Software Design Document (SDD)

Rev: A0-03

12 April, 2017

"Private Blockchain"

From Concept to Customer

Proficient In Application Development Services
We help you deliver real solution for real business problem
Proven Expertise In Networking
Our Latest Technology Standards helps you achieve faster time to market
Excellence In Embedded Design
VVDN is a technology innovation company which stands for Voice Video Data Networking
Pioneers In Video Technology
We drive innovative approach using latest Video technology standards

Revision History:

Date	Rev No.	Description	By
25-03-2017	A0-01	First Draft	VVDN
30-03-2017	A0-02	Second Draft	VVDN
12-04-2017	A0-03	Third Draft	VVDN

Table of Contents

Table of Figures

1 Introduction

This document describes SW design details of the “Private Blockchain” (EDUK_SMOB).
This SDD is made for the reference of:-

- Product managers at VVDN & Embedded Downloads to confirm the SW design & Architecture before implementation.
- Engineering Team at VVDN for implementation and validation of the EDUK_SMOB Project.
- System Integration and Verification team at VVDN/Embedded Downloads.

1.1 Product Overview

This product is to design and develop the Private Blockchain. Private Blockchain will get the secured data (can be media, text etc.) from the sender, verify data and then will deliver this secured data to the receiver.

Following are the key features of the proposed Private Blockchain:

- Provide secure https based communication between sender and the receiver.
- Ensures the validity of data by using the CRC check across multiple nodes in the network.
- Multiple Nodes in the network will be configured to work in mesh network and maintain the Blockchain.
- Each node will actively participate in the data validation process using the predefined algorithm i.e. CRC checksum.

2 Communication among Private Blockchain Nodes

Private Blockchain is used to make secure communication between sender and receiver. The Nodes in the network will also use secure transport for communication between themselves. Nodes will use broadcast mechanism to communicate with each other via TCP/IP protocol.

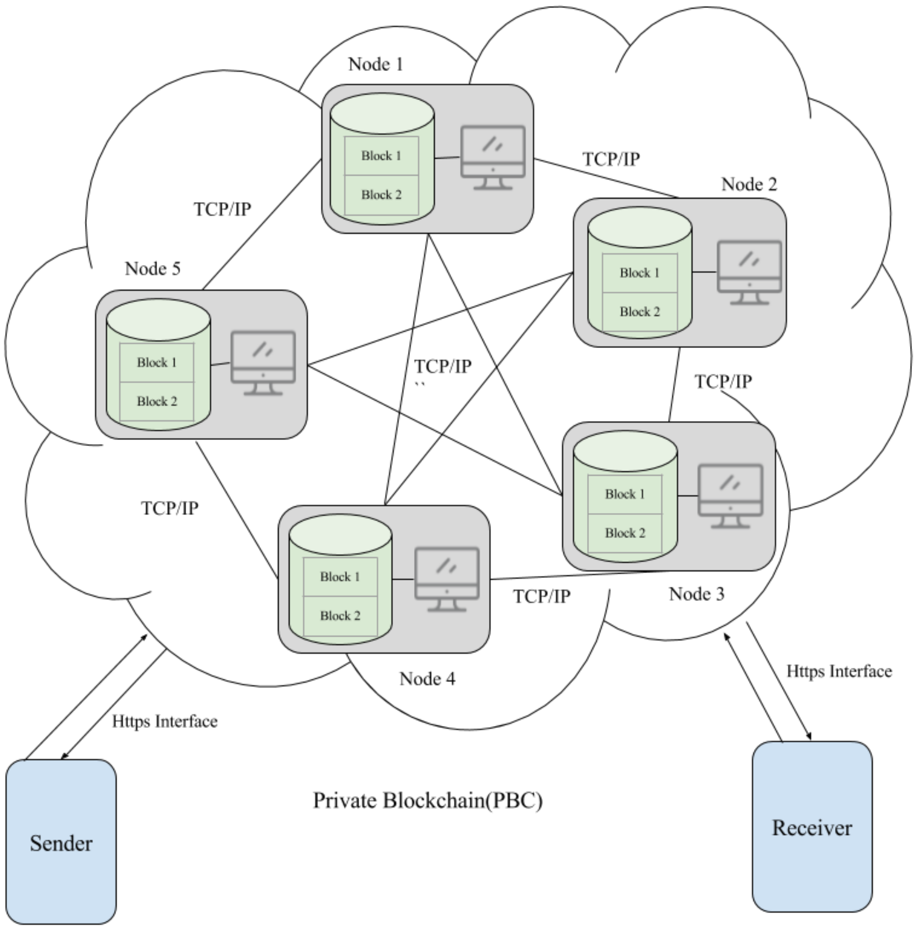


Figure 1: Private blockchain nodes communication

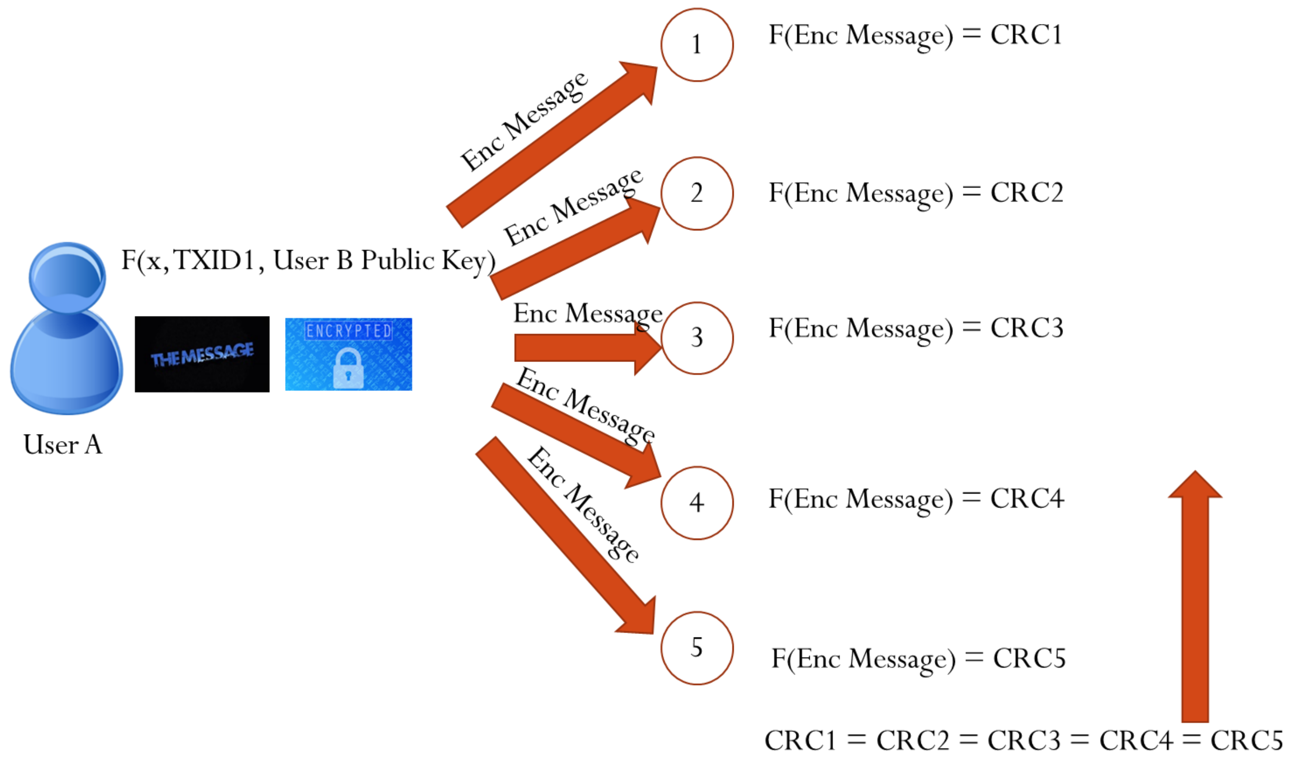
3 Private Blockchain Specifications

Private Blockchain has following functions:-

1. Secure communication between Sender and Nodes
2. Secure communication between Nodes.
3. Validation of data and adding new block in the Blockchain.
4. Secure communication between Receiver and Nodes.

3.1 Communication between Sender and Blockchain

Each node in the network will receive a Hash(Transaction Id), Encrypted Data , public address of receiver and a CRC from the sender and save it on its own local temporary file (for eg. temp_transactions.txt), then each node will validate the CRC locally. Each node also maintains a key-value pair through a map(collection) where key will be the Hash(Transaction ID) and value will be no of nodes those have verified CRC corresponding to that Hash(Transaction ID). As each node in the network verifies the CRC, and updates its local map. Nodes need to confirm CRC's of messages among each other – example - Node 1 needs to send CRC1 to Node 2, 3, 4, and 5. Node 2 needs to send CRC2 to Node1, 3, 4 ,5. Node 3 needs to send CRC 3 to Node 1,2, 4, 5. Node 4 needs to send CRC4 to Node 1, 2, 3, 5. Node 5 sends CRC5 to node 1, 2, 3, 4.
A block can be added to the blockchain by node 1 if node 1 has CRC1 = CRC2 = CRC3 = CRC4 = CRC5 (This counts as one confirmation). Now if node 2 has CRC1 = CRC2 = CRC3 = CRC4 = CRC5 this can count as a second confirmation and so on….



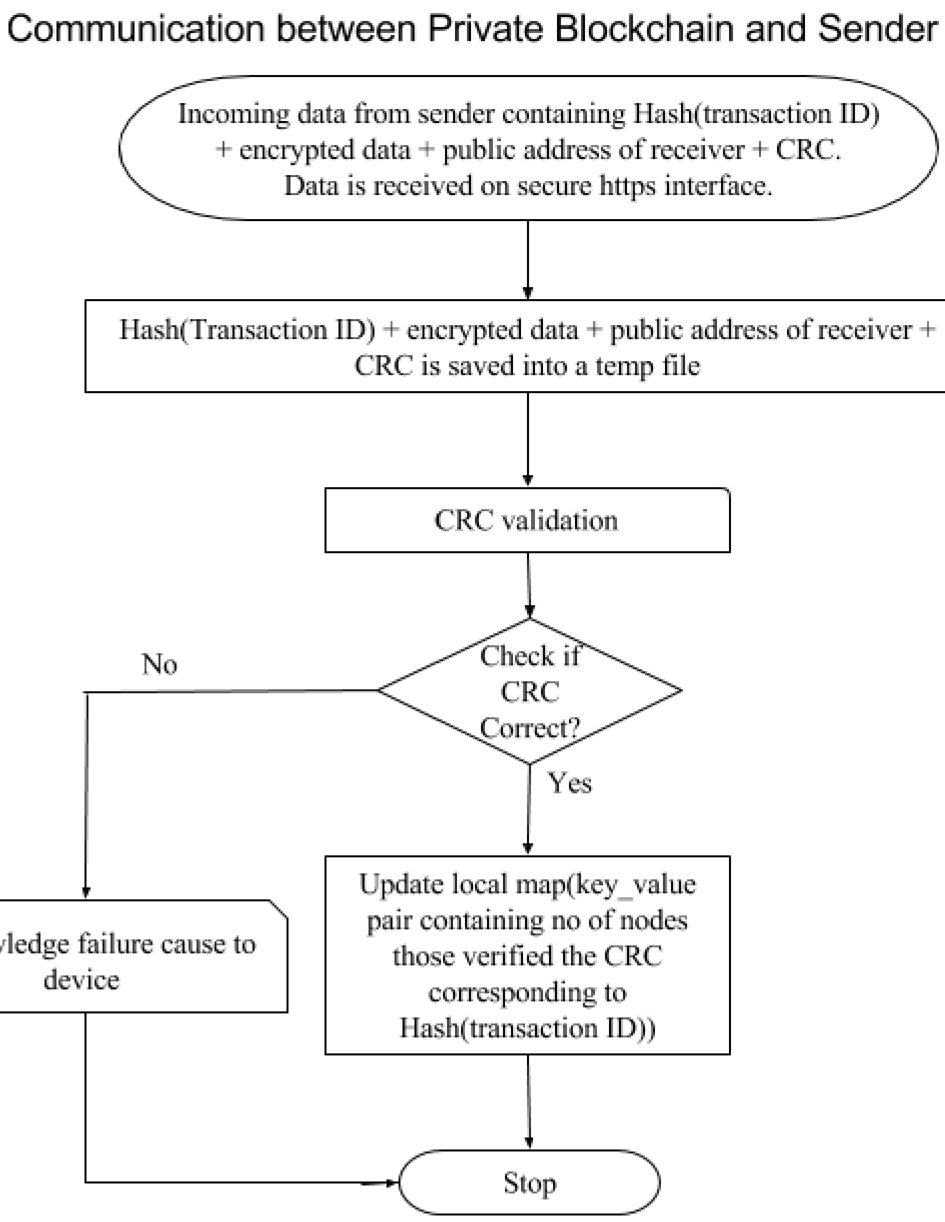

Communication between Private Blockchain and Sender
Incoming data from sender containing Hash(transaction ID) + encrypted data + public address of receiver + CRC. Data is received on secure https interface.
Hash(Transaction ID) + encrypted data + public address of receiver + CRC is saved into a temp file
CRC validation
Check if CRC Correct?
No
Yes
Acknowledge failure cause to device
Update local map(key_value pair containing no of nodes those verified the CRC corresponding to Hash(transaction ID))
Stop

Figure 2: Communication between Private Blockchain network and sender

3.1.1 CRC Validation

*A cyclic redundancy check (**CRC**) is an error-detecting code commonly used in digital networks and storage devices to detect accidental changes to raw data. Blocks of data entering these systems get a short check value attached, based on the remainder of a polynomial division of their contents.*

Every Node will calculate the CRC of the Encrypted Data and will compare with CRC sent by sender. If both CRC matches, node will update its local map.

3.2 Communication between nodes

Once the CRC has been validated by a node then that node will broadcast transaction along with its validation status to all other nodes using TCP/IP protocol. Each node will receive transaction and its status and update its local map. The entire blockchain needs to be synchronized between nodes every 1s. Every node keeps a copy of the blockchain.

3.3 Adding new block into Blockchain

Each node will check on the local map. Once it is found that CRC is verified by the sufficient number of nodes , it will initiate the process of adding the block to the Blockchain. Please check comments at 3.1 By default each node will expect minimum 80% (To be discussed) of nodes to verify the CRC of data.
Once the block is created , node will delete entry from its own temp file (temp_transactions.txt).

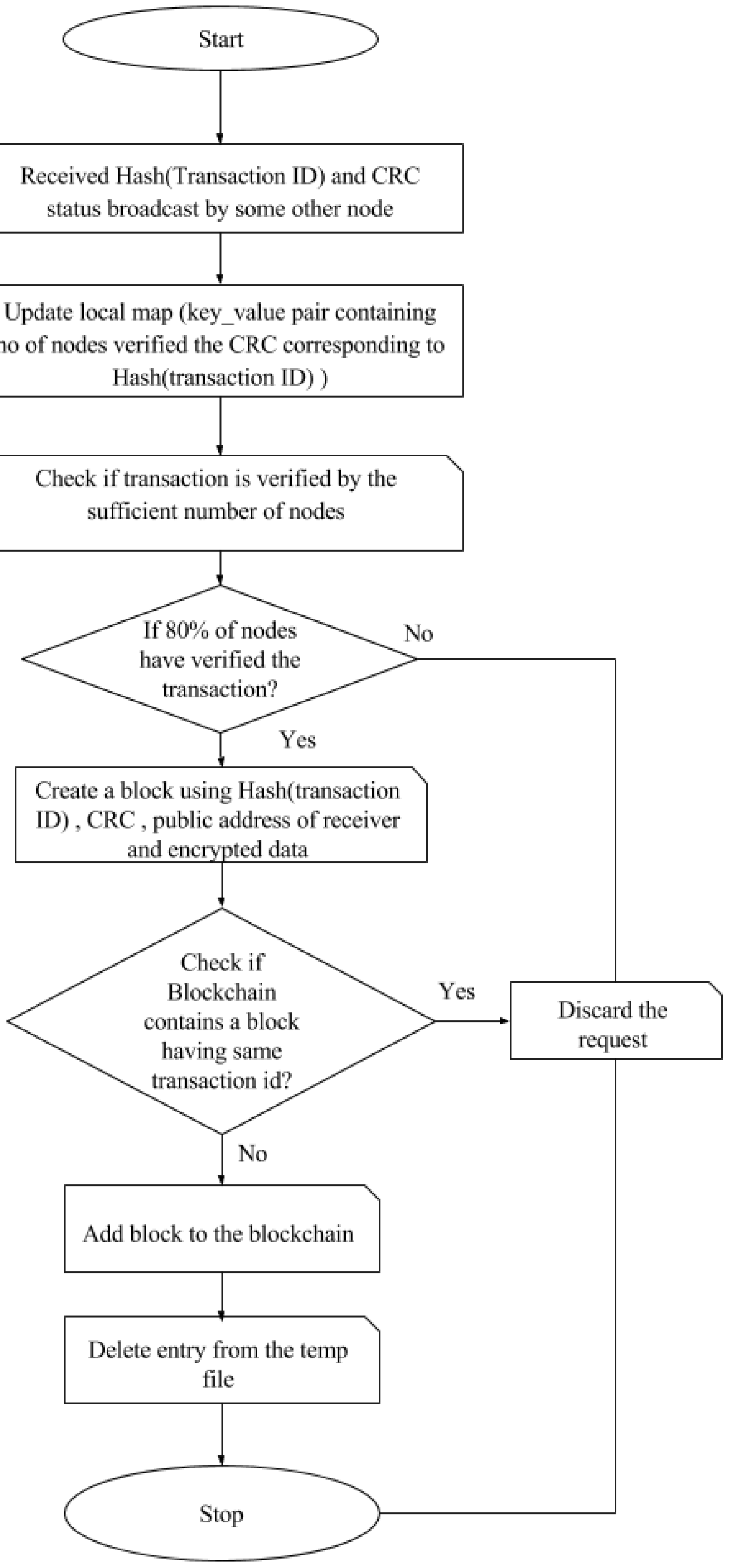

Start
Received Hash(Transaction ID) and CRC status broadcast by some other node
Update local map (key_value pair containing no of nodes verified the CRC corresponding to Hash(transaction ID))
Check if transaction is verified by the sufficient number of nodes
If 80% of nodes have verified the transaction?
No
Yes
Create a block using Hash(transaction ID) , CRC , public address of receiver and encrypted data
Check if Blockchain contains a block having same transaction id?
Yes
Discard the request
No
Add block to the blockchain
Delete entry from the temp file
Stop

Figure 3: Communication between nodes and addition of block to Blockchain

3.4 Block creation process

Once a node finds that transaction is verified by the sufficient number of nodes. It will create a block with Hash(transaction ID) , CRC and Encrypted Data.

A Block is divided into 3 parts:

1. Block Header : Block Header will contain the Hash of Previous block ie. The hash of Hash (Transaction ID) + Encrypted Data + public address of receiver + CRC of previous block and Timestamp.

2. Block Data : Block Data will contain Hash(Transaction ID) , Encrypted Data , Public address of receiver and the CRC of encrypted data.

3. Block Hash : Block Hash will contain the hash of the combination of Hash(Transaction ID), encrypted data , Public address of receiver and the CRC.

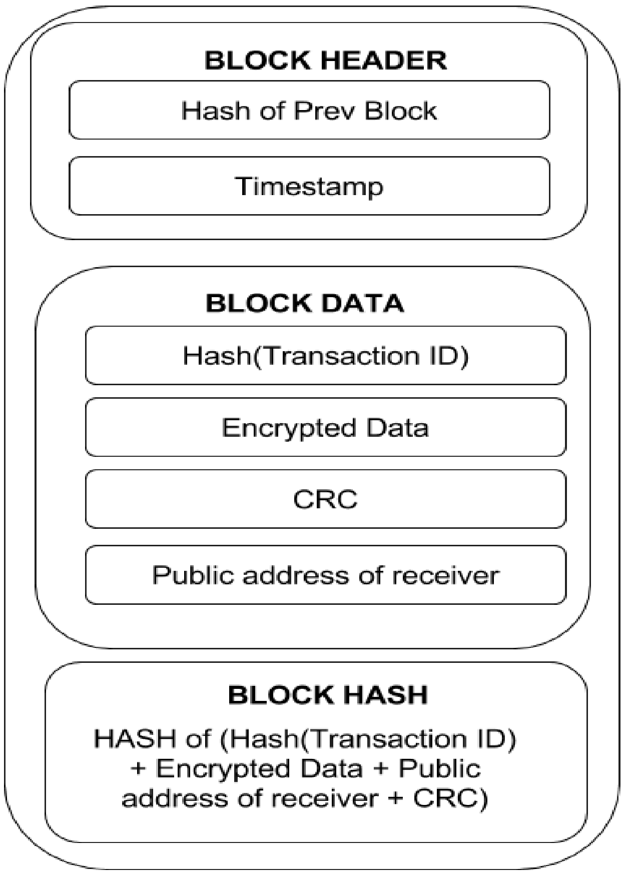

BLOCK HEADER
Hash of Prev Block
Timestamp
BLOCK DATA
Hash(Transaction ID)
Encrypted Data
CRC
Public address of receiver
BLOCK HASH
HASH of (Hash(Transaction ID) + Encrypted Data + Public address of receiver + CRC)

Figure 4: Block Creation Process

3.5 Communication between Receiver and Private Blockchain Network

Receiver will continuously poll the nodes with the Hash(Transaction ID). If a block having the same Hash(Transaction ID) exists in the Blockchain, it will return the encrypted data to the receiver.

After the successful transfer of encrypted data to receiver, the corresponding block from the Blockchain will be deleted.

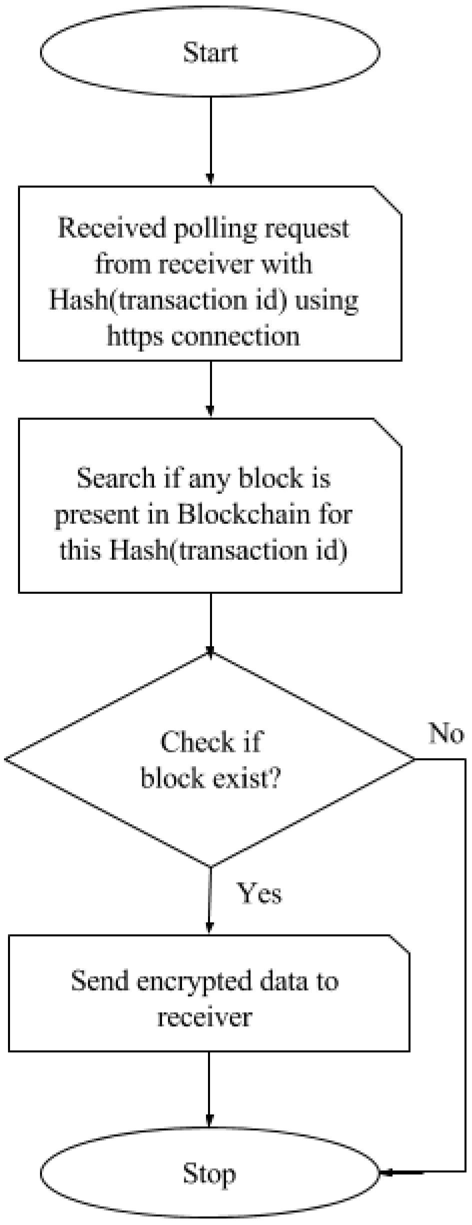

Start
Received polling request from receiver with Hash(transaction id) using https connection
Search if any block is present in Blockchain for this Hash(transaction id)
Check if block exist?
No
Yes
Send encrypted data to receiver
Stop

Figure 5: Communication between receiver and private Blockchain

3.6 Deletion of block from the Blockchain

Once the data is verified by the receiver, the receiver will send the acknowledgment ~~to a random node.~~ Node will delete the block from the Blockchain and broadcast the same to all other nodes.

Deletion of a block needs to happen the same way as creation of a block. So the delete command will be sent from the receiver to all the nodes on the network with a CRC. Only once the CRCs of all the nodes have been confirmed(same way as commented in 3.1, will the block be deleted).

On receiving the confirmation, each node will delete block from the Blockchain and update previous hash of next block.

4. Appearance of new node in the network

Whenever a new node appears in the network, node will broadcast its appearance in the network and will communicate with ~~a random node~~ to get the updated Blockchain. The new node needs to sync with multiple nodes, not just one random node. – If it syncs with only one node, the data can be manipulated.

If node is already present in the network then node will send the last hash of block it has in the Blockchain and will communicate with a random node to get the remaining blocks.

5 System requirement

- Java 1.8+

6 Services used to communicate with Sender and Receiver

Below is the list of services required for communicating with Sender and Receiver:

6.1 Send Data

Description	*To send the data to receiver through Private Blockchain.*
Method	*POST*
Input Params	*Data (Hash(Transaction ID) + Encrypted Data + Public address of receiver + CRC)* *Type : String*
Output Params	*status_code* *Type : I nteger*

6.2 ReceiveData

Description	*Request for data from node for the given Hash(transactionID)*
Method	*POST*
Input Params	*transaction_id : Hash(id of transaction)* *Type : String*
Output Params	*Data (encrypted)* *Type : String*

6.3 DeleteData

Description	*Request to delete the data from the Blockchain by the receiver*
Method	*POST*
Input Params	*transaction_id : Hash(id of transaction)* *Type : String*
Output Params	*status_code* *Type : Integer*

7. Broadcast Methods used to communicate among the nodes

Below is the list of methods required for communicating among the nodes:-

7.1 broadcastTransaction(String transactionId , String crc , boolean status)

Description	*After validating CRC , each node will broadcast transaction along with CRC to all other nodes*
Input Params	*transactionId: Hash(id of transaction)* *crc : generated CRC to validate the data* *status : status of validating CRC*

7.2 broadcastDeleteTransaction(String transactionId)

Description	Each node will receive delete information from the other node and delete entry from its local table corresponding to that node.
Input Params	*transactionId :Hash(id of transaction)*

7.3 getBlockchain()

Description	When a new node appears in the network, it will request whole Blockchain from a random node.
Output Params	*blockchain : blockchain available on random chosen node*

7.4 getBlocks(String blockHash)

Description	A node will request all the blocks after the block hash it has in Blockchain.
Input Params	*blockHash: the hash of block received by that node at last.*

7.5 broadcastNewNodeAppearance(String ipAddress)

Description	A node will broadcast its appearance in the network. All nodes will save its IP address into their nodes list.
Input Params	Ip address of new node.