Secure AODV in MANET

**Software:** NetSim Standard v13.2, Microsoft Visual Studio 2022

**Project Download Link:**

<https://github.com/NetSim-TETCOS/Secure_AODV_v13.1/archive/refs/heads/main.zip>

Follow the instructions specified in the following link to download and setup 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 providing security features like integrity and authentication. The reason only route discovery is secured by AODV is because 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 AODV project. RSA.c file is used to generate keys, encrypt, and decrypt the data. Users can implement their own encryption algorithms by changing RSA.c file. Malicious.c file is used to identify malicious nodes present in the network.

**Example**

1. The **Secure\_AODV\_Workspace** comes with a sample network configuration that are 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 bin folder which is part of NetSim’s installed directory. This is explained in the next section

**Secure AODV implementation**

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

Graphical user interface, text, application

Description automatically generated

Figure 1: Screenshot 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 solution and run the simulation.

Graphical user interface

Description automatically generated

Figure 2: 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 simulation of the given Configuration file, open packet animation. In the packet animation, users can notice **SAODV\_RREQ** and **SAODV\_RREP** control packets

Graphical user interface

Description automatically generated

**Figure 3:** NetSim Animation Window

The SAODV codes also logs certain details in SAODV.log (i.e. present in the NetSim setup install directory bin folder).

Graphical user interface, text, table

Description automatically generated

Figure 4: Secure\_AODV log file path

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 by **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 = 1,0,1,11.1.1.6,0,11.1.1.1,1**

**Encrypted Data = \*-Ÿ-\*-\*\*¡\*¡\*¡-Ÿ-\*\*¡\*¡\*¡\*-\* Decrypted Data = 1,0,1,11.1.1.6,0,11.1.1.1,1**

**..................................................................................**

**Malicious node implementation**

Here users can enable code to malicious node problem. Enable #define MALICIOUS\_ENABLE and comment #define SAODV\_ENABLE that are present inside AODV.h file.

Graphical user interface, text, application

Description automatically generated

**Figure 5:** Commit and Uncommit for SAODV and Malicious code

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

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

In packet animation, users can notice that malicious node will take all the packets and drops without forwarding to destination.

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 behavior.
* **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 solution and run the simulation.

**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 id’s of malicious nodes can be set using malicious\_node [ ] array present in malicious.c file. Comment the line #define SAODV\_ENABLE present in AODV.h file.
* Rebuild the solution and replace the dlls as explained before and run the simulation.
* If we run simulation without SAODV, we will get zero throughput because malicious node gets all the packets and drops without forwarding to destination. You can notice this in NetSim packet animation.

Chart

Description automatically generated

**Figure 6:** NetSim Animation Window

**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 solution and run the simulation.

**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.

The SAODV logs certain details in **Secure\_AODV.log**.

A picture containing graphical user interface

Description automatically generated

Figure 7: Secure AODV log file

The first line represents the packet type 30701 = RREQ. The second line is the message logged by SAODV when malicious node tries to decrypt the message.

# .........................................

**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**

**.........................................**

**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