC_amd64.deb ...
Unpacking solc:amd64 (1:0.8.25-0ubuntu1~STATIC) ...
Setting up solc:amd64 (1:0.8.25-0ubuntu1~STATIC) ...
ubuntu@group7:~$ npm install ganache --global

added 336 packages, and audited 366 packages in 34s

6 packages are looking for funding
  run `npm fund` for details

10 vulnerabilities (5 moderate, 4 high, 1 critical)

To address all issues, run:
  npm audit fix

Run `npm audit` for details.
npm notice
npm notice New major version of npm available! 8.5.0 -> 10.5.2
npm notice Changelog: https://github.com/npm/cli/releases/tag/v10.5.2
npm notice Run npm install -g npm@10.5.2 to update!
npm notice
ubuntu@group7:~$
```

```
ubuntu@group7:~/project3$ apt list solc
Listing... Done
solc/bionic,now 1:0.8.25-0ubuntu1~STATIC amd64 [installed]
N: There is 1 additional version. Please use the '-a' switch to see it
ubuntu@group7:~/project3$ apt list -a solc
Listing... Done
solc/bionic,now 1:0.8.25-0ubuntu1~STATIC amd64 [installed]
solc/bionic 1:0.7.5~develop-2020-11-18-1e08e4e0-0ubuntu1~bionic amd64

ubuntu@group7:~/project3$ sudo apt-get install solc=1:0.7.5~develop-2020-11-18-1e08e4e0-0ubuntu1~bionic
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  grub-pc-bin
```

Blockchain Snippet:

```
ubuntu@group7:~$ ganache
ganache v7.9.2 (@ganache/cli: 0.10.2, @ganache/core: 0.10.2)
Starting RPC server

Available Accounts
==================
(0) 0xC9d0fA289c51C34142Ed3bF51Bcc5155Cc099B3E (1000 ETH)
(1) 0x831Bbc1898326d88Ee3fAA6f528B3155a38a3711 (1000 ETH)
(2) 0x7e9731efC8F16F58033e94Ab154B44c3e995F1B8 (1000 ETH)
(3) 0x1Ed92FFB6C1C278C2C6C2b8cbaecaB8A81f2f157 (1000 ETH)
(4) 0x74113B9c3Af5C05A603b4fA900Fd039e14cCF692 (1000 ETH)
(5) 0x94a99b1E50c555c59C952b7028D4e10faD090f58 (1000 ETH)
(6) 0x61dDA683e6945810bC8a5108c4c8B2fc20982eA8 (1000 ETH)
(7) 0xd8Cc3Eb9F3547728161989448Cb7b907480c5470 (1000 ETH)
(8) 0x8607a7414fD88FdE5c6E6843De1C5E46a48a5323 (1000 ETH)
(9) 0xDeB97667B77b2F129Ceee886d20fF84428c1E2c4 (1000 ETH)

Private Keys
==================
(0) 0x7bd0cce428096878ff1d50b2ac808cffa836c630530e4c2a1497ac5859e0503e
(1) 0xeb245dada6670132e245a925f9a1a96f70c8754cf74cf9e12af03c92ea519949
(2) 0x5b95ee7aae4b77437c61bc56a0e5a961e2bc9b0a74212bc0f1b13db2e3b4140f
(3) 0xfc32057c36fd545eaa275003a444379a922a909e7d356c95684dfe71d2a7a162
```

```
Private Keys
==================
(0) 0x7bd0cce428096878ff1d50b2ac808cffa836c630530e4c2a1497ac5859e0503e
(1) 0xeb245dada6670132e245a925f9a1a96f70c8754cf74cf9e12af03c92ea519949
(2) 0x5b95ee7aae4b77437c61bc56a0e5a961e2bc9b0a74212bc0f1b13db2e3b4140f
(3) 0xfc32057c36fd545eaa275003a444379a922a909e7d356c95684dfe71d2a7a162
(4) 0x6c8b2eed27fc34a6d91001f76b4b448338704b2fda072751fa1a65148afd981
(5) 0xb617b97e4e03881e35449b70b1f82487cff8d766eb8beefe5eb4bbbf84dae475
(6) 0x3141482ed0bf84398f2aeafc1a6d28b8dfc67e9b3d54d3c615cfb52d80e4dcc6
(7) 0x5d0ddb2048dc8fe7bad9d3acdff87adf0662681197939719 27b15df184b1ca0f
(8) 0xd3a1198c870b87cc8987d53e8c196c0ee72217943708a2b4ee948076ceaf94ac
(9) 0xbd4c1e42f7207669bd28b96607fbd805747ac10464d33df3b4793b6b6937cd91

HD Wallet
==================
Mnemonic:      pave snap voyage winner catch file flock bone appear doll welcome mammal
Base HD Path:  m/44'/60'/0'/0/{account_index}

Default Gas Price
==================
2000000000

BlockGas Limit
==================
30000000

Call Gas Limit
==================
50000000

Chain
==================
Hardfork: shanghai
Id:       1337

RPC Listening on 127.0.0.1:8545
```

```
Hardfork: shanghai
Id:       1337

RPC Listening on 127.0.0.1:8545
eth_accounts
eth_getBlockByNumber
eth_chainId
eth_chainId
eth_estimateGas
eth_blockNumber
eth_getBlockByNumber
eth_sendTransaction

  Transaction: 0x04371a1fc9c0a41d9aa043097c31ca6a1ddc89758003aa139ff03ce5d28a6b38
  Contract created: 0xf76646565e5966176188b6041e43878e01c38429
  Gas usage: 394919
  Block number: 1
  Block time: Sat Apr 20 2024 02:32:31 GMT+0000 (Coordinated Universal Time)

eth_getTransactionReceipt
eth_chainId
eth_call
eth_accounts
eth_getBlockByNumber
eth_chainId
eth_chainId
eth_estimateGas
eth_blockNumber
eth_getBlockByNumber
eth_sendTransaction

  Transaction: 0x3b7ae61422695c0c3e4b0a7ae523e6366a89a51e745454df718553f00baed28b
  Contract created: 0x2f319bdab6544f16e9183b7d98ed3fc2b9135b1b
  Gas usage: 394919
  Block number: 2
  Block time: Sat Apr 20 2024 02:41:21 GMT+0000 (Coordinated Universal Time)
```

```
eth_call
eth_accounts
eth_getBlockByNumber
eth_chainId
eth_chainId
eth_estimateGas
eth_blockNumber
eth_getBlockByNumber
eth_sendTransaction

  Transaction: 0x3b7ae61422695c0c3e4b0a7ae523e6366a89a51e745454df718553f00baed28b
  Contract created: 0x2f319bdab6544f16e9183b7d98ed3fc2b9135b1b
  Gas usage: 394919
  Block number: 2
  Block time: Sat Apr 20 2024 02:41:21 GMT+0000 (Coordinated Universal Time)

eth_getTransactionReceipt
eth_chainId
eth_call
^[[A^[[A^[[A^[[A^[[A^[[A^[[A^[[A^[[Aeth_accounts
eth_getBlockByNumber
eth_chainId
eth_chainId
eth_estimateGas
eth_blockNumber
eth_getBlockByNumber
eth_sendTransaction

  Transaction: 0xc1e578eb3164d70984583c51395ab5c983737c270305a8a7dc09de2fc7ed7920
  Contract created: 0xbd7e7cd0330451d559b020f6c45877980b18ab7d
  Gas usage: 394919
  Block number: 3
  Block time: Sat Apr 20 2024 02:56:12 GMT+0000 (Coordinated Universal Time)

eth_getTransactionReceipt
eth_chainId
eth_call
```

Data Stored in Cassandra DB:

Keyspace: project3
Table: data

```
cqlsh:project3> select * from data;

 key | value
-----+-------
   C |    30
   B |    20
   A |   100
   D |    40

(4 rows)
cqlsh:project3>
```

Python code and the list of functions used

```
from web3 import Web3
from solcx import compile_source
import merkletools
import hashlib
from cassandra.cluster import Cluster
```

1. build_merkle_tree(data):
    - This function builds a Merkle tree over the provided data.
    - It iterates over the key-value pairs in the data dictionary and adds each value as a leaf node to the Merkle tree using the MerkleTools library.
    - After adding all the leaf nodes, it generates the Merkle root hash of the tree.

```
def build_merkle_tree(data):

    mt = merkletools.MerkleTools()
    for k, v in data.items():
        mt.add_leaf(v, True)
```

```
    mt.make_tree()
    return mt
```

2. get_merkle_index_by_key(key, key_index):
    - This function retrieves the Merkle tree leaf index associated with a given key.
    - It takes the key and a dictionary (key_index) that maps keys to their corresponding Merkle tree leaf indices.
    - It returns the leaf index associated with the given key.

```
def get_merkle_index_by_key(key, key_index):
    index = key_index.get(key)
    return index
```

3. get_merkle_proof_by_index(merkle_tree, index):
    - This function retrieves the Merkle proof associated with a given leaf index in the Merkle tree.
    - It takes the Merkle tree (merkle_tree) and the leaf index.
    - It returns the Merkle proof for the leaf node at the specified index.

```
def get_merkle_proof_by_index(merkle_tree, index):

    merkle_proof = merkle_tree.get_proof(index)
    return merkle_proof
```

4. query_data_by_key(key, data):
    - This function queries the value associated with a given key from the provided data dictionary.
    - It takes the key and the data dictionary.
    - It returns the value associated with the given key, or None if the key is not found in the dictionary.

```
def query_data_by_key(key, data):

    value = data.get(key)
    return value
```

5. store_data_in_cassandra(data):
    - This function stores key-value pairs in a Cassandra database.
    - It connects to the Cassandra cluster, creates a keyspace and table if they don't exist, and inserts the data into the table.

- It uses the Cassandra Python driver to execute CQL queries for interacting with Cassandra.

```
def store_data_in_cassandra(data):

    cluster = Cluster(['127.0.0.1'])  # Replace with your Cassandra
cluster IP address
    keyspace = "project3"
    table = "data"
    session = cluster.connect()

    # Create keyspace if it doesn't exist
    session.execute("CREATE KEYSPACE IF NOT EXISTS " + keyspace + " WITH
REPLICATION = {'class' : 'SimpleStrategy', 'replication_factor' : 1};")

    # Use the keyspace
    session.set_keyspace(keyspace)

    # Create table if it doesn't exist
    session.execute(f"CREATE TABLE IF NOT EXISTS {table} (key text PRIMARY
KEY, value text);")

    # Insert data into the table
    for key, value in data.items():
        session.execute(f"INSERT INTO {table} (key, value) VALUES
('{key}', '{value}');")

    # Close the session and cluster
    session.shutdown()
    cluster.shutdown()
```

6. query_value_by_key(key):

- This function queries the value associated with a given key from the Cassandra table.
- It connects to the Cassandra cluster, retrieves the value from the table using a CQL SELECT query, and returns it.
- If the key is not found in the table, it returns None.

```
def query_value_by_key(key):
```

```
    cluster = Cluster(['127.0.0.1'])  # Replace with your Cassandra
cluster IP address
    keyspace = "project3"
    table = "data"
    session = cluster.connect()

    # Use the keyspace
    session.set_keyspace(keyspace)

    # Retrieve the value from the table
    row = session.execute(f"SELECT value FROM {table} WHERE key =
'{key}';").one()

    # Close the session and cluster
    session.shutdown()
    cluster.shutdown()

    if row:
        return row.value
    else:
        return None
```

7. get_merkle_tree_from_cassandra(key_index):
    - This function retrieves all data from the Cassandra table and constructs a Merkle tree over its values.
    - It connects to the Cassandra cluster, retrieves the data from the table, builds the Merkle tree, and returns it.
    - It also reorders the data dictionary based on the provided key_index before building the Merkle tree.

```
def get_merkle_tree_from_cassandra(key_index):

    cluster = Cluster(['127.0.0.1'])  # Replace with your Cassandra
cluster IP address
    keyspace = "project3"
    table = "data"
    session = cluster.connect()

    # Use the keyspace
    session.set_keyspace(keyspace)
```

```
    # Retrieve all data from the table
    rows = session.execute(f"SELECT * FROM {table};")

    # Create a dictionary to store the retrieved data
    data = {}
    for row in rows:
        key = row.key
        value = row.value
        data[key] = value

    # Close the session and cluster
    session.shutdown()
    cluster.shutdown()
    data = {k: v for k, v in sorted(data.items(), key=lambda item:
key_index[item[0]])}
    # Build the Merkle Tree over the data
    merkle_tree = build_merkle_tree(data)

    return merkle_tree
```

8. get_merkle_proof_by_key(key, key_index, merkle_tree):

- This function retrieves the Merkle proof associated with a given key from the Merkle tree.
- It first gets the Merkle leaf index corresponding to the key using get_merkle_index_by_key function.
- Then, it retrieves the Merkle proof for that leaf index using get_merkle_proof_by_index function.

```
def get_merkle_proof_by_key(key, key_index, merkle_tree):

    index = get_merkle_index_by_key(key, key_index)
    merkle_proof = get_merkle_proof_by_index(merkle_tree, index)
    return merkle_proof
```

9.malicious_attempt(table, key, new_value):

- This function simulates a malicious attempt to modify data in the Cassandra database.
- It connects to the Cassandra cluster, updates the value associated with the specified key in the table to a new value, and then disconnects.

```
def malicious_attempt(table, key, new_value):

    cluster = Cluster(['127.0.0.1'])
    keyspace = "project3"
    session = cluster.connect()
    session.set_keyspace(keyspace)
    session.execute(f"UPDATE {table} SET value = %s WHERE key = %s",
(new_value, key))
```

10.validate_merkle_proof(value, merkle_proof, merkle_root):

- This function validates a value using its Merkle proof and Merkle root hash.
- It creates a MerkleTools object, sets the Merkle root hash, and then validates the Merkle proof against the value.

```
def validate_merkle_proof(value, merkle_proof, merkle_root):

    mt = merkletools.MerkleTools()
    mt.merkle_root = merkle_root
    return mt.validate_proof(merkle_proof, value)
```

```
    compiled_sol = compile_source(
        '''
        pragma solidity >0.5.0;
        contract Verify{
            string merkleRoot;

            function setMerkleRoot(string memory _merkleRoot) public {
                merkleRoot=_merkleRoot;
            }

            function getMerkleRoot()view public returns (string memory){
                return merkleRoot;
            }
        }
        ''',
        output_values=['abi', 'bin']
    )

    contract_id, contract_interface = compiled_sol.popitem()
    bytecode = contract_interface['bin']
    abi = contract_interface['abi']

    w3 = Web3(Web3.HTTPProvider('http://127.0.0.1:8545'))
    w3.eth.default_account = w3.eth.accounts[0]

    Verify = w3.eth.contract(abi=abi, bytecode=bytecode)

    deploy_tx_hash = Verify.constructor().transact()
    deploy_tx_receipt = w3.eth.wait_for_transaction_receipt(deploy_tx_hash)

    verify = w3.eth.contract(
        address=deploy_tx_receipt.contractAddress,
        abi=abi
```

The code demonstrates querying and setting the Merkle root hash in the blockchain smart contract. It then verifies the integrity of the data before and after the malicious attempt, comparing the Merkle root hash obtained from the Cassandra database with the one stored in the smart contract.

Main Program:

```
if __name__ == '__main__':
    # Original data
    ori_data = {
        'A': '10',
        'B': '20',
        'C': '30',
        'D': '40'
    }

    # Store data in Cassandra
    store_data_in_cassandra(ori_data)

    # Key-index mapping
    key_index = {
        'A': 0,
        'B': 1,
        'C': 2,
        'D': 3
    }

    # Build Merkle Tree
    merkle_tree = build_merkle_tree(ori_data)

    # Get Merkle root
    merkle_root = merkle_tree.get_merkle_root()

    # Compile Solidity contract
    compiled_sol = compile_source(
        '''
        pragma solidity >0.5.0;
        contract Verify{
            string merkleRoot;

            function setMerkleRoot(string memory _merkleRoot) public {
```

```
                merkleRoot=_merkleRoot;
            }

            function getMerkleRoot() view public returns (string memory){
                return merkleRoot;
            }
        }
        ''',
        output_values=['abi', 'bin']
    )

    contract_id, contract_interface = compiled_sol.popitem()
    bytecode = contract_interface['bin']
    abi = contract_interface['abi']

    # Connect to Ethereum node
    w3 = Web3(Web3.HTTPProvider('http://127.0.0.1:8545'))
    w3.eth.default_account = w3.eth.accounts[0]

    # Deploy contract
    Verify = w3.eth.contract(abi=abi, bytecode=bytecode)
    deploy_tx_hash = Verify.constructor().transact()
    deploy_tx_receipt =
w3.eth.wait_for_transaction_receipt(deploy_tx_hash)
    verify = w3.eth.contract(
        address=deploy_tx_receipt.contractAddress,
        abi=abi
    )

    # Set Merkle root in contract
    set_tx_hash = verify.functions.setMerkleRoot(merkle_root).transact()
    set_tx_recipient = w3.eth.wait_for_transaction_receipt(set_tx_hash)

    # Query data by key
    query_result = query_data_by_key('A', ori_data)
    print("Original Query Result:", query_result)

    # Get Merkle index by key
    merkle_index = get_merkle_index_by_key('A', key_index)
    print("Merkle Index:", merkle_index)
```

```
    # Get Merkle proof by index
    merkle_proof_index = get_merkle_proof_by_index(merkle_tree, 0)
    print("Original Merkle Proof Index:", merkle_proof_index)

    # Get Merkle root from contract
    merkle_root_contract = verify.functions.getMerkleRoot().call()
    print("Merkle Root from Contract:", merkle_root_contract)

    print()

    print("Before Malicious Attempt:")
    print("Value of A:", query_value_by_key('A'))

    # Get Merkle Tree from Cassandra
    mr = get_merkle_tree_from_cassandra(key_index)
    root = mr.get_merkle_root()
    print("Merkle Root from Cassandra:", root)

    # Get Merkle proof from Cassandra
    proof_from_cassandra = get_merkle_proof_by_key('A', key_index, mr)
    print("Merkle Proof from Cassandra:", proof_from_cassandra)

    # Validate Merkle proof from blockchain
    is_valid = merkle_tree.validate_proof(proof_from_cassandra,
hashlib.sha256(query_value_by_key('A').encode()).hexdigest(),
merkle_root_contract)
    print("Is Valid (Validating from Blockchain):", is_valid)

    print()

    # Perform malicious attempt
    malicious_attempt('data', 'A', '100')

    print("After Malicious Attempt:")
    print("Value of A:", query_value_by_key('A'))

    # Get Merkle proof from contract
    proof_from_contract = get_merkle_proof_by_key('A', key_index,
merkle_tree)
```

```
    print("Merkle Proof from Cassandra:", proof_from_cassandra)

    # Get Merkle Tree from Cassandra
    mr = get_merkle_tree_from_cassandra(key_index)
    root = mr.get_merkle_root()
    print("Merkle Root from Cassandra:", root)

    # Validate Merkle proof from blockchain
    is_valid = merkle_tree.validate_proof(proof_from_contract,
hashlib.sha256(query_value_by_key('A').encode()).hexdigest(),
merkle_root_contract)
    print("Is Valid (Validating from Blockchain):", is_valid)
```

Output Snippet - Client Verification of Proof

Comparison of merkle root hash stored in cassandra and the one stored in the blockchain.

```
ubuntu@group7:~/project3$ python3 p.py
Original Query Result: 10
Merkle Index: 0
Original Merkle Proof Index: [{'right': 'f5ca38f748a1d6eaf726b8a42fb575c3c71f1864a8143301782de13da2d9202b'}, {'righ
t': '8d7f6cff0755dc9b7f85cee8ac88d87f3a205bce0e61081b1a6438812241b28a'}]
Merkle Root from Contract: 012ec8f6e3357ddf6fbdf82fcae0303da3f5c6ee0348263f6ca2ab8a58295979

Before Malicious Attempt:
Value of A: 10
Merkle Root from Cassandra: 012ec8f6e3357ddf6fbdf82fcae0303da3f5c6ee0348263f6ca2ab8a58295979
Merkle Proof from Cassandra: [{'right': 'f5ca38f748a1d6eaf726b8a42fb575c3c71f1864a8143301782de13da2d9202b'}, {'righ
t': '8d7f6cff0755dc9b7f85cee8ac88d87f3a205bce0e61081b1a6438812241b28a'}]
Is Valid: True

After Malicious Attempt:
Value of A: 100
Merkle Proof from Cassandra: [{'right': 'f5ca38f748a1d6eaf726b8a42fb575c3c71f1864a8143301782de13da2d9202b'}, {'righ
t': '8d7f6cff0755dc9b7f85cee8ac88d87f3a205bce0e61081b1a6438812241b28a'}]
Merkle Root from Cassandra: 0e51e1bc0e7199eddd1d1746c1e4bfdafe2465249e2b6982f0ca99037cbe68d9
Is Valid: False
ubuntu@group7:~/project3$
```

Attack Validated.

```
eth_chainId
eth_estimateGas
eth_blockNumber
eth_getBlockByNumber
eth_sendTransaction

  Transaction: 0xa49b6fd2892206880bc8c71179d658bbdf8dbcab614182495a351fbf16b86654
  Gas usage: 89749
  Block number: 18
  Block time: Tue Apr 23 2024 11:47:17 GMT+0000 (Coordinated Universal Time)

eth_getTransactionReceipt
eth_chainId
eth_call
eth_accounts
eth_getBlockByNumber
eth_chainId
eth_chainId
eth_estimateGas
eth_blockNumber
eth_getBlockByNumber
eth_sendTransaction

  Transaction: 0x390d76d3a5e1672aab14e05f8116600a23b5f619db9bfd32e1dc9785af9f433f
  Contract created: 0x8ddcadb3b8d9934525d27223bc1c9cea3f740081
  Gas usage: 232046
  Block number: 19
  Block time: Tue Apr 23 2024 11:48:21 GMT+0000 (Coordinated Universal Time)

eth_getTransactionReceipt
eth_getBlockByNumber
eth_chainId
eth_chainId
eth_estimateGas
eth_blockNumber
eth_getBlockByNumber
eth_sendTransaction

  Transaction: 0x95790cc64d42d7dc61af0d3735ff652b0b9dfc6bc59e370784e9cf570f2a6a68
  Gas usage: 89749
  Block number: 20
  Block time: Tue Apr 23 2024 11:48:21 GMT+0000 (Coordinated Universal Time)

eth_getTransactionReceipt
eth_chainId
eth_call
```

The merkle root is stored and retrieved from the blockchain.