MODULE *DeCrypto*

LOCAL INSTANCE *Sequences*
LOCAL INSTANCE *Naturals*
LOCAL INSTANCE *SSWPacket*

LOCAL $HMACSIZE \triangleq 64$
LOCAL $START \triangleq$ "!"
LOCAL $MINMESSAGESIZE \triangleq 65$
LOCAL $PASSWORD \triangleq$ "lolpassword"
LOCAL $SignedMessages \triangleq \{isSSW,$
$notSSW,$
$\langle$ "D", "9", "2", "8", "D", "9", "2", "8",
"7", "5", "3", "0", "7", "5", "3", "0",
"9", "8", "5", "C", "9", "8", "5", "C",
"E", "B", "B", "A", "E", "B", "B", "A",
"D", "9", "2", "8", "D", "9", "2", "8",
"7", "5", "3", "0", "7", "5", "3", "0",
"9", "8", "5", "C", "9", "8", "5", "C",
"E", "B", "B", "A", "E", "B", "B", "A",
"9", "8", "5", "C", "9", "8", "5", "C",
"E", "B", "B", "A", "E", "B", "B", "A",
"D", "9", "2", "8", "D", "9", "2", "8",
"7", "5", "3", "0", "7", "5", "3", "0",
"9", "8", "5", "C", "9", "8", "5", "C" $\rangle\}$

$HMAC(str, pass) \triangleq \langle$ "l", "o", "l", "h", "m", "a", "c" $\rangle$ not concerned with the inner workings of *SHA*2

$SendMessage(str) \triangleq$ TRUE sending message to another cell. Assuming this works

--fair algorithm *DeCrypto*

variables $signedMessage \in SignedMessages,$
$bareMessage = \langle\rangle,$
$flag =$ FALSE,
$retreivedHMAC = \langle\rangle,$
$generatedHMAC = \langle\rangle,$
$result =$ FALSE,
$CompareHMAC \in$ BOOLEAN , since we dont model *SHA*2 this is random
$hmacsMatch =$ FALSE

begin

msgCheck: **if** $Len(signedMessage) \geq MINMESSAGESIZE$
then if $\langle Head(signedMessage)\rangle = \langle$ "!" $\rangle$
then $bareMessage := GetMessage(signedMessage)$;
$retreivedHMAC := GetHMAC(signedMessage)$;
$h1$: $generatedHMAC := HMAC(bareMessage, PASSWORD)$;

```
                    h2: hmacsMatch := CompareHMAC ;
                    h3: if hmacsMatch
                            then flag := TRUE ;
                                 result := SendMessage(bareMessage) ;
                        end if ;
                end if ;
            end if ;

end algorithm
```

BEGIN TRANSLATION

VARIABLES $signedMessage$, $bareMessage$, $flag$, $retreivedHMAC$, $generatedHMAC$, $result$, $CompareHMAC$, $hmacsMatch$, pc

$vars \triangleq \langle signedMessage, bareMessage, flag, retreivedHMAC, generatedHMAC, result, CompareHMAC, hmacsMatch, pc \rangle$

$Init \triangleq$ Global variables
$\land\ signedMessage \in SignedMessages$
$\land\ bareMessage = \langle\rangle$
$\land\ flag = \text{FALSE}$
$\land\ retreivedHMAC = \langle\rangle$
$\land\ generatedHMAC = \langle\rangle$
$\land\ result = \text{FALSE}$
$\land\ CompareHMAC \in \text{BOOLEAN}$
$\land\ hmacsMatch = \text{FALSE}$
$\land\ pc =$ "msgCheck"

$msgCheck \triangleq$ $\land\ pc =$ "msgCheck"
$\land$ IF $Len(signedMessage) \geq MINMESSAGESIZE$
THEN $\land$ IF $\langle Head(signedMessage) \rangle = \langle$ "!" $\rangle$
THEN $\land\ bareMessage' = GetMessage(signedMessage)$
$\land\ retreivedHMAC' = GetHMAC(signedMessage)$
$\land\ pc' =$ "h1"
ELSE $\land\ pc' =$ "Done"
$\land$ UNCHANGED $\langle bareMessage, retreivedHMAC \rangle$
ELSE $\land\ pc' =$ "Done"
$\land$ UNCHANGED $\langle bareMessage, retreivedHMAC \rangle$
$\land$ UNCHANGED $\langle signedMessage, flag, generatedHMAC, result, CompareHMAC, hmacsMatch \rangle$

$h1 \triangleq$ $\land\ pc =$ "h1"
$\land\ generatedHMAC' = HMAC(bareMessage, PASSWORD)$
$\land\ pc' =$ "h2"
$\land$ UNCHANGED $\langle signedMessage, bareMessage, flag, retreivedHMAC, result, CompareHMAC, hmacsMatch \rangle$

$h2 \triangleq$ $\land\ pc =$ "h2"

$\land\ hmacsMatch' = CompareHMAC$
$\land\ pc' = \text{“h3”}$
$\land$ UNCHANGED $\langle signedMessage,\ bareMessage,\ flag,\ retreivedHMAC,$
$generatedHMAC,\ result,\ CompareHMAC\rangle$

$h3 \triangleq \land\ pc = \text{“h3”}$
$\land$ IF $hmacsMatch$
THEN $\land\ flag' = \text{TRUE}$
$\land\ result' = SendMessage(bareMessage)$
ELSE $\land$ TRUE
$\land$ UNCHANGED $\langle flag,\ result\rangle$
$\land\ pc' = \text{“Done”}$
$\land$ UNCHANGED $\langle signedMessage,\ bareMessage,\ retreivedHMAC,\ generatedHMAC,$
$CompareHMAC,\ hmacsMatch\rangle$

$Next \triangleq\ msgCheck \lor h1 \lor h2 \lor h3$
$\lor$ Disjunct to prevent deadlock on termination
$(pc = \text{“Done”} \land \text{UNCHANGED } vars)$

$Spec \triangleq\ \land\ Init \land \Box[Next]_{vars}$
$\land\ \text{WF}_{vars}(Next)$

$Termination \triangleq \Diamond(pc = \text{“Done”})$

END TRANSLATION

$SAFETYCHECK \triangleq$
The password is never changed
$\land\ PASSWORD = \text{“lolpassword”}$
if we send something then it was valid SSW
$\land\ flag = \text{TRUE} \Rightarrow IsSSW(signedMessage)$
we forward a packet if and only if the $HMACs$ match
$\land\ result = \text{TRUE} \Rightarrow hmacsMatch = \text{TRUE}$

$LIVELINESS \triangleq$
if the $HMACs$ match then we eventually send something
$\land\ hmacsMatch = \text{TRUE} \rightsquigarrow result = \text{TRUE}$

* Modification History
* Last modified *Tue* May 08 03:43:33 *EDT* 2018 by *SabraouM*
* Created Sun May 06 09:03:31 *EDT* 2018 by *SabraouM*