Secure AODV in MANET

Software: NetSim Standard v14.0, Microsoft Visual Studio 2022

Project Download Link:

https://github.com/NetSim-TETCOS/Secure-AODV-v14.0/archive/refs/heads/main.zip

Follow the instructions specified in the following link to download and set up the Project in NetSim:

https://support.tetcos.com/en/support/solutions/articles/14000128666-downloading-and-setting-up-netsim-file-exchange-projects

Introduction:

SAODV is an extension of the AODV routing protocol that can be used to protect the route discovery mechanism by providing security features like integrity and authentication. The reason only route discovery is secured by AODV is that data messages can be protected using a point-to-point security protocol like IPSec. SAODV uses a key management system, and each node maintains public keys, encryption keys, and decryption keys.
To implement SAODV, we have added **Secure AODV.c, RSA.c, and Malicious.c** files in the AODV project. RSA.c file is used to generate keys, encrypt, and decrypt the data. Users can implement their own encryption algorithms by changing the RSA.c file. malicious.c file is used to identify malicious nodes present in the network.

Real-World Context:

In the context of military communication, the secure AODV protocol ensures the secure transmission of data from higher-level commands to lower-level commands. This protocol utilizes encryption and decryption keys to ensure that all devices can securely access and read the data. If a malicious device is present in the middle level, the data will not be accessible or readable to the malicious device.

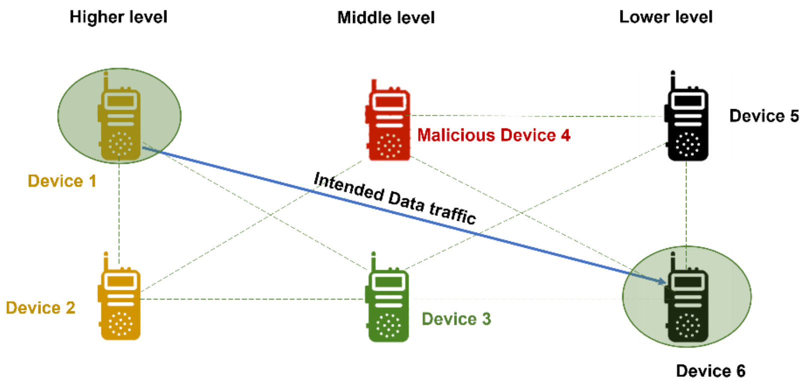


Figure 1:Secure AODV in Military communication using Manet

Secure AODV Overview:

1. To discover a route to Device 6, Device 1 sends an RREQ encrypted with Device 6's public key.
2. When Device 6 receives the RREQ, it decrypts it with its private key.
3. Device 6 then sends an RREP back to Device 1, encrypted with Device-1's public key.
4. Device 1 decrypts the RREP with its private key.

If a malicious device tries to intercept the RREQ or RREP, it will not be able to decrypt it because it does not have the private keys of Device 1 or Device 6. Additionally, if a malicious device tries to modify the RREQ or RREP, the other nodes in the network will be able to detect the modification and discard the packet.

Case 1:

Secure AODV implementation:

1. The **Secure-AODV-Workspace** comes with a sample network configuration that is already saved. To open this example, go to Your work in the home screen of NetSim and click on the **Secure-AODV-Example** from the list of experiments.
2. After running the simulation, a **Secure_AODV.log** file gets created in the Result Dashboard Window.

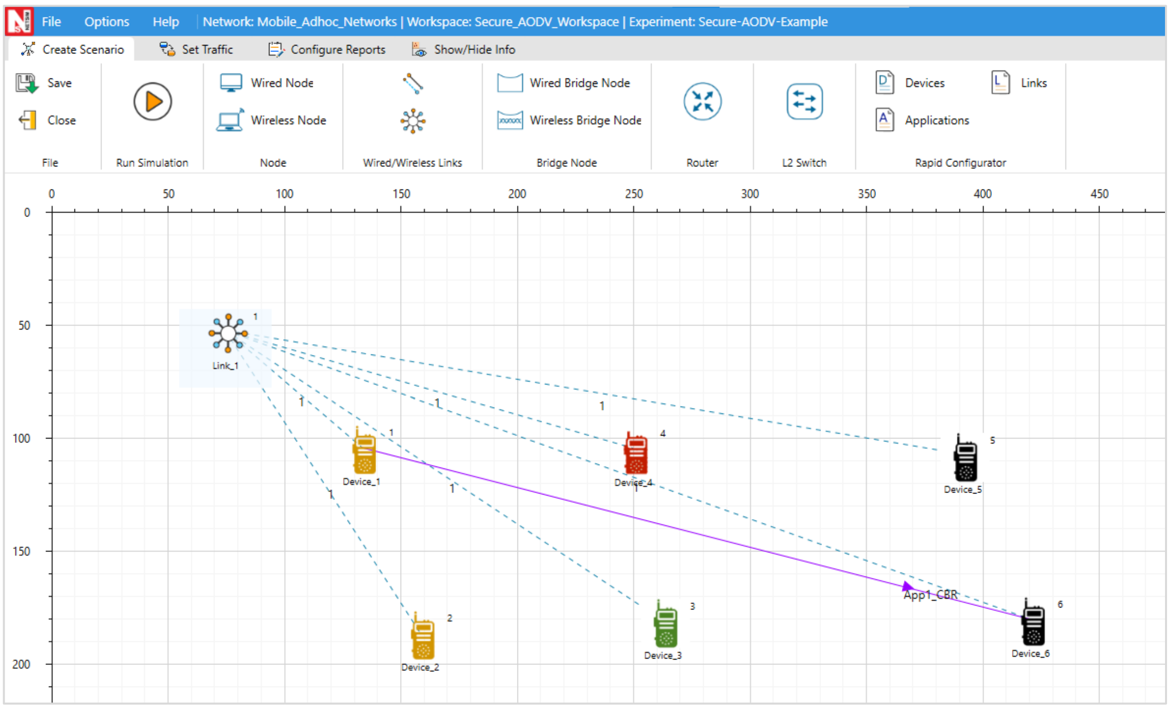
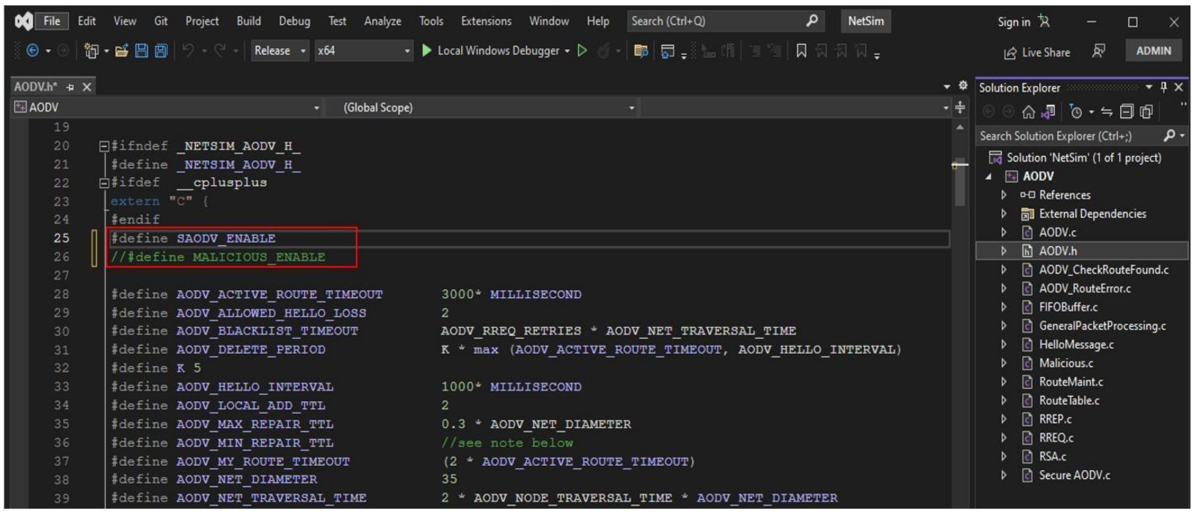


Figure 2: Network setup for Secure AODV in Manet

- Open the Source code in Visual Studio by going to Your work -> Source Code and Open code in the NetSim Home Screen window.
- Expand the AODV project.



```
#ifndef _NETSIM_AODV_H_
#define _NETSIM_AODV_H_
#ifdef __cplusplus
extern "C" {
#endif
#define SAODV_ENABLE
//#define MALICIOUS_ENABLE

#define AODV_ACTIVE_ROUTE_TIMEOUT       3000* MILLISECOND
#define AODV_ALLOWED_HELLO_LOSS         2
#define AODV_BLACKLIST_TIMEOUT          AODV_RREQ_RETRIES * AODV_NET_TRAVERSAL_TIME
#define AODV_DELETE_PERIOD              K * max (AODV_ACTIVE_ROUTE_TIMEOUT, AODV_HELLO_INTERVAL)
#define K 5
#define AODV_HELLO_INTERVAL             1000* MILLISECOND
#define AODV_LOCAL_ADD_TTL              2
#define AODV_MAX_REPAIR_TTL             0.3 * AODV_NET_DIAMETER
#define AODV_MIN_REPAIR_TTL             //see note below
#define AODV_MY_ROUTE_TIMEOUT           (2 * AODV_ACTIVE_ROUTE_TIMEOUT)
#define AODV_NET_DIAMETER               35
#define AODV_NET_TRAVERSAL_TIME         2 * AODV_NODE_TRAVERSAL_TIME * AODV_NET_DIAMETER
```

Figure 3: Screenshot of Solution Explorer of AODV project

- Here users can enable Secure AODV (Open **AODV.h** file).

  Uncomment the line **#define SAODV_ENABLE**,

  Comment the line **//#define MALICIOUS_ENABLE** present in **AODV.h** file.

Rebuild the AODV Project by right-clicking and Run the simulation for 100 sec in the Netsim GUI.

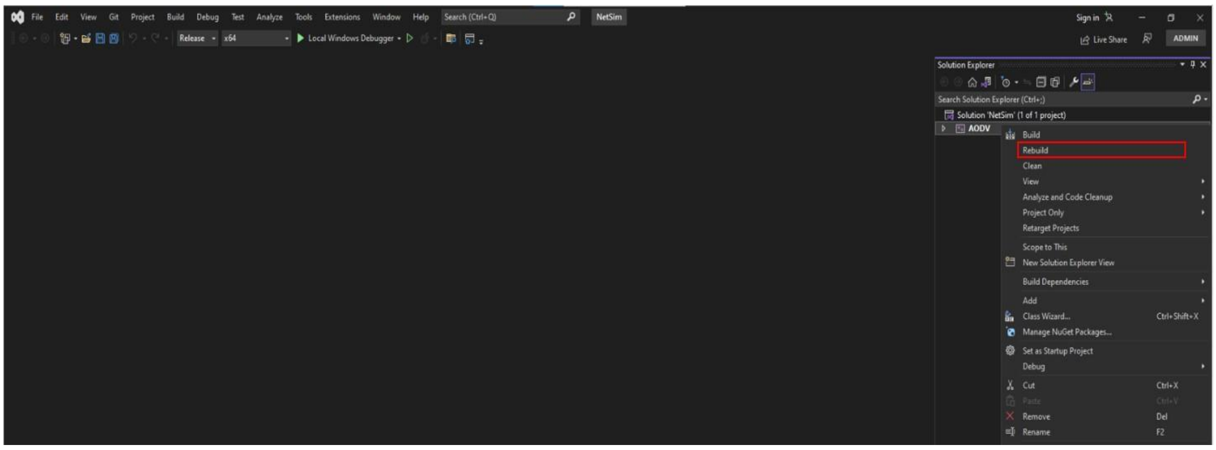


Figure 4: Screenshot of NetSim project Source Code in Visual Studio

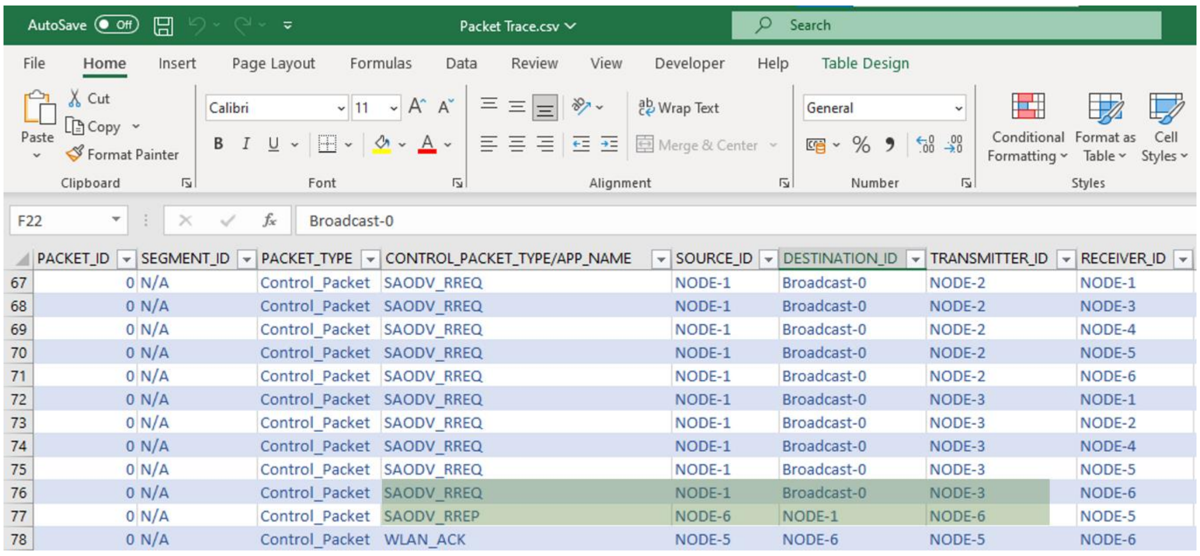
A **Secure_AODV.c** file is added to the AODV project which contains the following important functions:

- **saodv_encrypt_packet();** //This function is used to encrypt the control packet data
- **saodv_decrypt_packet();** //This function is used to decrypt the control packet data
- **get_rrep_str_data();** //This function is used to get the route reply data from AODV_RREP control packet
- **get_rreq_str_data();** //This function is used to get the route request data from AODV_RREQ control packet

- **get_saodv_ctrl_packet_type();** //This function is used to change the control packet type from AODV (AODV_RREQ, AODV_RREP) to SAODV (SAODV_RREQ, SAODV_RREP)
- **get_saodv_ctrl_packet();** //This function is called whenever a new control packet is generated
- **get_aodv_ctrl_packet();** //This function is called while processing the control packets

Results and discussion:

After Running the simulation of the given Configuration file, open the packet trace in the Result Dashboard Window. In the packet trace, filter the control packets to **SAODV_RREQ** and **SAODV_RREP)**

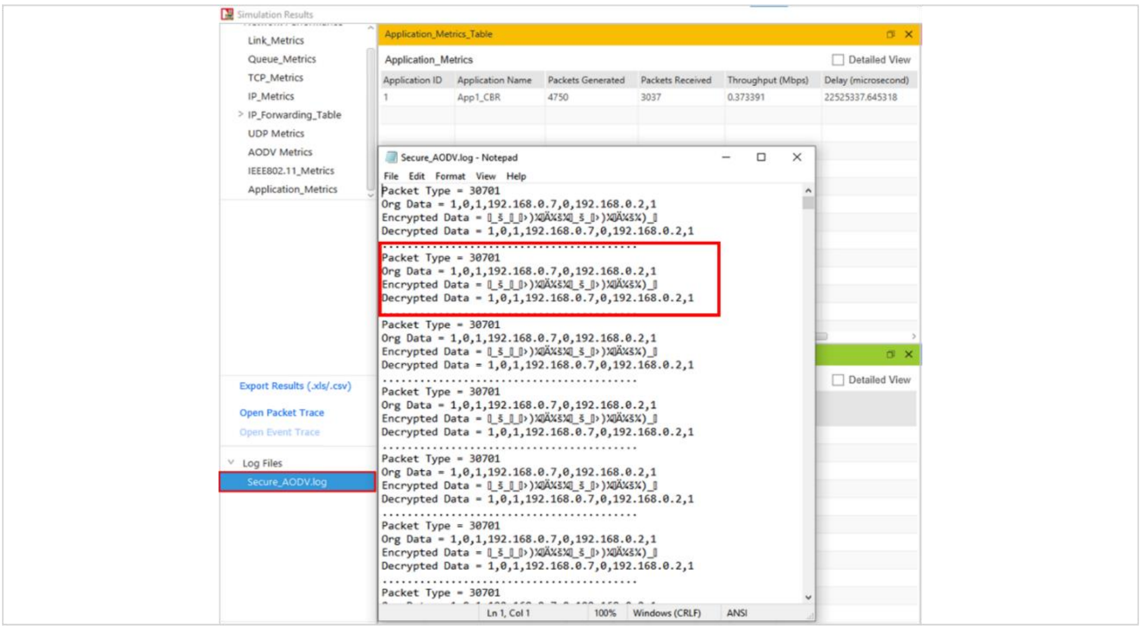


F22 — Broadcast-0

	PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYPE/APP_NAME	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID
67	0	N/A	Control_Packet	SAODV_RREQ	NODE-1	Broadcast-0	NODE-2	NODE-1
68	0	N/A	Control_Packet	SAODV_RREQ	NODE-1	Broadcast-0	NODE-2	NODE-3
69	0	N/A	Control_Packet	SAODV_RREQ	NODE-1	Broadcast-0	NODE-2	NODE-4
70	0	N/A	Control_Packet	SAODV_RREQ	NODE-1	Broadcast-0	NODE-2	NODE-5
71	0	N/A	Control_Packet	SAODV_RREQ	NODE-1	Broadcast-0	NODE-2	NODE-6
72	0	N/A	Control_Packet	SAODV_RREQ	NODE-1	Broadcast-0	NODE-3	NODE-1
73	0	N/A	Control_Packet	SAODV_RREQ	NODE-1	Broadcast-0	NODE-3	NODE-2
74	0	N/A	Control_Packet	SAODV_RREQ	NODE-1	Broadcast-0	NODE-3	NODE-4
75	0	N/A	Control_Packet	SAODV_RREQ	NODE-1	Broadcast-0	NODE-3	NODE-5
76	0	N/A	Control_Packet	SAODV_RREQ	NODE-1	Broadcast-0	NODE-3	NODE-6
77	0	N/A	Control_Packet	SAODV_RREP	NODE-6	NODE-1	NODE-6	NODE-5
78	0	N/A	Control_Packet	WLAN_ACK	NODE-5	NODE-6	NODE-5	NODE-6

Figure 5: Network Packet trace results for Secure AODV implementation

The SAODV logs certain details in **Secure_AODV.log**. The Log File can be observed in the Result Dashboard Window.



Application_Metrics

Application ID	Application Name	Packets Generated	Packets Received	Throughput (Mbps)	Delay (microsecond)
1	App1_CBR	4750	3037	0.373391	22525337.645318

```
Packet Type = 30701
Org Data = 1,0,1,192.168.0.7,0,192.168.0.2,1
Encrypted Data = □_š_□_□›)¼□Ä¼š¼□_š_□›)¼□Ä¼š¼)_□
Decrypted Data = 1,0,1,192.168.0.7,0,192.168.0.2,1
.........................................
Packet Type = 30701
Org Data = 1,0,1,192.168.0.7,0,192.168.0.2,1
Encrypted Data = □_š_□_□›)¼□Ä¼š¼□_š_□›)¼□Ä¼š¼)_□
Decrypted Data = 1,0,1,192.168.0.7,0,192.168.0.2,1
.........................................
Packet Type = 30701
Org Data = 1,0,1,192.168.0.7,0,192.168.0.2,1
Encrypted Data = □_š_□_□›)¼□Ä¼š¼□_š_□›)¼□Ä¼š¼)_□
Decrypted Data = 1,0,1,192.168.0.7,0,192.168.0.2,1
.........................................
Packet Type = 30701
Org Data = 1,0,1,192.168.0.7,0,192.168.0.2,1
Encrypted Data = □_š_□_□›)¼□Ä¼š¼□_š_□›)¼□Ä¼š¼)_□
Decrypted Data = 1,0,1,192.168.0.7,0,192.168.0.2,1
.........................................
Packet Type = 30701
Org Data = 1,0,1,192.168.0.7,0,192.168.0.2,1
Encrypted Data = □_š_□_□›)¼□Ä¼š¼□_š_□›)¼□Ä¼š¼)_□
Decrypted Data = 1,0,1,192.168.0.7,0,192.168.0.2,1
.........................................
Packet Type = 30701
Org Data = 1,0,1,192.168.0.7,0,192.168.0.2,1
Encrypted Data = □_š_□_□›)¼□Ä¼š¼□_š_□›)¼□Ä¼š¼)_□
Decrypted Data = 1,0,1,192.168.0.7,0,192.168.0.2,1
.........................................
Packet Type = 30701
```

Ln 1, Col 1 100% Windows (CRLF) ANSI

Figure 6: Secure AODV log file

The format of the log file is such that each control packet is logged. The first line represents the packet type and the numbering used in a NetSim internal numbering system where **30701 is RREQ and 30702 is RREP**. The second line is the message which is encrypted. The third line contains the encrypted message after running the RSA encryption algorithm. The fourth line is after decryption and if everything is OK, the 2nd and 4th lines must match.

....

Packet Type = 30701

Org Data = 1,0,1,192.168.0.7,0,192.168.0.2,1

Encrypted Data = š ššJÒÜšÀĐÜ Ü× šJÒÜšÀĐÜ ÜÒš

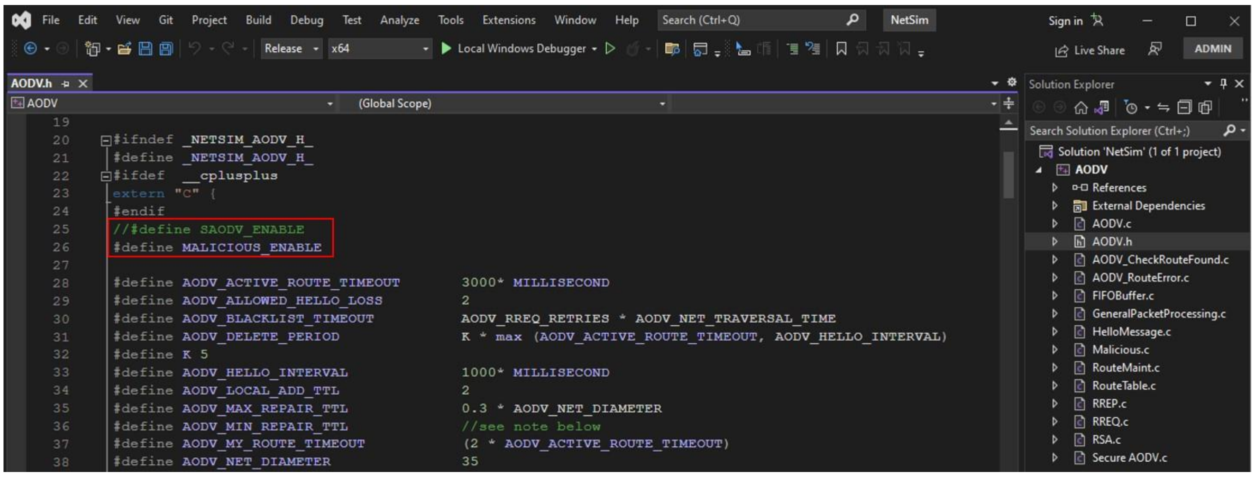
Decrypted Data = 1,0,1,192.168.0.7,0,192.168.0.2,1

..

Case 2 :

Malicious node implementation:

Here users can enable the code to implement malicious node
Uncomment #define MALICIOUS_ENABLE and
comment //#define SAODV_ENABLE that are present inside AODV.h file and Rebuild the Project.



```
#ifndef _NETSIM_AODV_H_
#define _NETSIM_AODV_H_
#ifdef __cplusplus
extern "C" {
#endif
//#define SAODV_ENABLE
#define MALICIOUS_ENABLE

#define AODV_ACTIVE_ROUTE_TIMEOUT        3000* MILLISECOND
#define AODV_ALLOWED_HELLO_LOSS          2
#define AODV_BLACKLIST_TIMEOUT           AODV_RREQ_RETRIES * AODV_NET_TRAVERSAL_TIME
#define AODV_DELETE_PERIOD               K * max (AODV_ACTIVE_ROUTE_TIMEOUT, AODV_HELLO_INTERVAL)
#define K 5
#define AODV_HELLO_INTERVAL              1000* MILLISECOND
#define AODV_LOCAL_ADD_TTL               2
#define AODV_MAX_REPAIR_TTL              0.3 * AODV_NET_DIAMETER
#define AODV_MIN_REPAIR_TTL              //see note below
#define AODV_MY_ROUTE_TIMEOUT            (2 * AODV_ACTIVE_ROUTE_TIMEOUT)
#define AODV_NET_DIAMETER                35
```

Figure 7: Comment and Uncomment the code of SAODV and Malicious

A malicious node advertises wrong routing information to produce itself as a specific node and receives whole network traffic.

After receiving the whole network traffic, it can either modify the packet information or drop them to make the network complicated.

A file **malicious.c** is added to the AODV project which contains the following functions:

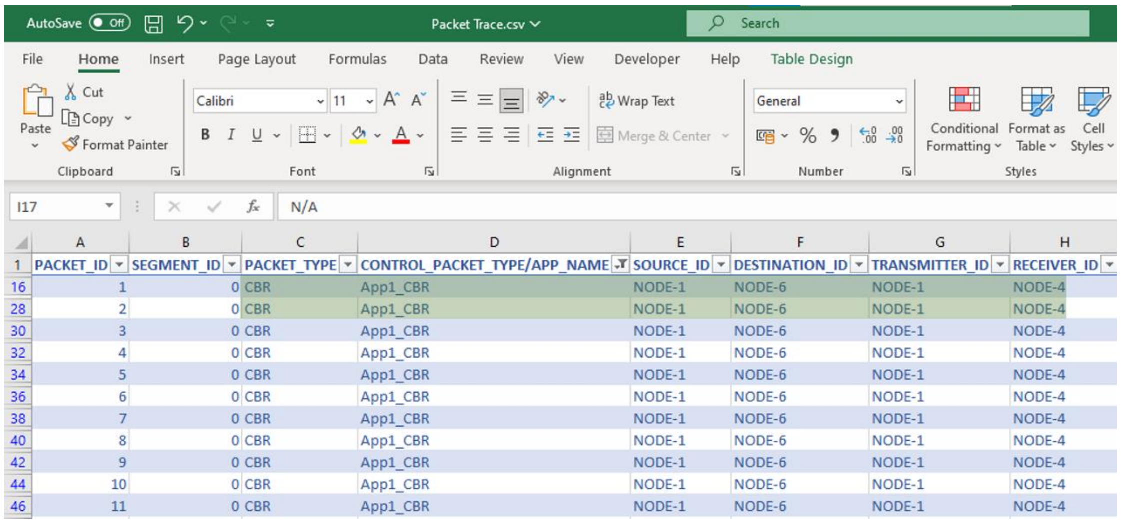
- **IsMaliciousNode();** //This function is used to identify whether a current device is malicious or not in-order to establish malicious behaviour.

- **fn_NetSim_AODV_MaliciousRouteAddToTable();** //This function is used to add a fake route entry into the route table of the malicious device with its next hop as the destination.
- **fn_NetSim_AODV_MaliciousProcessSourceRouteOption();** //This function is used to drop the received packets if the device is malicious, instead of forwarding the packet to the next hop

Rebuild the Project and Run the simulation for 100 seconds in Netsim GUI.

Results and discussion:

- You can set any device as a malicious node, and you can have more than one malicious node in a scenario.
- Device IDs of malicious nodes can be set using the malicious_node [ ] array present in malicious.c file. Comment the line #define SAODV_ENABLE present in AODV.h file.
- Rebuild the project and run the simulation.
- If we run the simulation without SAODV, we will get zero throughputs because the malicious node gets all the packets and drops without forwarding them to the destination. You can notice this in the NetSim packet trace by filtering the PACKET_TYPE to CBR.



	PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYPE/APP_NAME	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID
16	1	0	CBR	App1_CBR	NODE-1	NODE-6	NODE-1	NODE-4
28	2	0	CBR	App1_CBR	NODE-1	NODE-6	NODE-1	NODE-4
30	3	0	CBR	App1_CBR	NODE-1	NODE-6	NODE-1	NODE-4
32	4	0	CBR	App1_CBR	NODE-1	NODE-6	NODE-1	NODE-4
34	5	0	CBR	App1_CBR	NODE-1	NODE-6	NODE-1	NODE-4
36	6	0	CBR	App1_CBR	NODE-1	NODE-6	NODE-1	NODE-4
38	7	0	CBR	App1_CBR	NODE-1	NODE-6	NODE-1	NODE-4
40	8	0	CBR	App1_CBR	NODE-1	NODE-6	NODE-1	NODE-4
42	9	0	CBR	App1_CBR	NODE-1	NODE-6	NODE-1	NODE-4
44	10	0	CBR	App1_CBR	NODE-1	NODE-6	NODE-1	NODE-4
46	11	0	CBR	App1_CBR	NODE-1	NODE-6	NODE-1	NODE-4

Figure 8: NetSim Packet trace results for Malicious node implementation

Case 3:

Both Secure AODV and Malicious node implementation:

Enable (Uncomment) the below mentioned lines of code present in AODV.h file.

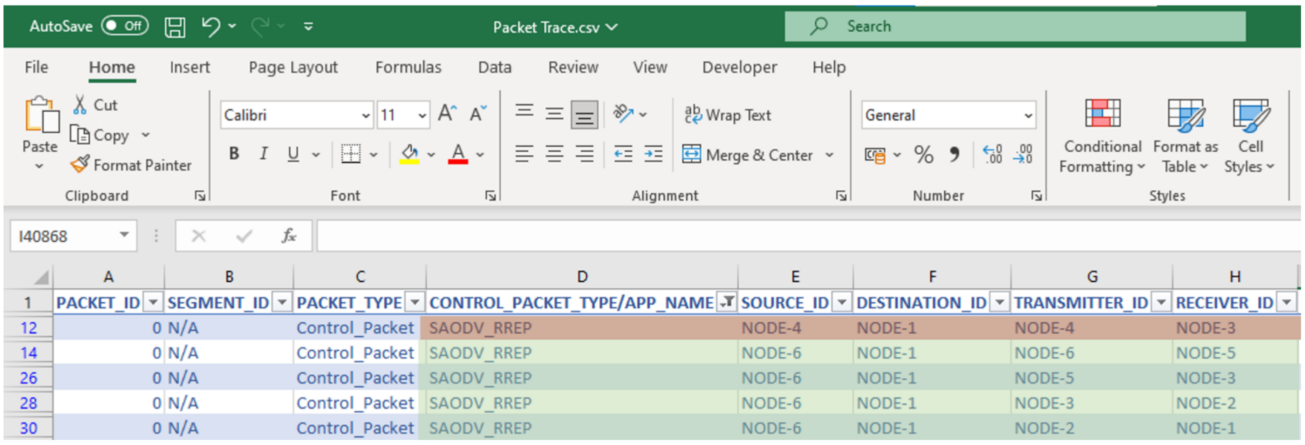
```
#define SAODV_ENABLE

#define MALICIOUS_ENABLE
```

Rebuild the Project and run the simulation for 100 sec in Netsim GUI.

Results and discussion:

Packets will be transmitted to the destination since SAODV helps in overcoming the Malicious Node problem. Route reply RREP from malicious node 4 will not be accepted by Node 1. It takes the Route reply from node 2 and forms the route. This can be observed in Packet Trace by filtering the CONTROL_PACKET_TYPE to SAODV_RREP. Malicious node 4 is transmitting the Route Reply (RREP) to node 3, but node 3 is not forwarding to any nodes .On the other hand, when it comes to node 6, it is sending the RREP to node 5, then from 5 to 3, from 3 to 2 and finally from 2 to 1.

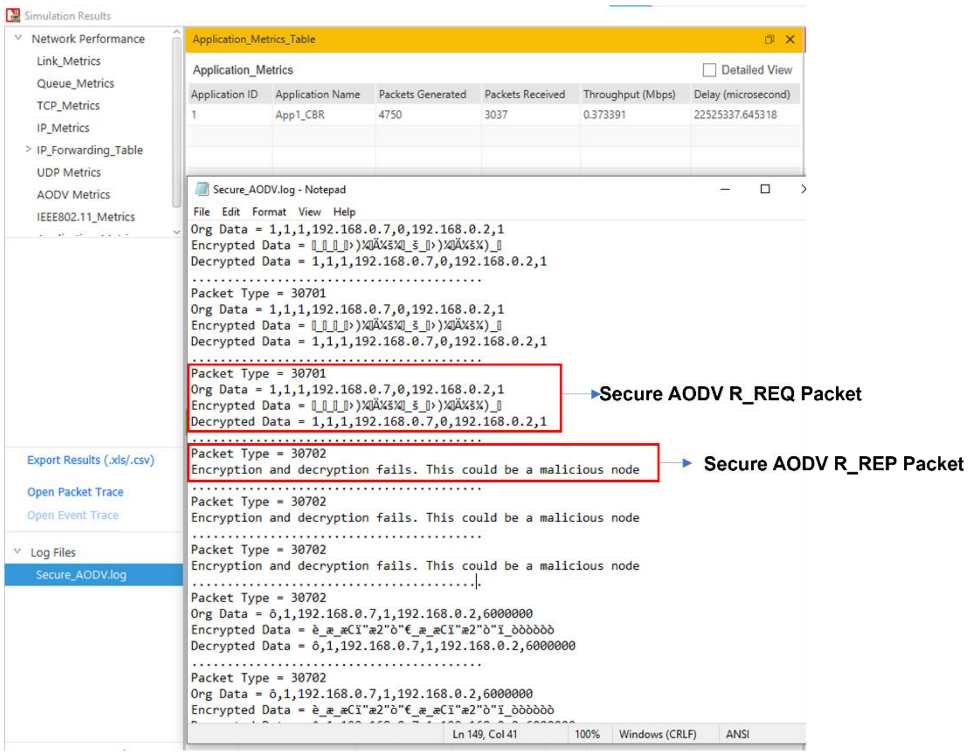
🟧	SAODV_RREP Packets from Malicious Node
🟩	SAODV_RREP from Destination node


AutoSave Off
Packet Trace.csv
Search
File Home Insert Page Layout Formulas Data Review View Developer Help
Paste Cut Copy Format Painter Clipboard
Calibri 11 Font
Wrap Text Merge & Center Alignment
General Number
Conditional Formatting Format as Table Cell Styles Styles
I40868

	PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYPE/APP_NAME	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID
12	0	N/A	Control_Packet	SAODV_RREP	NODE-4	NODE-1	NODE-4	NODE-3
14	0	N/A	Control_Packet	SAODV_RREP	NODE-6	NODE-1	NODE-6	NODE-5
26	0	N/A	Control_Packet	SAODV_RREP	NODE-6	NODE-1	NODE-5	NODE-3
28	0	N/A	Control_Packet	SAODV_RREP	NODE-6	NODE-1	NODE-3	NODE-2
30	0	N/A	Control_Packet	SAODV_RREP	NODE-6	NODE-1	NODE-2	NODE-1

Figure 9: NetSim Packet trace results for Secure AODV and Malicious node implementation

The SAODV logs certain details in **Secure_AODV.log**. The Log File can be observed in the Result Dashboard Window.



Application_Metrics

Application ID	Application Name	Packets Generated	Packets Received	Throughput (Mbps)	Delay (microsecond)
1	App1_CBR	4750	3037	0.373391	22525337.645318

```
Org Data = 1,1,1,192.168.0.7,0,192.168.0.2,1
Encrypted Data = ▯_▯_▯_▯›)¼▯Ä¼š¼▯_š_▯›)¼▯Ä¼š¼)_▯
Decrypted Data = 1,1,1,192.168.0.7,0,192.168.0.2,1
.........................................
Packet Type = 30701
Org Data = 1,1,1,192.168.0.7,0,192.168.0.2,1
Encrypted Data = ▯_▯_▯_▯›)¼▯Ä¼š¼▯_š_▯›)¼▯Ä¼š¼)_▯
Decrypted Data = 1,1,1,192.168.0.7,0,192.168.0.2,1
.........................................
Packet Type = 30701
Org Data = 1,1,1,192.168.0.7,0,192.168.0.2,1
Encrypted Data = ▯_▯_▯_▯›)¼▯Ä¼š¼▯_š_▯›)¼▯Ä¼š¼)_▯
Decrypted Data = 1,1,1,192.168.0.7,0,192.168.0.2,1
.........................................
Packet Type = 30702
Encryption and decryption fails. This could be a malicious node
.........................................
Packet Type = 30702
Encryption and decryption fails. This could be a malicious node
.........................................
Packet Type = 30702
Encryption and decryption fails. This could be a malicious node
.........................................
Packet Type = 30702
Org Data = ô,1,192.168.0.7,1,192.168.0.2,6000000
Encrypted Data = è_æ_æCï"æ2"ò"€_æ_æCï"æ2"ò"ï_òòòòòò
Decrypted Data = ô,1,192.168.0.7,1,192.168.0.2,6000000
.........................................
Packet Type = 30702
Org Data = ô,1,192.168.0.7,1,192.168.0.2,6000000
Encrypted Data = è_æ_æCï"æ2"ò"€_æ_æCï"æ2"ò"ï_òòòòòò
```

Figure 9: Secure AODV log file

The Packet type 30701 = RREQ is the request packet and the Packet type 30702=RREP is the reply packet, when the malicious node tries to decrypt the message.

...

Packet Type = 30702

Encryption and decryption fail. This could be a malicious node.

...

Appendix: NetSim source code modifications

We have added Secure_AODV.c, RSA.c and Malicious.c files, we have added the following macros code in AODV.h file within AODV project.

```
#define SAODV_ENABLE

#define  MALICIOUS_ENABLE
```

Then we have added the following lines of code in enum_AODV_Ctrl_Packet in AODV.h file

---

```
//#ifdef SAODV_ENABLE
 SAODV_RREQ,
 SAODV_RREP,
 SAODV_RERR,
//#endif
```

We have added the following function prototypes in AODV.h file, within AODV project.

```
#ifdef SAODV_ENABLE        void get_saodv_ctrl_packet(NetSim_PACKET*
packet);          void get_aodv_ctrl_packet(NetSim_PACKET* packet);        void
saodv_copy_packet(NetSim_PACKET* dest, NetSim_PACKET* src);        void
saodv_free_packet(NetSim_PACKET* packet);        void
remove_from_mapper(void* ptr, bool isfree);
```

#endif // SAODV_ENABLE bool

IsMaliciousNode(NETSIM_ID devId);

We have added the following function prototypes in AODV.c file int

fn_NetSim_AODV_MaliciousRouteAddToTable(NetSim_EVENTDETAILS*); int

fn_NetSim_AODV_MaliciousProcessSourceRouteOption(NetSim_EVENTDETAILS*);

Changes to NETWORK_IN event in fn_NetSim_AODV_Run() function in AODV.c file, within AODV project

---

```
 #ifdef SAODV_ENABLE switch (pstruEventDetails-
>pPacket->nControlDataType)
        {
                case SAODV_RREQ:
                case SAODV_RREP:
        case SAODV_RERR:
                get_aodv_ctrl_packet(pstruEventDetails->pPacket);
                break;
        }
        if (pstruEventDetails->pPacket == NULL)
        {
                return -1; //Decryption fail.
        }
#endif // SAODV_ENABLE
```

We have added the following lines of code in AODVctrlPacket_RREQ and default cases in NETWORK_IN event to check the current node is malicious or not.

```
if (IsMaliciousNode(pstruEventDetails->nDeviceId))
fn_NetSim_AODV_MaliciousRouteAddToTable(pstruEventDetails);
```

Changes code in fn_NetSim_AODV_CopyPacket () function, in AODV.c file, within AODV project

```
#ifdef SAODV_ENABLE switch(srcPacket->nControlDataType)
{
case  SAODV_RERR: case
SAODV_RREQ: case
SAODV_RREP:
saodv_copy_packet(destPacket,srcPacket);
return 0; break;
default:
#endif
            return  fn_NetSim_AODV_CopyPacket_F(destPacket,srcPacket);
#ifdef SAODV_ENABLE
            break;
            }
#endif
```

Changes code in int fn_NetSim_AODV_FreePacket () present in the AODV.c file, within AODV project

```
#ifdef SAODV_ENABLE        switch
(packet->nControlDataType)
          {
          case SAODV_RERR:
case SAODV_RREQ:           case
SAODV_RREP:
```

```
          saodv_free_packet(packet);
return 0;         break;
default:
                    remove_from_mapper(packet->pstruNetworkData->Packet_RoutingProtocol, true);
                    return 0;
break;
          }
#endif // SAODV_ENABLE
```

Changes code in fn_NetSim_AODV_GenerateRREQ (), fn_NetSim_AODV_RetryRREQ () and fn_NetSim_AODV_ForwardRREQ () functions present in RREQ.c file, within AODV project

```
#ifdef SAODV_ENABLE
            get_saodv_ctrl_packet(packet);
#endif
```

Changes code in fn_NetSim_AODV_GenerateRREP(), fn_NetSim_AODV_ForwardRREP () and fn_NetSim_AODV_GenerateRREPByIntermediate () functions present in RREP.c file, within AODV project

```
#ifdef SAODV_ENABLE
            get_saodv_ctrl_packet(packet);
#endif
```