MODULE *AVC_RVS*

EXTENDS *Naturals*, *Integers*, *FiniteSets*,
RVS_CryptoAbstraction, *Reputation_Game*

Clamp(*v*, *maxV*) $\triangleq$
IF $v < 0$ THEN 0 ELSE IF $v > maxV$ THEN *maxV* ELSE *v*

SumRep(*reputation*, *nodes*) $\triangleq$
LET *bound* $\triangleq$ *MaxNat*($\{reputation[n] : n \in nodes\}$)
IN
Cardinality(
$\{p \in (nodes \times (1 \,..\, bound)) : p[2] \leq reputation[p[1]]\}$
)

AverageRep(*reputation*) $\triangleq$
LET *nodes* $\triangleq$ DOMAIN *reputation*
card $\triangleq$ *Cardinality*(*nodes*)
IN IF $card = 0$ THEN 0 ELSE *SumRep*(*reputation*, *nodes*) $\div$ *card*

CanonicalReporters(*reputation*) $\triangleq$ DOMAIN *reputation*

CanonicalReports(*reputation*) $\triangleq$
CanonicalSelfReports(*CanonicalReporters*(*reputation*), *reputation*)

RecomputeReputation(*reputation*, *maxRep*, *decayNumerator*, *decayDenominator*) $\triangleq$
LET *nodes* $\triangleq$ DOMAIN *reputation*
IN
ReputationFromReports(
CanonicalReports(*reputation*),
reputation,
nodes,
nodes,
decayNumerator,
decayDenominator,
maxRep
)

CandidateReplicas(*reputation*, *threshold*) $\triangleq$
LET *allNodes* $\triangleq$ DOMAIN *reputation*
base $\triangleq$ $\{n \in allNodes : reputation[n] \geq threshold\}$
avg $\triangleq$ *AverageRep*(*reputation*)
weighted $\triangleq$ $\{n \in base : reputation[n] \geq avg\}$
IN
IF $weighted \neq \{\}$
THEN *weighted*
ELSE IF $base \neq \{\}$ THEN *base* ELSE *allNodes*

```
RVSKappa(candidates, threshold) ≜
    LET    card ≜ Cardinality(candidates)
           base ≜ IF threshold < 1 THEN 1 ELSE threshold
    IN
        IF card = 0
         THEN 1
         ELSE IF base ≤ card THEN base ELSE card

RVSContext(reputation, threshold) ≜
    LET cand ≜ CandidateReplicas(reputation, threshold)
        maxRep ≜ MaxNat({reputation[n] : n ∈ DOMAIN reputation})
    IN
        [
            candidates ↦ cand,
            maxRep ↦ maxRep,
            totalWeight ↦ SumRep(reputation, cand),
            kappa ↦ RVSKappa(cand, threshold)
        ]

RVSSelectPrimary(reputation, threshold, view) ≜
    LET ctx ≜ RVSContext(reputation, threshold)
        winners ≜
            ValidWinners(
                reputation,
                ctx.candidates,
                view,
                ctx.totalWeight,
                ctx.kappa,
                ctx.maxRep
            )
    IN
        PickWinner(
            reputation,
            winners,
            view,
            ctx.totalWeight,
            ctx.kappa,
            ctx.maxRep
        )

RVSPrimaryEvidence(reputation, threshold, view, leader) ≜
    LET ctx ≜ RVSContext(reputation, threshold)
        result ≜
            SortitionResult(
                reputation,
                leader,
```

```
            view,
            ctx.totalWeight,
            ctx.kappa,
            ctx.maxRep
        )
IN
    [ticket ↦ result.ticket, strikes ↦ result.strikes, proof ↦ result.proof]

RVSVerifyPrimary(reputation, threshold, view, leader, ticket, strikes, proof) ≜
LET ctx ≜ RVSContext(reputation, threshold)
    draws ≜ ctx.kappa
    result ≜
        SortitionResult(
            reputation,
            leader,
            view,
            ctx.totalWeight,
            draws,
            ctx.maxRep
        )
    winners ≜
        ValidWinners(
            reputation,
            ctx.candidates,
            view,
            ctx.totalWeight,
            draws,
            ctx.maxRep
        )
    verified ≜
        SortitionVerify(
            reputation,
            leader,
            view,
            ctx.totalWeight,
            draws,
            ctx.maxRep,
            result
        )
IN
    ∧ leader ∈ ctx.candidates
    ∧ leader ∈ winners
    ∧ result.ticket = ticket
    ∧ result.strikes = strikes
    ∧ result.proof = proof
```

```
        ∧ verified

TemporalDecay(previous, observed, decayNumerator, decayDenominator) ≜
    ((decayNumerator * previous) + ((decayDenominator − decayNumerator) * observed)) ÷ decayDenomin

DecayUpdateByObservation(reputation, node, observed, maxRep, decayNumerator, decayDenominator) ≜
    LET boundedObserved ≜ Clamp(observed, maxRep)
        blended ≜ TemporalDecay(reputation[node], boundedObserved, decayNumerator, decayDenomina
        base ≜ [reputation EXCEPT ![node] = Clamp(blended, maxRep)]
        recomputed ≜
            RecomputeReputation(
                base,
                maxRep,
                decayNumerator,
                decayDenominator
            )
    IN
        [n ∈ DOMAIN recomputed ↦ Clamp(recomputed[n], maxRep)]

DecayUpdate(reputation, node, isHonest, maxRep, decayNumerator, decayDenominator) ≜
    LET observed ≜ IF isHonest THEN maxRep ELSE 0
    IN  DecayUpdateByObservation(reputation, node, observed, maxRep, decayNumerator, decayDenomin
```