Modelling Authentication and Secrecy in Kerberos Protocol using NuSMV

UFCFYN-15-M - Analysis and Verification of Concurrent Systems

ASSIGNMENT

Prepared by

Kiran Kumar Mohanraj - 20054954

MSc Cyber Security

University of the West of England, Bristol

TABLE OF CONTENTS

1. SYSTEM DESIGN

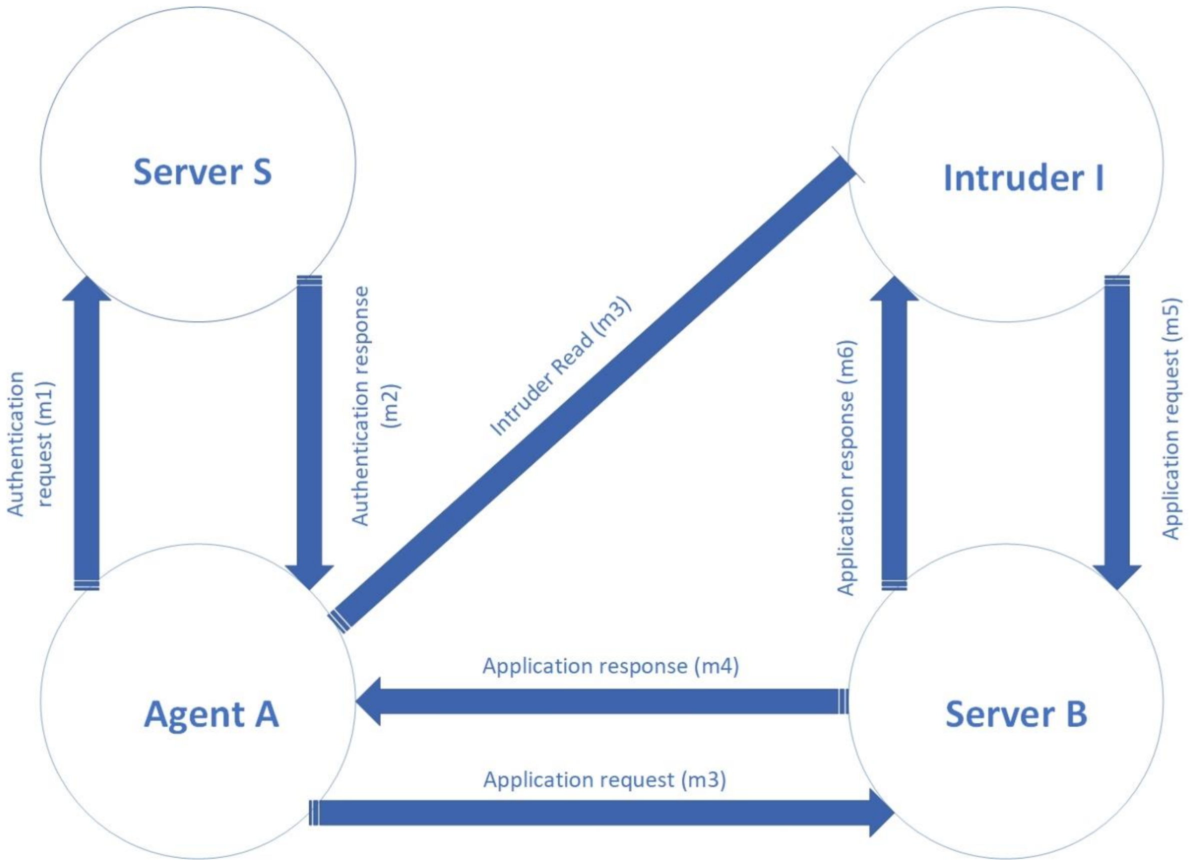
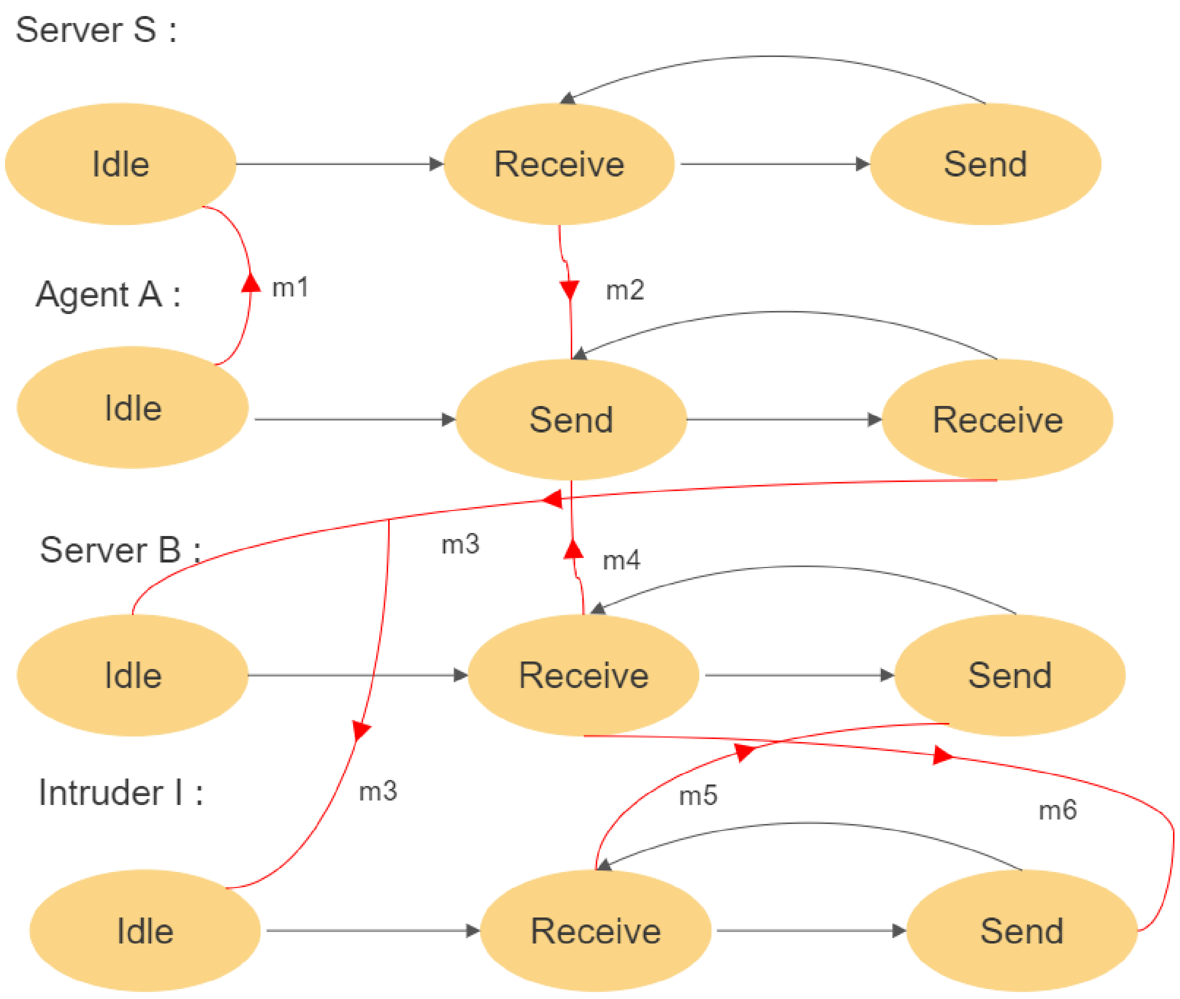

Server S
Intruder I
Agent A
Server B
Authentication request (m1)
Authentication response (m2)
Intruder Read (m3)
Application response (m6)
Application request (m5)
Application response (m4)
Application request (m3)

Figure 1 – System Design

The Kerberos protocol is based on four-step communication. It provides authentication between a client and a server through the authentication server. Figure 1 provides the system design of the Kerberos protocol under a replay attack. In the absence of Intruder, to enable communication between Agent A and Server B, Agent A sends a request to Authentication Server S for getting authentication key and Server S responds by providing the key. This key is sent to Server B and B verifies whether the key is provided by Server S and responds to Agent A.

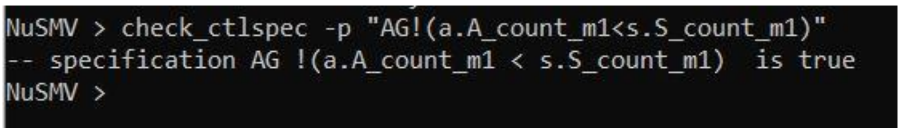
In the presence of Intruder, the intruder eavesdrops the message m3 (Application request) sent by Agent A to Server B. Once the communication between Agent A and Server B is completed, Intruder acts as Agent A and tries to create another session with Server B by sending the eavesdropped message. Server B provides response to Intruder thinking that Agent A has created another session. Thus, replay attack has been implemented.

2. STATE DIAGRAM



3. AUTHENTICATION PROPERTIES

- ***SPEC AG !(a.A_count_m1<s.S_count_m1)*** ***(1)***

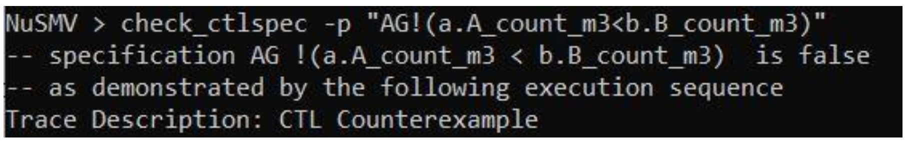


```
NuSMV > check_ctlspec -p "AG!(a.A_count_m1<s.S_count_m1)"
-- specification AG !(a.A_count_m1 < s.S_count_m1)  is true
NuSMV >
```

Figure 1

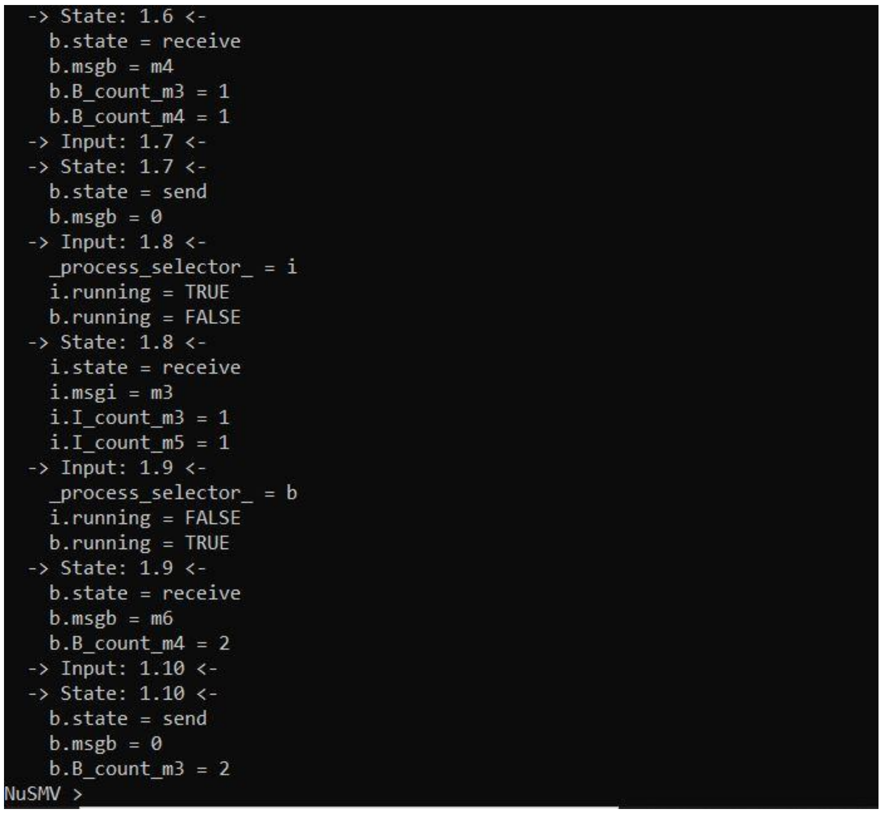
Figure 1 shows the authentication between Authentication Server and the Agent A. A_count_m1/S_count_m1 counts number of requests sent by A to Server S. For a system with no packet loss, number of requests sent by sender should be equal to number of requests received. Considering data loss, sender’s request can be greater than the requests received but not vice-versa. This property holds true because Server S receives request only from Agent A.

- ***SPEC AG!(a.A_count_m3<b.B_count_m3)*** ***(2)***



```
NuSMV > check_ctlspec -p "AG!(a.A_count_m3<b.B_count_m3)"
-- specification AG !(a.A_count_m3 < b.B_count_m3)  is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
```

Figure 2

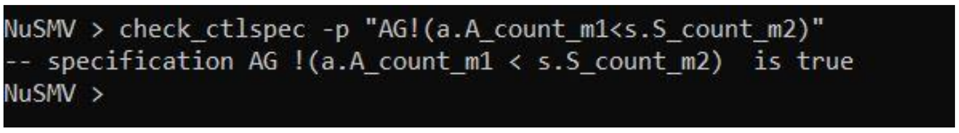


```
  -> State: 1.6 <-
    b.state = receive
    b.msgb = m4
    b.B_count_m3 = 1
    b.B_count_m4 = 1
  -> Input: 1.7 <-
  -> State: 1.7 <-
    b.state = send
    b.msgb = 0
  -> Input: 1.8 <-
    _process_selector_ = i
    i.running = TRUE
    b.running = FALSE
  -> State: 1.8 <-
    i.state = receive
    i.msgi = m3
    i.I_count_m3 = 1
    i.I_count_m5 = 1
  -> Input: 1.9 <-
    _process_selector_ = b
    i.running = FALSE
    b.running = TRUE
  -> State: 1.9 <-
    b.state = receive
    b.msgb = m6
    b.B_count_m4 = 2
  -> Input: 1.10 <-
  -> State: 1.10 <-
    b.state = send
    b.msgb = 0
    b.B_count_m3 = 2
NuSMV >
```

Figure 3

The above-shown CTL property is similar to (1), where the number of requests received by the receiver (Server B) should not be greater than the number of requests sent by Agent A. In this case, the Intruder I eavesdrop the request m3 from Agent A and establishes a communication with Server B. Thus, the property produces a false as result. Figure 3 shows the counter-example, the count value of request m3 becomes 2 because of Intruder's request.

- ***SPEC AG!(a.A_count_m1<s.S_count_m2)*** ***(3)***

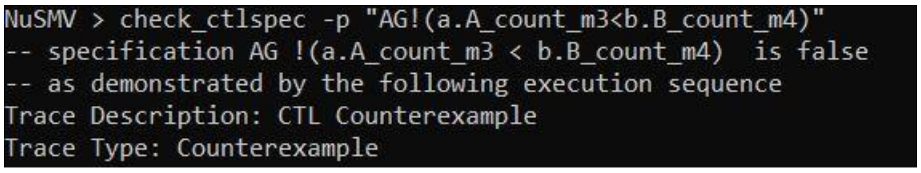


```
NuSMV > check_ctlspec -p "AG!(a.A_count_m1<s.S_count_m2)"
-- specification AG !(a.A_count_m1 < s.S_count_m2)  is true
NuSMV >
```

Figure 4

Figure 4 shows authenticity property that for every request sent there is equal or less response sent. Since there is no involvement of Intruder, this property holds true.

- ***SPEC AG!(a.A_count_m3<b.B_count_m4)*** ***(4)***



```
NuSMV > check_ctlspec -p "AG!(a.A_count_m3<b.B_count_m4)"
-- specification AG !(a.A_count_m3 < b.B_count_m4)  is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
```

Figure 5

The above-shown CTL property is similar to (3). The number of responses sent by Server B should not be greater than the number of requests sent by Agent A. Figure 5 shows that the specification is false because of the presence of Intruder I.

Figure 6 provides the counter-example, count value of response m4 becomes 2 (which is greater than the request) when it sends response m6 to the Intruder I.

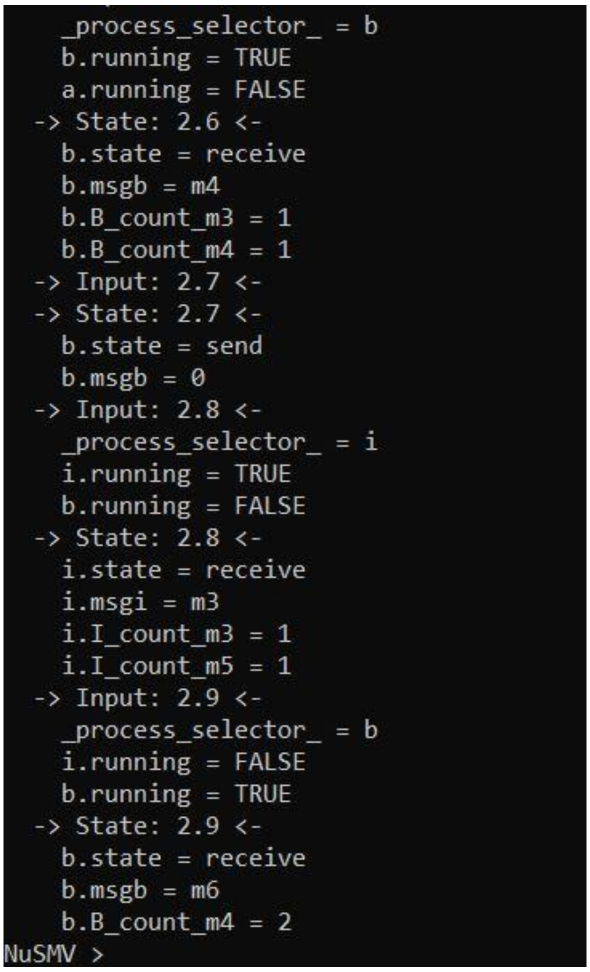
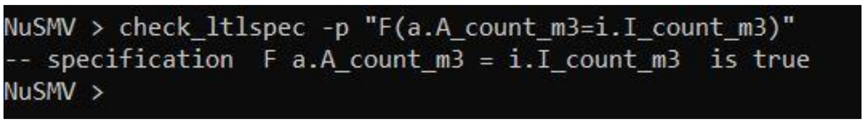

```
    _process_selector_ = b
    b.running = TRUE
    a.running = FALSE
  -> State: 2.6 <-
    b.state = receive
    b.msgb = m4
    b.B_count_m3 = 1
    b.B_count_m4 = 1
  -> Input: 2.7 <-
  -> State: 2.7 <-
    b.state = send
    b.msgb = 0
  -> Input: 2.8 <-
    _process_selector_ = i
    i.running = TRUE
    b.running = FALSE
  -> State: 2.8 <-
    i.state = receive
    i.msgi = m3
    i.I_count_m3 = 1
    i.I_count_m5 = 1
  -> Input: 2.9 <-
    _process_selector_ = b
    i.running = FALSE
    b.running = TRUE
  -> State: 2.9 <-
    b.state = receive
    b.msgb = m6
    b.B_count_m4 = 2
NuSMV >
```

Figure 6

3. SECRECY PROPERTIES

- ***SPEC F(a.A_count_m3=i.I_count_m3)***

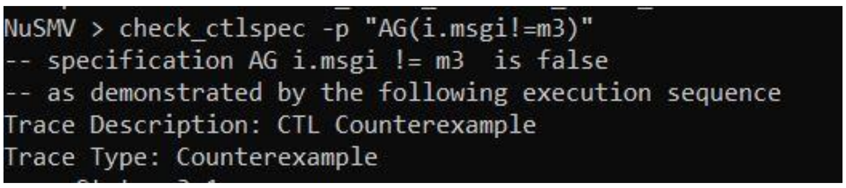


```
NuSMV > check_ltlspec -p "F(a.A_count_m3=i.I_count_m3)"
-- specification  F a.A_count_m3 = i.I_count_m3  is true
NuSMV >
```

Figure 7

The above-shown LTL property checks whether Intruder I have eavesdropped any request sent by Agent A. A_count_m3/I_count_m3 counts the number of requests sent by Agent A. In the future, if the Intruder eavesdrop the message, it would increment the I_count_m3 value and it will be equal to A_count_m3. Since there is presence of Intruder I, the property holds true.

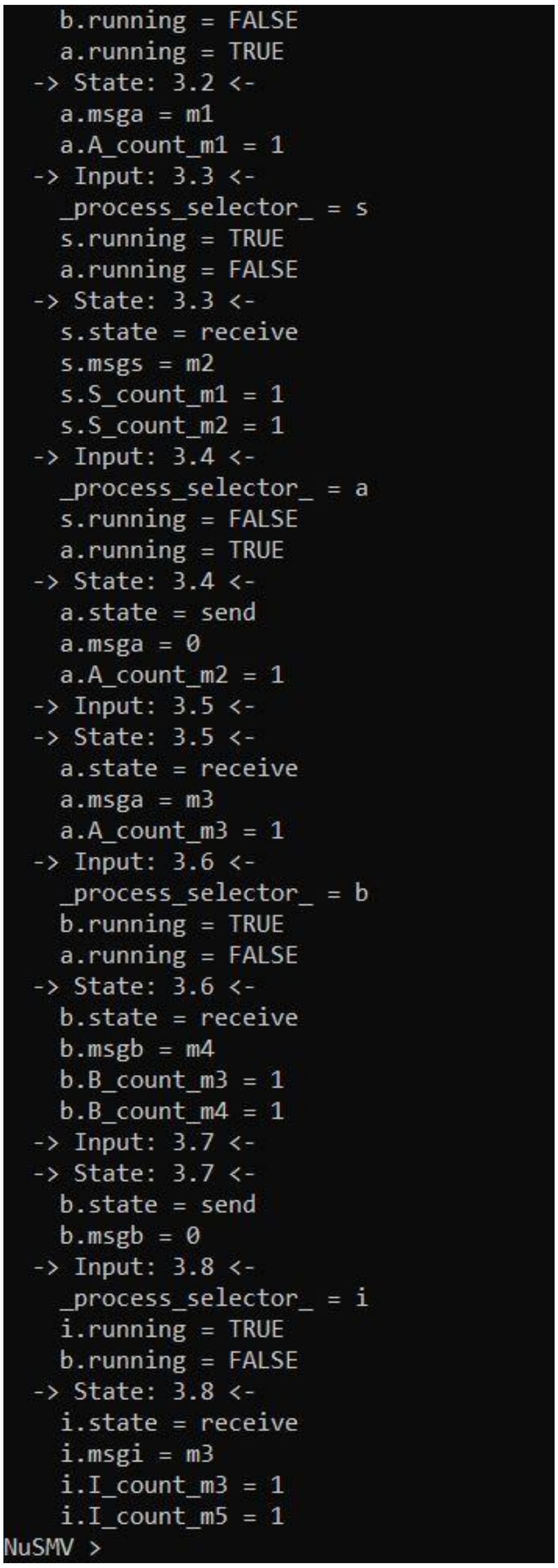
- ***SPEC AG(i.msgi!=m3)***



```
NuSMV > check_ctlspec -p "AG(i.msgi!=m3)"
-- specification AG i.msgi != m3  is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
```

Figure 8

This CTL property checks whether the Intruder I receive the request m3 sent by Agent A or not. Figure 9 shows the counter-example, when the Intruder I process starts, it receives the request m3 and switches to Receive state.



```
    b.running = FALSE
    a.running = TRUE
  -> State: 3.2 <-
    a.msga = m1
    a.A_count_m1 = 1
  -> Input: 3.3 <-
    _process_selector_ = s
    s.running = TRUE
    a.running = FALSE
  -> State: 3.3 <-
    s.state = receive
    s.msgs = m2
    s.S_count_m1 = 1
    s.S_count_m2 = 1
  -> Input: 3.4 <-
    _process_selector_ = a
    s.running = FALSE
    a.running = TRUE
  -> State: 3.4 <-
    a.state = send
    a.msga = 0
    a.A_count_m2 = 1
  -> Input: 3.5 <-
  -> State: 3.5 <-
    a.state = receive
    a.msga = m3
    a.A_count_m3 = 1
  -> Input: 3.6 <-
    _process_selector_ = b
    b.running = TRUE
    a.running = FALSE
  -> State: 3.6 <-
    b.state = receive
    b.msgb = m4
    b.B_count_m3 = 1
    b.B_count_m4 = 1
  -> Input: 3.7 <-
  -> State: 3.7 <-
    b.state = send
    b.msgb = 0
  -> Input: 3.8 <-
    _process_selector_ = i
    i.running = TRUE
    b.running = FALSE
  -> State: 3.8 <-
    i.state = receive
    i.msgi = m3
    i.I_count_m3 = 1
    i.I_count_m5 = 1
NuSMV >
```

Figure 9

5. REFERENCES

Adyanthaya, S., Rukmangada, S., Tiwari, A., & Singh, S. (2010). Modelingfreshness concept to overcome replay attackin Kerberos protocol using NuSMV. *2010 International Conference on Computer and Communication Technology, ICCCT-2010*, 125–129. https://doi.org/10.1109/ICCCT.2010.5640425

M.Panti, M.Spalazzi, S.Tacconi. Using the NuSMV model checker to verify the Kerberos protocol.