MODULE *AccessControlManagement*

EXTENDS *Naturals*, *Sequences*

CONSTANTS *ProcessCount*, *ResourceCount*

$Processes \triangleq 1 \,..\, ProcessCount$
$Resources \triangleq 1 \,..\, ResourceCount$

$NULL \triangleq$ "NULL"
$ALLOWED \triangleq$ "ALLOWED"
$REJECTED \triangleq$ "REJECTED"
$REQUESTED \triangleq$ "REQUESTED"
$IN\_USE \triangleq$ "IN_USE"

$URI\_PERMISSION \triangleq$ "URI_PERMISSION"
$CUSTOM\_PERMISSION \triangleq$ "CUSTOM_PERMISSION"

$NORMAL \triangleq$ "NORMAL"
$SIGNATURE \triangleq$ "SIGNATURE"
$DANGEROUS \triangleq$ "DANGEROUS"

$Boolean \triangleq \{\text{TRUE}, \text{FALSE}\}$
$ResourceStatus \triangleq \{REQUESTED, ALLOWED, REJECTED, IN\_USE\}$
$PermissionTypes \triangleq \{URI\_PERMISSION, CUSTOM\_PERMISSION\}$
$PermissionLevels \triangleq \{NORMAL, SIGNATURE, DANGEROUS\}$

```
** --algorithm AccessControlManagement
{
    variables Acl_Status = [a ∈ Processes ↦ [r ∈ Resources ↦ NULL]] ;
              Acl_PermissionType = [a ∈ Processes ↦ [r ∈ Resources ↦ NULL]] ;
              Acl_PermissionLevel = [a ∈ Processes ↦ [r ∈ Resources ↦ NULL]] ;
              Consent = [a ∈ Processes ↦ [r ∈ Resources ↦ FALSE]] ;
              Grid = [a ∈ Processes ↦ [a2 ∈ Processes ↦ FALSE]] ;

    macro Define( p, r )
    {
     with ( t ∈ PermissionTypes ) { Acl_PermissionType[p][r] := t ; } ;
     Acl_PermissionLevel[p][r] := NORMAL ;
     }

    macro Request( p, r ) { Acl_Status[p][r] := REQUESTED ; }

    macro Decide( p, r )
    {
      with ( b ∈ Boolean )
      {
       if ( b = TRUE )
       {
```

```
      Acl_Status[p][r] := ALLOWED ;
      Consent[p][r] := TRUE ;
      }
     else
     {
      Acl_Status[p][r] := REJECTED ;
      Consent[p][r] := FALSE ;
      }
     }
 }

macro Revoke( p, r )
{
  Acl_Status[p][r] := NULL ;
  Consent[p][r] := FALSE ;
 }

macro Use( p, r )
{
  Acl_Status[p][r] := IN_USE ;
 }

macro Connect( a1, a2 )
{
 Grid[a1][a2] := TRUE ;
 }

procedure Update( p2 )
variable ResourceList = Resources ;
{
 UPDATE:
 while ( ResourceList ≠ {} )
 {
  with ( r ∈ ResourceList )
  {
   if ( Acl_PermissionLevel[p2][r] = NORMAL )
   {
    Acl_PermissionLevel[p2][r] := DANGEROUS ;
    Consent[p2][r] := FALSE ;
    } ;

   ResourceList := ResourceList \ {r} ;
   }
  } ;
 return ;
 }
```

```
procedure Delegate( app1, app2 )
variable ResourceList = Resources ;
{
 DELEGATE:
 while ( ResourceList ≠ {} )
 {
  with ( r ∈ ResourceList )
  {
   if ( Acl_PermissionType[app2][r] ≠ NULL ∧ Acl_PermissionType[app1][r] = NULL )
    Acl_PermissionType[app1][r] := Acl_PermissionType[app2][r] ;

   if ( Acl_Status[app2][r] ≠ NULL ∧ Acl_Status[app1][r] = NULL )
    Acl_Status[app1][r] := Acl_Status[app2][r] ;

   ResourceList := ResourceList \ {r} ;
  }
 } ;
 return ;
 }

fair process ( AcmNext ∈ Processes )
variable Resource
{
    s0: Grid[self][self] := TRUE ;

    s1: while (TRUE)
    {
     s2: with (R ∈ Resources){Resource := R ; } ;

     s3: with (B ∈ Boolean){if (B = TRUE)call Update(self) ; } ;

     s4: if (Acl_PermissionType[self][Resource] = NULL){Define(self , Resource)}

         else if (Acl_Status[self][Resource] = NULL){Request(self , Resource) ; }

         else if (Acl_Status[self][Resource] = REQUESTED){Decide(self , Resource) ; }

         else if (Acl_Status[self][Resource] = ALLOWED)
         {
          either {Revoke(self , Resource) ; }
          or {Use(self , Resource) ; }
         } ;

     s5: with (rand ∈ Boolean)
     {
      if (rand = TRUE)
      {
       with (a ∈ (Processes \ {self}))
```

```
            {
             Connect(self, a);
             call Delegate(self, a);
            }
           }
          }
         }
        }
    }
}
```

**

BEGIN TRANSLATION ($chksum(pcal) =$ "68482efa" $\land\ chksum(tla) =$ "b9184269")

Procedure variable *ResourceList* of procedure *Update* at line 77 col 14 changed to *ResourceList_*

CONSTANT *defaultInitValue*

VARIABLES *Acl_Status*, *Acl_PermissionType*, *Acl_PermissionLevel*, *Consent*, *Grid*, *pc*, *stack*, *p2*, *ResourceList_*, *app1*, *app2*, *ResourceList*, *Resource*

$vars \triangleq \langle Acl\_Status,\ Acl\_PermissionType,\ Acl\_PermissionLevel,\ Consent,\ Grid,\ pc,\ stack,\ p2,\ ResourceList\_,\ app1,\ app2,\ ResourceList,\ Resource \rangle$

$ProcSet \triangleq (Processes)$

$Init \triangleq$ Global variables
$\land\ Acl\_Status = [a \in Processes \mapsto [r \in Resources \mapsto NULL]]$
$\land\ Acl\_PermissionType = [a \in Processes \mapsto [r \in Resources \mapsto NULL]]$
$\land\ Acl\_PermissionLevel = [a \in Processes \mapsto [r \in Resources \mapsto NULL]]$
$\land\ Consent = [a \in Processes \mapsto [r \in Resources \mapsto \text{FALSE}]]$
$\land\ Grid = [a \in Processes \mapsto [a2 \in Processes \mapsto \text{FALSE}]]$
Procedure *Update*
$\land\ p2 = [self \in ProcSet \mapsto defaultInitValue]$
$\land\ ResourceList\_ = [self \in ProcSet \mapsto Resources]$
Procedure *Delegate*
$\land\ app1 = [self \in ProcSet \mapsto defaultInitValue]$
$\land\ app2 = [self \in ProcSet \mapsto defaultInitValue]$
$\land\ ResourceList = [self \in ProcSet \mapsto Resources]$
Process *AcmNext*
$\land\ Resource = [self \in Processes \mapsto defaultInitValue]$
$\land\ stack = [self \in ProcSet \mapsto \langle\rangle]$
$\land\ pc = [self \in ProcSet \mapsto$ "s0"$]$

$UPDATE(self) \triangleq \land\ pc[self] =$ "UPDATE"
$\land$ IF $ResourceList\_[self] \neq \{\}$
THEN $\land \exists\, r \in ResourceList\_[self] :$
$\land$ IF $Acl\_PermissionLevel[p2[self]][r] = NORMAL$
THEN $\land\ Acl\_PermissionLevel' = [Acl\_PermissionLevel$ EXCEPT $![p$
$\land\ Consent' = [Consent$ EXCEPT $![p2[self]][r] =$ FALSE$]$

```
                                ELSE ∧ TRUE
                                     ∧ UNCHANGED ⟨Acl_PermissionLevel,
                                                  Consent⟩
                             ∧ ResourceList_′ = [ResourceList_ EXCEPT ![self] = ResourceList_[self] \
                        ∧ pc′ = [pc EXCEPT ![self] = "UPDATE"]
                        ∧ UNCHANGED ⟨stack, p2⟩
                   ELSE ∧ pc′ = [pc EXCEPT ![self] = Head(stack[self]).pc]
                        ∧ ResourceList_′ = [ResourceList_ EXCEPT ![self] = Head(stack[self]).Resou
                        ∧ p2′ = [p2 EXCEPT ![self] = Head(stack[self]).p2]
                        ∧ stack′ = [stack EXCEPT ![self] = Tail(stack[self])]
                        ∧ UNCHANGED ⟨Acl_PermissionLevel, Consent⟩
              ∧ UNCHANGED ⟨Acl_Status, Acl_PermissionType, Grid, app1,
                           app2, ResourceList, Resource⟩

Update(self) ≜ UPDATE(self)

DELEGATE(self) ≜ ∧ pc[self] = "DELEGATE"
                 ∧ IF ResourceList[self] ≠ {}
                      THEN ∧ ∃ r ∈ ResourceList[self] :
                                ∧ IF Acl_PermissionType[app2[self]][r] ≠ NULL ∧ Acl_PermissionTyp
                                     THEN ∧ Acl_PermissionType′ = [Acl_PermissionType EXCEPT
                                     ELSE ∧ TRUE
                                          ∧ UNCHANGED Acl_PermissionType
                                ∧ IF Acl_Status[app2[self]][r] ≠ NULL ∧ Acl_Status[app1[self]][r] = N
                                     THEN ∧ Acl_Status′ = [Acl_Status EXCEPT ![app1[self]][r] = A
                                     ELSE ∧ TRUE
                                          ∧ UNCHANGED Acl_Status
                                ∧ ResourceList′ = [ResourceList EXCEPT ![self] = ResourceList[self] \
                           ∧ pc′ = [pc EXCEPT ![self] = "DELEGATE"]
                           ∧ UNCHANGED ⟨stack, app1, app2⟩
                      ELSE ∧ pc′ = [pc EXCEPT ![self] = Head(stack[self]).pc]
                           ∧ ResourceList′ = [ResourceList EXCEPT ![self] = Head(stack[self]).Resou
                           ∧ app1′ = [app1 EXCEPT ![self] = Head(stack[self]).app1]
                           ∧ app2′ = [app2 EXCEPT ![self] = Head(stack[self]).app2]
                           ∧ stack′ = [stack EXCEPT ![self] = Tail(stack[self])]
                           ∧ UNCHANGED ⟨Acl_Status, Acl_PermissionType⟩
                 ∧ UNCHANGED ⟨Acl_PermissionLevel, Consent, Grid, p2,
                              ResourceList_, Resource⟩

Delegate(self) ≜ DELEGATE(self)

s0(self) ≜ ∧ pc[self] = "s0"
           ∧ Grid′ = [Grid EXCEPT ![self][self] = TRUE]
           ∧ pc′ = [pc EXCEPT ![self] = "s1"]
           ∧ UNCHANGED ⟨Acl_Status, Acl_PermissionType,
                        Acl_PermissionLevel, Consent, stack, p2,
```

$ResourceList\_,\ app1,\ app2,\ ResourceList,\ Resource\rangle$

$$
\begin{array}{l}
s1(self) \triangleq \ \land\ pc[self] = \text{"s1"} \\
\qquad\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{"s2"}] \\
\qquad\land \text{UNCHANGED } \langle Acl\_Status,\ Acl\_PermissionType, \\
\qquad\qquad Acl\_PermissionLevel,\ Consent,\ Grid,\ stack,\ p2, \\
\qquad\qquad ResourceList\_,\ app1,\ app2,\ ResourceList,\ Resource\rangle
\end{array}
$$

$$
\begin{array}{l}
s2(self) \triangleq \ \land\ pc[self] = \text{"s2"} \\
\qquad\land\ \exists\, R \in Resources : \\
\qquad\quad Resource' = [Resource \text{ EXCEPT } ![self] = R] \\
\qquad\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{"s3"}] \\
\qquad\land \text{UNCHANGED } \langle Acl\_Status,\ Acl\_PermissionType, \\
\qquad\qquad Acl\_PermissionLevel,\ Consent,\ Grid,\ stack,\ p2, \\
\qquad\qquad ResourceList\_,\ app1,\ app2,\ ResourceList\rangle
\end{array}
$$

$$
\begin{array}{l}
s3(self) \triangleq \ \land\ pc[self] = \text{"s3"} \\
\qquad\land\ \exists\, B \in Boolean : \\
\qquad\quad \text{IF } B = \text{TRUE} \\
\qquad\qquad \text{THEN } \land\ \land\ p2' = [p2 \text{ EXCEPT } ![self] = self] \\
\qquad\qquad\qquad\qquad \land\ stack' = [stack \text{ EXCEPT } ![self] = \langle [procedure \mapsto \text{"Update"}, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad pc \mapsto \text{"s4"}, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad ResourceList\_ \mapsto ResourceList\_[self], \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad p2 \mapsto p2[self]]\rangle \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \circ\ stack[self]] \\
\qquad\qquad\qquad\quad \land\ ResourceList\_' = [ResourceList\_ \text{ EXCEPT } ![self] = Resources] \\
\qquad\qquad\qquad\quad \land\ pc' = [pc \text{ EXCEPT } ![self] = \text{"UPDATE"}] \\
\qquad\qquad \text{ELSE } \land\ pc' = [pc \text{ EXCEPT } ![self] = \text{"s4"}] \\
\qquad\qquad\qquad\quad \land \text{UNCHANGED } \langle stack,\ p2,\ ResourceList\_\rangle \\
\qquad\land \text{UNCHANGED } \langle Acl\_Status,\ Acl\_PermissionType, \\
\qquad\qquad Acl\_PermissionLevel,\ Consent,\ Grid,\ app1,\ app2, \\
\qquad\qquad ResourceList,\ Resource\rangle
\end{array}
$$

$$
\begin{array}{l}
s4(self) \triangleq \ \land\ pc[self] = \text{"s4"} \\
\qquad\land \text{IF } Acl\_PermissionType[self][Resource[self]] = NULL \\
\qquad\qquad \text{THEN } \land\ \exists\, t \in PermissionTypes : \\
\qquad\qquad\qquad\quad Acl\_PermissionType' = [Acl\_PermissionType \text{ EXCEPT } ![self][Resource[self]] = t] \\
\qquad\qquad\qquad \land\ Acl\_PermissionLevel' = [Acl\_PermissionLevel \text{ EXCEPT } ![self][Resource[self]] = N \\
\qquad\qquad\qquad \land \text{UNCHANGED } \langle Acl\_Status,\ Consent\rangle \\
\qquad\qquad \text{ELSE } \land \text{IF } Acl\_Status[self][Resource[self]] = NULL \\
\qquad\qquad\qquad\qquad \text{THEN } \land\ Acl\_Status' = [Acl\_Status \text{ EXCEPT } ![self][Resource[self]] = REQUES \\
\qquad\qquad\qquad\qquad\qquad\quad \land \text{UNCHANGED } Consent \\
\qquad\qquad\qquad\qquad \text{ELSE } \land \text{IF } Acl\_Status[self][Resource[self]] = REQUESTED \\
\qquad\qquad\qquad\qquad\qquad\qquad \text{THEN } \land\ \exists\, b \in Boolean : \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{IF } b = \text{TRUE} \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{THEN } \land\ Acl\_Status' = [Acl\_Status \text{ EXCEPT } ![self][R
\end{array}
$$

$\land\ Consent' = [Consent \ \text{EXCEPT}\ ![self][Resou$

$\text{ELSE}\ \ \land\ Acl\_Status' = [Acl\_Status \ \text{EXCEPT}\ ![self][R$

$\land\ Consent' = [Consent \ \text{EXCEPT}\ ![self][Resou$

$\text{ELSE}\ \ \land\ \text{IF}\ Acl\_Status[self][Resource[self]] = ALLOWED$

$\text{THEN}\ \ \land\ \lor\ \land\ Acl\_Status' = [Acl\_Status \ \text{EXCEPT}\ ![sel$

$\land\ Consent' = [Consent \ \text{EXCEPT}\ ![self][Re$

$\lor\ \land\ Acl\_Status' = [Acl\_Status \ \text{EXCEPT}\ ![sel$

$\land\ \text{UNCHANGED}\ Consent$

$\text{ELSE}\ \ \land\ \text{TRUE}$

$\land\ \text{UNCHANGED}\ \langle Acl\_Status, Consent\rangle$

$\land\ \text{UNCHANGED}\ \langle Acl\_PermissionType, Acl\_PermissionLevel\rangle$

$\land\ pc' = [pc \ \text{EXCEPT}\ ![self] = \text{“s5”}]$

$\land\ \text{UNCHANGED}\ \langle Grid, stack, p2, ResourceList\_, app1, app2, ResourceList, Resource\rangle$

$$
\begin{aligned}
s5(self) \triangleq\ & \land\ pc[self] = \text{“s5”} \\
& \land\ \exists\, rand \in Boolean : \\
& \quad \text{IF}\ rand = \text{TRUE} \\
& \quad\quad \text{THEN}\ \land\ \exists\, a \in (Processes \setminus \{self\}) : \\
& \quad\quad\quad\quad \land\ Grid' = [Grid \ \text{EXCEPT}\ ![self][a] = \text{TRUE}] \\
& \quad\quad\quad\quad \land\ \land\ app1' = [app1 \ \text{EXCEPT}\ ![self] = self] \\
& \quad\quad\quad\quad\quad \land\ app2' = [app2 \ \text{EXCEPT}\ ![self] = a] \\
& \quad\quad\quad\quad\quad \land\ stack' = [stack \ \text{EXCEPT}\ ![self] = \langle[procedure \mapsto \text{“Delegate”}, \\
& \quad\quad\quad\quad\quad\quad\quad pc \mapsto \text{“s1”}, \\
& \quad\quad\quad\quad\quad\quad\quad ResourceList \mapsto ResourceList[self], \\
& \quad\quad\quad\quad\quad\quad\quad app1 \mapsto app1[self], \\
& \quad\quad\quad\quad\quad\quad\quad app2 \mapsto app2[self]]\rangle \\
& \quad\quad\quad\quad\quad\quad\quad \circ\ stack[self]] \\
& \quad\quad\quad\quad \land\ ResourceList' = [ResourceList \ \text{EXCEPT}\ ![self] = Resources] \\
& \quad\quad\quad\quad \land\ pc' = [pc \ \text{EXCEPT}\ ![self] = \text{“DELEGATE”}] \\
& \quad\quad \text{ELSE}\ \land\ pc' = [pc \ \text{EXCEPT}\ ![self] = \text{“s1”}] \\
& \quad\quad\quad\quad \land\ \text{UNCHANGED}\ \langle Grid, stack, app1, app2, ResourceList\rangle \\
& \land\ \text{UNCHANGED}\ \langle Acl\_Status, Acl\_PermissionType, \\
& \quad\quad Acl\_PermissionLevel, Consent, p2, ResourceList\_, \\
& \quad\quad Resource\rangle
\end{aligned}
$$

$$
\begin{aligned}
AcmNext(self) \triangleq\ & s0(self) \lor s1(self) \lor s2(self) \lor s3(self) \lor s4(self) \\
& \lor s5(self)
\end{aligned}
$$

Allow infinite stuttering to prevent deadlock on termination.

$$
\begin{aligned}
Terminating \triangleq\ & \land\ \forall\, self \in ProcSet : pc[self] = \text{“Done”} \\
& \land\ \text{UNCHANGED}\ vars
\end{aligned}
$$

$$
\begin{aligned}
Next \triangleq\ & (\exists\, self \in ProcSet : Update(self) \lor Delegate(self)) \\
& \lor (\exists\, self \in Processes : AcmNext(self))
\end{aligned}
$$

$\lor\ Terminating$

$Spec \triangleq \land Init \land \Box[Next]_{vars}$
$\land \forall\, self \in Processes : \mathrm{WF}_{vars}(AcmNext(self)) \land \mathrm{WF}_{vars}(Update(self)) \land \mathrm{WF}_{vars}(Delegate(self))$

$Termination \triangleq \Diamond(\forall\, self \in ProcSet : pc[self] = \text{“Done”})$

END TRANSLATION

$AcmTypeOK \triangleq \land Acl\_Status \in [Processes \to [Resources \to ResourceStatus \cup \{NULL\}]]$
$\land Acl\_PermissionType \in [Processes \to [Resources \to PermissionTypes \cup \{NULL\}]]$
$\land Acl\_PermissionLevel \in [Processes \to [Resources \to PermissionLevels \cup \{NULL\}]]$
$\land Consent \in [Processes \to [Resources \to Boolean]]$
$\land Grid \in [Processes \to [Processes \to Boolean]]$

$AcmRedelegation \triangleq \neg(\exists\, p \in Processes :$
$\exists\, r \in Resources :$
$\land Acl\_Status[p][r] = IN\_USE$
$\land Consent[p][r] \neq \text{TRUE})$

$AcmLiveness \triangleq \Diamond(\exists\, p \in Processes :$
$\exists\, r \in Resources :$
$Acl\_Status[p][r] = ALLOWED \lor Acl\_Status[p][r] = REJECTED)$

* Modification History
* Last modified Sat *Jun* 10 23:11:47 *GMT*+03:30 2023 by *Amirhosein*
* Created *Thu Mar* 23 07:45:26 *GMT*+03:30 2023 by *Amirhosein*