EXTENDS $FiniteSets$, $Integers$, $Utils$

CONSTANTS
  P the set of processes
, B the set of malicious processes
, $tAdv$ the time it takes for a malicious process to produce a message
, tWB the time it takes for a well-behaved process to produce a message

ASSUME $B \subseteq P$ malicious processes are a subset of all processes
$W \triangleq P \setminus B$ the set of well-behaved processes

$Tick \triangleq Nat$ a tick is a real-time clock tick
$Round \triangleq Nat$ a round is just a tag on a message

Processes build a DAG of messages. The message-production rate of well-behaved processes is of 1 message per tWB ticks, and that of malicious processes is of 1 message per $tAdv$ ticks. We require that, collectively, well-behaved processes produce messages at a rate strictly higher than that of malicious processes.

ASSUME $Cardinality(W) * tAdv > Cardinality(B) * tWB$

 A message consists of a unique ID, a round number, and a coffer containing the IDs of a set of predecessor messages:
$MessageID \triangleq P \times Nat$
$Message \triangleq [id : MessageID, round : Round, coffer : \text{SUBSET } MessageID]$
$sender(m) \triangleq m.id[1]$

A strongly consistent chain is a subset of the messages in the DAG that potentially has some dangling pointers (*i.e.* messages that have predecessors not in the chain) and that satisfies the following recursive predicate:

 * Any set of messages which all have a round of 0 is a strongly consistent chain.

 * A set of messages C with some non-zero rounds and maximal round r is a strongly consistent chain when, with Tip being the set of messages in the chain that have round r and $Pred$ being the set of messages in the chain with round $r - 1$, $Pred$ is a strict majority of the set of predecessors of each message in Tip and $C \setminus Tip$ is a consistent chain.

 The max round of a set of messages is the maximal round of its messages:
$MaxRound(M) \triangleq MaxInteger(\{m.round : m \in M\})$

$StronglyConsistentChain(M) \triangleq$
  $\wedge\ M \neq \{\}$
  $\wedge\ \vee\ MaxRound(M) = 0$
    $\vee\ \forall\, r \in 1 \ .. \ MaxRound(M) :$
     LET $Tip \triangleq \{m \in M : m.round = r\}$
       $Pred \triangleq \{m \in M : m.round = r - 1\}$
     IN $\wedge\ Tip \neq \{\}$
        $\wedge\ \forall\, m \in Tip :$
         $\wedge\ \{m2.id : m2 \in Pred\} \subseteq m.coffer$

$$\wedge \; 2 * Cardinality(Pred) > Cardinality(m.coffer)$$

A weaker version of the above:

$ConsistentChain(M) \triangleq$
  $\wedge \; M \neq \{\}$
  $\wedge \; \vee \; MaxRound(M) = 0$
   $\vee \; \forall\, r \in 1\,..\,MaxRound(M) :$
    LET $Tip \triangleq \{m \in M : m.round = r\}$
     $Pred \triangleq \{m \in M : m.round = r - 1\}$
   IN $\wedge \; Tip \neq \{\}$
    $\wedge \; \forall\, m \in Tip : \exists\, Maj \in$ SUBSET $Pred :$
     $\wedge \{m2.id : m2 \in Maj\} \subseteq m.coffer$
     $\wedge \; 2 * Cardinality(Maj) > Cardinality(m.coffer)$

$SubsetsWithMaxRound(M,\, r) \triangleq \{M2 \in$ SUBSET $M : \exists\, m \in M2 : m.round = r\}$

The set of all consistent chains that can be found in M:

$ConsistentChains(M,\, r) \triangleq$
  $\{C \in SubsetsWithMaxRound(M,\, r) : ConsistentChain(C)\}$

The set of all strongly consistent chains that can be found in M:

$StronglyConsistentChains(M,\, r) \triangleq$
  $\{C \in SubsetsWithMaxRound(M,\, r) : StronglyConsistentChain(C)\}$

We can rank the chains by wheight, *i.e.* just their cardinality, or we can consider the maximal or minimal one in the subset order:

$HeaviestConsistentChains(M,\, r) \triangleq MaxCardinalitySets(ConsistentChains(M,\, r))$
$HeaviestStronglyConsistentChains(M,\, r) \triangleq MaxCardinalitySets(StronglyConsistentChains(M,\, r))$
$MinimalStronglyConsistentChains(M,\, r) \triangleq MinimalSets(StronglyConsistentChains(M,\, r))$
$MaximalStronglyConsistentChains(M,\, r) \triangleq MaximalSets(StronglyConsistentChains(M,\, r))$

Two chains are disjoint when there is a round smaller than their max round in which they have no messages in common.

$DisjointChains(C1,\, C2) \triangleq$
  LET $rmax \triangleq MaxRound(C1 \cup C2)$
  IN $\exists\, r \in 0\,..\,(rmax - 1) :$
    $\{m \in C1 : m.round = r\} \cap \{m \in C2 : m.round = r\} = \{\}$

Acceptance rule

$AcceptedMessages(M,\, r) \triangleq \{m \in M :$
  $\wedge \; m.round = r - 1$
  $\wedge$ LET $CCs \triangleq MinimalStronglyConsistentChains(M,\, r - 1)$IN   This looks promising!
   $\wedge \; \exists\, C \in CCs : m \in C$
   $\wedge \; \forall\, C1,\, C2 \in CCs :$
    $\wedge \; m \in C1$
    $\wedge \; m \notin C2$

$$\land\ DisjointChains(C1,\ C2)$$
$$\Rightarrow Cardinality(C2) \le Cardinality(C1)\}$$

M does not have dangling edges:
$$Closed(M)\ \triangleq\ \forall\, m \in M : \forall\, i \in m.coffer : \exists\, m2 \in M : m2.id = i$$

Now we specify the algorithm

--algorithm $Algo\{$
  variables
    $messages = \{\}\,;$
    $tick = 0\,;$
    $phase =$ "start"$\,;$ each tick has two phases: "start" and "end"
    $donePhase = [p \in P \mapsto$ "end"$]\,;$
    $pendingMessage = [p \in P \mapsto \langle\rangle]\,;$ message we're computing the *VDF* on
    $messageCount = [p \in P \mapsto 0]\,;$ used to generate unique message *IDs*
  define {
    $currentRound\ \triangleq\ tick \div tWB$ round of well-behaved processes
    $wellBehavedMessages\ \triangleq\ \{m \in messages : sender(m) \in P \setminus B\}$
     possible sets of messages received by a well-behaved process:
    $receivedMsgsSets\ \triangleq$
      ignore messages from future rounds:
      LET $msgs\ \triangleq\ \{m \in messages : m.round < currentRound\}$IN
      $\{M \in$ SUBSET $msgs :$
        don't use a set of messages that has dangling edges (messages in coffers that are missing):
        $\land\ Closed(M)$
        $\land\ wellBehavedMessages \subseteq\ M\}$
  }
  macro $sendMessage(\ m\ )$ {
    $messages := messages \cup \{m\}$
  }
  process ($clock \in \{$"clock"$\}$) {
$tick:$ **while** (TRUE) {
    await $\forall\, p \in P : donePhase[p] = phase\,;$
    if ($phase =$ "start")
      $phase :=$ "end"
    else {
      $phase :=$ "start"$\,;$
      $tick := tick + 1$
    }
  }
  }
  process ($proc \in P \setminus B$) a well-behaved process
  {
$l1:$ **while** (TRUE) {

3

```
            await phase = "start" ;
            if ( tick%tWB = 0 ) {
                  Start the VDF computation for the next message:
                  with ( msgs ∈ receivedMsgsSets )
                  with ( predMsgs = AcceptedMessages(msgs, currentRound) ) {
                      pendingMessage[self] := [
                          id ↦ ⟨self, messageCount[self] + 1⟩,
                          round ↦ currentRound,
                          coffer ↦ {m.id : m ∈ predMsgs}] ;
                      messageCount[self] := messageCount[self] + 1
                  }
            } ;
            donePhase[self] := "start" ;
l2:         await phase = "end" ;
            if ( tick%tWB = tWB − 1 )
                  it's the end of the tWB period, the VDF has been computed
                  sendMessage(pendingMessage[self]) ;
            donePhase[self] := "end" ;
        }
    }
    process ( byz ∈ B )   a malicious process
    {
lb1:    while ( TRUE ) {
            await phase = "start" ;
            if ( tick%tAdv = 0 ) {
                  Start the VDF computation for the next message:
                  with ( maxRound = Max({m.round : m ∈ messages} ∪ {0}, ≤ ) )
                  with ( rnd ∈ {maxRound, maxRound + 1} )
                  with ( predMsgs ∈ SUBSET {m ∈ messages : m.round = rnd − 1} ) {
                      when rnd > 0 ⇒ predMsgs ≠ {} ;
                      pendingMessage[self] := [
                          id ↦ ⟨self, messageCount[self] + 1⟩,
                          round ↦ rnd,
                          coffer ↦ {m.id : m ∈ predMsgs}] ;
                      messageCount[self] := messageCount[self] + 1
                  }
            } ;
            donePhase[self] := "start" ;
lb2:        await phase = "end" ;
            if ( tick%tAdv = tAdv − 1 )
                  sendMessage(pendingMessage[self]) ;
            donePhase[self] := "end" ;
        } ;
    }
}
```

Invariant describing the type of the variables:

$TypeOK \triangleq$
- $\wedge\ messages \in \text{SUBSET } Message$
- $\wedge\ pendingMessage \in [P \rightarrow Message \cup \{\langle\rangle\}]$
- $\wedge\ tick \in Tick$
- $\wedge\ phase \in \{\text{"start"}, \text{"end"}\}$
- $\wedge\ donePhase \in [P \rightarrow \{\text{"start"}, \text{"end"}\}]$
- $\wedge\ messageCount \in [P \rightarrow Nat]$

The main property we want to establish is that, each round, for each message m of a well-behaved process, the messages of well-behaved processes from the previous round are all in m's coffer and consist of a strict majority of m's coffer.

$Safety \triangleq \forall\, p \in P \setminus B : \text{LET } m \triangleq pendingMessage[p] \text{ IN}$
- $\wedge\ m \neq \langle\rangle$
- $\wedge\ m.round > 0$
- $\Rightarrow$
- $\wedge\ \forall\, m2 \in wellBehavedMessages : m2.round = m.round - 1 \Rightarrow m2.id \in m.coffer$
- $\wedge\ \text{LET } M \triangleq \{m2 \in wellBehavedMessages : m2.round = m.round - 1\}$
  - $\text{IN}\quad 2 * Cardinality(M) > Cardinality(m.coffer)$

helper definition:

$messageWithID(id) \triangleq \text{CHOOSE } m \in messages : m.id = id$

Basic well-formedness properties:

$Inv1 \triangleq \forall\, m \in messages :$
- $\wedge\ \forall\, m2 \in messages : m \neq m2 \Rightarrow m.id \neq m2.id$
- $\wedge\ \forall\, id\ \in m.coffer\ :$
  - $\wedge\ \exists\, m2 \in messages : m2.id = id$
  - $\wedge\ messageWithID(id).round = m.round - 1$