I. Vetomint consensus algorithm

Algorithm 1 Vetomint consensus algorithm

```
Initialization:
  $round_p := 0$                                                    /* current round number */
  $step_p \in \{propose, prevote, precommit\}$
  $decision_p := nil$
  $lockedValue_p := nil$
  $lockedRound_p := -1$
  $validValue_p := nil$
  $validRound_p := -1$
upon start do $StartRound(0)$
Function $StartRound(round)$ :
  $round_p \leftarrow round$
  $step_p \leftarrow propose$
  if $\text{proposer}(round_p) = p$ then
    if $validValue_p \neq nil$ then
      $proposal \leftarrow validValue_p$
    else
      $proposal \leftarrow getValue()$
    broadcast $\langle \text{PROPOSAL}, round_p, proposal, validRound_p \rangle$
  else
    schedule $OnTimeoutPropose(round_p)$ to be executed after $timeoutPropose(round_p)$

// on-proposal
upon $\langle \text{PROPOSAL}, round_p, v, -1 \rangle$ from $\text{proposer}(round_p)$ while $step_p = propose$ do
  if $valid(v) \wedge (lockedValue_p = v \vee (favor(v) \wedge lockedRound_p = -1))$ then
    broadcast $\langle \text{PREVOTE}, round_p, id(v) \rangle$
  else
    broadcast $\langle \text{PREVOTE}, round_p, nil \rangle$
  $step_p \leftarrow prevote$

// on-4f-non-nil-prevote-in-propose-step
upon $\langle \text{PROPOSAL}, round_p, v, vr \rangle$ from $\text{proposer}(round_p)$ AND $4f+1$ $\langle \text{PREVOTE}, vr, id(v) \rangle$ while $step_p = propose \wedge (vr \geq 0 \wedge vr < round_p)$ do
  if $valid(v) \wedge ((favor(v) \wedge lockedRound_p < vr) \vee lockedValue_p = v)$ then
    broadcast $\langle \text{PREVOTE}, round_p, id(v) \rangle$
  else
    broadcast $\langle \text{PREVOTE}, round_p, nil \rangle$
  $step_p \leftarrow prevote$

// on-4f-non-nil-prevote-in-prevote-step
upon $\langle \text{PROPOSAL}, round_p, v, * \rangle$ from $\text{proposer}(round_p)$ AND $4f+1$ $\langle \text{PREVOTE}, round_p, id(v) \rangle$ while $valid(v) \wedge step_p \geq prevote$ for the first time do
  if $step_p = prevote$ then
    $lockedValue_p \leftarrow v$
    $lockedRound_p \leftarrow round_p$
    broadcast $\langle \text{PRECOMMIT}, round_p, id(v)) \rangle$
    $step_p \leftarrow precommit$
  $validValue_p \leftarrow v$
  $validRound_p \leftarrow round_p$
// on-4f-nil-prevote
upon $4f+1$ $\langle \text{PREVOTE}, round_p, nil \rangle$ while $step_p = prevote$ do
  broadcast $\langle \text{PRECOMMIT}, round_p, nil \rangle$
  $step_p \leftarrow precommit$

// on-5f-prevote                                        /* Early termination of prevote phase */
upon $5f+1$ $\langle \text{PREVOTE}, round_p, * \rangle$ while $step_p = prevote$ do
  if $4f+1$ $\langle \text{PREVOTE}, round_p, id(v) \rangle$ is received then
    broadcast $\langle \text{PRECOMMIT}, round_p, id(v) \rangle$
  else
    broadcast $\langle \text{PRECOMMIT}, round_p, nil \rangle$
  $step_p \leftarrow precommit$

// on-5f-precommit
upon $5f+1$ $\langle \text{PRECOMMIT}, round_p, * \rangle$ for the first time do
  schedule $OnTimeoutPrecommit(round_p)$ to be executed after $timeoutPrecommit(round_p)$

// on-4f-non-nil-precommit
upon $\langle \text{PROPOSAL}, r, v, * \rangle$ from $\text{proposer}(r)$ AND $4f+1$ $\langle \text{PRECOMMIT}, r, id(v) \rangle$ while $decision_p = nil$ do
  if $valid(v)$ then
    update height, reset all, and call $StartRound(0)$
Function $OnTimeoutPropose(round)$ :
  if $round = round_p \wedge step_p = propose$ then
    broadcast $\langle \text{PREVOTE}, round_p, nil \rangle$
    $step_p \leftarrow prevote$
Function $OnTimeoutPrecommit(round)$ :
  if $round = round_p$ then
    $StartRound(round_p + 1)$
```

We assume $6f+1$ voting power where at most f of it is byzantine.