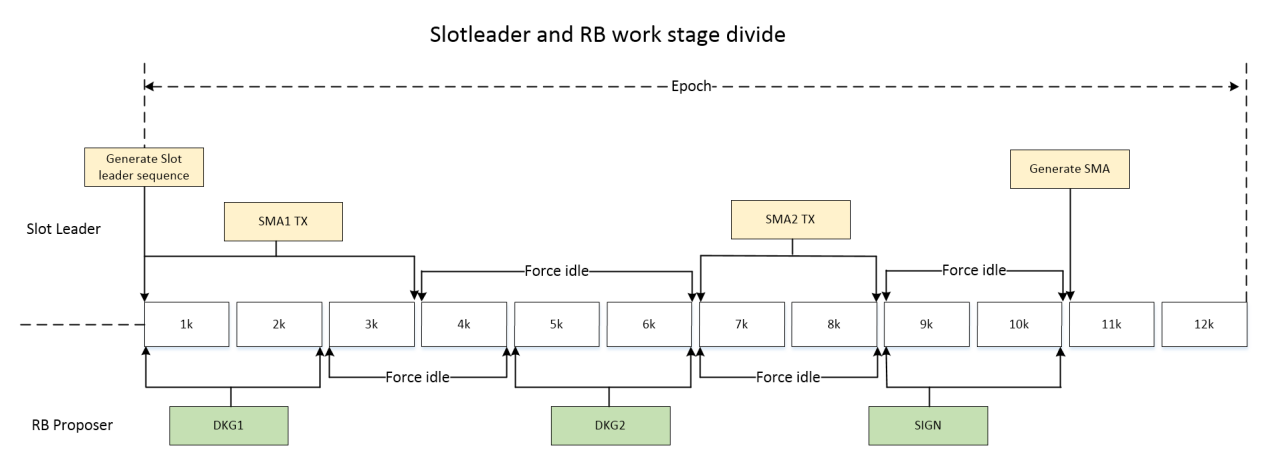
# Random Beacon structure

## Introduction

Random Beacon is the random generator simulated by the Random Number Proposer Group. It outputs a random number in epochn(n >= 1) for the epochn+1.

The total process can be split three stages : DKG1, DKG2, SIG. Every stage is fixed to predefined period of each epoch (AS following chart).



The reason of 2k slots between stages each other is that, 2k slots at least have k blocks, and k is the Persistence height to prevent block chain roll back.

Random proposers nodes create and send transactions according to their identities in the corresponding stage. The transactions should be verified and executed by precompiled contract, then to be saved on the block chain. At the last stage (SIG), precompiled contract should compute the random number according the block chain data saved in the previous stage, and save the random number to block chain.

There are two class in the code : **RandomBeacon** and **RandomBeaconContract**. RandomBeacon is used to send transaction, RandomBeaconContract is the precompiled contract, used to execute transaction and compute the random number.

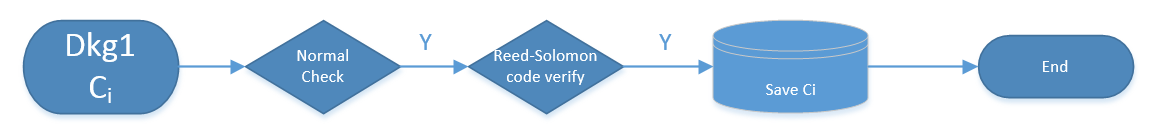
Every miner node has a timer to trigger RandomBeacon::Loop once for each slot. RandomBeacon should send transaction or wait next stage or do nothing according to the current stage.

(chart)

## DKG1

DKG1 is fixed to 1k~2k of each epoch. The random proposer node’s **RandomBeacon** should create and send DKG1 transaction, which include DKG commitment info.

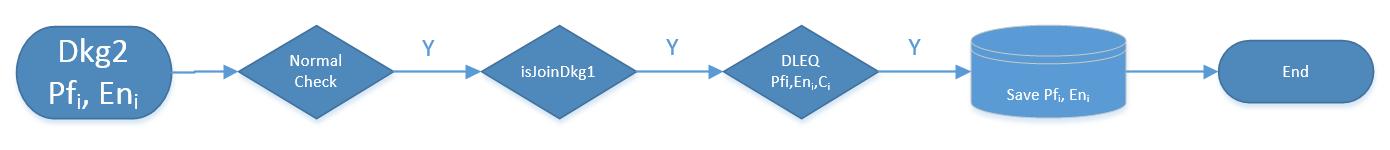
**RandomBeaconContract** should first do NormalCheck which contains whether in the right time, has the authority, and so on. Then verify Ci. If passed it will save Ci in the state database.



## DKG2

DKG2 is fixed to 5k~6k of each epoch. The random proposer node’s **RandomBeacon** should create and send DKG2 transaction, which include DKG encrypt share info and DLEQ proof info.

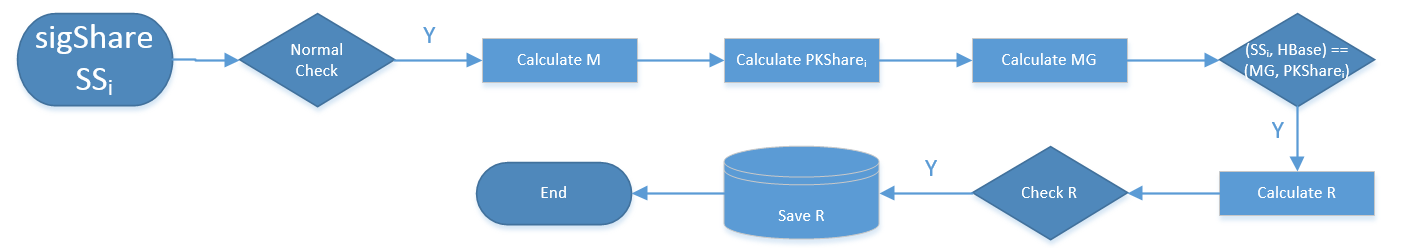
**RandomBeaconContract** should first do NormalCheck, then check whether the proposer participated in the dkg1. Then verify DLEQ to make sure Proofi, Enshare i is valid. If passed it will save Proofi, Enshare i in the state database.



## SIG

SIG is fixed to 9k~10k of each epoch. The random proposer node’s **RandomBeacon** should create and send SIG transaction, which include signature share info.

**RandomBeaconContract** should first do NormalCheck, then check the validation of parameter SSi. If passed it will save SSi in the state database. If there is enough people commit the signature, we calculate the Random for the next epoch, then verify the Random, if passed it will save the Random in the state database.



1. **total graph**

