**Blockchain System with Credit-Based Consensus Mechanism for Industrial Internet of Things: An Application of Industry 4.0**

Now-a-days all industries like banking (ATM’s, sensor-based transaction devices), hospitals (transaction Machines) are using IIOT (Industrial Internet of Things which means small devices communicate with centralized servers to exchange data) devices to communicate with their servers. These devices will have internet connection to communicate with industries servers and these devices will have limited battery power so heavy cryptographic algorithms cannot be implemented to provide/improve security. To overcome from this issue author, us using Blockchain based algorithms such as POW (proof of work) and credit consensus. Entire Blockchain technique cannot be implemented as these devices are small and run on battery so author is using POW and Credit Consensus concept from Blockchain technique.

Below 3 problems make author to utilize only POW and Credit Consensus concept from entire Blockchain technique.

* Efficiency and Security: All transactions are safe under block chain Credit Consensus and if we use entire Blockchain then efficiency problem will raise in devices (sensors) to run entire Blockchain technique. This makes author to use only Credit Consensus.
* Transparency and Privacy: All transaction done in Credit Consensus are available publicly and there is no privacy for data. So, to provide security to data author is using symmetric encryption technique to hide data from public and can only be decrypted by industrial manager. When sensors or devices setup then industrial manager share public keys with sensors via GATEWAYS. All sensors encrypt data using public key and send to GATEWAY and GATEWAY will store at industrial server where manager can decrypt all data using keys.
* High concurrency and low throughput: As sensors report huge data to servers so concurrent requests will arrive from all sensors and then server can produce low throughput or output. To increase throughput, we are using DAG (directed acyclic graph architecture) concept. In DAG each transaction referred as node instead of maintaining multiple blocks. Running transaction as nodes take less time compare to blocks generation.

In this application three types of devices will work

* Sensors: These are small devices which interact with GATEWAYS to send/receive data. Keys also collected from GATEWAYS. Sensor will encrypt data and then generate hash code on transaction and then send to gateway. Gateway/industrial server will authenticate hash and check all transactions contains unique hash value, if hash value unique then sensor credit will increase and this hash values will be used as Proof Of work for transactions. While sending transactions sensors can report two types of attacks called ‘Lazy Tips and Double Spending’ and these two attacks can be easily detected with Credit Consensus Algorithm.
* Lazy Tips: In this malicious sensor report same hash values for all transactions and Credit Consensus POW look for new hash values. If same hash value detected for all transaction, then Lazy Tips attack or abnormal behaviour detected.
* Double Spending: In this technique sensors report success hash values of previous transactions and if POW Consensus detect old hash value, then this abnormal behaviour will be detected.

By using above two values credit positive and negative value will be calculated. Sensors also called as light node

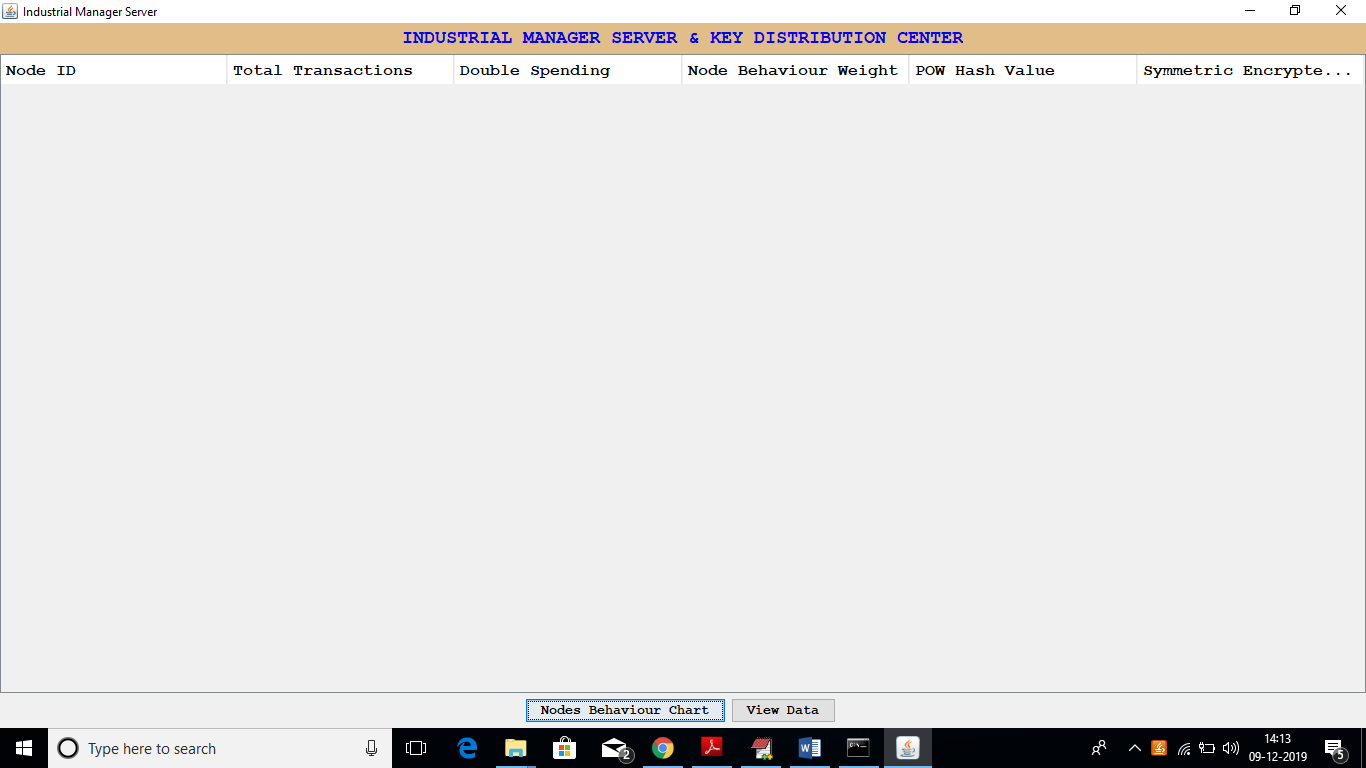
* Gateways: Also called as Full Node because it will have high energy compare to normal sensors. Gateways receive request from manager authorized sensors and then send to credit consensus POW algorithm to check sensor behaviour and then send response data to manager.
* Manager: Manager will generate public and secret keys and store it in gateways to exchange public keys with sensors. All sensors’ data can be access by this manager by using secret keys. Sensors will send data to gateways and gateways store received/processed data at manager server.

To implement above concept, we are using SHA256 for hashing and AES for data encryption to provide privacy. Here, we have designed two applications called ‘IndustrialManager and Wireless\_Sensors’.

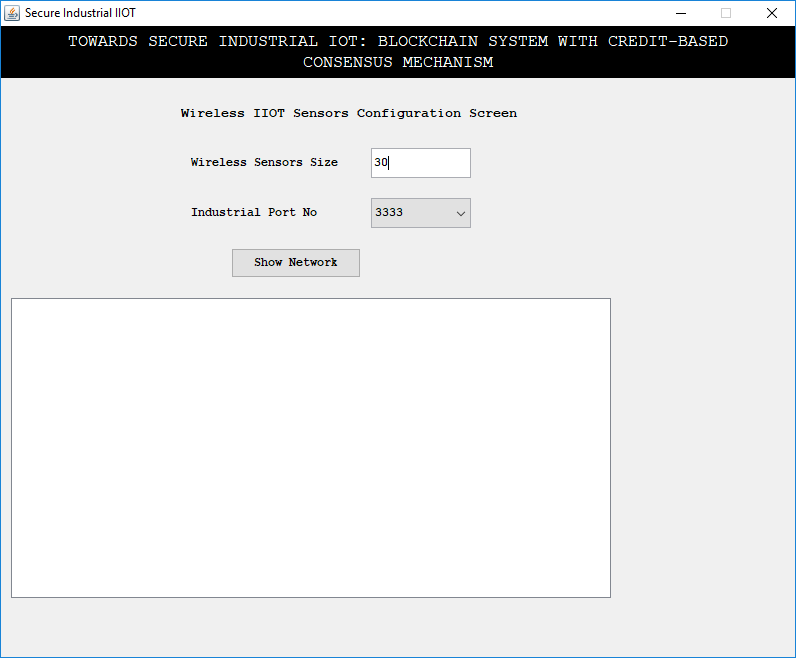
* IndustrialManager: This application responsible to generate keys for sensors and then run Credit Consensus POW algorithm to process/check each transaction send by sensors.
* Wireless\_Sensors: This is a simulation-based application which request gateways to receive keys and then send encrypted transaction to gateways for processing.

**SCREEN SHOTS**

First double click on ‘run.bat’ file from ‘IndustrialManager’ to get below screen and let it run



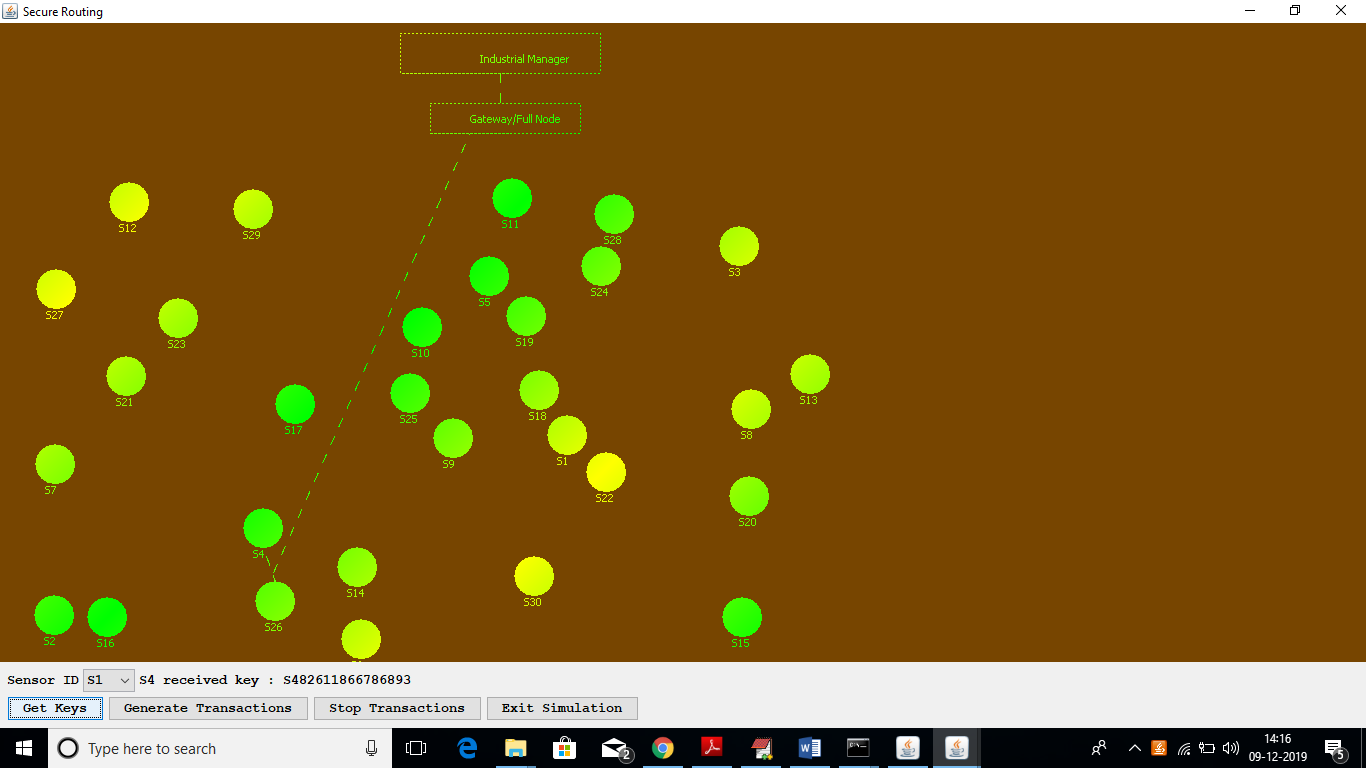
In above screen we can see each transaction details from each node and then monitor node to detect its normal or abnormal behaviour’. Now double click on ‘run.bat’ file from ‘Wireless\_Sensors’ folder to get below screen



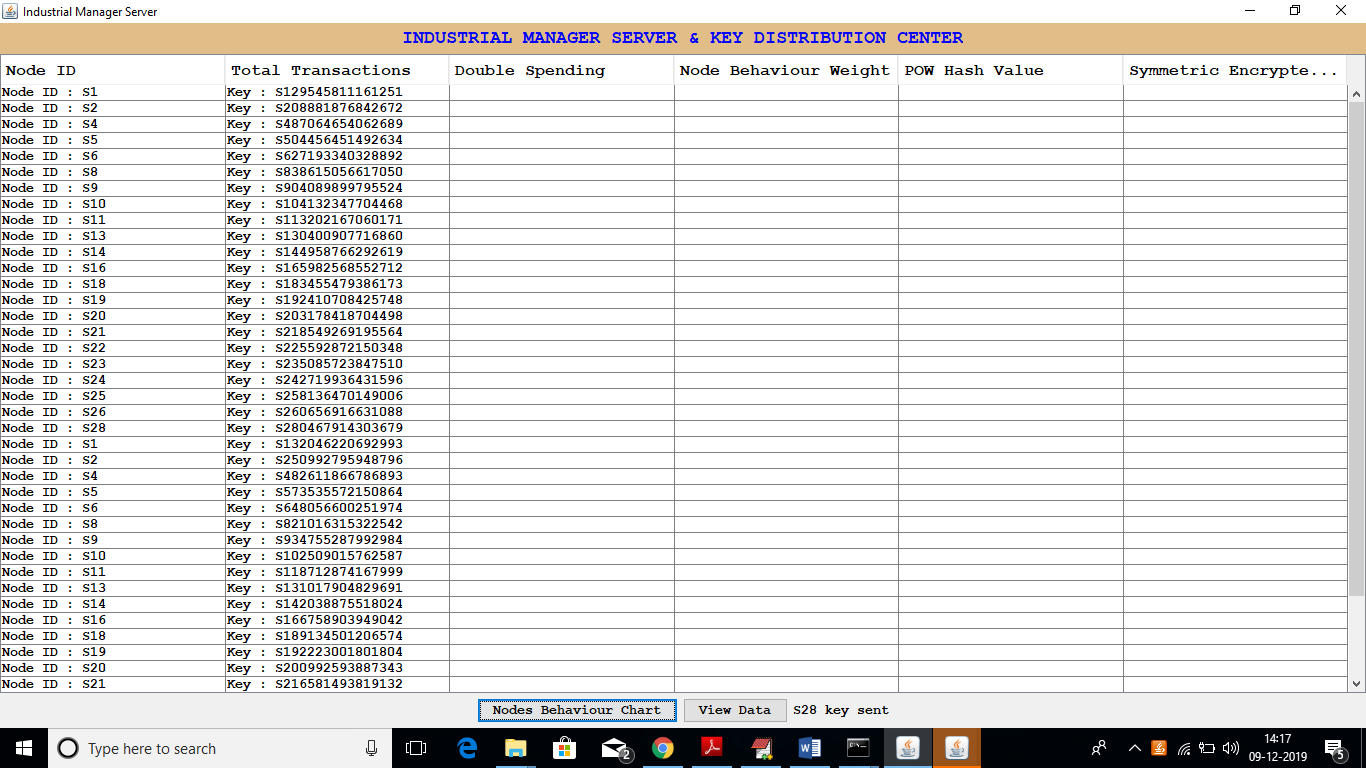
In above screen enter number of sensors and then click on ‘Show Network’ screen to get below screen



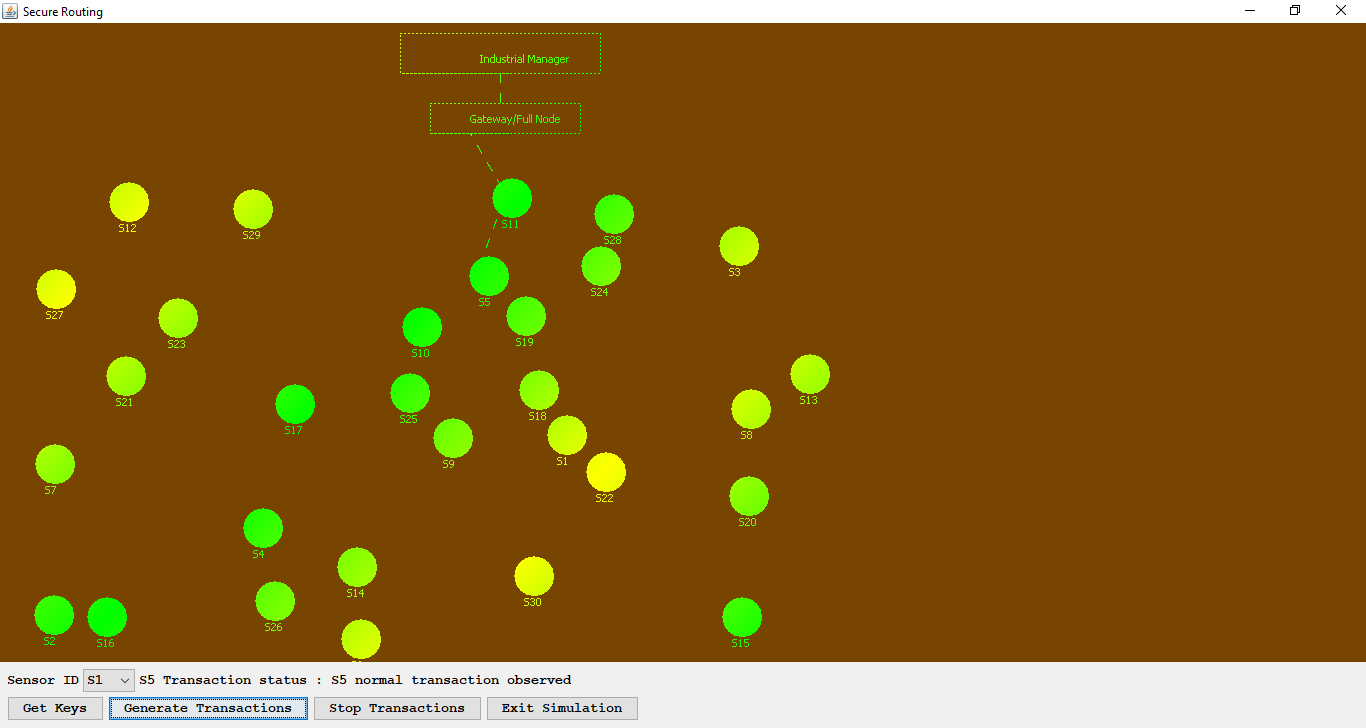
In above screen click on ‘Get Keys’ button to allow all sensors to obtain keys from gateways



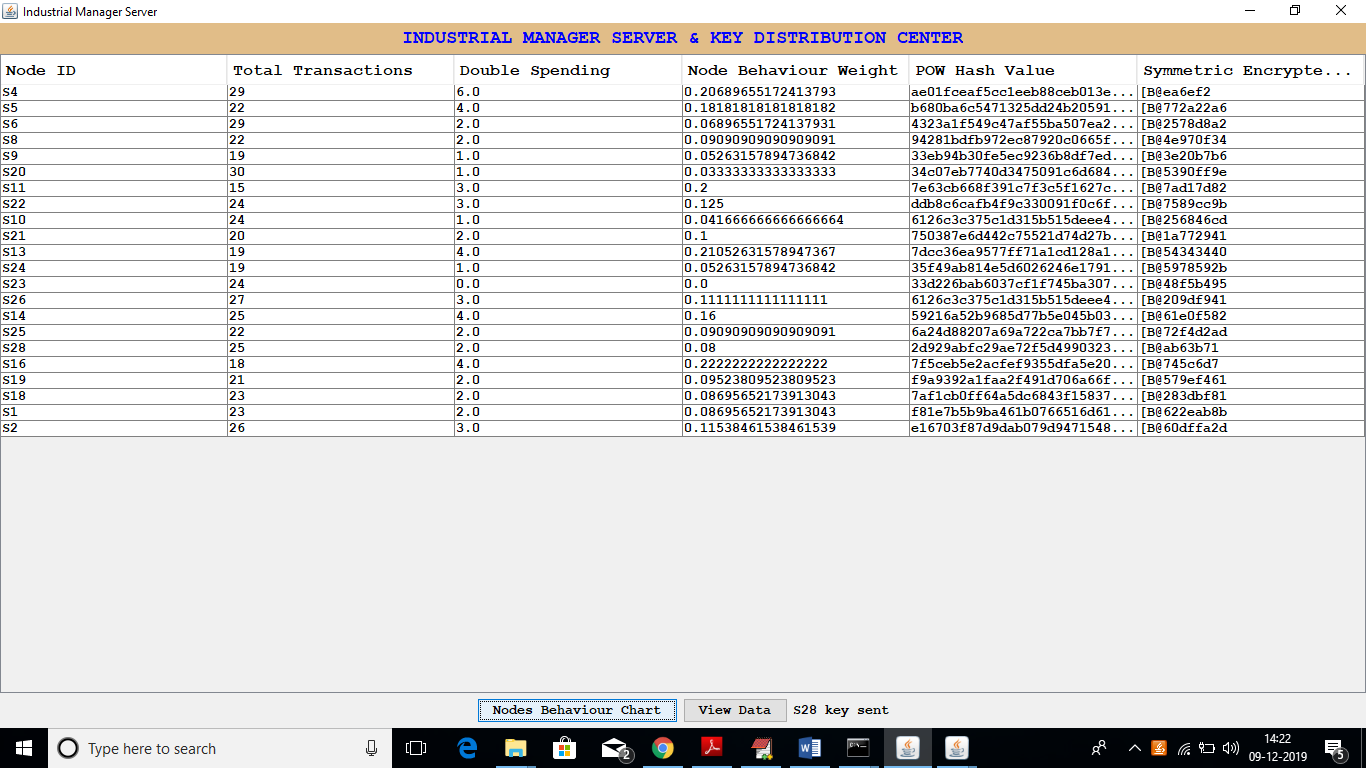
In above screen we can see each node is getting keys from gateway and these key details we can see at ‘manager screen’ also



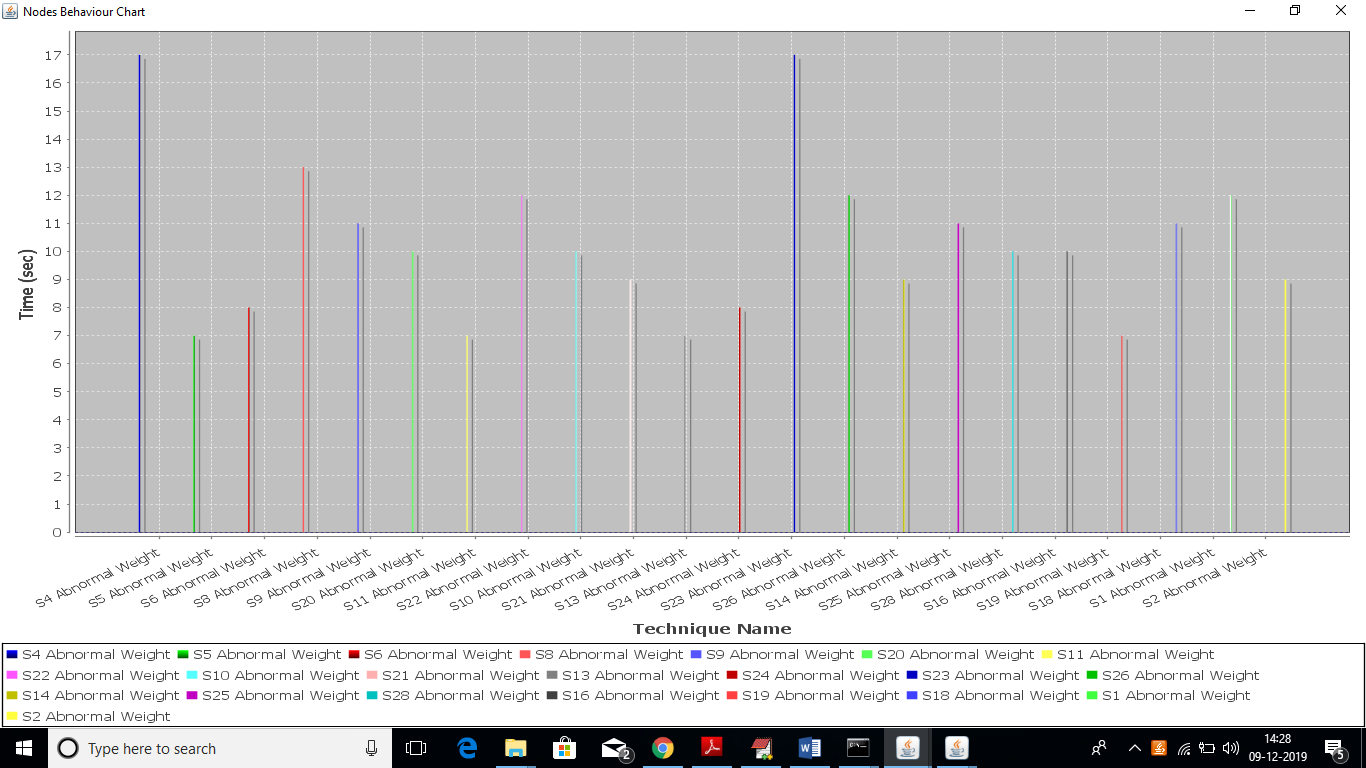
Now go to simulation screen and click on ‘Generate Transactions’ button to select random nodes and to send random transaction data to gateway. Due to random data sometime nodes will report same transaction then POW detect it as abnormal transaction. This random data and continuous data sending concept just I am using to make some node to report same data and POW can record it. After some time, you can click on ‘Stop Transaction’ to stop it.



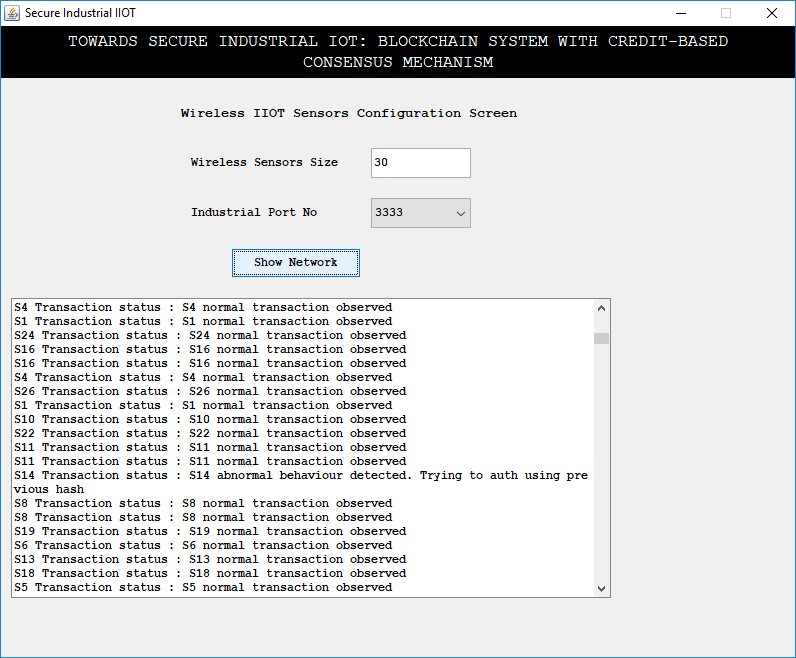
In above screen we can see transaction sending to gateway for processing. Now we can see each transaction process status at below manager screen



In above screen each node data report is recording and their hash values checking to collect their behaviour, if they send old transaction data hash value then it will be considering as ‘abnormal behaviour’. In above screen I am showing all nodes sending abnormal attack data and in real time this will not happen. Just to show the concept of old hash values I sent random continuous request and all nodes send repeated data and becomes in abnormal behaviour. From above screen we can see first nodes sent total 29 transaction and out of that 6-transaction report old hash values then it will detect as abnormal behaviour. If it reports 1 or 2 times then it can be managed and consider as normal behaviour. Now in above screen click on ‘Node Behaviour Chart’ button to see which nodes report same old hash value more no of times.



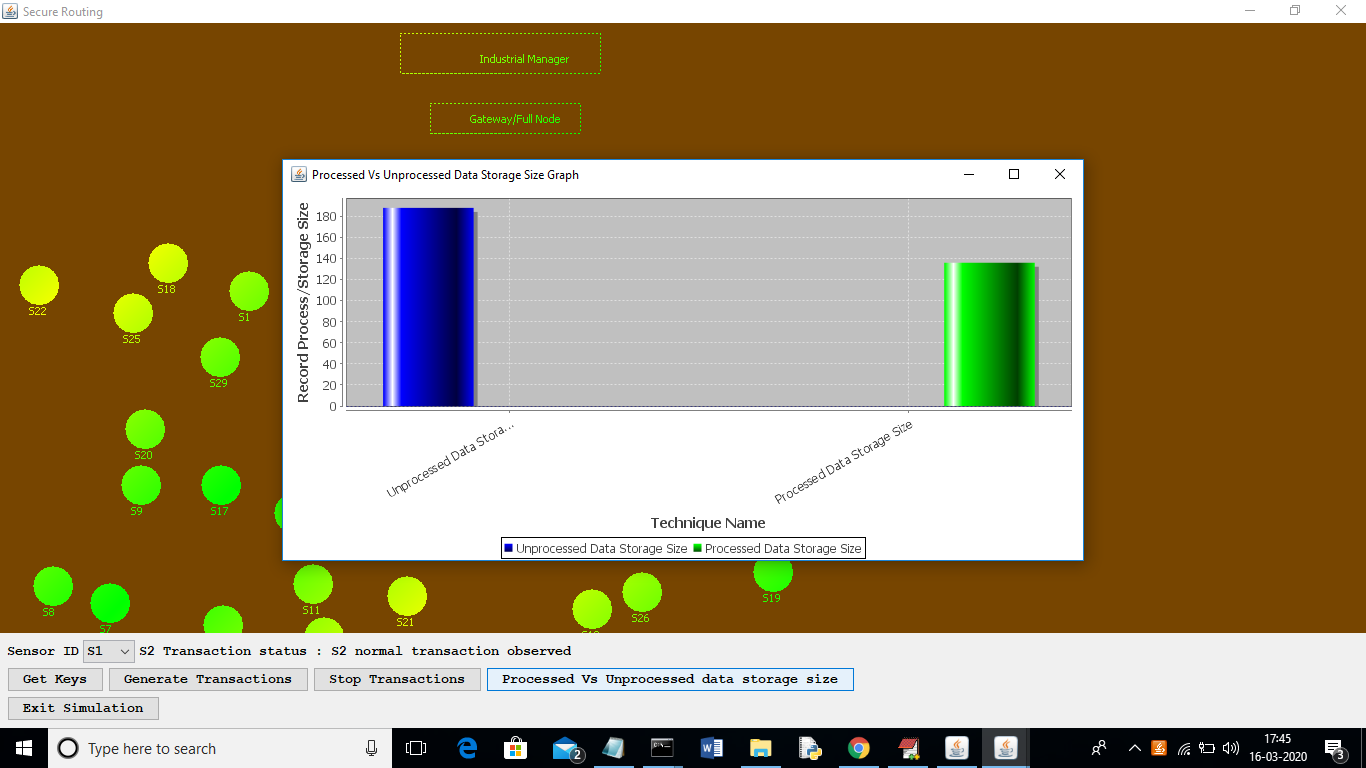
In above screen only 2 nodes report old hash values a greater number of time and he consider as abnormal nodes. S4 and S23 are the two nodes whose Double Spending Weight is 17 and other are not up to that. In above graph x-axis represents node id and y-axis represents Double Spending Weight



In above screen also we can see normal or abnormal behaviour.

As an addition to this project, we are adding techniques for processing and cleaning sensor data so that we don't have to pay extra for storage and processing when we get bad or corrupted data. Because of attacks or problems, sensors may sometimes produce corrupted data, and it can cost a lot to process and store this data. In almost every field, sensors are used to collect data. For example, in health care, sensors will be implanted in patients and send data collected to a hospital server to be processed. This processing includes running a complicated algorithm to encrypt the data and then storing it. The storage space and processing power will be wasted if we work with damaged data. So, before we process that kind of data, we will find it and get rid of it.

So, in health care, for example, the sensor has to be able to read body temperature between 10 and 105. If it reads something else, we will drop it. We can save money on storage and processing by using this method.



In the above graph, I'm getting rid of data that isn't related while sensing it, which lowers the cost of storage. The x-axis in the above graph shows the cost of storing all data and process data that is not related, and the y-axis shows the size. Based on the graph above, we can say that process data can lower both costs.

In this project, I'm also looking at values between 0 and 150 as valid sense data. Values above 150 will be thrown out as irrelevant data.