PSEUDO CODE

1. Validity checks for cryptographic values

Using nacl.signing for digital signature and verification

```
    private_key 🡐 generate_private_key()
    public_key 🡐 generate_public_key()
    share_public_to_validators()
        Function broadcast_msg()
            signature 🡐 sign_message(private_key)
            send(msg, signature, to=destination)
    Function receive_msg()
            public_key.verify(signature)
```

-

2. “sync up” replicas that got behind

```
In Block_tree:
Function process_vote:
        If node is leader & vote_count == 2*f+1:
// By broadcasting leader’s state list to every validator once we know the leader is a loyal
one by getting 2f+1 votes for its proposal, validators can sync up
            broadcast_leader_state_list()
Function receive_state_list(states):
        If not compare(states, local.states):
            Update state
            Update high_commit_qc
```

-

3. Client requests: de-duplication

In Mempool, maintain state of transactions and compare when inserting a new transaction into the mempool

```
Function insert(transaction t):
    If not find(t , transaction_map):
        transaction_map.add(t, initial_state)
```

4. Verification of submitted command committed to the ledger

When leader commits a transaction into ledger, it broadcasts committed message to client of the transaction .

```
Function process_qc():
    If node is leader:
        Client_id = get_client_id(qc.vote_info.tid)
        broadcast(“Commit”, {tid, safety.signature}, to=client_id)
```

In Client:

```
Function receive_commit():
    verify_commit_message()
    update_commit_status(tid)
```

-