MODULE *auth_device*

EXTENDS *Sequences*, *FiniteSets*, *Naturals*, *TLC*, *Modbus*, *ASCII*, *SSWPacket*

LOCAL $HMACSIZE \triangleq 64$
LOCAL $MINMESSAGESIZE \triangleq 1$
LOCAL $MINMACMESSAGESIZE \triangleq 65$
LOCAL $PASSWORD \triangleq$ "lolpassword"
LOCAL INSTANCE *Hex*
WITH $natValue \leftarrow 0$, $hexValue \leftarrow \langle 0 \rangle$
LOCAL $Range(T) \triangleq \{T[x] : x \in \text{DOMAIN } T\}$
$MessagesToSerialPort \triangleq$ these are in *ASCII* but they are converted to decimal before being used below. See *StrTupleToN*
⟨ *StrTupleToNumTuple*(⟨":", "J", "G", "P", "9", "4", "3", "2", "J", "3", "9", "J", "G", "W", "I", "R", "W"⟩),
StrTupleToNumTuple(⟨":", "1", "1", "0", "3", "0", "0", "6", "B", "0", "0", "0", "3", "7", "E", "\r", "\
StrTupleToNumTuple(⟨":", "1", "1", "0", "3", "0", "0", "6", "B", "0", "0", "0", "3", "7", "E", "C", "R"
StrTupleToNumTuple(⟨":", "1", "1", "0", "3", "0", "0", "6", "B", "0", "0", "0", ":", "1", "1", "0", "3", "0", "0", "6", "B",
*this one starts a new modbus packet half way through the message
StrTupleToNumTuple(⟨"!", "m", "Q", "I", "N", "B", "F", "u", "O", "v", "x", "M", "B", "E", "A", "D", "g", "D", "B", "Q",

⟨⟩,⟨1⟩,⟨2⟩,⟨3⟩,⟨4⟩,⟨5⟩,⟨6⟩,⟨7⟩,⟨8⟩,⟨9⟩,⟨10⟩,⟨11⟩,⟨12⟩,⟨13⟩, * all possible values
⟨14⟩,⟨15⟩,⟨16⟩,⟨17⟩,⟨18⟩,⟨19⟩,⟨20⟩,⟨21⟩,⟨22⟩,⟨23⟩,⟨24⟩,⟨25⟩, *that could come across the serial line
⟨26⟩,⟨27⟩,⟨28⟩,⟨29⟩,⟨30⟩,⟨31⟩,⟨32⟩,⟨33⟩,⟨34⟩,⟨35⟩,⟨36⟩,⟨37⟩,⟨38⟩,⟨39⟩,⟨40⟩,⟨41⟩,
⟨42⟩,⟨43⟩,⟨44⟩,⟨45⟩,⟨46⟩,⟨47⟩,⟨48⟩,⟨49⟩,⟨50⟩,⟨51⟩,⟨52⟩,⟨53⟩,⟨54⟩,⟨55⟩,⟨56⟩,⟨57⟩,
⟨58⟩,⟨59⟩,⟨60⟩,⟨61⟩,⟨62⟩,⟨63⟩,⟨64⟩,⟨65⟩,⟨66⟩,⟨67⟩,⟨68⟩,⟨69⟩,⟨70⟩,⟨71⟩,⟨72⟩,⟨73⟩,
⟨74⟩,⟨75⟩,⟨76⟩,⟨77⟩,⟨78⟩,⟨79⟩,⟨80⟩,⟨81⟩,⟨82⟩,⟨83⟩,⟨84⟩,⟨85⟩,⟨86⟩,⟨87⟩,⟨88⟩,⟨89⟩,
⟨90⟩,⟨91⟩,⟨92⟩,⟨93⟩,⟨94⟩,⟨95⟩,⟨96⟩,⟨97⟩,⟨98⟩,⟨99⟩,⟨100⟩,⟨101⟩,⟨102⟩,⟨103⟩,⟨104⟩,⟨105⟩,
⟨106⟩,⟨107⟩,⟨108⟩,⟨109⟩,⟨110⟩,⟨111⟩,⟨112⟩,⟨113⟩,⟨114⟩,⟨115⟩,⟨116⟩,⟨117⟩,⟨118⟩,⟨119⟩,⟨120⟩,⟨121⟩,

⟨122⟩,⟨123⟩,⟨124⟩,⟨125⟩,⟨126⟩,⟨127⟩,⟨128⟩,⟨129⟩,⟨130⟩,⟨131⟩,⟨132⟩,⟨133⟩,⟨134⟩,⟨135⟩,⟨136⟩,⟨137⟩,

⟨138⟩,⟨139⟩,⟨140⟩,⟨141⟩,⟨142⟩,⟨143⟩,⟨144⟩,⟨145⟩,⟨146⟩,⟨147⟩,⟨148⟩,⟨149⟩,⟨150⟩,⟨151⟩,⟨152⟩,⟨153⟩,

⟨154⟩,⟨155⟩,⟨156⟩,⟨157⟩,⟨158⟩,⟨159⟩,⟨160⟩,⟨161⟩,⟨162⟩,⟨163⟩,⟨164⟩,⟨165⟩,⟨166⟩,⟨167⟩,⟨168⟩,⟨169⟩,

⟨170⟩,⟨171⟩,⟨172⟩,⟨173⟩,⟨174⟩,⟨175⟩,⟨176⟩,⟨177⟩,⟨178⟩,⟨179⟩,⟨180⟩,⟨181⟩,⟨182⟩,⟨183⟩,⟨184⟩,⟨185⟩,

⟨186⟩,⟨187⟩,⟨188⟩,⟨189⟩,⟨190⟩,⟨191⟩,⟨192⟩,⟨193⟩,⟨194⟩,⟨195⟩,⟨196⟩,⟨197⟩,⟨198⟩,⟨199⟩,⟨200⟩,⟨201⟩,

⟨202⟩,⟨203⟩,⟨204⟩,⟨205⟩,⟨206⟩,⟨207⟩,⟨208⟩,⟨209⟩,⟨210⟩,⟨211⟩,⟨212⟩,⟨213⟩,⟨214⟩,⟨215⟩,⟨216⟩,⟨217⟩,

⟨218⟩,⟨219⟩,⟨220⟩,⟨221⟩,⟨222⟩,⟨223⟩,⟨224⟩,⟨225⟩,⟨226⟩,⟨227⟩,⟨228⟩,⟨229⟩,⟨230⟩,⟨231⟩,⟨232⟩,⟨233⟩,

⟨234⟩,⟨235⟩,⟨236⟩,⟨237⟩,⟨238⟩,⟨239⟩,⟨240⟩,⟨241⟩,⟨242⟩,⟨243⟩,⟨244⟩,⟨245⟩,⟨246⟩,⟨247⟩,⟨248⟩,⟨249⟩,

⟨250⟩,⟨251⟩,⟨252⟩,⟨253⟩,⟨254⟩,⟨255⟩

⟩
$MessagesToUntrustNet \triangleq$ ⟨*StrTupleToNumTuple*(⟨"!", "m", "Q", "I", "N", "B", "F", "u", "O", "v", "x", "M
StrTupleToNumTuple(⟨":", "1", "1", "0", "3", "0", "0", "6", "B", "0", "0", "0", "

$\rangle$

$HMAC(str,\ pass) \triangleq \langle$ "I", "K", "o", "W", "L", "9", "v", "G", "U", "h", "S", "1", "q", "t", "Z", "f", "4", "5"

$FindPartnerMessage(msg,\ messages) \triangleq$ CHOOSE $x \in messages : x.id = msg.id$

--fair algorithm *auth_device*

variables

$chan = [trustnet\_out \mapsto \langle\rangle,\ sign \mapsto \langle\rangle,\ verify \mapsto \langle\rangle,\ messagecheck \mapsto \langle\rangle,$
$untrustnet\_in \mapsto \langle\rangle,\ untrustnet\_out \mapsto \langle\rangle,\ finished\_untrustnet \mapsto \langle\rangle,\ finished\_trustnet \mapsto \langle\rangle],$

IPC calls

macro *send*(*dest*, *msg*)
begin
print "sending to " $\circ$ *dest*;
$chan[dest] := Append(chan[dest],\ msg)$;
end macro ;

macro *receive*(*channel*, *msg*)
begin
print *channel* $\circ$ " received *msg*";
await $Len(chan[channel]) > 0$;
$msg := Head(chan[channel])$;
$chan[channel] := Tail(chan[channel])$;
end macro ;

Signing process.

fair process *sign* = "sign"

variables $msg = \langle\rangle,$
$generated\_hmac = \langle\rangle$;

begin

*sign*1: **while** TRUE **do**

receive("sign", *msg*) ;
*sign*2: $generated\_hmac := HMAC(msg.text,\ PASSWORD)$; hash it and the password
*sign*3: *send*("untrustnet_out", $[id \mapsto msg.id,\ hmac \mapsto generated\_hmac,\ source \mapsto$ "sign", *isValid* $\mapsto$
end while ;

end process

Check validitity of the underlying protocol. In this case *Modbus*

fair process *msgchk* = "msgchk"

variables $msg = \langle\rangle$

begin

*modbus*1: **while** TRUE **do**

```
                receive("messagecheck", msg);
                if IsModbus(NumTupleToStrTuple(msg.text)) then
                    if msg.source = "trustnet_in" then
                        mod1: send("untrustnet_out", [id ↦ msg.id, isValid ↦ TRUE, source ↦ "msgchk", text
                    elsif (msg.source = "untrust_in") then
                        mod2: send("trustnet_out", [id ↦ msg.id, isValid ↦ TRUE, source ↦ "msgchk", text ↦
                    end if ;

                else
                    if msg.source = "trustnet_in" then
                        mod3: send("untrustnet_out", [id ↦ msg.id, isValid ↦ FALSE, source ↦ "msgchk", tex
                    elsif msg.source = "untrust_in" then
                        mod4: send("trustnet_out", [id ↦ msg.id, isValid ↦ FALSE, source ↦ "msgchk", text
                    end if ;
                end if ;
            end while ;

end process
```

Check the valididity of the signature

```
fair process verify = "verify"

variables   msg = ⟨⟩,
            bareMessage = ⟨⟩,
            retreivedHMAC  = ⟨⟩,
            generatedHMAC = ⟨⟩,
            result = FALSE,
            CompareHMAC ∈ BOOLEAN ,   since we dont model SHA2 this is random
            hmacsMatch = FALSE
begin
verify1:  while TRUE do
              receive("verify", msg);
              verify2: retreivedHMAC  := GetHMAC(msg.text);
              verify3: generatedHMAC := HMAC(msg.text, PASSWORD);
              verify4: hmacsMatch := CompareHMAC ;
              if hmacsMatch then
                  verify5: send("trustnet_out", [id ↦ msg.id, isValid ↦ TRUE, source ↦ "verify", text ↦ msg
              else
                  verify6: send("trustnet_out", [id ↦ msg.id, isValid ↦ FALSE, source ↦ "verify", text ↦ ms
              end if ;
          end while ;

end process
```

Receive plaintext modbus from the trusted serial port

```
fair process trustnet_in = "trustnet_in"

variables   rxBuf = ⟨⟩,
            incomingMessages = ⟨⟩,   MessagesToSerialPort,
            guid = ⟨0⟩,
            msg = ⟨⟩,
            msgid = ⟨⟩,
            rx = FALSE,
            rxReg = ⟨⟩,
            last2 = ⟨3, 3⟩,   dummy numbers
            incByte = ⟨⟩,

begin
  wait for something to appear in the buffer
trustnet_in1:  while Len(incomingMessages) > 0 do
                   if Len(incomingMessages) > 1 then
                       uti1: msg := Head(incomingMessages) ;
                       uti2: incomingMessages := Tail(incomingMessages) ;
                    else
                       uti3: msg := incomingMessages[1] ;
                       uti4: incomingMessages := ⟨⟩ ;
                   end if ;
               start:   while Len(msg) > 0 do
                        if Len(rxBuf) = MAXMODBUSSIZE then
                            rxBuf := ⟨⟩ ;
                            rxReg := ⟨⟩ ;
                            incByte := ⟨⟩ ;
                            incMessage := ⟨⟩ ;
                            goto start ;
                        end if ;
               inc:     incByte := ⟨Head(msg)⟩ ;
                        msg := Tail(msg) ;
                        if incByte = ⟨⟩ then
                         else
                            rxReg := incByte ;
                        end if ;


               receive:          a ":" character indicates the start of a new message
                                 if rxReg = StrTupleToNumTuple(⟨":"⟩)
                                     then rxBuf := ⟨⟩ ;
                                 end if ;
                                 r0:  last2  := Tail(last2 ∘ rxReg) ;
                                 r1:  rxBuf := rxBuf ∘ rxReg ;   put the contents of the register into the buffer
                                   empty the register
                                 r2: rxReg := ⟨⟩ ;
```

```
                check:    if we get the end of the modbus "/r/n" then ship it
                          if NumTupleToStrTuple(last2) = ⟨"\r", "\n"⟩ then   convert back to ASCII before che
                              check0: msgid := ⟨guid[1]⟩ ∘ ⟨"t", "n", "i"⟩ ;
                              check1: guid[1] := guid[1] + 1 ;
                              check2: send("messagecheck", [id ↦ msgid, text ↦ rxBuf, source ↦ "trustnet_i
                              check3: send("sign", [id ↦ msgid, text ↦ rxBuf]) ;
                              check4: rxBuf := ⟨⟩ ;
                              rxReg := ⟨⟩ ;
                              incByte := ⟨⟩ ;
                              incMessage := ⟨⟩ ;
          end if ;
       end while ;
    end while ;

end process ;
```

```
 Receive signed messages from untrusted serial port
fair process untrustnet_in = "untrustnet_in"

variables   rxBuf = ⟨⟩,
            guid = 0,
            incomingMessages = MessagesToUntrustNet,   MessagesToSerialPort,
            msg = ⟨⟩,
            msgid = ⟨⟩,
            rx = FALSE,
            rxReg = ⟨⟩,
            last2  = ⟨3, 3⟩,   dummy numbers
            incMessage = ⟨⟩,
            incByte = ⟨⟩,

begin
  wait for something to appear in the buffer
untrustnet_in1:   while Len(incomingMessages) > 0 do
                    if Len(incomingMessages) > 1 then
                        uti1: msg := Head(incomingMessages) ;
                        uti2: incomingMessages := Tail(incomingMessages) ;
                     else
                        uti3: msg := incomingMessages[1] ;
                        uti4: incomingMessages := ⟨⟩ ;
                    end if ;
                      print (⟨"u", "t", "i", ":", ""⟩ ∘ NumTupleToStrTuple(msg));
                start:  while Len(msg) > 0 do
                inc:      incByte :=  ⟨Head(msg)⟩ ;
                          incMessage := Tail(msg) ;
                          if incByte = ⟨⟩ then
```

```
                    else
                        rxReg := incByte ;
                    end if ;

            receive:    a "!" character indicates the start of a new message
                    if rxReg = StrTupleToNumTuple(⟨"!"⟩)
                        then rxBuf := ⟨⟩ ;
                    end if ;
                    r0:  last2  := Tail(last2 ∘ rxReg) ;
                    r1:  rxBuf := rxBuf ∘ rxReg ;   put the contents of the register into the buffer
                      empty the register
                    r2: rxReg := ⟨⟩ ;
            check:  if we get the end of the modbus "/r/n" then ship it
                    if NumTupleToStrTuple(last2) = ⟨"\r", "\n"⟩ then   convert back to ASCII before chec
                        check0: msgid := guid ∘ "untrustnet_in" ;
                        check1: guid := guid + 1 ;
                        check2: send("messagecheck", [id ↦ msgid, text ↦ rxBuf, source ↦ "untrustnet
                        check3: send("verify", [id ↦ msgid, text ↦ rxBuf]) ;
                        check4: rxBuf := ⟨⟩ ;
                        rxReg := ⟨⟩ ;
                        incByte := ⟨⟩ ;
                        incMessage := ⟨⟩ ;
        end if ;
    end while ;
end while ;

end process ;
```

```
 process to send modbus out the trusted serial port
fair process trustnet_out = "trustnet_out"

variables   msg = ⟨⟩,
            txBuf = ⟨⟩,
            txReg = ⟨⟩,
            adder = 0,
            validMessages = {}

begin
    to1: while TRUE do
        receive("trustnet_out", msg) ;
        if msg.isValid then   if the message is valid then look for another message in the validMessages set with the same i
            if ∃ x ∈ validMessages : x.id = msg.id then   if one exists then both portions of the message were verified a
                txBuf := msg.text ;
                transmit: send("finished_trustnet", NumTupleToStrTuple(txBuf)) ;   converting back to characters fo
                to2: validMessages := validMessages \ {x ∈ validMessages : x.id = msg.id} ;   remove sent message
```

```
            else
                validMessages := validMessages ∪ {msg} ;   if a message with the same id is not found then add this me
            end if ;
        end if ;
        finished:   txReg := ⟨⟩ ;
        txBuf := ⟨⟩ ;
    end while ;

end process ;
```

---

process to send modbus out the trusted serial port

```
fair process untrustnet_out = "untrustnet_out"

variables   msg = ⟨⟩,
            txBuf = ⟨⟩,
            txReg = ⟨⟩,
            adder = 0,
            validMessages = {}

begin
    uto1: while TRUE do
        receive("untrustnet_out", msg) ;
        if msg.isValid then   if the message is valid then look for another message in the validMessages set with the same i
            if ∃ x ∈ validMessages : x.id = msg.id then   if one exists then both portions of the message were verified a
                if msg.source = "sign" then
                    txBuf := StrTupleToNumTuple(⟨"!"⟩) ∘ StrTupleToNumTuple(msg.hmac) ∘ msg.text ;
                else
                    txBuf := StrTupleToNumTuple(⟨"!"⟩) ∘ StrTupleToNumTuple(FindPartnerMessage(msg, va
                end if ;
                transmit:   send("finished_untrustnet", NumTupleToStrTuple(txBuf)) ;
                uto2:   validMessages := validMessages \ {x ∈ validMessages : x.id = msg.id} ;   remove sent mess
            else
                validMessages := validMessages ∪ {msg} ;   if a message with the same id is not found then add this me
            end if ;
        end if ;
        finished:   txReg := ⟨⟩ ;
        txBuf := ⟨⟩ ;
    end while ;

end process ;



end algorithm ;
```

BEGIN TRANSLATION

Label *uti*1 of process *trustnet_in* at line 149 col 31 changed to *uti*1_

Label *uti*2 of process *trustnet_in* at line 150 col 31 changed to *uti*2_
Label *uti*3 of process *trustnet_in* at line 152 col 31 changed to *uti*3_
Label *uti*4 of process *trustnet_in* at line 153 col 31 changed to *uti*4_
Label start of process *trustnet_in* at line 155 col 25 changed to *start_*
Label *inc* of process *trustnet_in* at line 163 col 25 changed to *inc_*
Label receive of process *trustnet_in* at line 172 col 29 changed to *receive_*
Label *r*0 of process *trustnet_in* at line 175 col 33 changed to *r*0_
Label *r*1 of process *trustnet_in* at line 176 col 33 changed to *r*1_
Label *r*2 of process *trustnet_in* at line 178 col 33 changed to *r*2_
Label check of process *trustnet_in* at line 180 col 29 changed to *check_*
Label *check*0 of process *trustnet_in* at line 181 col 41 changed to *check*0_
Label *check*1 of process *trustnet_in* at line 182 col 41 changed to *check*1_
Label *check*2 of process *trustnet_in* at line 53 col 9 changed to *check*2_
Label *check*3 of process *trustnet_in* at line 53 col 9 changed to *check*3_
Label *check*4 of process *trustnet_in* at line 185 col 41 changed to *check*4_
Label transmit of process *trustnet_out* at line 53 col 9 changed to *transmit_*
Label finished of process *trustnet_out* at line 276 col 19 changed to *finished_*
Process variable *msg* of process sign at line 67 col 13 changed to *msg_*
Process variable *msg* of process *msgchk* at line 82 col 13 changed to *msg_m*
Process variable *msg* of process verify at line 109 col 13 changed to *msg_v*
Process variable *rxBuf* of process *trustnet_in* at line 135 col 13 changed to *rxBuf_*
Process variable *incomingMessages* of process *trustnet_in* at line 136 col 13 changed to *incomingMessages_*
Process variable *guid* of process *trustnet_in* at line 137 col 13 changed to *guid_*
Process variable *msg* of process *trustnet_in* at line 138 col 13 changed to *msg_t*
Process variable *msgid* of process *trustnet_in* at line 139 col 13 changed to *msgid_*
Process variable *rx* of process *trustnet_in* at line 140 col 13 changed to *rx_*
Process variable *rxReg* of process *trustnet_in* at line 141 col 13 changed to *rxReg_*
Process variable *last*2 of process *trustnet_in* at line 142 col 13 changed to *last*2_
Process variable *incByte* of process *trustnet_in* at line 143 col 13 changed to *incByte_*
Process variable *msg* of process *untrustnet_in* at line 202 col 13 changed to *msg_u*
Process variable *msg* of process *trustnet_out* at line 258 col 13 changed to *msg_tr*
Process variable *txBuf* of process *trustnet_out* at line 259 col 13 changed to *txBuf_*
Process variable *txReg* of process *trustnet_out* at line 260 col 13 changed to *txReg_*
Process variable adder of process *trustnet_out* at line 261 col 13 changed to *adder_*
Process variable *validMessages* of process *trustnet_out* at line 262 col 13 changed to *validMessages_*

VARIABLES *chan*, *pc*, *msg_*, *generated_hmac*, *msg_m*, *msg_v*, *bareMessage*, *retreivedHMAC*, *generatedHMAC*, *result*, *CompareHMAC*, *hmacsMatch*, *rxBuf_*, *incomingMessages_*, *guid_*, *msg_t*, *msgid_*, *rx_*, *rxReg_*, *last*2_, *incByte_*, *rxBuf*, *guid*, *incomingMessages*, *msg_u*, *msgid*, *rx*, *rxReg*, *last*2, *incMessage*, *incByte*, *msg_tr*, *txBuf_*, *txReg_*, *adder_*, *validMessages_*, *msg*, *txBuf*, *txReg*, *adder*, *validMessages*

$vars \triangleq \langle chan, pc, msg\_, generated\_hmac, msg\_m, msg\_v, bareMessage, retreivedHMAC, generatedHMAC, result, CompareHMAC, hmacsMatch, rxBuf\_, incomingMessages\_, guid\_, msg\_t, msgid\_, rx\_, rxReg\_,$

$last2\_$, $incByte\_$, $rxBuf$, $guid$, $incomingMessages$, msg_u, $msgid$, rx,
$rxReg$, $last2$, $incMessage$, $incByte$, msg_tr, $txBuf\_$, $txReg\_$, $adder\_$,
$validMessages\_$, msg, $txBuf$, $txReg$, $adder$, $validMessages\rangle$

$ProcSet \triangleq \{\text{“sign”}\} \cup \{\text{“msgchk”}\} \cup \{\text{“verify”}\} \cup \{\text{“trustnet\_in”}\} \cup \{\text{“untrustnet\_in”}\} \cup \{\text{“trustnet\_out”}\} \cup \{\text{“un}$

$Init \triangleq$ Global variables
$\land\ chan = [trustnet\_out \mapsto \langle\rangle,\ sign \mapsto \langle\rangle,\ verify \mapsto \langle\rangle,\ messagecheck \mapsto \langle\rangle,$
$\quad untrustnet\_in \mapsto \langle\rangle,\ untrustnet\_out \mapsto \langle\rangle,\ finished\_untrustnet \mapsto \langle\rangle,\ finished\_trustnet \mapsto \langle\rangle]$
Process sign
$\land\ msg\_ = \langle\rangle$
$\land\ generated\_hmac = \langle\rangle$
Process *msgchk*
$\land\ msg\_m = \langle\rangle$
Process verify
$\land\ msg\_v = \langle\rangle$
$\land\ bareMessage = \langle\rangle$
$\land\ retreivedHMAC\ = \langle\rangle$
$\land\ generatedHMAC = \langle\rangle$
$\land\ result = \text{FALSE}$
$\land\ CompareHMAC \in \text{BOOLEAN}$
$\land\ hmacsMatch = \text{FALSE}$
Process *trustnet_in*
$\land\ rxBuf\_ = \langle\rangle$
$\land\ incomingMessages\_ = MessagesToSerialPort$
$\land\ guid\_\ = \langle 0\rangle$
$\land\ msg\_t = \langle\rangle$
$\land\ msgid\_ = \langle\rangle$
$\land\ rx\_ = \text{FALSE}$
$\land\ rxReg\_ = \langle\rangle$
$\land\ last2\_\ = \langle 3, 3\rangle$
$\land\ incByte\_ = \langle\rangle$
Process *untrustnet_in*
$\land\ rxBuf = \langle\rangle$
$\land\ guid = 0$
$\land\ incomingMessages = MessagesToUntrustNet$
$\land\ msg\_u = \langle\rangle$
$\land\ msgid\ = \langle\rangle$
$\land\ rx = \text{FALSE}$
$\land\ rxReg = \langle\rangle$
$\land\ last2\ = \langle 3, 3\rangle$
$\land\ incMessage = \langle\rangle$
$\land\ incByte = \langle\rangle$
Process *trustnet_out*
$\land\ msg\_tr = \langle\rangle$

$\land\ txBuf\_ = \langle\rangle$
$\land\ txReg\_ = \langle\rangle$
$\land\ adder\_ = 0$
$\land\ validMessages\_ = \{\}$
Process *untrustnet_out*
$\land\ msg = \langle\rangle$
$\land\ txBuf = \langle\rangle$
$\land\ txReg = \langle\rangle$
$\land\ adder = 0$
$\land\ validMessages = \{\}$
$\land\ pc = [self \in ProcSet \mapsto$ CASE $self =$ "sign" $\to$ "sign1"
$\square$ $self =$ "msgchk" $\to$ "modbus1"
$\square$ $self =$ "verify" $\to$ "verify1"
$\square$ $self =$ "trustnet_in" $\to$ "trustnet_in1"
$\square$ $self =$ "untrustnet_in" $\to$ "untrustnet_in1"
$\square$ $self =$ "trustnet_out" $\to$ "to1"
$\square$ $self =$ "untrustnet_out" $\to$ "uto1"]

$sign1 \triangleq$ $\land\ pc[$"sign"$] =$ "sign1"
$\land\ Len(chan[$"sign"$]) > 0$
$\land\ msg\_' = Head(chan[$"sign"$])$
$\land\ chan' = [chan$ EXCEPT $![$"sign"$] = Tail(chan[$"sign"$])]$
$\land\ pc' = [pc$ EXCEPT $![$"sign"$] =$ "sign2"$]$
$\land$ UNCHANGED $\langle generated\_hmac, msg\_m, msg\_v, bareMessage, retreivedHMAC, generatedHMAC, result, CompareHMAC, hmacsMatch, rxBuf\_, incomingMessages\_, guid\_, msg\_t, msgid\_, rx\_, rxReg\_, last2\_, incByte\_, rxBuf, guid, incomingMessages, msg\_u, msgid, rx, rxReg, last2, incMessage, incByte, msg\_tr, txBuf\_, txReg\_, adder\_, validMessages\_, msg, txBuf, txReg, adder, validMessages\rangle$

$sign2 \triangleq$ $\land\ pc[$"sign"$] =$ "sign2"
$\land\ generated\_hmac' = HMAC(msg\_.text, PASSWORD)$
$\land\ pc' = [pc$ EXCEPT $![$"sign"$] =$ "sign3"$]$
$\land$ UNCHANGED $\langle chan, msg\_, msg\_m, msg\_v, bareMessage, retreivedHMAC, generatedHMAC, result, CompareHMAC, hmacsMatch, rxBuf\_, incomingMessages\_, guid\_, msg\_t, msgid\_, rx\_, rxReg\_, last2\_, incByte\_, rxBuf, guid, incomingMessages, msg\_u, msgid, rx, rxReg, last2, incMessage, incByte, msg\_tr, txBuf\_, txReg\_, adder\_, validMessages\_, msg, txBuf, txReg, adder, validMessages\rangle$

$sign3 \triangleq$ $\land\ pc[$"sign"$] =$ "sign3"
$\land\ chan' = [chan$ EXCEPT $![$"untrustnet_out"$] = Append(chan[$"untrustnet_out"$], ([id \mapsto msg\_.id, hm$

$\land\ pc' = [pc \text{ EXCEPT } ![\text{“sign”}] = \text{“sign1”}]$
$\land$ UNCHANGED $\langle msg\_, generated\_hmac, msg\_m, msg\_v, bareMessage,$
$retreivedHMAC, generatedHMAC, result, CompareHMAC,$
$hmacsMatch, rxBuf\_, incomingMessages\_, guid\_, msg\_t,$
$msgid\_, rx\_, rxReg\_, last2\_, incByte\_, rxBuf, guid,$
$incomingMessages, msg\_u, msgid, rx, rxReg, last2,$
$incMessage, incByte, msg\_tr, txBuf\_, txReg\_, adder\_,$
$validMessages\_, msg, txBuf, txReg, adder,$
$validMessages\rangle$

$sign \triangleq sign1 \lor sign2 \lor sign3$

$modbus1 \triangleq \land pc[\text{“msgchk”}] = \text{“modbus1”}$
$\land Len(chan[\text{“messagecheck”}]) > 0$
$\land msg\_m' = Head(chan[\text{“messagecheck”}])$
$\land chan' = [chan \text{ EXCEPT } ![\text{“messagecheck”}] = Tail(chan[\text{“messagecheck”}])]$
$\land$ IF $IsModbus(NumTupleToStrTuple(msg\_m'.text))$
THEN $\land$ IF $msg\_m'.source = \text{“trustnet\_in”}$
THEN $\land\ pc' = [pc \text{ EXCEPT } ![\text{“msgchk”}] = \text{“mod1”}]$
ELSE $\land$ IF $(msg\_m'.source = \text{“untrust\_in”})$
THEN $\land\ pc' = [pc \text{ EXCEPT } ![\text{“msgchk”}] = \text{“mod2”}]$
ELSE $\land\ pc' = [pc \text{ EXCEPT } ![\text{“msgchk”}] = \text{“modbus1”}]$
ELSE $\land$ IF $msg\_m'.source = \text{“trustnet\_in”}$
THEN $\land\ pc' = [pc \text{ EXCEPT } ![\text{“msgchk”}] = \text{“mod3”}]$
ELSE $\land$ IF $msg\_m'.source = \text{“untrust\_in”}$
THEN $\land\ pc' = [pc \text{ EXCEPT } ![\text{“msgchk”}] = \text{“mod4”}]$
ELSE $\land\ pc' = [pc \text{ EXCEPT } ![\text{“msgchk”}] = \text{“modbus1”}]$
$\land$ UNCHANGED $\langle msg\_, generated\_hmac, msg\_v, bareMessage,$
$retreivedHMAC, generatedHMAC, result, CompareHMAC,$
$hmacsMatch, rxBuf\_, incomingMessages\_, guid\_, msg\_t,$
$msgid\_, rx\_, rxReg\_, last2\_, incByte\_, rxBuf, guid,$
$incomingMessages, msg\_u, msgid, rx, rxReg, last2,$
$incMessage, incByte, msg\_tr, txBuf\_, txReg\_, adder\_,$
$validMessages\_, msg, txBuf, txReg, adder,$
$validMessages\rangle$

$mod1 \triangleq \land pc[\text{“msgchk”}] = \text{“mod1”}$
$\land chan' = [chan \text{ EXCEPT } ![\text{“untrustnet\_out”}] = Append(chan[\text{“untrustnet\_out”}], ([id \mapsto msg\_m.id, is$
$\land pc' = [pc \text{ EXCEPT } ![\text{“msgchk”}] = \text{“modbus1”}]$
$\land$ UNCHANGED $\langle msg\_, generated\_hmac, msg\_m, msg\_v, bareMessage,$
$retreivedHMAC, generatedHMAC, result, CompareHMAC,$
$hmacsMatch, rxBuf\_, incomingMessages\_, guid\_, msg\_t,$
$msgid\_, rx\_, rxReg\_, last2\_, incByte\_, rxBuf, guid,$
$incomingMessages, msg\_u, msgid, rx, rxReg, last2,$
$incMessage, incByte, msg\_tr, txBuf\_, txReg\_, adder\_,$
$validMessages\_, msg, txBuf, txReg, adder,$

$validMessages\rangle$

$mod2 \triangleq \land pc[\text{“msgchk”}] = \text{“mod2”}$
$\land chan' = [chan \text{ EXCEPT } ![\text{“trustnet\_out”}] = Append(chan[\text{“trustnet\_out”}], ([id \mapsto msg\_m.id, isVali$
$\land pc' = [pc \text{ EXCEPT } ![\text{“msgchk”}] = \text{“modbus1”}]$
$\land \text{UNCHANGED } \langle msg\_, generated\_hmac, msg\_m, msg\_v, bareMessage,$
$retreivedHMAC, generatedHMAC, result, CompareHMAC,$
$hmacsMatch, rxBuf\_, incomingMessages\_, guid\_, msg\_t,$
$msgid\_, rx\_, rxReg\_, last2\_, incByte\_, rxBuf, guid,$
$incomingMessages, msg\_u, msgid, rx, rxReg, last2,$
$incMessage, incByte, msg\_tr, txBuf\_, txReg\_, adder\_,$
$validMessages\_, msg, txBuf, txReg, adder,$
$validMessages\rangle$

$mod3 \triangleq \land pc[\text{“msgchk”}] = \text{“mod3”}$
$\land chan' = [chan \text{ EXCEPT } ![\text{“untrustnet\_out”}] = Append(chan[\text{“untrustnet\_out”}], ([id \mapsto msg\_m.id, is$
$\land pc' = [pc \text{ EXCEPT } ![\text{“msgchk”}] = \text{“modbus1”}]$
$\land \text{UNCHANGED } \langle msg\_, generated\_hmac, msg\_m, msg\_v, bareMessage,$
$retreivedHMAC, generatedHMAC, result, CompareHMAC,$
$hmacsMatch, rxBuf\_, incomingMessages\_, guid\_, msg\_t,$
$msgid\_, rx\_, rxReg\_, last2\_, incByte\_, rxBuf, guid,$
$incomingMessages, msg\_u, msgid, rx, rxReg, last2,$
$incMessage, incByte, msg\_tr, txBuf\_, txReg\_, adder\_,$
$validMessages\_, msg, txBuf, txReg, adder,$
$validMessages\rangle$

$mod4 \triangleq \land pc[\text{“msgchk”}] = \text{“mod4”}$
$\land chan' = [chan \text{ EXCEPT } ![\text{“trustnet\_out”}] = Append(chan[\text{“trustnet\_out”}], ([id \mapsto msg\_m.id, isVali$
$\land pc' = [pc \text{ EXCEPT } ![\text{“msgchk”}] = \text{“modbus1”}]$
$\land \text{UNCHANGED } \langle msg\_, generated\_hmac, msg\_m, msg\_v, bareMessage,$
$retreivedHMAC, generatedHMAC, result, CompareHMAC,$
$hmacsMatch, rxBuf\_, incomingMessages\_, guid\_, msg\_t,$
$msgid\_, rx\_, rxReg\_, last2\_, incByte\_, rxBuf, guid,$
$incomingMessages, msg\_u, msgid, rx, rxReg, last2,$
$incMessage, incByte, msg\_tr, txBuf\_, txReg\_, adder\_,$
$validMessages\_, msg, txBuf, txReg, adder,$
$validMessages\rangle$

$msgchk \triangleq modbus1 \lor mod1 \lor mod2 \lor mod3 \lor mod4$

$verify1 \triangleq \land pc[\text{“verify”}] = \text{“verify1”}$
$\land Len(chan[\text{“verify”}]) > 0$
$\land msg\_v' = Head(chan[\text{“verify”}])$
$\land chan' = [chan \text{ EXCEPT } ![\text{“verify”}] = Tail(chan[\text{“verify”}])]$
$\land pc' = [pc \text{ EXCEPT } ![\text{“verify”}] = \text{“verify2”}]$
$\land \text{UNCHANGED } \langle msg\_, generated\_hmac, msg\_m, bareMessage,$

$retreivedHMAC$, $generatedHMAC$, $result$, $CompareHMAC$,
$hmacsMatch$, $rxBuf\_$, $incomingMessages\_$, $guid\_$, msg_t,
$msgid\_$, $rx\_$, $rxReg\_$, $last2\_$, $incByte\_$, $rxBuf$, $guid$,
$incomingMessages$, msg_u, $msgid$, rx, $rxReg$, $last2$,
$incMessage$, $incByte$, msg_tr, $txBuf\_$, $txReg\_$, $adder\_$,
$validMessages\_$, msg, $txBuf$, $txReg$, $adder$,
$validMessages\rangle$

$verify2 \triangleq$ $\land\ pc[\text{“verify”}] = \text{“verify2”}$
$\land\ retreivedHMAC' = GetHMAC(msg\_v.text)$
$\land\ pc' = [pc \text{ EXCEPT } ![\text{“verify”}] = \text{“verify3”}]$
$\land$ UNCHANGED $\langle chan$, $msg\_$, $generated\_hmac$, msg_m, msg_v,
$bareMessage$, $generatedHMAC$, $result$, $CompareHMAC$,
$hmacsMatch$, $rxBuf\_$, $incomingMessages\_$, $guid\_$, msg_t,
$msgid\_$, $rx\_$, $rxReg\_$, $last2\_$, $incByte\_$, $rxBuf$, $guid$,
$incomingMessages$, msg_u, $msgid$, rx, $rxReg$, $last2$,
$incMessage$, $incByte$, msg_tr, $txBuf\_$, $txReg\_$, $adder\_$,
$validMessages\_$, msg, $txBuf$, $txReg$, $adder$,
$validMessages\rangle$

$verify3 \triangleq$ $\land\ pc[\text{“verify”}] = \text{“verify3”}$
$\land\ generatedHMAC' = HMAC(msg\_v.text, PASSWORD)$
$\land\ pc' = [pc \text{ EXCEPT } ![\text{“verify”}] = \text{“verify4”}]$
$\land$ UNCHANGED $\langle chan$, $msg\_$, $generated\_hmac$, msg_m, msg_v,
$bareMessage$, $retreivedHMAC$, $result$, $CompareHMAC$,
$hmacsMatch$, $rxBuf\_$, $incomingMessages\_$, $guid\_$, msg_t,
$msgid\_$, $rx\_$, $rxReg\_$, $last2\_$, $incByte\_$, $rxBuf$, $guid$,
$incomingMessages$, msg_u, $msgid$, rx, $rxReg$, $last2$,
$incMessage$, $incByte$, msg_tr, $txBuf\_$, $txReg\_$, $adder\_$,
$validMessages\_$, msg, $txBuf$, $txReg$, $adder$,
$validMessages\rangle$

$verify4 \triangleq$ $\land\ pc[\text{“verify”}] = \text{“verify4”}$
$\land\ hmacsMatch' = CompareHMAC$
$\land$ IF $hmacsMatch'$
THEN $\land\ pc' = [pc \text{ EXCEPT } ![\text{“verify”}] = \text{“verify5”}]$
ELSE $\land\ pc' = [pc \text{ EXCEPT } ![\text{“verify”}] = \text{“verify6”}]$
$\land$ UNCHANGED $\langle chan$, $msg\_$, $generated\_hmac$, msg_m, msg_v,
$bareMessage$, $retreivedHMAC$, $generatedHMAC$, $result$,
$CompareHMAC$, $rxBuf\_$, $incomingMessages\_$, $guid\_$,
msg_t, $msgid\_$, $rx\_$, $rxReg\_$, $last2\_$, $incByte\_$, $rxBuf$,
$guid$, $incomingMessages$, msg_u, $msgid$, rx, $rxReg$,
$last2$, $incMessage$, $incByte$, msg_tr, $txBuf\_$, $txReg\_$,
$adder\_$, $validMessages\_$, msg, $txBuf$, $txReg$, $adder$,
$validMessages\rangle$

$verify5 \triangleq \land pc[\text{“verify”}] = \text{“verify5”}$
$\land chan' = [chan \text{ EXCEPT } ![\text{“trustnet\_out”}] = Append(chan[\text{“trustnet\_out”}], ([id \mapsto msg\_v.id, isVal$
$\land pc' = [pc \text{ EXCEPT } ![\text{“verify”}] = \text{“verify1”}]$
$\land \text{UNCHANGED } \langle msg\_, generated\_hmac, msg\_m, msg\_v, bareMessage,$
$retreivedHMAC, generatedHMAC, result, CompareHMAC,$
$hmacsMatch, rxBuf\_, incomingMessages\_, guid\_, msg\_t,$
$msgid\_, rx\_, rxReg\_, last2\_, incByte\_, rxBuf, guid,$
$incomingMessages, msg\_u, msgid, rx, rxReg, last2,$
$incMessage, incByte, msg\_tr, txBuf\_, txReg\_, adder\_,$
$validMessages\_, msg, txBuf, txReg, adder,$
$validMessages\rangle$

$verify6 \triangleq \land pc[\text{“verify”}] = \text{“verify6”}$
$\land chan' = [chan \text{ EXCEPT } ![\text{“trustnet\_out”}] = Append(chan[\text{“trustnet\_out”}], ([id \mapsto msg\_v.id, isVal$
$\land pc' = [pc \text{ EXCEPT } ![\text{“verify”}] = \text{“verify1”}]$
$\land \text{UNCHANGED } \langle msg\_, generated\_hmac, msg\_m, msg\_v, bareMessage,$
$retreivedHMAC, generatedHMAC, result, CompareHMAC,$
$hmacsMatch, rxBuf\_, incomingMessages\_, guid\_, msg\_t,$
$msgid\_, rx\_, rxReg\_, last2\_, incByte\_, rxBuf, guid,$
$incomingMessages, msg\_u, msgid, rx, rxReg, last2,$
$incMessage, incByte, msg\_tr, txBuf\_, txReg\_, adder\_,$
$validMessages\_, msg, txBuf, txReg, adder,$
$validMessages\rangle$

$verify \triangleq verify1 \lor verify2 \lor verify3 \lor verify4 \lor verify5 \lor verify6$

$trustnet\_in1 \triangleq \land pc[\text{“trustnet\_in”}] = \text{“trustnet\_in1”}$
$\land \text{IF } Len(incomingMessages\_) > 0$
$\text{THEN } \land \text{IF } Len(incomingMessages\_) > 1$
$\text{THEN } \land pc' = [pc \text{ EXCEPT } ![\text{“trustnet\_in”}] = \text{“uti1\_”}]$
$\text{ELSE } \land pc' = [pc \text{ EXCEPT } ![\text{“trustnet\_in”}] = \text{“uti3\_”}]$
$\text{ELSE } \land pc' = [pc \text{ EXCEPT } ![\text{“trustnet\_in”}] = \text{“Done”}]$
$\land \text{UNCHANGED } \langle chan, msg\_, generated\_hmac, msg\_m, msg\_v,$
$bareMessage, retreivedHMAC, generatedHMAC,$
$result, CompareHMAC, hmacsMatch, rxBuf\_,$
$incomingMessages\_, guid\_, msg\_t, msgid\_, rx\_,$
$rxReg\_, last2\_, incByte\_, rxBuf, guid,$
$incomingMessages, msg\_u, msgid, rx, rxReg,$
$last2, incMessage, incByte, msg\_tr, txBuf\_,$
$txReg\_, adder\_, validMessages\_, msg, txBuf,$
$txReg, adder, validMessages\rangle$

$start\_ \triangleq \land pc[\text{“trustnet\_in”}] = \text{“start\_”}$
$\land \text{IF } Len(msg\_t) > 0$
$\text{THEN } \land \text{IF } Len(rxBuf\_) = MAXMODBUSSIZE$
$\text{THEN } \land rxBuf\_' = \langle\rangle$

$\land\ rxReg\_' = \langle\rangle$
$\land\ incByte\_' = \langle\rangle$
$\land\ incMessage' = \langle\rangle$
$\land\ pc' = [pc \text{ EXCEPT } ![\text{“trustnet\_in”}] = \text{“start\_”}]$
ELSE $\land\ pc' = [pc \text{ EXCEPT } ![\text{“trustnet\_in”}] = \text{“inc\_”}]$
$\land$ UNCHANGED $\langle rxBuf\_, rxReg\_, incByte\_, incMessage\rangle$
ELSE $\land\ pc' = [pc \text{ EXCEPT } ![\text{“trustnet\_in”}] = \text{“trustnet\_in1”}]$
$\land$ UNCHANGED $\langle rxBuf\_, rxReg\_, incByte\_, incMessage\rangle$
$\land$ UNCHANGED $\langle chan, msg\_, generated\_hmac, msg\_m, msg\_v, bareMessage, retreivedHMAC, generatedHMAC, result, CompareHMAC, hmacsMatch, incomingMessages\_, guid\_, msg\_t, msgid\_, rx\_, last2\_, rxBuf, guid, incomingMessages, msg\_u, msgid, rx, rxReg, last2, incByte, msg\_tr, txBuf\_, txReg\_, adder\_, validMessages\_, msg, txBuf, txReg, adder, validMessages\rangle$

$inc\_ \triangleq$ $\land\ pc[\text{“trustnet\_in”}] = \text{“inc\_”}$
$\land\ incByte\_' = \langle Head(msg\_t)\rangle$
$\land\ msg\_t' = Tail(msg\_t)$
$\land$ IF $incByte\_' = \langle\rangle$
THEN $\land$ UNCHANGED $rxReg\_$
ELSE $\land\ rxReg\_' = incByte\_'$
$\land\ pc' = [pc \text{ EXCEPT } ![\text{“trustnet\_in”}] = \text{“receive\_”}]$
$\land$ UNCHANGED $\langle chan, msg\_, generated\_hmac, msg\_m, msg\_v, bareMessage, retreivedHMAC, generatedHMAC, result, CompareHMAC, hmacsMatch, rxBuf\_, incomingMessages\_, guid\_, msgid\_, rx\_, last2\_, rxBuf, guid, incomingMessages, msg\_u, msgid, rx, rxReg, last2, incMessage, incByte, msg\_tr, txBuf\_, txReg\_, adder\_, validMessages\_, msg, txBuf, txReg, adder, validMessages\rangle$

$receive\_ \triangleq$ $\land\ pc[\text{“trustnet\_in”}] = \text{“receive\_”}$
$\land$ IF $rxReg\_ = StrTupleToNumTuple(\langle\text{“:”}\rangle)$
THEN $\land\ rxBuf\_' = \langle\rangle$
ELSE $\land$ TRUE
$\land$ UNCHANGED $rxBuf\_$
$\land\ pc' = [pc \text{ EXCEPT } ![\text{“trustnet\_in”}] = \text{“r0\_”}]$
$\land$ UNCHANGED $\langle chan, msg\_, generated\_hmac, msg\_m, msg\_v, bareMessage, retreivedHMAC, generatedHMAC, result, CompareHMAC, hmacsMatch, incomingMessages\_, guid\_, msg\_t, msgid\_, rx\_, rxReg\_, last2\_, incByte\_, rxBuf, guid, incomingMessages, msg\_u, msgid, rx, rxReg, last2, incMessage, incByte, msg\_tr, txBuf\_,$

$txReg\_$, $adder\_$, $validMessages\_$, msg, $txBuf$, $txReg$,
$adder$, $validMessages\rangle$

$r0\_ \triangleq$ $\land\ pc[\text{“trustnet\_in”}] = \text{“r0\_”}$
$\land\ last2\_' = Tail(last2\_ \circ rxReg\_)$
$\land\ pc' = [pc \text{ EXCEPT } ![\text{“trustnet\_in”}] = \text{“r1\_”}]$
$\land$ UNCHANGED $\langle chan, msg\_, generated\_hmac, msg\_m, msg\_v, bareMessage,$
$retreivedHMAC, generatedHMAC, result, CompareHMAC,$
$hmacsMatch, rxBuf\_, incomingMessages\_, guid\_, msg\_t,$
$msgid\_, rx\_, rxReg\_, incByte\_, rxBuf, guid,$
$incomingMessages, msg\_u, msgid, rx, rxReg, last2,$
$incMessage, incByte, msg\_tr, txBuf\_, txReg\_, adder\_,$
$validMessages\_, msg, txBuf, txReg, adder, validMessages\rangle$

$r1\_ \triangleq$ $\land\ pc[\text{“trustnet\_in”}] = \text{“r1\_”}$
$\land\ rxBuf\_' = rxBuf\_ \circ rxReg\_$
$\land\ pc' = [pc \text{ EXCEPT } ![\text{“trustnet\_in”}] = \text{“r2\_”}]$
$\land$ UNCHANGED $\langle chan, msg\_, generated\_hmac, msg\_m, msg\_v, bareMessage,$
$retreivedHMAC, generatedHMAC, result, CompareHMAC,$
$hmacsMatch, incomingMessages\_, guid\_, msg\_t, msgid\_,$
$rx\_, rxReg\_, last2\_, incByte\_, rxBuf, guid,$
$incomingMessages, msg\_u, msgid, rx, rxReg, last2,$
$incMessage, incByte, msg\_tr, txBuf\_, txReg\_, adder\_,$
$validMessages\_, msg, txBuf, txReg, adder, validMessages\rangle$

$r2\_ \triangleq$ $\land\ pc[\text{“trustnet\_in”}] = \text{“r2\_”}$
$\land\ rxReg\_' = \langle\rangle$
$\land\ pc' = [pc \text{ EXCEPT } ![\text{“trustnet\_in”}] = \text{“check\_”}]$
$\land$ UNCHANGED $\langle chan, msg\_, generated\_hmac, msg\_m, msg\_v, bareMessage,$
$retreivedHMAC, generatedHMAC, result, CompareHMAC,$
$hmacsMatch, rxBuf\_, incomingMessages\_, guid\_, msg\_t,$
$msgid\_, rx\_, last2\_, incByte\_, rxBuf, guid,$
$incomingMessages, msg\_u, msgid, rx, rxReg, last2,$
$incMessage, incByte, msg\_tr, txBuf\_, txReg\_, adder\_,$
$validMessages\_, msg, txBuf, txReg, adder, validMessages\rangle$

$check\_ \triangleq$ $\land\ pc[\text{“trustnet\_in”}] = \text{“check\_”}$
$\land$ IF $NumTupleToStrTuple(last2\_) = \langle \text{“\textbackslash r”}, \text{“\textbackslash n”} \rangle$
THEN $\land\ pc' = [pc \text{ EXCEPT } ![\text{“trustnet\_in”}] = \text{“check0\_”}]$
ELSE $\land\ pc' = [pc \text{ EXCEPT } ![\text{“trustnet\_in”}] = \text{“start\_”}]$
$\land$ UNCHANGED $\langle chan, msg\_, generated\_hmac, msg\_m, msg\_v,$
$bareMessage, retreivedHMAC, generatedHMAC, result,$
$CompareHMAC, hmacsMatch, rxBuf\_, incomingMessages\_,$
$guid\_, msg\_t, msgid\_, rx\_, rxReg\_, last2\_, incByte\_,$
$rxBuf, guid, incomingMessages, msg\_u, msgid, rx,$
$rxReg, last2, incMessage, incByte, msg\_tr, txBuf\_,$

$txReg\_$, $adder\_$, $validMessages\_$, msg, $txBuf$, $txReg$,
$adder$, $validMessages\rangle$

$check0\_ \triangleq$ $\land\ pc[\text{“trustnet\_in”}] = \text{“check0\_”}$
$\land\ msgid\_' = \langle guid\_[1]\rangle \circ \langle \text{“t”}, \text{“n”}, \text{“i”}\rangle$
$\land\ pc' = [pc \text{ EXCEPT } ![\text{“trustnet\_in”}] = \text{“check1\_”}]$
$\land$ UNCHANGED $\langle chan, msg\_, generated\_hmac, msg\_m, msg\_v,$
$bareMessage, retreivedHMAC, generatedHMAC, result,$
$CompareHMAC, hmacsMatch, rxBuf\_, incomingMessages\_,$
$guid\_, msg\_t, rx\_, rxReg\_, last2\_, incByte\_, rxBuf,$
$guid, incomingMessages, msg\_u, msgid, rx, rxReg,$
$last2, incMessage, incByte, msg\_tr, txBuf\_, txReg\_,$
$adder\_, validMessages\_, msg, txBuf, txReg, adder,$
$validMessages\rangle$

$check1\_ \triangleq$ $\land\ pc[\text{“trustnet\_in”}] = \text{“check1\_”}$
$\land\ guid\_' = [guid\_ \text{ EXCEPT } ![1] = guid\_[1] + 1]$
$\land\ pc' = [pc \text{ EXCEPT } ![\text{“trustnet\_in”}] = \text{“check2\_”}]$
$\land$ UNCHANGED $\langle chan, msg\_, generated\_hmac, msg\_m, msg\_v,$
$bareMessage, retreivedHMAC, generatedHMAC, result,$
$CompareHMAC, hmacsMatch, rxBuf\_, incomingMessages\_,$
$msg\_t, msgid\_, rx\_, rxReg\_, last2\_, incByte\_, rxBuf,$
$guid, incomingMessages, msg\_u, msgid, rx, rxReg,$
$last2, incMessage, incByte, msg\_tr, txBuf\_, txReg\_,$
$adder\_, validMessages\_, msg, txBuf, txReg, adder,$
$validMessages\rangle$

$check2\_ \triangleq$ $\land\ pc[\text{“trustnet\_in”}] = \text{“check2\_”}$
$\land\ chan' = [chan \text{ EXCEPT } ![\text{“messagecheck”}] = Append(chan[\text{“messagecheck”}], ([id \mapsto msgid\_, text$
$\land\ pc' = [pc \text{ EXCEPT } ![\text{“trustnet\_in”}] = \text{“check3\_”}]$
$\land$ UNCHANGED $\langle msg\_, generated\_hmac, msg\_m, msg\_v, bareMessage,$
$retreivedHMAC, generatedHMAC, result, CompareHMAC,$
$hmacsMatch, rxBuf\_, incomingMessages\_, guid\_, msg\_t,$
$msgid\_, rx\_, rxReg\_, last2\_, incByte\_, rxBuf, guid,$
$incomingMessages, msg\_u, msgid, rx, rxReg, last2,$
$incMessage, incByte, msg\_tr, txBuf\_, txReg\_, adder\_,$
$validMessages\_, msg, txBuf, txReg, adder,$
$validMessages\rangle$

$check3\_ \triangleq$ $\land\ pc[\text{“trustnet\_in”}] = \text{“check3\_”}$
$\land\ chan' = [chan \text{ EXCEPT } ![\text{“sign”}] = Append(chan[\text{“sign”}], ([id \mapsto msgid\_, text \mapsto rxBuf\_]))]$
$\land\ pc' = [pc \text{ EXCEPT } ![\text{“trustnet\_in”}] = \text{“check4\_”}]$
$\land$ UNCHANGED $\langle msg\_, generated\_hmac, msg\_m, msg\_v, bareMessage,$
$retreivedHMAC, generatedHMAC, result, CompareHMAC,$
$hmacsMatch, rxBuf\_, incomingMessages\_, guid\_, msg\_t,$
$msgid\_, rx\_, rxReg\_, last2\_, incByte\_, rxBuf, guid,$

$incomingMessages$, msg_u, $msgid$, rx, $rxReg$, $last2$,
$incMessage$, $incByte$, msg_tr, $txBuf\_$, $txReg\_$, $adder\_$,
$validMessages\_$, msg, $txBuf$, $txReg$, $adder$,
$validMessages\rangle$

$check4\_ \triangleq$ $\land\ pc[\text{“trustnet\_in”}] = \text{“check4\_”}$
$\land\ rxBuf\_' = \langle\rangle$
$\land\ rxReg\_' = \langle\rangle$
$\land\ incByte\_' = \langle\rangle$
$\land\ incMessage' = \langle\rangle$
$\land\ pc' = [pc \text{ EXCEPT } ![\text{“trustnet\_in”}] = \text{“start\_”}]$
$\land$ UNCHANGED $\langle chan$, $msg\_$, $generated\_hmac$, msg_m, msg_v,
$bareMessage$, $retreivedHMAC$, $generatedHMAC$, $result$,
$CompareHMAC$, $hmacsMatch$, $incomingMessages\_$, $guid\_$,
msg_t, $msgid\_$, $rx\_$, $last2\_$, $rxBuf$, $guid$,
$incomingMessages$, msg_u, $msgid$, rx, $rxReg$, $last2$,
$incByte$, msg_tr, $txBuf\_$, $txReg\_$, $adder\_$,
$validMessages\_$, msg, $txBuf$, $txReg$, $adder$,
$validMessages\rangle$

$uti1\_ \triangleq$ $\land\ pc[\text{“trustnet\_in”}] = \text{“uti1\_”}$
$\land\ msg\_t' = Head(incomingMessages\_)$
$\land\ pc' = [pc \text{ EXCEPT } ![\text{“trustnet\_in”}] = \text{“uti2\_”}]$
$\land$ UNCHANGED $\langle chan$, $msg\_$, $generated\_hmac$, msg_m, msg_v, $bareMessage$,
$retreivedHMAC$, $generatedHMAC$, $result$, $CompareHMAC$,
$hmacsMatch$, $rxBuf\_$, $incomingMessages\_$, $guid\_$, $msgid\_$,
$rx\_$, $rxReg\_$, $last2\_$, $incByte\_$, $rxBuf$, $guid$,
$incomingMessages$, msg_u, $msgid$, rx, $rxReg$, $last2$,
$incMessage$, $incByte$, msg_tr, $txBuf\_$, $txReg\_$, $adder\_$,
$validMessages\_$, msg, $txBuf$, $txReg$, $adder$,
$validMessages\rangle$

$uti2\_ \triangleq$ $\land\ pc[\text{“trustnet\_in”}] = \text{“uti2\_”}$
$\land\ incomingMessages\_' = Tail(incomingMessages\_)$
$\land\ pc' = [pc \text{ EXCEPT } ![\text{“trustnet\_in”}] = \text{“start\_”}]$
$\land$ UNCHANGED $\langle chan$, $msg\_$, $generated\_hmac$, msg_m, msg_v, $bareMessage$,
$retreivedHMAC$, $generatedHMAC$, $result$, $CompareHMAC$,
$hmacsMatch$, $rxBuf\_$, $guid\_$, msg_t, $msgid\_$, $rx\_$, $rxReg\_$,
$last2\_$, $incByte\_$, $rxBuf$, $guid$, $incomingMessages$,
msg_u, $msgid$, rx, $rxReg$, $last2$, $incMessage$, $incByte$,
msg_tr, $txBuf\_$, $txReg\_$, $adder\_$, $validMessages\_$, msg,
$txBuf$, $txReg$, $adder$, $validMessages\rangle$

$uti3\_ \triangleq$ $\land\ pc[\text{“trustnet\_in”}] = \text{“uti3\_”}$
$\land\ msg\_t' = incomingMessages\_[1]$
$\land\ pc' = [pc \text{ EXCEPT } ![\text{“trustnet\_in”}] = \text{“uti4\_”}]$

$\land$ UNCHANGED $\langle chan,\ msg\_,\ generated\_hmac,\ msg\_m,\ msg\_v,\ bareMessage,$
$retreivedHMAC,\ generatedHMAC,\ result,\ CompareHMAC,$
$hmacsMatch,\ rxBuf\_,\ incomingMessages\_,\ guid\_,\ msgid\_,$
$rx\_,\ rxReg\_,\ last2\_,\ incByte\_,\ rxBuf,\ guid,$
$incomingMessages,\ msg\_u,\ msgid,\ rx,\ rxReg,\ last2,$
$incMessage,\ incByte,\ msg\_tr,\ txBuf\_,\ txReg\_,\ adder\_,$
$validMessages\_,\ msg,\ txBuf,\ txReg,\ adder,$
$validMessages\rangle$

$uti4\_ \triangleq \land pc[\text{“trustnet\_in”}] = \text{“uti4\_”}$
$\land\ incomingMessages\_' = \langle\rangle$
$\land\ pc' = [pc \text{ EXCEPT } ![\text{“trustnet\_in”}] = \text{“start\_”}]$
$\land$ UNCHANGED $\langle chan,\ msg\_,\ generated\_hmac,\ msg\_m,\ msg\_v,\ bareMessage,$
$retreivedHMAC,\ generatedHMAC,\ result,\ CompareHMAC,$
$hmacsMatch,\ rxBuf\_,\ guid\_,\ msg\_t,\ msgid\_,\ rx\_,\ rxReg\_,$
$last2\_,\ incByte\_,\ rxBuf,\ guid,\ incomingMessages,$
$msg\_u,\ msgid,\ rx,\ rxReg,\ last2,\ incMessage,\ incByte,$
$msg\_tr,\ txBuf\_,\ txReg\_,\ adder\_,\ validMessages\_,\ msg,$
$txBuf,\ txReg,\ adder,\ validMessages\rangle$

$trustnet\_in \triangleq trustnet\_in1 \lor start\_ \lor inc\_ \lor receive\_ \lor r0\_ \lor r1\_$
$\lor r2\_ \lor check\_ \lor check0\_ \lor check1\_ \lor check2\_$
$\lor check3\_ \lor check4\_ \lor uti1\_ \lor uti2\_ \lor uti3\_ \lor uti4\_$

$untrustnet\_in1 \triangleq \land pc[\text{“untrustnet\_in”}] = \text{“untrustnet\_in1”}$
$\land$ IF $Len(incomingMessages) > 0$
THEN $\land$ IF $Len(incomingMessages) > 1$
THEN $\land\ pc' = [pc \text{ EXCEPT } ![\text{“untrustnet\_in”}] = \text{“uti1”}]$
ELSE $\land\ pc' = [pc \text{ EXCEPT } ![\text{“untrustnet\_in”}] = \text{“uti3”}]$
ELSE $\land\ pc' = [pc \text{ EXCEPT } ![\text{“untrustnet\_in”}] = \text{“Done”}]$
$\land$ UNCHANGED $\langle chan,\ msg\_,\ generated\_hmac,\ msg\_m,\ msg\_v,$
$bareMessage,\ retreivedHMAC,\ generatedHMAC,$
$result,\ CompareHMAC,\ hmacsMatch,\ rxBuf\_,$
$incomingMessages\_,\ guid\_,\ msg\_t,\ msgid\_,\ rx\_,$
$rxReg\_,\ last2\_,\ incByte\_,\ rxBuf,\ guid,$
$incomingMessages,\ msg\_u,\ msgid,\ rx,\ rxReg,$
$last2,\ incMessage,\ incByte,\ msg\_tr,\ txBuf\_,$
$txReg\_,\ adder\_,\ validMessages\_,\ msg,\ txBuf,$
$txReg,\ adder,\ validMessages\rangle$

$start \triangleq \land pc[\text{“untrustnet\_in”}] = \text{“start”}$
$\land$ IF $Len(msg\_u) > 0$
THEN $\land\ pc' = [pc \text{ EXCEPT } ![\text{“untrustnet\_in”}] = \text{“inc”}]$
ELSE $\land\ pc' = [pc \text{ EXCEPT } ![\text{“untrustnet\_in”}] = \text{“untrustnet\_in1”}]$
$\land$ UNCHANGED $\langle chan,\ msg\_,\ generated\_hmac,\ msg\_m,\ msg\_v,\ bareMessage,$
$retreivedHMAC,\ generatedHMAC,\ result,\ CompareHMAC,$

$hmacsMatch$, $rxBuf\_$, $incomingMessages\_$, $guid\_$, msg_t, $msgid\_$, $rx\_$, $rxReg\_$, $last2\_$, $incByte\_$, $rxBuf$, $guid$, $incomingMessages$, msg_u, $msgid$, rx, $rxReg$, $last2$, $incMessage$, $incByte$, msg_tr, $txBuf\_$, $txReg\_$, $adder\_$, $validMessages\_$, msg, $txBuf$, $txReg$, $adder$, $validMessages\rangle$

$inc \triangleq \land pc[\text{“untrustnet\_in”}] = \text{“inc”}$
$\land incByte' = \langle Head(msg\_u)\rangle$
$\land incMessage' = Tail(msg\_u)$
$\land$ IF $incByte' = \langle\rangle$
THEN $\land rxReg' = rxReg$
ELSE $\land rxReg' = incByte'$
$\land pc' = [pc$ EXCEPT $![\text{“untrustnet\_in”}] = \text{“receive”}]$
$\land$ UNCHANGED $\langle chan, msg\_, generated\_hmac, msg\_m, msg\_v, bareMessage, retreivedHMAC, generatedHMAC, result, CompareHMAC, hmacsMatch, rxBuf\_, incomingMessages\_, guid\_, msg\_t, msgid\_, rx\_, rxReg\_, last2\_, incByte\_, rxBuf, guid, incomingMessages, msg\_u, msgid, rx, last2, msg\_tr, txBuf\_, txReg\_, adder\_, validMessages\_, msg, txBuf, txReg, adder, validMessages\rangle$

$receive \triangleq \land pc[\text{“untrustnet\_in”}] = \text{“receive”}$
$\land$ IF $rxReg = StrTupleToNumTuple(\langle\text{“!”}\rangle)$
THEN $\land rxBuf' = \langle\rangle$
ELSE $\land$ TRUE
$\land rxBuf' = rxBuf$
$\land pc' = [pc$ EXCEPT $![\text{“untrustnet\_in”}] = \text{“r0”}]$
$\land$ UNCHANGED $\langle chan, msg\_, generated\_hmac, msg\_m, msg\_v, bareMessage, retreivedHMAC, generatedHMAC, result, CompareHMAC, hmacsMatch, rxBuf\_, incomingMessages\_, guid\_, msg\_t, msgid\_, rx\_, rxReg\_, last2\_, incByte\_, guid, incomingMessages, msg\_u, msgid, rx, rxReg, last2, incMessage, incByte, msg\_tr, txBuf\_, txReg\_, adder\_, validMessages\_, msg, txBuf, txReg, adder, validMessages\rangle$

$r0 \triangleq \land pc[\text{“untrustnet\_in”}] = \text{“r0”}$
$\land last2' = Tail(last2 \circ rxReg)$
$\land pc' = [pc$ EXCEPT $![\text{“untrustnet\_in”}] = \text{“r1”}]$
$\land$ UNCHANGED $\langle chan, msg\_, generated\_hmac, msg\_m, msg\_v, bareMessage, retreivedHMAC, generatedHMAC, result, CompareHMAC, hmacsMatch, rxBuf\_, incomingMessages\_, guid\_, msg\_t, msgid\_, rx\_, rxReg\_, last2\_, incByte\_, rxBuf, guid, incomingMessages, msg\_u, msgid, rx, rxReg, incMessage, incByte, msg\_tr, txBuf\_, txReg\_, adder\_, validMessages\_,$

msg, $txBuf$, $txReg$, $adder$, $validMessages\rangle$

$r1 \triangleq$ $\land\ pc[\text{“untrustnet\_in”}] = \text{“r1”}$
$\land\ rxBuf' = rxBuf \circ rxReg$
$\land\ pc' = [pc \text{ EXCEPT } ![\text{“untrustnet\_in”}] = \text{“r2”}]$
$\land$ UNCHANGED $\langle chan, msg\_, generated\_hmac, msg\_m, msg\_v, bareMessage, retreivedHMAC, generatedHMAC, result, CompareHMAC, hmacsMatch, rxBuf\_, incomingMessages\_, guid\_, msg\_t, msgid\_, rx\_, rxReg\_, last2\_, incByte\_, guid, incomingMessages, msg\_u, msgid, rx, rxReg, last2, incMessage, incByte, msg\_tr, txBuf\_, txReg\_, adder\_, validMessages\_, msg, txBuf, txReg, adder, validMessages\rangle$

$r2 \triangleq$ $\land\ pc[\text{“untrustnet\_in”}] = \text{“r2”}$
$\land\ rxReg' = \langle\rangle$
$\land\ pc' = [pc \text{ EXCEPT } ![\text{“untrustnet\_in”}] = \text{“check”}]$
$\land$ UNCHANGED $\langle chan, msg\_, generated\_hmac, msg\_m, msg\_v, bareMessage, retreivedHMAC, generatedHMAC, result, CompareHMAC, hmacsMatch, rxBuf\_, incomingMessages\_, guid\_, msg\_t, msgid\_, rx\_, rxReg\_, last2\_, incByte\_, rxBuf, guid, incomingMessages, msg\_u, msgid, rx, last2, incMessage, incByte, msg\_tr, txBuf\_, txReg\_, adder\_, validMessages\_, msg, txBuf, txReg, adder, validMessages\rangle$

$check \triangleq$ $\land\ pc[\text{“untrustnet\_in”}] = \text{“check”}$
$\land$ IF $NumTupleToStrTuple(last2) = \langle \text{“\textbackslash r”}, \text{“\textbackslash n”} \rangle$
THEN $\land\ pc' = [pc \text{ EXCEPT } ![\text{“untrustnet\_in”}] = \text{“check0”}]$
ELSE $\land\ pc' = [pc \text{ EXCEPT } ![\text{“untrustnet\_in”}] = \text{“start”}]$
$\land$ UNCHANGED $\langle chan, msg\_, generated\_hmac, msg\_m, msg\_v, bareMessage, retreivedHMAC, generatedHMAC, result, CompareHMAC, hmacsMatch, rxBuf\_, incomingMessages\_, guid\_, msg\_t, msgid\_, rx\_, rxReg\_, last2\_, incByte\_, rxBuf, guid, incomingMessages, msg\_u, msgid, rx, rxReg, last2, incMessage, incByte, msg\_tr, txBuf\_, txReg\_, adder\_, validMessages\_, msg, txBuf, txReg, adder, validMessages\rangle$

$check0 \triangleq$ $\land\ pc[\text{“untrustnet\_in”}] = \text{“check0”}$
$\land\ msgid' = guid \circ \text{“untrustnet\_in”}$
$\land\ pc' = [pc \text{ EXCEPT } ![\text{“untrustnet\_in”}] = \text{“check1”}]$
$\land$ UNCHANGED $\langle chan, msg\_, generated\_hmac, msg\_m, msg\_v, bareMessage, retreivedHMAC, generatedHMAC, result, CompareHMAC, hmacsMatch, rxBuf\_, incomingMessages\_, guid\_, msg\_t, msgid\_, rx\_, rxReg\_, last2\_, incByte\_, rxBuf, guid, incomingMessages, msg\_u, rx, rxReg, last2, incMessage, incByte, msg\_tr, txBuf\_, txReg\_,$

$adder\_$, $validMessages\_$, msg, $txBuf$, $txReg$, $adder$,
$validMessages\rangle$

$check1 \triangleq \land pc[\text{“untrustnet\_in”}] = \text{“check1”}$
$\land guid' = guid + 1$
$\land pc' = [pc \text{ EXCEPT } ![\text{“untrustnet\_in”}] = \text{“check2”}]$
$\land \text{UNCHANGED } \langle chan, msg\_, generated\_hmac, msg\_m, msg\_v,$
$bareMessage, retreivedHMAC, generatedHMAC, result,$
$CompareHMAC, hmacsMatch, rxBuf\_, incomingMessages\_,$
$guid\_, msg\_t, msgid\_, rx\_, rxReg\_, last2\_, incByte\_,$
$rxBuf, incomingMessages, msg\_u, msgid, rx, rxReg,$
$last2, incMessage, incByte, msg\_tr, txBuf\_, txReg\_,$
$adder\_, validMessages\_, msg, txBuf, txReg, adder,$
$validMessages\rangle$

$check2 \triangleq \land pc[\text{“untrustnet\_in”}] = \text{“check2”}$
$\land chan' = [chan \text{ EXCEPT } ![\text{“messagecheck”}] = Append(chan[\text{“messagecheck”}], ([id \mapsto msgid, text \mapsto$
$\land pc' = [pc \text{ EXCEPT } ![\text{“untrustnet\_in”}] = \text{“check3”}]$
$\land \text{UNCHANGED } \langle msg\_, generated\_hmac, msg\_m, msg\_v, bareMessage,$
$retreivedHMAC, generatedHMAC, result, CompareHMAC,$
$hmacsMatch, rxBuf\_, incomingMessages\_, guid\_, msg\_t,$
$msgid\_, rx\_, rxReg\_, last2\_, incByte\_, rxBuf, guid,$
$incomingMessages, msg\_u, msgid, rx, rxReg, last2,$
$incMessage, incByte, msg\_tr, txBuf\_, txReg\_, adder\_,$
$validMessages\_, msg, txBuf, txReg, adder,$
$validMessages\rangle$

$check3 \triangleq \land pc[\text{“untrustnet\_in”}] = \text{“check3”}$
$\land chan' = [chan \text{ EXCEPT } ![\text{“verify”}] = Append(chan[\text{“verify”}], ([id \mapsto msgid, text \mapsto rxBuf]))]$
$\land pc' = [pc \text{ EXCEPT } ![\text{“untrustnet\_in”}] = \text{“check4”}]$
$\land \text{UNCHANGED } \langle msg\_, generated\_hmac, msg\_m, msg\_v, bareMessage,$
$retreivedHMAC, generatedHMAC, result, CompareHMAC,$
$hmacsMatch, rxBuf\_, incomingMessages\_, guid\_, msg\_t,$
$msgid\_, rx\_, rxReg\_, last2\_, incByte\_, rxBuf, guid,$
$incomingMessages, msg\_u, msgid, rx, rxReg, last2,$
$incMessage, incByte, msg\_tr, txBuf\_, txReg\_, adder\_,$
$validMessages\_, msg, txBuf, txReg, adder,$
$validMessages\rangle$

$check4 \triangleq \land pc[\text{“untrustnet\_in”}] = \text{“check4”}$
$\land rxBuf' = \langle\rangle$
$\land rxReg' = \langle\rangle$
$\land incByte' = \langle\rangle$
$\land incMessage' = \langle\rangle$
$\land pc' = [pc \text{ EXCEPT } ![\text{“untrustnet\_in”}] = \text{“start”}]$
$\land \text{UNCHANGED } \langle chan, msg\_, generated\_hmac, msg\_m, msg\_v,$

$bareMessage$, $retreivedHMAC$, $generatedHMAC$, $result$, $CompareHMAC$, $hmacsMatch$, $rxBuf\_$, $incomingMessages\_$, $guid\_$, msg_t, $msgid\_$, $rx\_$, $rxReg\_$, $last2\_$, $incByte\_$, $guid$, $incomingMessages$, msg_u, $msgid$, rx, $last2$, msg_tr, $txBuf\_$, $txReg\_$, $adder\_$, $validMessages\_$, msg, $txBuf$, $txReg$, $adder$, $validMessages\rangle$

$uti1 \triangleq \land pc[\text{“untrustnet\_in”}] = \text{“uti1”}$
$\land msg\_u' = Head(incomingMessages)$
$\land pc' = [pc \text{ EXCEPT } ![\text{“untrustnet\_in”}] = \text{“uti2”}]$
$\land \text{UNCHANGED } \langle chan, msg\_, generated\_hmac, msg\_m, msg\_v, bareMessage, retreivedHMAC, generatedHMAC, result, CompareHMAC, hmacsMatch, rxBuf\_, incomingMessages\_, guid\_, msg\_t, msgid\_, rx\_, rxReg\_, last2\_, incByte\_, rxBuf, guid, incomingMessages, msgid, rx, rxReg, last2, incMessage, incByte, msg\_tr, txBuf\_, txReg\_, adder\_, validMessages\_, msg, txBuf, txReg, adder, validMessages\rangle$

$uti2 \triangleq \land pc[\text{“untrustnet\_in”}] = \text{“uti2”}$
$\land incomingMessages' = Tail(incomingMessages)$
$\land pc' = [pc \text{ EXCEPT } ![\text{“untrustnet\_in”}] = \text{“start”}]$
$\land \text{UNCHANGED } \langle chan, msg\_, generated\_hmac, msg\_m, msg\_v, bareMessage, retreivedHMAC, generatedHMAC, result, CompareHMAC, hmacsMatch, rxBuf\_, incomingMessages\_, guid\_, msg\_t, msgid\_, rx\_, rxReg\_, last2\_, incByte\_, rxBuf, guid, msg\_u, msgid, rx, rxReg, last2, incMessage, incByte, msg\_tr, txBuf\_, txReg\_, adder\_, validMessages\_, msg, txBuf, txReg, adder, validMessages\rangle$

$uti3 \triangleq \land pc[\text{“untrustnet\_in”}] = \text{“uti3”}$
$\land msg\_u' = incomingMessages[1]$
$\land pc' = [pc \text{ EXCEPT } ![\text{“untrustnet\_in”}] = \text{“uti4”}]$
$\land \text{UNCHANGED } \langle chan, msg\_, generated\_hmac, msg\_m, msg\_v, bareMessage, retreivedHMAC, generatedHMAC, result, CompareHMAC, hmacsMatch, rxBuf\_, incomingMessages\_, guid\_, msg\_t, msgid\_, rx\_, rxReg\_, last2\_, incByte\_, rxBuf, guid, incomingMessages, msgid, rx, rxReg, last2, incMessage, incByte, msg\_tr, txBuf\_, txReg\_, adder\_, validMessages\_, msg, txBuf, txReg, adder, validMessages\rangle$

$uti4 \triangleq \land pc[\text{“untrustnet\_in”}] = \text{“uti4”}$
$\land incomingMessages' = \langle\rangle$
$\land pc' = [pc \text{ EXCEPT } ![\text{“untrustnet\_in”}] = \text{“start”}]$
$\land \text{UNCHANGED } \langle chan, msg\_, generated\_hmac, msg\_m, msg\_v, bareMessage,$

$retreivedHMAC$, $generatedHMAC$, $result$, $CompareHMAC$, $hmacsMatch$, $rxBuf\_$, $incomingMessages\_$, $guid\_$, msg_t, $msgid\_$, $rx\_$, $rxReg\_$, $last2\_$, $incByte\_$, $rxBuf$, $guid$, msg_u, $msgid$, rx, $rxReg$, $last2$, $incMessage$, $incByte$, msg_tr, $txBuf\_$, $txReg\_$, $adder\_$, $validMessages\_$, msg, $txBuf$, $txReg$, $adder$, $validMessages\rangle$

$untrustnet\_in \triangleq untrustnet\_in1 \lor start \lor inc \lor receive \lor r0 \lor r1 \lor r2 \lor check \lor check0 \lor check1 \lor check2 \lor check3 \lor check4 \lor uti1 \lor uti2 \lor uti3 \lor uti4$

$$
\begin{array}{l}
to1 \triangleq \land pc[\text{“trustnet\_out”}] = \text{“to1”} \\
\quad \land Len(chan[\text{“trustnet\_out”}]) > 0 \\
\quad \land msg\_tr' = Head(chan[\text{“trustnet\_out”}]) \\
\quad \land chan' = [chan \text{ EXCEPT } ![\text{“trustnet\_out”}] = Tail(chan[\text{“trustnet\_out”}])] \\
\quad \land \text{IF } msg\_tr'.isValid \\
\qquad \text{THEN } \land \text{IF } \exists x \in validMessages\_ : x.id = msg\_tr'.id \\
\qquad\qquad \text{THEN } \land txBuf\_' = msg\_tr'.text \\
\qquad\qquad\qquad \land pc' = [pc \text{ EXCEPT } ![\text{“trustnet\_out”}] = \text{“transmit\_”}] \\
\qquad\qquad\qquad \land \text{UNCHANGED } validMessages\_ \\
\qquad\qquad \text{ELSE } \land validMessages\_' = (validMessages\_ \cup \{msg\_tr'\}) \\
\qquad\qquad\qquad \land pc' = [pc \text{ EXCEPT } ![\text{“trustnet\_out”}] = \text{“finished\_”}] \\
\qquad\qquad\qquad \land \text{UNCHANGED } txBuf\_ \\
\qquad \text{ELSE } \land pc' = [pc \text{ EXCEPT } ![\text{“trustnet\_out”}] = \text{“finished\_”}] \\
\qquad\qquad \land \text{UNCHANGED } \langle txBuf\_, validMessages\_\rangle \\
\quad \land \text{UNCHANGED } \langle msg\_, generated\_hmac, msg\_m, msg\_v, bareMessage, \\
\qquad\qquad retreivedHMAC, generatedHMAC, result, CompareHMAC, \\
\qquad\qquad hmacsMatch, rxBuf\_, incomingMessages\_, guid\_, msg\_t, \\
\qquad\qquad msgid\_, rx\_, rxReg\_, last2\_, incByte\_, rxBuf, guid, \\
\qquad\qquad incomingMessages, msg\_u, msgid, rx, rxReg, last2, \\
\qquad\qquad incMessage, incByte, txReg\_, adder\_, msg, txBuf, txReg, \\
\qquad\qquad adder, validMessages\rangle
\end{array}
$$

$$
\begin{array}{l}
finished\_ \triangleq \land pc[\text{“trustnet\_out”}] = \text{“finished\_”} \\
\quad \land txReg\_' = \langle\rangle \\
\quad \land txBuf\_' = \langle\rangle \\
\quad \land pc' = [pc \text{ EXCEPT } ![\text{“trustnet\_out”}] = \text{“to1”}] \\
\quad \land \text{UNCHANGED } \langle chan, msg\_, generated\_hmac, msg\_m, msg\_v, \\
\qquad\qquad bareMessage, retreivedHMAC, generatedHMAC, result, \\
\qquad\qquad CompareHMAC, hmacsMatch, rxBuf\_, \\
\qquad\qquad incomingMessages\_, guid\_, msg\_t, msgid\_, rx\_, \\
\qquad\qquad rxReg\_, last2\_, incByte\_, rxBuf, guid, \\
\qquad\qquad incomingMessages, msg\_u, msgid, rx, rxReg, last2, \\
\qquad\qquad incMessage, incByte, msg\_tr, adder\_, \\
\qquad\qquad validMessages\_, msg, txBuf, txReg, adder, \\
\qquad\qquad validMessages\rangle
\end{array}
$$

$transmit\_ \triangleq \land pc[\text{“trustnet\_out”}] = \text{“transmit\_”}$
$\land chan' = [chan \text{ EXCEPT } ![\text{“finished\_trustnet”}] = Append(chan[\text{“finished\_trustnet”}], (NumTupleT$
$\land pc' = [pc \text{ EXCEPT } ![\text{“trustnet\_out”}] = \text{“to2”}]$
$\land \text{UNCHANGED } \langle msg\_, generated\_hmac, msg\_m, msg\_v, bareMessage,$
$retreivedHMAC, generatedHMAC, result, CompareHMAC,$
$hmacsMatch, rxBuf\_, incomingMessages\_, guid\_,$
$msg\_t, msgid\_, rx\_, rxReg\_, last2\_, incByte\_,$
$rxBuf, guid, incomingMessages, msg\_u, msgid, rx,$
$rxReg, last2, incMessage, incByte, msg\_tr, txBuf\_,$
$txReg\_, adder\_, validMessages\_, msg, txBuf, txReg,$
$adder, validMessages\rangle$

$to2 \triangleq \land pc[\text{“trustnet\_out”}] = \text{“to2”}$
$\land validMessages\_' = validMessages\_ \setminus \{x \in validMessages\_ : x.id = msg\_tr.id\}$
$\land pc' = [pc \text{ EXCEPT } ![\text{“trustnet\_out”}] = \text{“finished\_”}]$
$\land \text{UNCHANGED } \langle chan, msg\_, generated\_hmac, msg\_m, msg\_v, bareMessage,$
$retreivedHMAC, generatedHMAC, result, CompareHMAC,$
$hmacsMatch, rxBuf\_, incomingMessages\_, guid\_, msg\_t,$
$msgid\_, rx\_, rxReg\_, last2\_, incByte\_, rxBuf, guid,$
$incomingMessages, msg\_u, msgid, rx, rxReg, last2,$
$incMessage, incByte, msg\_tr, txBuf\_, txReg\_, adder\_,$
$msg, txBuf, txReg, adder, validMessages\rangle$

$trustnet\_out \triangleq to1 \lor finished\_ \lor transmit\_ \lor to2$

$uto1 \triangleq \land pc[\text{“untrustnet\_out”}] = \text{“uto1”}$
$\land Len(chan[\text{“untrustnet\_out”}]) > 0$
$\land msg' = Head(chan[\text{“untrustnet\_out”}])$
$\land chan' = [chan \text{ EXCEPT } ![\text{“untrustnet\_out”}] = Tail(chan[\text{“untrustnet\_out”}])]$
$\land \text{IF } msg'.isValid$
$\text{THEN } \land \text{IF } \exists x \in validMessages : x.id = msg'.id$
$\text{THEN } \land \text{IF } msg'.source = \text{“sign”}$
$\text{THEN } \land txBuf' = StrTupleToNumTuple(\langle\text{“!”}\rangle) \circ StrTupleToNumTuple$
$\text{ELSE } \land txBuf' = StrTupleToNumTuple(\langle\text{“!”}\rangle) \circ StrTupleToNumTuple$
$\land pc' = [pc \text{ EXCEPT } ![\text{“untrustnet\_out”}] = \text{“transmit”}]$
$\land \text{UNCHANGED } validMessages$
$\text{ELSE } \land validMessages' = (validMessages \cup \{msg'\})$
$\land pc' = [pc \text{ EXCEPT } ![\text{“untrustnet\_out”}] = \text{“finished”}]$
$\land txBuf' = txBuf$
$\text{ELSE } \land pc' = [pc \text{ EXCEPT } ![\text{“untrustnet\_out”}] = \text{“finished”}]$
$\land \text{UNCHANGED } \langle txBuf, validMessages\rangle$
$\land \text{UNCHANGED } \langle msg\_, generated\_hmac, msg\_m, msg\_v, bareMessage,$
$retreivedHMAC, generatedHMAC, result, CompareHMAC,$
$hmacsMatch, rxBuf\_, incomingMessages\_, guid\_, msg\_t,$
$msgid\_, rx\_, rxReg\_, last2\_, incByte\_, rxBuf, guid,$
$incomingMessages, msg\_u, msgid, rx, rxReg, last2,$

$incMessage$, $incByte$, msg_tr, $txBuf\_$, $txReg\_$, $adder\_$,
$validMessages\_$, $txReg$, $adder\rangle$

$finished \triangleq \land pc[\text{“untrustnet\_out”}] = \text{“finished”}$
$\land txReg' = \langle\rangle$
$\land txBuf' = \langle\rangle$
$\land pc' = [pc \text{ EXCEPT } ![\text{“untrustnet\_out”}] = \text{“uto1”}]$
$\land \text{UNCHANGED } \langle chan, msg\_, generated\_hmac, msg\_m, msg\_v,$
$bareMessage, retreivedHMAC, generatedHMAC, result,$
$CompareHMAC, hmacsMatch, rxBuf\_, incomingMessages\_,$
$guid\_, msg\_t, msgid\_, rx\_, rxReg\_, last2\_,$
$incByte\_, rxBuf, guid, incomingMessages, msg\_u,$
$msgid, rx, rxReg, last2, incMessage, incByte,$
$msg\_tr, txBuf\_, txReg\_, adder\_, validMessages\_,$
$msg, adder, validMessages\rangle$

$transmit \triangleq \land pc[\text{“untrustnet\_out”}] = \text{“transmit”}$
$\land chan' = [chan \text{ EXCEPT } ![\text{“finished\_untrustnet”}] = Append(chan[\text{“finished\_untrustnet”}], (NumTu$
$\land pc' = [pc \text{ EXCEPT } ![\text{“untrustnet\_out”}] = \text{“uto2”}]$
$\land \text{UNCHANGED } \langle msg\_, generated\_hmac, msg\_m, msg\_v, bareMessage,$
$retreivedHMAC, generatedHMAC, result, CompareHMAC,$
$hmacsMatch, rxBuf\_, incomingMessages\_, guid\_,$
$msg\_t, msgid\_, rx\_, rxReg\_, last2\_, incByte\_,$
$rxBuf, guid, incomingMessages, msg\_u, msgid, rx,$
$rxReg, last2, incMessage, incByte, msg\_tr, txBuf\_,$
$txReg\_, adder\_, validMessages\_, msg, txBuf, txReg,$
$adder, validMessages\rangle$

$uto2 \triangleq \land pc[\text{“untrustnet\_out”}] = \text{“uto2”}$
$\land validMessages' = validMessages \setminus \{x \in validMessages : x.id = msg.id\}$
$\land pc' = [pc \text{ EXCEPT } ![\text{“untrustnet\_out”}] = \text{“finished”}]$
$\land \text{UNCHANGED } \langle chan, msg\_, generated\_hmac, msg\_m, msg\_v, bareMessage,$
$retreivedHMAC, generatedHMAC, result, CompareHMAC,$
$hmacsMatch, rxBuf\_, incomingMessages\_, guid\_, msg\_t,$
$msgid\_, rx\_, rxReg\_, last2\_, incByte\_, rxBuf, guid,$
$incomingMessages, msg\_u, msgid, rx, rxReg, last2,$
$incMessage, incByte, msg\_tr, txBuf\_, txReg\_, adder\_,$
$validMessages\_, msg, txBuf, txReg, adder\rangle$

$untrustnet\_out \triangleq uto1 \lor finished \lor transmit \lor uto2$

$Next \triangleq sign \lor msgchk \lor verify \lor trustnet\_in \lor untrustnet\_in$
$\lor trustnet\_out \lor untrustnet\_out$

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

$\land \mathrm{WF}_{vars}(msgchk)$
$\land \mathrm{WF}_{vars}(verify)$
$\land \mathrm{WF}_{vars}(trustnet\_in)$
$\land \mathrm{WF}_{vars}(untrustnet\_in)$
$\land \mathrm{WF}_{vars}(trustnet\_out)$
$\land \mathrm{WF}_{vars}(untrustnet\_out)$

END TRANSLATION

$SAFETYCHECK \triangleq$

modbus check module:

$\land \forall m \in Range(chan[\text{“messagecheck”}]) : Len(m.text) \leq MAXMODBUSSIZE$ only messages with a valid lengt

messages going to untrusted network

$\land \forall m \in validMessages : m.isValid = \text{TRUE}$ message parts waiting for their counterpart are valid

$\land \forall m \in Range(chan[\text{“finished\_untrustnet”}]) : GetHMAC(m) = HMAC(m,\ m)$ only properly signed messages a

$\land \forall m \in Range(chan[\text{“finished\_untrustnet”}]) : IsModbus(GetMessage(m))$ only properly formed modbus is sent

messages going to trusted network:

$\land \forall m \in validMessages\_ : m.isValid = \text{TRUE}$ message parts waiting for their counterpart are valid

$\land \forall m \in Range(chan[\text{“finished\_trustnet”}]) : IsModbus(m)$ only properly formed modbus is sent to trustnet

$LIVENESS \triangleq$

$\land \Diamond(Len(incomingMessages\_) = 0)$
$\land \Diamond(Len(incomingMessages) = 0)$
$\land \Diamond(Len(chan[\text{“finished\_untrustnet”}]) > 0)$
$\land \Diamond(Len(chan[\text{“finished\_trustnet”}]) > 0)$

counterpart will eventually come

$\land \forall p \in validMessages\_ : \Diamond(msg\_tr.id = p.id \land msg\_tr.source \neq p.source)$

modbus check module:

$\land \forall p \in Range(chan[\text{“messagecheck”}]), \exists q \in Range(chan[\text{“untrustnet\_out”}]) : p.source = \text{“trustnet\_in”} \rightsquigarrow : p.id = q.id$

* Modification History
* Last modified Sun *Dec* 30 12:24:13 *EST* 2018 by *mssabr*01
* Last modified *Wed Oct* 17 11:32:47 *EDT* 2018 by *userMehdi*
* Created *Tue Oct* 02 17:14:28 *EDT* 2018 by *mssabr*01