# Slot leader selection

## Introduction

Slot leader selection use to select unique miner of slots from epoch leaders. Slot leader selection mainly select a sequence, and use this sequence to match previous epoch leaders to get the unique slot leader.

The slot leader should be issue block anonymously and the block should be verified publicly.

## Time divide

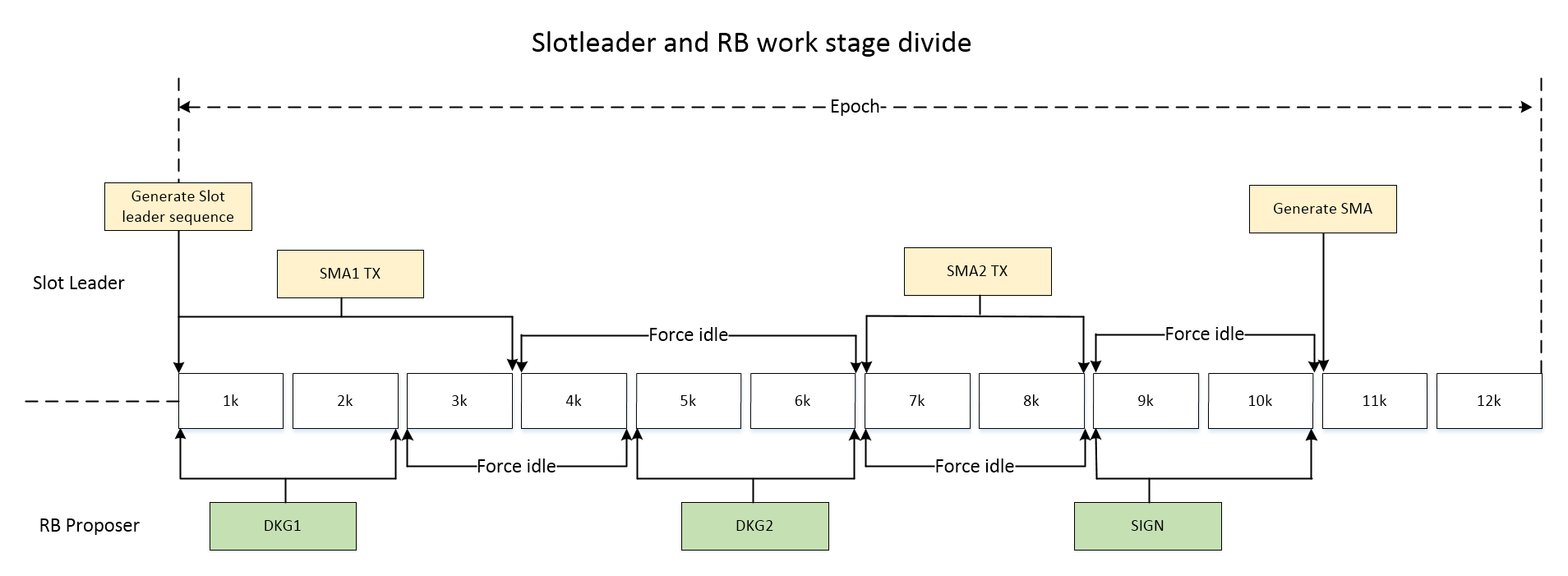
Slot leader selection’s execution period based on 1 epoch, and the behavior is the same in each epoch.

Divide the slots in each epoch into 10 groups, each include K slots.

The selection cycle is designed to be send Stage 1 transactions in 1K, and forced idle in 2K, 3K~4K to send Stage2 transactions, 5K forced idle, and in 6K to calculate SMA. This SMA value is stored in the local LevelDB and is used by the next epoch to calculate the slot leader. Since the RB calculates the random number at 7k, the slot leader selection need to generate the SMA to end before 7k to ensure that the result cannot be changed.

If the transaction arrives at the precompiled contract within the forced idle time range, the information is not allowed to write to StateDB. For example, only the Stage2 transaction received during the 3k~4k process allows data to be written to the StateDB, and the Stage2 transactions received at other times are not written to the StateDB.

As shown below:



The main steps include:

1. Generate slot leader sequence by RB and PKs. Use the Epoch Leader generated by the previous Epoch as the basis to match the slot leader sequence, and calculate the current block of each slot of the epoch;

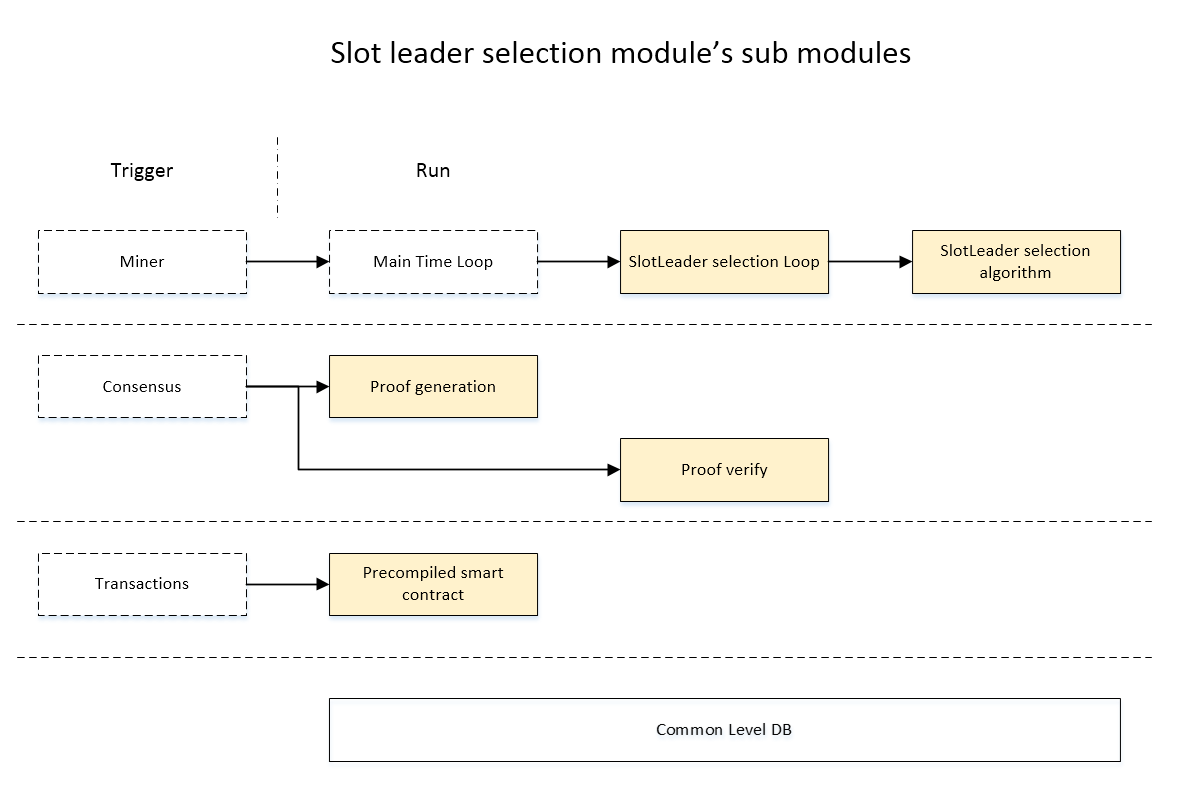
2. The epoch Leader selection module (external) generates the epoch Leader of the current epoch for transaction sending and next epoch mine;

3. Node in current epoch leader sends a transaction;

4. Node in current epoch leader calculate the SMA for the next Epoch slot selection before starting the generation of the RB random number;

## Code structure

The slot leader function code is mainly composed of the following parts: work loop, selection algorithm, pre-compiled contract, Proof generation, proof verification, and LevelDB encapsulation shared by POS.



Code is in these paths below:

|  |
| --- |
| pos/slotleader/\*.go  core/vm/slot\_leader\_selection\_contract.go  pos/posdb/\*.go  pos/postools/slottools/\*.go |

## Work loop

The work loop is periodically triggered by the Main Time loop. During the running process, the current epoch ID and slot ID are obtained. For each epoch ID, four execution phases are divided according to the slot ID. They are:

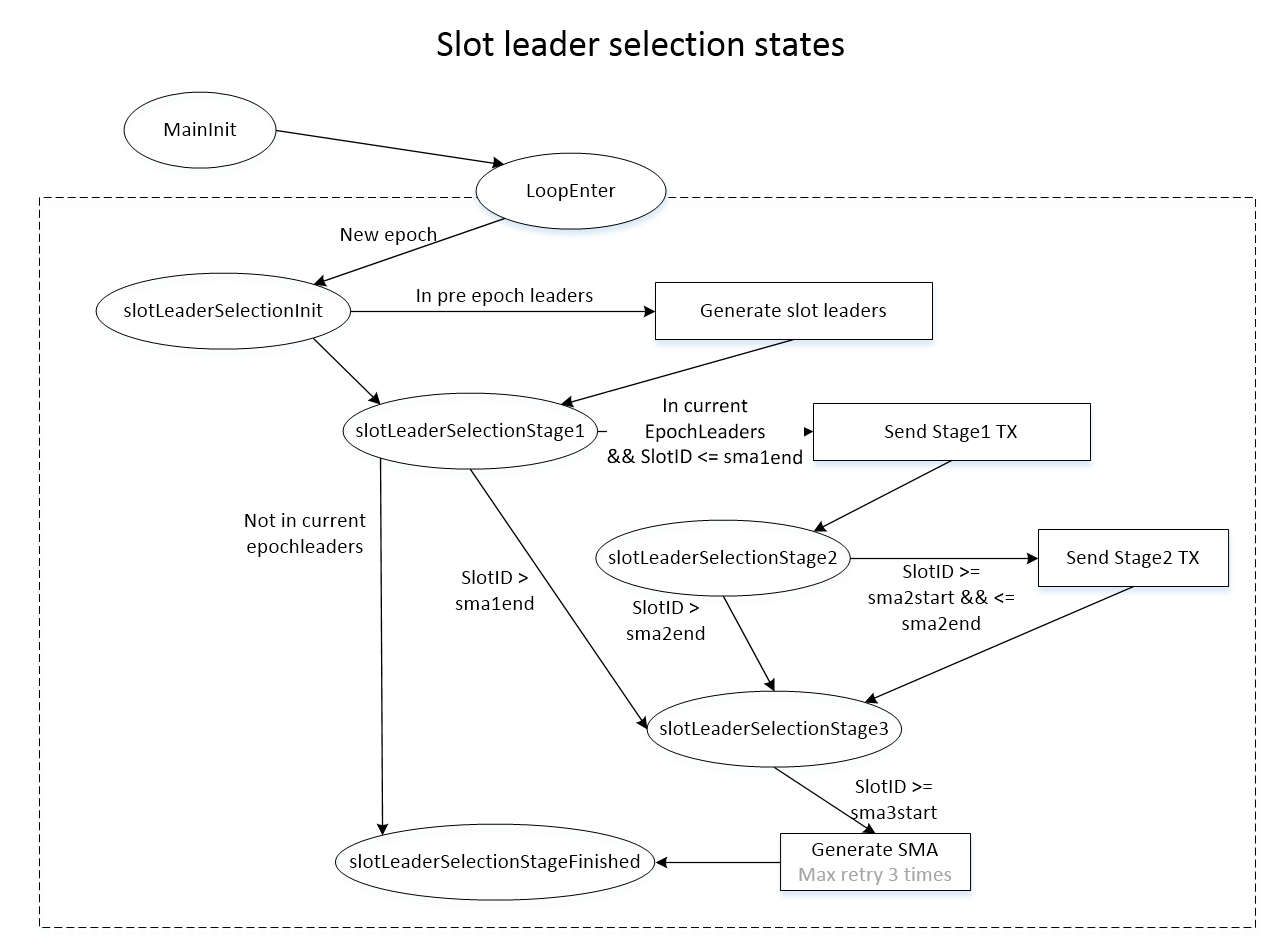
Stage 1: slot leader selection initial;

Stage 2: generate slot leader; Send Stage1 TX;

Stage 3: Send Stage2 TX;

Stage 4: Compute and Generate SMA;

Stage 5: Complete, wait for the next epoch.



The corresponding code is in the following location:

|  |
| --- |
| pos/slotleader/workflow.go |

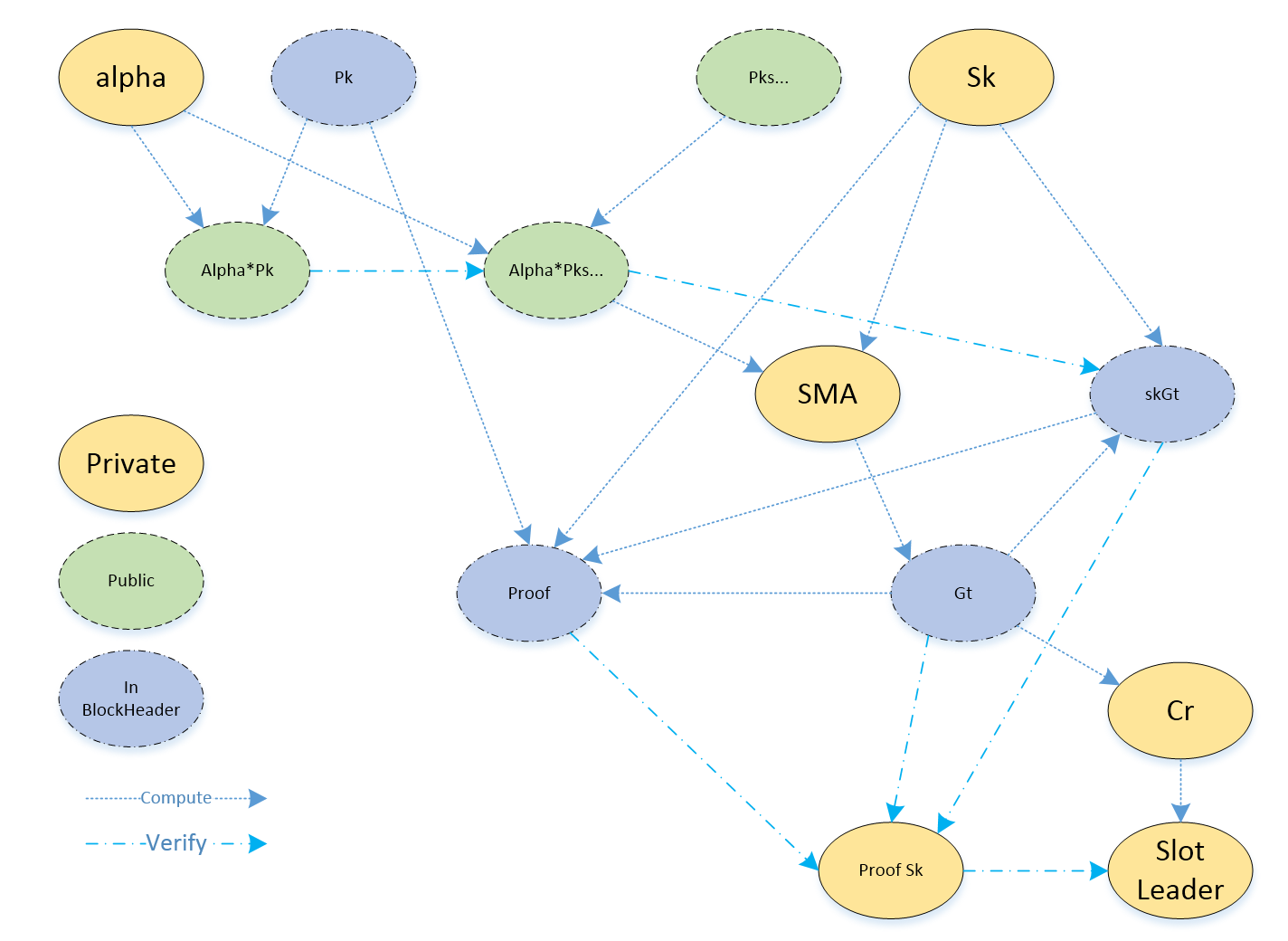
## Selection algorithm

（Wait for Zhao xiaofeng fill）

## Precompile contract

（Wait for Zhao xiaofeng fill）

## Proof generate



（Wait for Zhao xiaofeng fill）

## Proof verify

（Wait for Zhao xiaofeng fill）