MODULE *VDFConsensus*

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:

$MaxRound(M) \triangleq MaxInteger(\{m.round : m \in M\})$

$$
\begin{array}{l}
StronglyConsistentChain(M) \triangleq \\
\quad \land\ M \neq \{\} \\
\quad \land\ \lor\ MaxRound(M) = 0 \\
\quad \ \lor\ \forall r \in 1 \,..\, MaxRound(M) : \\
\qquad\qquad \text{LET } Tip \triangleq \{m \in M : m.round = r\} \\
\qquad\qquad\qquad Pred \triangleq \{m \in M : m.round = r - 1\} \\
\qquad\qquad \text{IN} \quad \land\ Tip \neq \{\} \\
\qquad\qquad\qquad \land\ \forall m \in Tip : \\
\qquad\qquad\qquad\quad \land\ \{m2.id : m2 \in Pred\} \subseteq m.coffer
\end{array}
$$

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

A weaker version of the above:

$$
\begin{array}{l}
ConsistentChain(M) \triangleq \\
\quad \land\ M \neq \{\} \\
\quad \land\ \lor\ MaxRound(M) = 0 \\
\qquad \lor\ \forall r \in 1 \,..\, MaxRound(M) : \\
\qquad\quad \text{LET } Tip \triangleq \{m \in M : m.round = r\} \\
\qquad\qquad\quad Pred \triangleq \{m \in M : m.round = r - 1\} \\
\qquad\quad \text{IN} \quad \land\ Tip \neq \{\} \\
\qquad\qquad\quad \land\ \forall m \in Tip : \exists Maj \in \text{SUBSET } Pred : \\
\qquad\qquad\qquad \land \{m2.id : m2 \in Maj\} \subseteq m.coffer \\
\qquad\qquad\qquad \land\ 2 * Cardinality(Maj) > Cardinality(m.coffer)
\end{array}
$$

$$
\begin{array}{l}
Chains(M, r) \triangleq \{C \in \text{SUBSET } M : \\
\quad \land\ \forall m \in C : m.round \leq r \\
\quad \land\ \forall r2 \in 0 \,..\, r : \exists m \in C : m.round = r2\}
\end{array}
$$

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

$$
\begin{array}{l}
ConsistentChains(M, r) \triangleq \\
\quad \{C \in Chains(M, r) : ConsistentChain(C)\}
\end{array}
$$

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

$$
\begin{array}{l}
StronglyConsistentChains(M, r) \triangleq \\
\quad \{C \in Chains(M, r) : StronglyConsistentChain(C)\}
\end{array}
$$

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:

$$
\begin{array}{l}
HeaviestConsistentChains(M, r) \triangleq MaxCardinalitySets(ConsistentChains(M, r)) \\
HeaviestStronglyConsistentChains(M, r) \triangleq MaxCardinalitySets(StronglyConsistentChains(M, r)) \\
MinimalStronglyConsistentChains(M, r) \triangleq MinimalSets(StronglyConsistentChains(M, r)) \\
MaximalStronglyConsistentChains(M, r) \triangleq MaximalSets(StronglyConsistentChains(M, r))
\end{array}
$$

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

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

Acceptance rule

$$
\begin{array}{l}
AcceptedMessages(M, r) \triangleq \{m \in M : \\
\quad \land\ m.round = r - 1 \\
\quad \land\ \text{LET } CCs \triangleq MaximalStronglyConsistentChains(M, r - 1) \text{IN} \qquad \text{This looks promising!} \\
\qquad \land\quad \exists C \in CCs : m \in C \\
\qquad \land\quad \forall C1, C2 \in CCs :
\end{array}
$$

$\land\ m \in C1$
$\land\ m \notin C2$
$\land\ \mathit{DisjointChains}(C1,\ C2)$
$\Rightarrow \mathit{Cardinality}(C2) \leq \mathit{Cardinality}(C1)\}$

M does not have dangling edges:

$\mathit{Closed}(M) \triangleq \forall\, m \in M : \forall\, i \in m.\mathit{coffer} : \exists\, m2 \in M : m2.\mathit{id} = i$

Now we specify the algorithm

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

```
    {
l1:   while ( TRUE ) {
        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"   ;
        when currentRound < 2 ;   TODO temporary hack
        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:

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

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.

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

helper definition:

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

Basic well-formedness properties:

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