# Consensus Algorithms in Wireless Blockchain System

## 1 Consensus Algorithm in Each Round

**Algorithm 1** The SWIB Blockchain Consensus Protocol

**while** true **do**
  //** Iteration for round $r$
  ▷ Initialization:
  //** Broadcast transactions
  **for** $j < K$ slots **do**
    $BroadcastMSG()$
    $j = j + 1$
  **end for**
  ▷ Consensus Process:
  $Rds^r = GenerateRandomValue(r, B_H^{r-1}, sig_{full}^{r-1})$
  $result, proof$ = Block Proposer Election$(sk, Rdm^r)$ ▷ Block Proposer Election
  **if** $result == True$ **then** ▷ Block Generation
    $B^r$ = Generate Block$(B^{r-1}, Txs)$
    $sig_v^r = Sign(B_H^r)$
  **end if**
  **while** $!finalized$ **do** ▷ Block Verification and Finalization
    //** Broadcast signatures
    $BroadcastMSG()$
    $B^r, proof, sig_u^r, sig_{full}^r = RcvMSG()$
    //**Check the Finalization of new block
    **if** $isValid(sig_{full}^r)$ **then**
      $AddSig(B^r, sig_{full}^r)$
      $Append(BC, B^r)$
      $finalized = True$
    **else if** $Count(Sigs^r) \geq \lceil \frac{N+1}{2} \rceil$ **then**
      $sig_{full}^r$ = Recover Full Signature$(Sigs^r)$
      $broadcast(sig_{full}^r)$ with probability $p_{max}$ and power $P_{max}$
      $AddSig(B^r, sig_{full}^r)$
      $Append(BC, B^r)$
      $finalized = True$
    **else if** $sig_u^r \notin Signs^r$ **then**
      Append Signature$(Sigs^r, sig_u^r)$
    **else**
      //**Check the validation of new block
      **if** $isValid(B^r, pk_{BP}, proof, Rdm^r)$ **then**
        $sig_v^r$ = Generate Signature$(B_H^r, sk_v)$
      **end if**
    **end if**
  **end while**
  $r = r + 1$
**end while**

**Algorithm 2** BroadcastMSG subroutine

**if** $v$ decided to send a transaction based on $\hat{p}_v$ **then**
  $broadcast(MSGs)$ with power $P_t$
**else**
  **if** channel is idle **then**
    //** Count idle slots within $T_v$
    $e_v = e_v + 1$
  **else**
    Receive a message from others
  **end if**
**end if**
//**maintain the estimate of adversary time window
$count_v = count_v + 1$
**if** $count_v > T_v$ **then**
  $count_v = 1$
  **if** $e_v == 0$ **then**
    $\hat{p}_v = \hat{p}_v/(1+\gamma)$
    $\hat{T}_v = \hat{T}_v + 2$
  **else if** $e_v >= 1$ **then**
    $\hat{p}_v = \hat{p}_v * (1+\gamma)$
    $\hat{T}_v = \hat{T}_v - 1$
  **end if**
**end if**

**Algorithm 3** The SWIB Blockchain Consensus Protocol for each node $v$

**while** true **do**
  //** Iteration for round $r$
  ▷ Initialization:
  **for** $j < K$ slots **do**
    $BroadcastMSG()$
    $j = j + 1$
  **end for**
  $Rds^r = GenerateRandomValue(r, B_H^{r-1}, sig_{full}^{r-1})$
  ▷ Consensus Process:
  Block Proposer Election();
  Block Verification();
  Block Finalization();
  $r = r + 1$
**end while**

Table 1: Summary of Notations

| Symbol | Description |
|---|---|
| $N$ | network size |
| $V$ | set of nodes |
| $V_f$ | set of nodes that fail to transmit a message to a receiver |
| $d_{u,v}$ | Euclidean distance between $Node_u$ and $Node_v$ |
| $H_{u,v}$ | channel gain from $Node_u$ to $Node_v$ |
| $P_t$ | transmission power utilized in broadcast protocol |
| $P_n$ | additive white Gaussian noise power |
| $\tau_o$ | time interval of a single time slot |
| $\hat{p}_v$ | send probability of $Node_v$ |
| $\epsilon_{u,v}$ | communication interruption probability between $Node_u$ and $Node_v$ |
| $p_{comm,v}$ | communication failure probability of $Node_v$ |
| $BC$ | blockchain |
| $B^r$ | block generated in the r-th round |
| $B_H$ | hash value of block $B$ |
| $tx$ | a trasnaction |
| $T$ | time window of adversary |
| $\hat{T}_v$ | estimate of $T$ by $Node_v$ |
| $\delta$ | proportion of non-jammed slots |
| $Rdm^r$ | random value generated in the r-th round |
| $T_v, \rho_v$ | active time and active time ratio of $Node_v$, respectively |
| $N_v, r_v$ | number of blocks generated by $Node_v$ in the latest K blocks and consensus ratio |
| $S_v$ | stability of $Node_v$ |
| $p_v$ | elected probability of $Node_v$ |
| $pk, sk$ | public key and private key, respetively |
| $sig_v^r$ | partial signature of $Node_v$ in the r-th round |
| $sig_{full}^r$ | full signature of $Node_v$ in the r-th round |

**Algorithm 4** Block Verification for each node $v$

$B^r, proof = RcvMSG()$
//**Check the validation of new block
$result_v$ = Verify Block Proposer($pk_{BP}, proof, Rdm^r$)
**if** $result_v == True$ **then**
    **if** $H_{pre}^r == B_H^{r-1}$ **then**
        **if** isvalid(Txs) **then**
            $sig_v^r$ = Generate Signature($B_H^r, sk_v$)
        **end if**
    **end if**
**end if**

**Algorithm 5** Block Finalization for each node $v$

**while** $!finalized$ **do**
  $BroadcastMSG()$
  $sig_u^r, sig_{full}^r = RcvMSG()$
  //**Check the Finalization of new block
  **if** $isValid(sig_{full}^r)$ **then**
    $AddSig(B^r, sig_{full}^r)$
    $Append(BC, B^r)$
    $finalized = True$
  **else if** $Count(Sigs^r) \geq \lceil \frac{N+1}{2} \rceil$ **then**
    $sig_{full}^r =$ Recover Full Signature$(Sigs^r)$
    $broadcast(sig_{full}^r)$ with probability $p_{max}$ and power $P_{max}$
    $AddSig(B^r, sig_{full}^r)$
    $Append(BC, B^r)$
    $finalized = True$
  **else if** $sig_u^r \notin Signs^r$ **then**
    Append Signature$(Sigs^r, sig_u^r)$
  **end if**
**end while**

**Algorithm 6** Stable Wireless Blockchain Protocol

▷ Initialization:
$Sortition(PKs^r, S^r)$
$Rds^r = GenerateRandomness(r, B^{r-1}_{hash}, sig^r_{final})$
▷ Leader Election and Block Proposal:
$result = BlockProposerSelection(sk, Rds^r)$
**if** $result == True$ **then** ▷ As Block Proposer
    $B^r = GenerateBlock(B^{r-1}, Txs)$
    $sig^r_{partial} = Sign(B^r_{hash})$
    $broadcast(B^r, sig^r_{partial})$ with probability $p$
**else** ▷ As Ordinary Nodes
    Waiting to receive new Block
**end if**
▷ Block Verification and Finalization:
**while** $!finalized$ **do** ▷ All Consensus Nodes
    $(B^r, Signs^r, sig^r_{full}, Tx) = RcvMSG()$
    //**Check the validation of new block
    **if** $isValid(B^r)$ and $VerifyBlockProposer(pk_{BP}, Rds^r)$ **then**
        $sig^r_v = GenerateSignature(B^r_{hash}, sk_v)$
    **end if**
    **if** $isValid(sig^r_{full})$ **then**
        $\sigma^r_F = sig^r_{full}$
        $broadcast(\sigma^r_F)$ with probability $p$
        $Append(B^r, \sigma^r_F)$
        $finalized = True$
    **else if** $Count(Signs^r) >= \lceil \frac{N}{2} \rceil$ **then**
        $\sigma^r_F = RecoverFullSignature(Signs^r)$
        $broadcast(\sigma^r_F)$ with probability $p$
        $Append(B^r, \sigma^r_F)$
        $finalized = True$
    **else if** $sig^r_u \notin Signs^r$ **then**
        $Signs^r = AppendSignature(sig^r_u)$
    **else if** $v$ did not broadcast its partial signature **then**
        $broadcast(sig^r_v)$ with probability $p$
    **else**
        $broadcast(Tx)$ with probability $p$
    **end if**
    $count = count + 1$
    **if** $count > T$ **then**
        $count = 1$
        **if** Received $T$ consecutive transactions in the past $T$ rounds **then**
            $p = p * (1 + \delta)^{-1}$
            $T = T + 2$
        **end if**
    **end if**
**end while**

46: **function** RecNewBlock($m_B, \sigma_v$)
47: **if** $\sigma_v \notin sigShares$ **then**
48: $sigShares = AppendSignature(\sigma_v)$
49: **end if**
50: **if** $Count(sigShares) > K$ **then**
51: $FinalSig = RecoverFinalSig(sigShares)$
52: **else**
53: $FinalSig = null$
54: **end if**
55: **return** $sigShares, FinalSig, B_v^{new}$
56: **end function**
57: **function** AppendSignature($\sigma_v$)
58: **if** $\sigma_v \notin sigShares$ **then**
59: $sigShares \leftarrow sigShares + \sigma_v)$
60: **end if**
61: **return** $sigShares$
62: **end function**